ISec Lab #13

Hardening básico de Linux
Permisos y Configuraciones

marzo de 2011

Ferran Pichel Llaquet
fpichel<arroba>isecauditors.com

Introducción

En esta ocasión nuestro IsecLab tratará sobre un aspecto muy importante a la hora de administrar un servidor, su securización (ing: hardening). Cómo los sistemas operativos de la familia *nix son los más usados y los que nos permiten una mayor configuración, el presente artículo se referirá a dichos sistemas.

La configuración de un sistema operativo no es algo trivial y requiere un cuidado minucioso de cada uno de los aspectos. Por esa razón se introducirán los aspectos más básicos como son los permisos y configuraciones base.

En el presente texto se intentará dar una explicación detallada de como funcionan los permisos y atributos de los ficheros y como configurarlos correctamente para evitar accesos inapropiados a datos sensibles en el servidor. Asimismo, se hablará de aspectos relacionados con la configuración, tanto del sistema como de posibles servicios que en el se alojen. Se intentara enfocar el texto tanto para el acceso remoto como el local, ya que ambos aspectos son sumamente importantes para el buen funcionamiento de éste.

INTRODUCCIÓN..... 2

1. PERMISOS..... 4
   1.1. Permisos..... 4
   1.2. Modificación..... 5
   1.3. Asignación coherente..... 7
2. CONFIGURACIONES..... 11
   2.1. Seguridad..... 11
      2.1.1. Seguridad Física..... 11
      2.1.2. Limitando el acceso físico..... 11
      2.1.3. Configurando el gestor de arranque..... 11
   2.2. Seguridad Local..... 12
      2.2.1. Seguridad del sistema de ficheros..... 12
      2.2.2. Cambio periódico de contraseña..... 13
      2.2.3. Características de las contraseñas..... 14
      2.2.4. Controlando RunLevels (rcX.d)..... 14
      2.2.5. Inhabilitando Ctrl.+Alt+Del..... 15
      2.2.6. Estableciendo los límites..... 15
      2.2.7. Opciones de montaje de las particiones..... 17
      2.2.8. Cron..... 17
      2.2.9. Generando Logs..... 19
      2.2.10. Rotación de Logs..... 20
      2.2.11. Notificaciones de Logs..... 21
3. CONCLUSIONES..... 22

1. PERMISOS

1.1. Permisos

Linux es un sistema multiusuario, lo que conlleva a mantener la privacidad de estos y un control general para que no todos puedan hacer lo que quieran y comprometer así el sistema. En la familia de operativos *nix se utilizan los permisos para ello, permitiendo o no realizar acciones concretas al resto de usuarios no propietarios del fichero, o incluso al propio propietario.

Cada usuario tiene un identificador UID (User *Identification*) y un grupo identificativo GID (Group *Identification*). Estos usuarios pueden realizar tres acciones distintas en un fichero:

- Ejecutar (ing: *Execute*).
- Leer (ing; *Read*).
- Escribir (ing: *Write*).

Cuando un usuario crea un fichero, el UID y GID propietarios de éste son los del propio usuario, como es lógico. Pero tenemos más permisos que podemos configurar, de hecho si hacemos “ls -l” veremos una cadena, al inicio de cada línea parecida a la siguiente:

-rwx------

Estos permisos nos indican que el usuario propietario del fichero tiene permisos de Lectura (R), Escritura (W) y Ejecución (X), que no es un directorio y que los miembros del grupo y el resto de usuarios del sistema no pueden hacerle nada.

Para interpretar correctamente esta cadena se debe entender que cada espacio corresponde a un Bit de permisos, y estos al mismo tiempo se pueden agrupar en cuatro bloques:

-	Indica si es o no un directorio.
rwx	Permisos para el propietario del Fichero.
---	Permisos para los usuarios del grupo del Fichero.
---	Permisos para el resto de los usuarios.

Bit 0: En caso de que el fichero sea un directorio nos aparecerá una “d”.

Los Bits de permisos tienen distinto comportamiento según se trate de un fichero o de un directorio, a continuación se describen las diferencias:

Bit	Fichero	Directorio
R	Lectura del fichero.	Lectura del contenido del directorio.
W	Escritura del fichero.	Mover y borrar ficheros en un directorio.
X	Ejecución del fichero.	Poder acceder a un directorio.

Veamos a continuación un par de bits algo especiales.

SUID (S)

El bit SUID es un flag especial que sirve para modificar temporalmente los privilegios del usuario que ejecuta un programa. Si este bit se activa, el usuario que ejecute el fichero pasará a tener los mismos privilegios que el propietario de este.

Imaginemos por un momento que el comando "/bin/ls" tiene este bit activo, lo cual es una barbaridad. Ahora cualquier usuario que pueda ejecutar "/bin/ls", es decir, prácticamente todos, podrá ver el contenido de todos y cada uno de los directorios aunque no tenga permisos para ello. ¿Por qué ocurre esto? Basta con mirar el propietario de dicho comando:

```
-rwsr-xr-x 1 root root 89632 2006-05-05 20:15 /bin/ls
```

Ahí se puede observar que el propietario es root, lo que significa que el "ls" no se ejecutará como usuario, sino como root, indiferentemente de que usuario lo esté ejecutando. Además vemos activado el bit SUID representado por la letra "s", que se encuentra en el lugar de "x", simplemente porque esta directamente relacionado.

Este bit se puede asignar para que al ejecutar la aplicación se cambie el usuario, el grupo o ambas características del usuario que ejecuta.

STICKY BIT (T)

El sticky bit es un flag adicional que actualmente no tiene función alguna en los ficheros, pero en cambio si la tiene cuando se trata de directorios. Se representa con la letra "T".

Si este bit se encuentra activo en un directorio, los ficheros del directorio pueden ser borrados o renombrados únicamente por el propietario del directorio o por el propietario del fichero, lo cual puede ser de gran ayuda cuando hablamos de directorios donde se encuentran ficheros sensibles del sistema u otra información necesaria para el buen funcionamiento de éste.

Veamos ahora como modificar estas características cómodamente.

1.2. Modificación

Después de esta ojeada rápida a los permisos veamos como modificarlos de una manera cómoda e intuitiva. Como el bit 0 nos indica si se trata o no de un directorio, no podemos modificarlo, sino que es asignado en el momento de crear el fichero/directorio.

Como ya se ha dicho, cuando un fichero es creado se le asigna el propietario (ing: owner) y el grupo de ese usuario. Estos valores pueden ser cambiados utilizando el comando "chown" (Change Owner). La sintaxis básica se muestra a continuación:

```
chown <usuario>:<grupo> fichero
```

De esta manera conseguimos que el fichero pertenezca ahora a <usuario> y al grupo <grupo>. Si uno de los dos campos no se especifica, queda sin cambiar. Obviamente este comando debe ser ejecutado por el anterior propietario del fichero.

Veamos ahora que ocurre con los bits de permisos. Tenemos tres bloques con tres bits cada uno: “rwx”. Para modificarlos debemos usar el comando “chmod” (Change Modes), este comando tiene varias sintaxis, a continuación se comentan las más usadas:

- **Mediante la suma del valor de cada BIT:**

  Si pensamos que cada bloque de tres bits es independiente del resto, podemos diferenciar cada uno según su valor en binario, que dependerá de la posición en la que se encuentre. Es decir, si tenemos “rxw” (todos los bits a 1) y hacemos la suma binaria de:

  100 + 010 + 001 = 111 -> 7 (en base 10)

  Dicho de otra manera, cada permiso tiene un único valor, y basta con sumarlos para asignárselos a un fichero:

Permiso	*Binario*	*Decimal*
Read (r)	100	4
Write (w)	010	2
Execute (x)	001	1

  Teniendo en cuenta que son tres bloques de bits, ahora solo se debe elegir que permisos dar a cada bloque. Por ejemplo, si se quiere dar “rxw” (7) al propietario, “rw” (6) al grupo y “r” (4) al resto, basta con poner:

  chmod 764 <fichero>

  Y luego al hacer “ls -l <fichero>” veremos que los permisos han sido modificados:

  -rwxrw-r--

  Hasta ahora se jugado solamente con los bits “rwx”, veamos como activar y desactivar el bit SUID i STICKY.

  Para poder jugar con estos bits hay que añadir un dígito al comando anterior, por ejemplo:

  # chmod 1755 <fichero>

  En este caso se esta asignando:

1	Se le asigna el bit STICKY
7	“r”+”w”+”x” para el propietario
5	“r”+”x” para el grupo
5	“r”+”x” para el resto de usuarios.

  A continuación se muestran los bits que se asignan dependiendo del primer dígito:

1	Sticky Bit
2	SUID del grupo
4	SUID del usuario

Sumando los valores podemos especificar las opciones necesarias, el funcionamiento es análogo al de los permisos "rwx".

- **Mediante la letra y el bloque que queremos utilizar:**

  Otra manera de hacer lo mismo es utilizando las letras directamente, "r", "w", "x" para los permisos y luego también podemos referirnos a un bloque específico:

u	***User (Propietario)***
g	Grupo (Group)
o	El resto (Other)
a	Todos (All)

  Finalmente para asignar los mismos permisos que en el ejemplo anterior, debemos ejecutar "chmod" una vez para cada bloque de bits usando la siguiente sintaxis:

```
chmod u=rwx <fichero>
chmod g=rw <fichero>
chmod o=r <fichero>
```

  Podemos también desactivar permisos o bien añadirlos utilizando los símbolos "-" y "+" respectivamente. Por ejemplo:

  - Quitamos permisos de ejecución a todos: chmod a-x <fichero>
  - Añadimos permisos de ejecución a todos: chmod a+x <fichero>

    Si se quisiera poner los mismos permisos, por ejemplo, para el grupo y el propietario, se puede hacer con una sola sentencia:

```
chmod ug=x <fichero>
```

    Así se ha asignado ejecución tanto al propietario como al grupo. Esta sintaxis también es posible con el operando "+" y "-".

    Finalmente, podemos asignar el bit SUID y el STICKY utilizando las letras "s" y "t" respectivamente, y siempre usando la misma sintaxis:

```
chmod u+s <fichero>
chmod +t <fichero>
```

    Si no se especifica el bloque de bits ("u", "g", "o" o "a") se toma por defecto "a", con lo que se asigna el permiso a todos los bloques.

1.3. Asignación coherente

Aunque la asignación de permisos a un fichero es algo prácticamente trivial, hacerlo adecuadamente no lo es tanto. No es nada raro encontrar servidores donde se permite listar directorios o incluso ver código de aplicaciones y scripts que se usan aunque "en teoría" no se tenga acceso a ellos. Este tipo de problemas acostumbran a ser causados por una mala gestión de los permisos en los ficheros sensibles.

Vamos a empezar por el fichero donde se guardan los usuarios y passwords del sistema encriptados, el fichero "/etc/shadow". Aunque los permisos por defecto varían según la distribución que se esté usando, no es así en cuanto al uso que se hace de éste.

Cuando un usuario cambia el password o bien se registra en el sistema, lo hace utilizando el programa “/bin/login”, el cual tiene activado el atributo, o bit, SUID. Dicho de otra manera, mientras “login” esta en ejecución, el usuario que lo ejecuta es “root”.

Ahora si miramos los permisos del fichero “/etc/shadow” veremos algo así:

```
-rw-r----- 1 root shadow 735 2006-09-29 12:51 /etc/shadow
```

Puede variar el grupo o incluso los permisos. Ahora tengamos en cuenta que uso se hará de este fichero, es decir, que usuario va a utilizarlo y para qué. Como se ha comentado hace un momento este fichero solamente será leído y escrito por root, a no ser que el administrador quiera que alguno más lo haga. ¿Entonces para que queremos que el grupo también sea capaz de leer el fichero ?

A menos que sea realmente necesario, se debería eliminar todo acceso a este fichero por parte de cualquiera, exceptuando “root”. Precisamente hará un año o así, salió un bug en la syscall “chown” que permitía cambiar el grupo a cualquier fichero del sistema, por lo tanto, si tenemos permiso de lectura en el “/etc/shadow” para el grupo, y este es cambiado, el nuevo grupo también los tendrá y por lo tanto podrá leer el fichero.

Para poder evitar estas situaciones antes de que no haya marcha atrás, lo mejor es darle permisos de lectura y escritura sólo y exclusivamente al usuario root:

```
# chmod 600 /etc/shadow
# ls -l /etc/shadow
-rw------- 1 root shadow 735 2006-09-29 12:51 /etc/shadow
```

Sigamos ahora con la gestión multiusuario. En la familia de operativos *nix los usuarios tienen su “home” en “/home/<usuario>/” por defecto. Exceptuando el administrador “root”, el home del cual se encuentra en “/root” generalmente. Veamos como mantener la privacidad de todos los usuarios y como proteger el “home” del administrador para que los usuarios no puedan ver que hay en el.

Por defecto, muchas distribuciones de Linux permiten listar el contenido del directorio “/root” al resto de usuarios del sistema y eso no es bueno. Debemos pues realizar cambios para que esto no ocurra, simplemente basta con asignar permisos de ejecución, lectura y escritura exclusivamente al usuario root:

```
# chmod 700 /root
```

Y ya tenemos el directorio protegido contra el listado de ficheros. Sería recomendable realizar el mismo comando a todos los “homes” de los usuarios para evitar que unos vean el contenido de los otros:

```
# chmod 700 /home/*
```

En cambio si queremos hacer grupos de usuarios, y queremos que todos los que formen parte de dicho grupo puedan acceder a sus “homes” mutuamente. Simplemente debemos añadir los usuarios a un grupo y hacer lo siguiente:

```
chown :<grupo> /home/user1
chown :<grupo> /home/user2
...
chown :<grupo> /home/userN
```

Y finalmente debemos darles permiso para que puedan listar y ejecutar, por ejemplo, en todos los homes de los usuarios del grupo:

```
chmod g+wx /home/user1
...
chmod g+wx /home/userN
```

Con estos 4 comandos ya tenemos configurados los permisos para los usuarios y para el propio root. A continuación veremos como configurar los logs del sistema para evitar lecturas por parte de otros usuarios distintos a root o el encargado de los logs.

Supongamos el caso que un usuario es el encargado de los logs, llamemos le “logger”, y queremos que sea el único con acceso. Bastará entonces con cambiar el propietario de los logs a “logger” y darle los permisos adecuados, tal y como se ha hecho con los “homes” de los usuarios. Claro que a lo mejor interesa que “logger” solo pueda leer los ficheros, sin llegar a modificarlos, y no queremos que pueda cambiarse los privilegios porque entonces podría saltarse esa prohibición.

Entonces se podría crear un grupo llamado “logs”, añadirle el usuario “logger” y cambiar el grupo del directorio de los logs, además de asignarle permisos de solo lectura para el grupo. Pues vamos allá.

Primeramente vamos a crear el grupo:

```
# addgroup logs
Añadiendo el grupo «logs» (1002)...
Hecho.
```

Segundo, añadimos el usuario especificado a dicho grupo:

```
# gpasswd -a logger logs
Añadiendo al usuario logger al grupo logs
```

Ahora cambiamos el grupo del directorio de los logs, generalmente “/var/log”:

```
# chown :logs /var/log/
```

Y asignamos los permisos pertinentes (u=rwx , g=rx, o=-):

```
# chmod 750 /var/log
```

Ahora que ya tenemos el directorio securizado para el resto de usuarios sólo es necesario permitir leer todos los ficheros del directorio al usuario “logger”, como no es el propietario ni forma parte del grupo de muchos de los ficheros, se deberá dar permisos de lectura al resto de usuarios “o”. La gracia está en que con los permisos del directorio “/var/log” ya no se permite la entrada a éste, entonces dar permisos de lectura al resto, significa dar permisos de lectura a “logger”, porque los demás usuarios no podrán entrar en el directorio:

```
# chmod o+r /var/log/*
```

Finalmente vemos los permisos del directorio y de su contenido:

```
# ls -ld /var/log
drwxr-x--- 5 root logs 4096 2006-10-03 10:08 /var/log
# ls -l /var/log
[..]
-rw-r--r-- 1 root adm  133812 2006-10-03 11:08 messages
[..]
```

Ya tenemos los logs protegidos contra los ojos curiosos que puedan haber en nuestro servidor. Hay que tener en cuenta que bloquear el acceso al recurso “/var/log” puede acarrear problemas en alguno de los comandos que cogen información de ahí, como por ejemplo “lastlog”, sólo podrán realizar correctamente dicho comando el usuario “root” y los miembros del grupo “logs”.

Este pequeño ejemplo realizado sobre el directorio de logs, es aplicable a cualquier otra operación de este estilo, donde un usuario es el encargado de un recurso en concreto. Además si queremos añadir un segundo usuario para realizar la misma acción bastará con añadirlo al grupo, “logs” en este caso, y ya tendrá los mismos privilegios que el primero.

Dependiendo del tipo de servidor, los permisos deberán ser más o menos restrictivos, eso ya depende de cada caso en concreto y del administrador que esté al cargo.

Lo que si es generalizable es el minucioso cuidado que se debe tener con los ejecutables con el bit SUID activo, se deben extremar las precauciones y mantenerlo fuera del alcance de los usuarios. De lo contrario, si existiera un bug en algún ejecutable de este tipo o se hiciera un mal uso de éste, un usuario podría llegar a realizar una escalada de privilegios y ser entonces “root”, lo que comprometería el sistema.

2. Configuraciones

2.1. Seguridad

La seguridad local puede atacarse desde dos puntos de vista, el acceso físico al sistema y el acceso virtual. A lo largo del capítulo se irán comentando aplicaciones que se encargan de gestionar sucesos. Éstas utilizan, por lo general, el mail local para comunicarse con el administrador del sistema, por esa razón sería necesario instalar un servidor de correo para permitirle dicha comunicación. De lo contrario, algunas aplicaciones comentadas a continuación podrían no funcionar correctamente.

Comentaremos a continuación sobre como podemos proteger nuestro sistema de las manos maliciosas de la gente que lo envuelve.

2.1.1. Seguridad Física

Aunque parezca un aspecto poco importante cuando se habla de securización de un sistema, la realidad es que son pocos, o muy pocos, los servidores o sistemas, en general, que tienen una buena política de seguridad en este aspecto.

2.1.2. Limitando el acceso físico

Aunque parezca algo obvio, se debe mencionar. Algunos administradores creen, y en casos es así, que la mayoría del personal que se mueve por su alrededor no saben realmente como funciona el servidor, y que, consecuentemente, no van a tocar nada porque ni tan solo sabrían encenderlo. Pero la información cada día se extiende más y poco a poco la gente va conociendo…

Basta con imaginar solo por un momento, la típica máquina que contiene los credenciales de acceso en un post-it al lateral. ¿Qué pasaría si alguien con suficientes conocimientos pudiera entrar en la sala? Quedaría toda la seguridad comprometida aun teniendo todos los servicios y sistemas actualizados.

Sería interesante desarrollar unas políticas de acceso para controlar los usos y usuarios de la sala afectada.

2.1.3. Configurando el gestor de arranque

Otro punto importante a tener en cuenta es el inicio del operativo, ya que cualquier persona podría acceder a los parámetros de arranque del gestor y forzar un inicio en modo “singular”, es decir, en modo de recuperación del sistema, por lo que sería superusuario y tendría el control total de la maquina.

Otro posible escenario de intrusión es la funcionalidad de edición añadida cuando se muestra el menú, es decir, poder modificar los parámetros correspondientes al arranque del sistema. Si esta edición no se encuentra protegida bajo contraseña, un usuario mal intencionado sería capaz de ejecutar un comando justo arrancar el sistema, y como superusuario. Por ejemplo, en la imagen del kernel de Linux hay un parámetro de arranque: “init”, el cual nos permite ejecutar un comando al iniciar.

Supongamos el caso de un gestor de inicio GRUB, en el que añadimos una entrada configurada de la siguiente manera:

```
title    Linux
root     (hd0,0)
kernel   /vmlinuz root=/dev/hda4 ro quiet splash
boot
```

Si no estuviera protegido con contraseña, al arrancar "GRUB" podríamos presionar la tecla "e" y editar estas líneas. Simplemente con la siguiente modificación ya tendríamos control total del sistema:

```
kernel   /vmlinuz root=/dev/hda4 ro quiet splash init=/bin/bash
```

Para evitar esto, basta con fijar una contraseña de en el gestor de arranque, tal y como ya se ha comentado, o bien utilizar el password de la BIOS.

Para activar el password al gestor de arranque GRUB se debe:

1. Acceder al fichero de configuración. p.e. '/boot/grub/menu.lst'
2. Activar la opción de contraseña añadiendo una de estas líneas:
   a. En claro (inseguro y totalmente desaconsejable):
      password <CONTRASEÑA>
   b. Encriptada en md5 (recomendado):
      password –md5 <HASH MD5 de la Contraseña>

Tanto una solución como la otra supone un punto negativo en entornos de servidores de administración remota ya que no podemos forzar el reinicio de la máquina a distancia. Y la encriptación de la contraseña en el fichero de configuración de GRUB es realmente recomendable, ya que en muchos casos, por defecto, los usuarios pueden visualizar el contenido de dicho fichero.

2.2. Seguridad Local

Veamos a continuación unas pinceladas sobre la securización local para los usuarios, una vez ya tienen la sesión activa.

2.2.1. Seguridad del sistema de ficheros

Se deberían asignar los permisos adecuados a los directorios sensibles del sistema, para evitar miradas de terceros, tanto propietarios como permisos. De esta manera aseguramos la privacidad de los usuarios:

```
# chmod 700 <fichero>                      (permisos)
# chown <usuario>:<grupo> <fichero>   (propietario)
```

Adicionalmente podemos utilizar un cifrado de cualquier tipo securizar ficheros concretos o bien montar una partición cifrada de manera que la operación de cifrado sea transparente. Algunas herramientas interesantes para realizar estas acciones:

Aplicación	Home Page	Descripción
Enc FS	http://arg0.net/encfs	Creación de directories cifrados.
Loop AES	http://sourceforge.net/projects/loop-aes/	Creación de particiones cifradas.
GNU Privacy Guard (GPG)	http://www.gnupg.org/	Version de PGP libre.
OpenSSL	http://www.openssl.org/	Versión Abierta de SSL

Veamos a continuación un pequeño ejemplo básico de uso de la última herramienta mostrada en la tabla, OpenSSL. Esta tiene multitud de opciones que por motivos de espacio no van a ser descritas en este texto. Primero creamos un fichero y lo ciframos:

```
# openssl enc –e –bf –in texto_plano.txt –out texto_cifrado.sec
enter bf-cbc encryption password: <contraseña>
Verifying - enter bf-cbc encryption password: <contraseña>
```

Una vez hecho esto el fichero cifrado es “texto_cifrado.sec”. En este caso se ha usado “blue-Fis.” (-bf) como algoritmo de cifrado. A continuación se muestra como descifrar el fichero:

```
# openssl enc –d –bf –in texto_cifrado.sec  –out texto_original.txt
enter bf-cubic encryption password: <contraseña>
```

Ahora ya volvemos a tener el texto original.

2.2.2. Cambio periódico de contraseña

Por mucho que se proteja un sistema, algunos usuarios son descuidados y no se preocupan lo suficiente a la hora de asignar una contraseña o bien a la hora de conservarla. Cuando eso ocurre, un potencial atacante ya habría superado la primera barrera, la remota, y por lo tanto tendría acceso al sistema gracias a la cuenta débil proporcionada por el usuario legítimo.

Para evitar esta clase de problemas sería interesante obligar a los usuarios a redefinir su contraseña cada cierto periodo de tiempo, así aseguramos que aunque el usuario pierda la contraseña o alguien malévolo la posea, al cabo de cierto tiempo esos credenciales se invalidarán, consiguiendo así que el atacante no puedo reutilizarlos. Una aplicación creada para este fin es “chage” que la podemos encontrar prácticamente en todo sistema *nix instalada por defecto.

Su ayuda es bastante explicita y clarificante, se ejecuta de la siguiente manera:

```
# chage <opciones> <usuario>
```

Con las opciones podemos asignar un día concreto de expiración, inhabilitar una cuenta después de cierto tiempo en desuso, el número máximo y mínimo en los que habrá que cambiar la clave,.... (man chage). A continuación se muestra un pequeño ejemplo:

```
# chage –M 30 –W 5 usuario
```

En este caso estamos diciendo que el usuario “usuario” deberá cambiar la contraseña en un máximo de 30 días, y se le avisará durante los 5 días anteriores de que debe hacerlo. Finalmente podemos comprobar que realmente estas opciones han sido asignadas con la opción “-l” que nos lista las opciones concretas para el usuario especificado:

```
# chage -l usuario
Último cambio de contraseña:                                    may 28, 2007
La contraseña caduca:                                           jun 27, 2007
Contraseña inactiva:                                            nunca
La cuenta caduca:                                               nunca
Número de días mínimo entre cambio de contraseña:               0
Número de días máximo entre cambio de contraseñas:              30
Número de días de aviso antes de que expire la contraseña:      5
```

2.2.3. Características de las contraseñas

Aunque en el apartado anterior se comentaba como realizar cambios periódicos y forzar a los usuarios a cambiar la clave, también sería interesante, que esta clave tuviera un mínimo de robustez. Para ello podemos hacer uso de las “cracklib”, basta con instalarlas en el sistema y así podremos especificar las opciones que creamos convenientes para la creación de nuevas contraseñas en el sistema.

Podemos bajarlo utilizando algún gestor de paquetes propio de la distribución usada, o bien de la web: http://sourceforge.net/projects/cracklib

Una vez instalado en el sistema, podemos pasar a la configuración, nos dirigimos al fichero “/etc/pam.d/passwd”. Ahí podemos añadir la siguiente línea (o modificarla en caso de existir):

```
password required pam_cracklib.so difok=3 retry=2 minlen=8
```

Con esto especificamos que ha de haber una diferencia mínima de 3 letras, habrá un máximo de 2 reintentos y la longitud mínima será de 8. Es posible que aun habiendo añadido esta línea no funcione correctamente, esto puede deberse a que haya ya otra directiva anterior que sobrescriba la nuestra, por ejemplo podría haber una como la siguiente:

```
password required pam_unix.so nulloc obscure min=4 max=8 md5
```

Basta con eliminar/comentar dicha línea para que todo funcione correctamente.

2.2.4. Controlando RunLevels (rcX.d)

Es importante para poder securizar un sistema, entender como funciona. En el caso de los sistemas *nix se producen cambios de estado, por ejemplo, cuando se inicia se encuentra a un estado distinto a cuando se esta apagando el sistema.

Pues bien, simplemente añadiendo/borrando/modificando algunos ficheros del sistema podemos customizar cada uno de los estados, ejecutando lo que queremos al inicio o fin de un estado concreto. El directorio del que hablamos varía según la distribución, algunos posibles son:

Directorios posibles	
/etc/rcX.d/	La X es un dígito que nos indica el runlevel al que afecta el directorio.
/etc/init.d/rc.d/rcX.d	

La lista de RunLevels es la siguiente:

RunLevel	*Descripción del estado*
0	Apagado
1	Modo de un solo usuario
2-5	Modo multiusuario (habitual).
6	Reinicio
S	Inicio del sistema por defecto.

Nota: El runlevel inicial puede variar con lo que se debería mirar /etc/inittab la acción “sysinit”. En el siguiente apartado se comenta más profundamente inittab.

Si entramos en cualquier de estos directorios veremos un conjunto de scripts encargados de configurar e inicializar lo necesario. Si nos fijamos en el patrón que siguen todos los nombres, veremos que es el siguiente:

```
<S|K><número><NombreScript>
```

Campo	Descripción
S ó K	S: Se ejecuta al iniciar el runlevel. K: Se ejecuta al finalizar el runlevel.
número	Determina el orden de ejecución de entre el grupo S ó K
NombreScript	El script concretamente

Finalmente basta con crear el script que deseemos y ponerlo en el directorio que se prefiera. Finalmente, el comando "runlevel" nos muestra el RunLevel en el que nos encontramos.

Por ejemplo:

```
# runlevel
N 2
```

En este caso nos encontramos en el RunLevel 2, con lo que podríamos añadir cualquier fichero que quisiéramos ejecutar al inicio de nuestro runlevel simplemente creando el fichero: */etc/rc2.d/S13ScriptPropio* con un script en su interior. Evidentemente, para que toda la ejecución transcurra sin problemas, dicho fichero debe tener permisos de ejecución: *chmod +x /etc/rc2.d/S13ScriptPropio*

2.2.5. Inhabilitando Ctrl.+Alt+Del

Dependiendo del entorno en el que se encuentre el servidor, puede ser aconsejable deshabilitar el reinicio por teclado utilizando la secuencia de teclas Ctrl.+Alt+Del.

Para hacerlo basta con ir al fichero "/etc/inittab" que se encarga del comportamiento de los RunLevel, o estados del sistema. Ahí podemos encontrar la siguiente línea:

```
# What to do when CTRL-ALT-DEL is pressed.
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
```

Parámetros	Significado	Valor
Identificador	Id único en /etc/inittab	ca
RunLevels	RunLevels a los que afecta	12345
Acción	En que acción se ejecutará el proceso	Ctrlaltdel
Proceso	Proceso a ejecutar	/sbin/shutdown
Parámetros	Parámetros del proceso	-t1 –a –r now

Esta sentencia define la acción que se llevará a cabo cuando se produzca la acción "CtrlAltDel" en cualquiera de los runlevel (1-5), excepto el estado de apagado (0) y reboot(6). Lo que ocurrirá es que se llamara a "/sbin/shutdown" con los parámetros especificados y por lo tanto se cerrará el sistema "-r now" (al instante).
Añadiendo el carácter '#' al inicio de la segunda línea ya es suficiente. Es muy interesante todo el contenido de este fichero de configuración, se debería estudiar minuciosamente cada uno de sus parámetros y opciones para amoldarlo mejor a las necesidades de cada caso concreto.

Para poder conseguir la lista de todas las funcionalidades y "acciones" posibles basta con echarle una ojeada al "man inittab".

2.2.6. Estableciendo los límites

Para poder tener un buen control sobre el uso que hacen los usuarios del sistema, en los sistemas *nix existen herramientas para especificar barreras. Por ejemplo, podemos se puede definir un máximo de logias simultáneos, un tamaño máximo de espacio, un número máximo de procesos,….

Para empezar podemos ver todas las opciones activadas que hay actualmente y su valor utilizando el comando virtual (forma parte de bash): # ulimit -a

La salida del comando muestra la información siguiente:

Descripción	Argumento	Valor Actual
core file size (blocks)	-c	0
data seg size (kbytes)	-d	Unlimited
max nice	-e	0
file size (blocks)	-f	Unlimited
max memory size (kbytes)	-i	Unlimited
pending signals	-m	Unlimited
open files	-n	1024
cpu time (seconds)	-t	Unlimited
max user processes	-u	Unlimited
virtual memory (kbytes)	-v	Unlimited

En el output podemos ver el nombre del parámetro, juntamente con el argumento necesario para variarlo y finalmente el valor. Por ejemplo, ahora hay máximo de fichero que se pueden abrir a 1024, para cambiarlo a por ejemplo a 20, bastará con ejecutar:

```
# ulimit –n 20
```

Tan simple como efectivo. Por otra parte tenemos el fichero de configuración que se encuentra en "/etc/security/limits.conf". En este fichero de configuración se puede observar una explicación detallada de todos los campos y como usarlos.

Las sentencias son de la estructura siguiente:

<domain> <type> <item> <value>

Opción	Value
Domain	Área de afectación del límite (usuario, grupo,…).
Type	Soft: Se avisa de que se ha superado el límite pero no se ejecuta ninguna acción. Hard: Al pasar el límite se corta la ejecución del proceso que lo sobrepasa.
Item	Propiedad a modificar (lista completa en el fichero de configuración).
Value	El valor con el que queremos inicializar.

Hay que tener en cuenta que un usuario no podrá modificar sus propios privilegios para que sean MENOS restrictivos. Por otra parte, cuando se habla de la gestión de multiusuario, serán de vital importancia los siguientes "ítems":

Item	Descripción
nproc	Número máximo de procesos.
maxlogins	Máximo de Logias simultáneos.
priority	Prioridad

Ahora solo hace falta estudiar cuales son las mejores limitaciones para cada caso.

2.2.7. Opciones de montaje de las particiones

Cuando se monta un sistema servidor hay que tener en cuenta la posibilidad de dividirlo en varias particiones y asignando en cada una de ellas una serie de propiedades.

Gracias a la configuración que se encuentra en el fichero "/etc/fstab" es posible asignar varias opciones a cada una de las particiones, algunas de las opciones se presentan a continuación:

Opción	*Descripción*
nodev	Inhabilita la interpretación de caracteres o bloques de dispositivos. (para realizar la acción contraria: dev)
noexec	Inhabilita la ejecución de binarios en la partición. (para realizar la acción contraria: exec)
nouser	No permite que un usuario pueda montar la partición. (para realizar la acción contraria: user ó users)
nosuid	No permite ficheros SUID en la partición. (para realizar la acción contraria: suid)
Rw	Monta la partición con Escritura y Lectura (R+W)
Ro	Monta la partición como Sólo Lectura (Read Only)
noauto	Obliga a montar a mano la partición, no automáticamente al inicio. (para realizar la acción contraria: auto)
Default	Opciones por defecto que por lo general incluyen: Rw, suid, dev, exec, auto, nouse

Nota: Para una lista completa: man 8 mount

Gracias a todos estos parámetros y opciones es posible diseñar una estructura óptima y coherente. Una posibilidad podría ser particionar los directorios donde se alojan los binarios "/bin" y "/sbin", y que estas fueran las únicas particiones en las que se permitiera el "exec". O, en sistemas cerrados y compactos, no permitir la escritura en particiones del sistema donde no debería haber más cambios, añadiendo la opción "ro".

Como conclusión podemos decir que hay un sinfín de propiedades que pueden ser asignadas y por lo tanto dan muchas configuraciones distintas posibles.

2.2.8. Cron

El daemon cron permite realizar acciones regularmente. Así conseguimos que el sistema se actualice, por ejemplo, una vez a la semana. Para poder utilizarlo basta con instalarlo utilizando el repositorio de paquetes de la distribución pertinente o bien bajarlo de la web oficial: ftp://ftp.isc.org/isc/cron/

Una vez instalado en el sistema solo hace falta configurarlo y luego añadirlo al arranque. Para un administrador, cron podría dividirse en dos funcionalidades, la gestión general del sistema y la particular para cada usuario.

Empecemos con la gestión que se puede llevar a cabo en el propio sistema gracias a este daemon. La configuración general del programa se encuentra en el fichero "/etc/crontab", si se abre el fichero se pueden observar las siguientes opciones:

```
#m h dom mon dow user command
```

Dónde:

Campo	*Significado*
m	Minuto
h	Hora
dom (Day of Month)	Día del mes
mon (Month)	Mes
dow (Day of the Week)	Día de la semana
user	Usuario que ejecuta el comando
command	Comando

Como se puede deducir de la multitud de campos que encontramos en las opciones, se puede llevar un cálculo minucioso de cada programación. Por ejemplo la siguiente sentencia haría que se hiciera un backup en una partición a parte de todos los homes de los usuarios cada domingo a las 7 de la tarde:

```
#m h dom mon dow user command
0   19   *     *     sun    root    tar –cjvf /mnt/partición/homes.tar.bz2   /home
```

Cuando se especifica un asterisco (*) significa que ese campo carece de importancia para la ejecución, en este caso en concreto, da igual el mes o el día del mes, ya que lo único que se quiere es realizar una copia comprimida de los "home" cada domingo a las 7, indiferentemente del día del mes que sea.

Como se ha comentado al empezar el apartado, también hay una funcionalidad añadida de "cron" que permite que usuarios no-privilegiados del sistema sean capaces de realizar tareas regularmente. Para ello tenemos el binario "crontab" que se encarga de hacer una copia para el usuario de un fichero de configuración. Si ejecutamos el comando "crontab" veremos las opciones que este tiene:

Arg.	*Descripción*
-e	Edita el fichero
-l	Lista el fichero
-r	Borra el fichero

Si ejecutamos "contrab –e" simplemente se nos abrirá el fichero con un editor y podremos así editarlo, el formato es exactamente el mismo que en el general, sólo que no hay campo de usuario. Cada usuario tendrá su propio fichero de configuración y los comandos se ejecutarán con sus privilegios.

¿Pero que pasa si precisamente esta característica nos supone un problema? Talvez sea preferible que los usuarios no puedan programar tareas en el sistema, ya sea por la falta de necesidad o por los peligros que esto conlleva (bombas DoS programadas a través del crontab). Por suerte también hay solución para este problema, si miramos el man de "cron", veremos como se nombran dos ficheros: "/etc/cron.allow" y "/etc/cron.deny". Estos sirven precisamente para gestionar los accesos a los usuarios.

Por defecto estos ficheros no existen y por lo tanto todos los usuarios pueden hacer lo que quieran. En concreto su funcionamiento es el siguiente: el fichero "/etc/cron.allow", en caso de existir, es el único al que "cron" hace caso, es decir, los usuarios que no aparezcan en esta lista no tendrán acceso a "crontab". En cambio si en vez de querer evitar a todos los usuarios deseamos evitar solo a unos pocos, basta con añadir su nombre al fichero "/etc/cron.deny" e inhabilitar el fichero "/etc/cron.allow" borrándolo o moviéndolo.

Por ejemplo, si queremos vetar a todo el mundo y creamos el "/etc/cron.allow", cuando algún usuario intente usar "crontab" recibirá el siguiente mensaje:

```
root # touch /etc/cron.allow

usuario $ crontab –e
You (usuario) are not allowed to use this program (crontab)
See crontab(1) for more information
```

Ya tenemos suficiente información para poder configurar este servicio como mejor nos parezca.

2.2.9. Generando Logs

En el sistema es muy importante estar al corriente de lo que sucede, para ello existen aplicaciones de "Logs" que se encargan de guardarlos en ficheros separados para una mejor organización.

Una buena política de seguridad empieza por una organización centralizada de los logs, es decir, enviando los eventos/registros a una máquina central que se encargará de mantener a salvo todo lo relacionado con el comportamiento de cada uno de los servidores. Evitando así que estos sean comprometidos a causa de una intrusión realizada por un usuario malévolo. Para poder analizar y comprender correctamente todo este flujo de información basada en marcas de tiempo (ing: timestamp) es recomendable disponer de un servidor ntpd (NTP: Network Time Protocol) para que todos los hosts estén sincronizados en tiempo. Y así los logs sean más coherentes.

Algunas herramientas que permiten realizar toda esta gestión son "syslogd" y "syslog-ng". A continuación veremos una configuración básica de "syslogd" por ser el más habitual.

El fichero de configuración sigue el esquema siguiente:

```
[tipo de servicio].[nivel de alerta]          [fichero|host destino]
```

Dónde el tipo de servicio puede ser desde el propio "cron" al "mail", temas de autentificaciones…. Los tipos de alerta en cambio pueden tener varios niveles, algunos más destacados son: "info" (informativos), "warn" (avisos) y "err" (errores). Finalmente el tercer campo corresponde al fichero destino del sistema o al host destino. Por ejemplo la siguiente línea:

```
mail.info                    /var/log/mail.info
mail.warn                    /var/log/mail.warn
mail.err                     /var/log/mail.err
```

En este caso se están enviando todas las alertas informativas del servicio de correo a "/var/log/mail.info" y bueno, el resto ya se puede deducir. Finalmente, si quisiéramos enviar todos los logs de mail (indiferentemente del nivel) a un host central, bastaría con poner la siguiente línea:

```
mail.*                       @Host_Central
```

Y tener en el /etc/hosts especificado dicho nombre. Finalmente decir que Syslogd no soporta encriptación en el envío de logs, lo que puede suponer una baja importante. Por esa razón existe "syslog-ng" que tiene multitud de opciones y configuraciones extra que "syslogd" no.

Como curiosidad decir que hay una serie de servicios, 8 en concreto, que van de local0 a local7, que representan servicios del administrador, no generalizados, con lo que si hacemos alguna aplicación podemos utilizar esos identificadores para comunicarnos con syslog.

2.2.10. Rotación de Logs

Para poder controlar fácilmente y de forma ordenada los Logs generados por el sistema, es interesante tener en cuenta la opción de rotarlos cada cierto tiempo. De esta manera tenemos una lista de ficheros parecida a:

fichero.1.gz .. fichero.N.gz.

Dónde el más viejo es el fichero N y el más nuevo el fichero 1. Cuando se rota se realizan las siguientes acciones:

Mover e incrementar en 1 los ficheros ya existentes con los anteriores:

Log.1.gz -> Log.2.gz .. Log.N.gz -> Log.N+1.gz

Generar un nuevo “Log.1.gz” con el fichero de Logs actual.

Los parámetros tanto de tiempo como el máximo de ficheros que se deben almacenar antes de ir moviendo/eliminando el último pueden ser configurados.

La aplicación encargada de todas estas acciones y que nos permite configurar todos estos parámetros es “logrotate”. Una vez instalada, si nos dirigimos al fichero de configuración, normalmente ubicado en:

/etc/logrotate.conf

Se pueden observar varias líneas de configuración de servicios del sistema, como por ejemplo la asociada al fichero “wtmp” (logs de logias):

```
/var/log/wtmp {
missingok
monthly
create 0664 root utmp
rotate 1
}
```

Si miramos el man de la aplicación veremos lo que estas opciones especifican:

Opción	*Descripción*
missingok	Si el fichero de log no existe, se sigue con el siguiente sin mostrar error.
monthly	Se rotarán mensualmente.
create 0664 root utmp	El fichero nuevo se creará para el usuario “root”, grupo “utmp” y permisos 0664.
rotate 1	El número de rotaciones que se harán antes de eliminar o enviar por mail el fichero más viejo.

Hay que tener en cuenta que hay muchísimas más opciones comentadas en detalle en el manual, como por ejemplo la posibilidad de ejecutar algún script antes o después de hacer la rotación.

Vamos a centrarnos un minuto en el fichero de configuración, si nos fijamos veremos la siguiente línea:

include /etc/logrorate.d

Si buscamos esta dirección, veremos que se trata de un directorio, dentro del cual hay varios ficheros. Pues bien, en cada uno de estos ficheros se encuentra la configuración de la aplicación o servicio al que hacen referencia. Esto permite que cuando se instale un servicio, como por ejemplo apache, éste lleve incorporadas sus políticas de rotación por defecto, y cree el fichero: /etc/logrotate.d/apache. Con las opciones dentro.

Un ejemplo de configuración para “syslogd” sería la siguiente:

```
/var/log/syslog {
  missingok
  notifempty
    postrotate
  /usr/bin/killall -HUP syslogd > /dev/null 2>&1 || true
    endscript
  weekly
}
```

En este caso vemos que aparecen algunas opciones nuevas:

Opción	*Descripción*
notifempty	Si el fichero esta vacío no rota.
postrotate endscript	Lo que se encuentra entre estas dos premisas se ejecutará después de rotar los logs.
/usr/bin/killall -HUP syslogd > /dev/null 2>&1 || true	Este comando se ejecuta después de rotar los logs.

En este caso el comando que se ejecuta sirve para enviar la señal SIGHUP que realizará una recarga del proceso, sin necesidad de reiniciar el servicio. Así se consigue crear el nuevo fichero de log que acabamos de eliminar en la rotación.

2.2.11. Notificaciones de Logs

En este punto ya existe una gestión de logs, ahora faltará visualizar la información relevante cómodamente. Para ello tenemos varias opciones.

La primera y más simple consiste en mirar los ficheros directamente a partir de herramientas del sistema como: grep, cat… O sus correspondientes parejas para ficheros comprimidos “.gz”: zgrep, zcat….

Esta primera opción es poco cómoda, ya que el administrador ha de ir revisando cada uno de los ficheros regularmente. Existen otras posibilidades, como: logwatch, logcheck…

Por ejemplo, “logcheck” es una aplicación que se encarga de ir analizando los ficheros de logs en base a unos patrones predefinidos o configurables, ya que son simples expresiones regulares, y en caso de que se cumpla se enviará un mail notificando.

Primero de todo, en el fichero “/etc/logcheck/logcheck.logfiles” podemos encontrar los ficheros que analizará, basta con añadir los que nos parezcan oportunos.

3. Conclusiones

Aunque para muchos será algo muy básico todo lo anteriormente comentado, la realidad muestra que son pocos los administradores que se preocupan verdaderamente por los permisos en el sistema y las configuraciones de las aplicaciones y servicios, dejando la privacidad de los usuarios indefensa ante los ojos curiosos de algunos otros algo más experimentados. O lo que es peor aun, dejando ver ficheros sensibles del sistema con lo que éste queda comprometido por completo.

Aunque este texto no pretende ser un “howto” completo, si está planteado para solucionar posibles dudas de los administradores noveles y mostrar, aunque sea de forma superficial, algunas de las opciones más comunes para una configuración correcta y los conceptos básicos asociados a los permisos de usuarios sobre ficheros.

En cuanto a las configuraciones, por supuesto que no se encuentran todas las opciones en este “howto”, de hecho sería altamente recomendable, si no obligatorio, repasar las manpages de cada una de las aplicaciones, tanto de administración como servicios, buscando las mejores opciones y funcionalidades que puedan ser útiles para nuestros objetivos.

Para finalizar, destacar la importancia de una política de actualización buena, para evitar posibles intrusiones a causa de un fallo interno del programa y que no podemos controlar. Asimismo se deberá tener especial cuidado en la asignación de permisos a los diferentes usuarios y ficheros del sistema.

Un saludo.

por: Ferran Pichel fpichel@isecauditors.com
y la colaboración de: Angel Puigventós apuigventos@isecauditors.com