El objetivo de este articulo es presentar algunas técnicas sencillas para proteger los sistemas Linux...

Seguridad básica en Linux

1. Introducción.

La progresiva disminución de costes del hardware, así como la constante mejora de su funcionalidad, ha generado un aumento considerable de instalación de redes tanto a nivel empresarial como doméstico.

Por otra parte, la conexión de dichas redes a Internet ha experimento en estos últimos años un crecimiento exponencial debido, igualmente, al bajo coste de las tarifas de conexión. Este acceso masivo de redes interconectadas provoca un grave problema de seguridad y es a partir de finales de los años '80 cuando comienzan a aparecer los primeros gusanos de Internet (un gusano es un programa informático que se autoduplica y autopropaga).

De esta forma salta la alarma respecto de la seguridad de los sistemas informáticos y comienzan a aparecer organizaciones cuya misión es ofrecer soluciones frente a problemas de seguridad. Tal es el caso del CERT (Computer Emergency Response Team) creado por la agencia DARPA (Defense Advanced Research Projects Agency).

Los perjuicios económicos provocados por los ataques mediante todo tipo de técnicas, ya sean virus, troyanos, gusanos, exploits, etc, o usuarios malintencionados, son tales que la seguridad en cualquier instalación informática se ha convertido en un tema prioritario. Para cualquier administrador de un sistema informático, la seguridad es crítica.

Respecto de un sistema informático se puede decir que es seguro si está 'fuera de peligro', es decir está protegido. Pero, conseguir llegar a esta situación es prácticamente imposible. No existe un sistema completamente seguro. Todos los sistemas tienen sus vulnerabilidades, bien del propio sistema operativo o de las aplicaciones ejecutadas sobre él. Y todo lo que se puede hacer es aumentar la dificultad para que el sistema quede comprometido.

Otro punto que hay que tener en cuenta es que, en la medida en que el sistema es mas seguro se pierde funcionalidad. Hay que encontrar, por tanto, un equilibrio entre seguridad y funcionalidad. Desactivar todos aquellos servicios que no sean necesarios y proteger el sistema en todo aquello que sea básico es imprescindible.

El objetivo de este articulo es presentar algunas técnicas sencillas para proteger los sistemas Linux. Lógicamente la seguridad entendida por una entidad bancaria no es igual que en una fabrica o en un centro de cálculo o en una instalación militar estratégica. Pero todas ellas deben garantizar la integridad y confidencialidad de sus datos, así como asegurar la disponibilidad de los mismos. Para ello los objetivos de sus normas o medidas de seguridad deben ser la prevención, la detección y la recuperación. Y, básicamente la consecución de estos objetivos puede resumirse en la denegación del acceso a usuarios no autorizados.

Como es sabido, en un sistema Linux el usuario root es el que tiene el control total sobre el sistema. Puede hacer lo que quiera. Lógicamente, a un pirata informático lo que le interesa es obtener ese estatus de usuario root y a partir de ahí utilizar la máquina para los fines que el persiga, como utilizarla como pasarela para dejar falsas pistas sobre sus 'malas acciones', o junto con otras máquinas llevar a cabo un ataque distribuido de denegación de servicio (DDoS) , o ejecutar sus programas con la CPU de la máquina atacada, obtener datos para luego comercializar con ellos,...

La ventaja que tiene Linux frente a otros sistemas operativos es que es libre (Open Source) y su código fuente esta disponible para todo aquel que quiera verlo, estudiarlo, probarlo y corregirlo. Cualquier versión beta que se publique del núcleo de Linux es probada por una gran cantidad de programadores que hacen de banco de pruebas, bajo las mas diversas condiciones y que de esa forma colaboran a que el producto final sea de calidad. No existe la prisa comercial omnipresente en el desarrollo de software propietario.

2. Seguridad.

Y ¿qué se entiende por seguridad?. La norma ISO (Organización Internacional de Normalización) dice que la seguridad consiste en *minimizar la vulnerabilidad de bienes y recursos* .

La seguridad en un sistema se basa en los mecanismos de protección que ese sistema proporciona. Estos mecanismos deben permitir controlar qué usuarios tienen acceso a los recursos del sistema y qué tipo de operaciones pueden realizar sobre esos recursos.

Todo mecanismo de protección debe manejar 2 conceptos:

1.

Recursos: son las partes del sistema utilizadas por los procesos.

2.

Dominios de protección: son el conjunto de recursos y operaciones sobre estos recursos que podrán utilizar todos aquellos procesos que se ejecuten sobre él.

En general, en un sistema LINUX, el conjunto de recursos está formado por todos los archivos del sistema, y el dominio será un usuario y los procesos que el ejecuta y que, por tanto, tengan el mismo UID efectivo.

Para controlar el acceso de los dominios a los recursos se utilizan las Listas de Control de Acceso por cada recurso. La Lista de Control de Acceso (ACL) especifica qué dominios tienen acceso al recurso y qué operaciones asociadas al recurso pueden utilizar. El problema que plantea la lista de control de acceso es su tamaño variable, ya que depende del numero de dominios que tengan acceso al recurso y de las operaciones que pueda realizar cada uno de ellos.

En Linux, para conseguir Listas de tamaño constante, se utilizan 2 técnicas:

1.

Reducir el numero de operaciones posibles sobre un recurso (archivo): podemos controlar 3 operaciones sobre los archivos, que son la lectura (r), escritura (w) y la ejecución (x).

2.

Reducir el número de dominios que aparecen en la lista. Esto se consigue mediante el concepto de grupos de usuarios.

Todos los usuarios de un sistema Linux deben pertenecer, al menos, a un grupo. Existen 3 grupos o categorías en la relación entre un dominio (usuario) y un recurso (archivo):

-

Propietario: indica quién creó el archivo

-

Grupo del propietario: reúne a todos los usuarios que pertenecen al grupo del propietario.

-

Resto de usuarios: los que no crearon el archivo y no pertenecen al grupo del propietario.

Con estos 2 mecanismos la Lista de control de acceso se reduce a 9 bits, organizados en 3 grupos de 3. Esta Lista de control de acceso está situada en la Palabra de protección del nodo-i, que es donde se mantienen todos los atributos asociados a un archivo.

3. Control de acceso al sistema.

Cuando encendemos el ordenador se ejecuta un software, llamado BIOS (Basic Input/Output System) cuya función es determinar la configuración de la máquina y proporcionar un sistema básico de control sobre los dispositivos. Este software se suele almacenar en memoria ROM (Read o-nly Memory) o FLASH que permite la actualización de la BIOS sin cambiar el chip.

En el momento del arranque se puede acceder a la BIOS de la máquina mediante la pulsación de una tecla o combinación de ellas. Desde la interfaz de dicho software es posible modificar parámetros del sistema, como la fecha y hora del sistema, secuencia de arranque, etc. Lógicamente el acceso a la BIOS puede estar protegida con contraseña, si se habilita. Por lo tanto, una primera medida de seguridad sera habilitar la contraseña de la BIOS, y que no coincida con otras contraseñas utilizadas.

Frente a esta medida de seguridad hay que decir que la contraseña BIOS se puede anular provocando un cortocircuito en la batería de la CMOS. Incluso hay fabricantes que, en la placa base, incluyen un jumper que permite borrar la BIOS. Además, existen también utilidades de captura de contraseñas BIOS como son: !BIOS de Bluefish (conjunto de herramientas de propósito general para atacar a la BIOS que incluye utilidades de captura y herramientas de descifrado y que normalmente tiene éxito con la mayoría de las BIOS), CMOS PASSWORDS RECOVERY TOOLS , KILLCMOS, etc.

Una vez arrancada la máquina se ejecuta el gestor de arranque, que puede ser LILO (Linux Loader) o GRUB GNU (Grand Unified Boot Loader) y que es el responsable de la carga del sistema operativo seleccionado.

LILO acepta una serie de opciones en línea cuando se ejecuta. En concreto, el argumento *single* es el más peligroso, ya que arranca Linux en modo monousuario y por defecto, la mayoría de las distribuciones vuelcan a un prompt de root en una shell de comandos sin pedir ningún tipo de contraseña. Es una puerta de entrada al sistema que se debe de cerrar.

También se puede conseguir una shell para cambiar la password del root introduciendo en las opciones en línea del LILO lo siguiente:

linux init=/bin/bash rw

Estos agujeros de seguridad se puede minimizar añadiendo al archivo de configuración de LILO */etc/lilo.conf* las opciones siguientes:

1.

delay = 0 que controla la cantidad de tiempo (en décimas de segundo) que LILO espera a que el usuario introduzca datos antes de arrancar la opción por defecto.

2.

prompt fuerza al usuario a introducir algo y LILO no arranca el sistema automáticamente. Puede ser útil en servidores para eliminar los reinicios desatendidos.

3.

restricted, situada en la imagen/es a proteger, pide una contraseña si se pasan opciones en tiempo de arranque (como linux single).

4.

password=CLAVE (contraseña no cifrada). Junto con la opción anterior y situadas dentro de la sección de la partición Linux a proteger evitan el acceso en modo 'single' desde el prompt del LILO. Si no se añade la opción *restricted*, la password se pedirá en cualquier caso.

Es importante eliminar el permiso de lectura sobre */etc/lilo.conf* para el grupo y los otros, y de esa forma que no se pueda 'ver' la contraseña de LILO.

El comando *chattr* permite transformar el archivo */etc/lilo.conf* en invariable, lo cual significa que, tenga los permisos que tenga, el archivo no se puede borrar, renombrar, escribir en él, o crearle enlaces. Solo el root puede hacerlo.

```
# chattr +i /etc/lilo.conf
```

Para luego introducir cualquier modificación en este archivo habrá que quitar el flag de invariable:

```
# chattr -i /etc/lilo.conf
```

Nada de lo dicho tiene sentido si no hemos desactivado desde la BIOS la opción de arranque desde diskette, ya que en él puede haber un LILO completamente distinto. O también se puede arrancar con un disco de una distribución y a continuación arrancar el núcleo como Linux single.

Si el gestor de arranque utilizado es el GRUB GNU el nivel de seguridad introducido es mayor.

GRUB es un gestor multiarranque de carga directa. No tiene un numero limitado de núcleos arrancables (bootables) y permite arrancar cualquier tipo de sistema operativo sin necesidad de conocer la posición física de su núcleo en el disco, solo la partición correspondiente. En http://www.gnu.org/software/grub
se puede encontrar toda la información referente a esta herramienta.

GRUB presenta varias ventajas respecto de LILO, como, por ejemplo, no es necesario instalarlo cada vez que hacen modificaciones en su archivo de configuracion. Si que es necesario ejecutar la orden *grub-install* cuando se ha ejecutado un *apt-get upgrade*, para cargar la nueva version del paquete grub.

Para crear un diskette de arranque de GRUB hay que copiar los archivos 'stage1' y 'stage2' (etapas de ejecución del GRUB) del directorio */boot/grub/* en el primero y segundo bloque del diskette:

```
# dd if=stage1 of=/dev/fd0 bs=512 count=1
```

```
# dd if=stage2 of=/dev/fd0 bs=512 seek=1
```

Es importante disponer de este diskette por si hay algún fallo en el MBR y no arranca el equipo. Otra forma mas sencilla de obtener este diskette es, sin montar la unidad floppy, ejecutar la orden:

```
# cat /boot/grub/stage1 /boot/grub/stage2 > /dev/fd0
```

De la misma forma, es tambien interesante hacer una copia del MBR a diskette (o cualquier otro medio), sobre todo si se tienen varios sistemas operativos instalados.

```
# dd if=/dev/hda of=/floppy/arranque.mbr count=1
```

Si, por seguridad, se quieren suprimir las operaciones interactivas desde GRUB, añadir la directiva ***password*** en el archivo de configuración de GRUB ***/boot/grub/menu.lst*** de la forma:

```
password --md5 PASSWORD
```

GRUB de esta forma desactiva cualquier control interactivo hasta que se pulse *p* y se introduzca la contraseña correcta. El argumento
md5
indica a GRUB que almacene la password encriptada. Si no se indica estará en texto plano. Ejecutar

```
# grub
```

```
grub> md5crypt
```

pide la password y la devuelve encriptada. Se debe copiar en */boot/grub/menu.lst*.

Otra medida de control de acceso al sistema puede consistir en provocar rearranques sucesivos del sistema cuando se intente entrar en el en modo 'single'. Para ello habrá que añadir, en el archivo de configuración /etc/inittab, alguna de las líneas siguientes:

```
ls:S:wait: shutdown -t0 -r now
```

ls:S:off:/sbin/sh

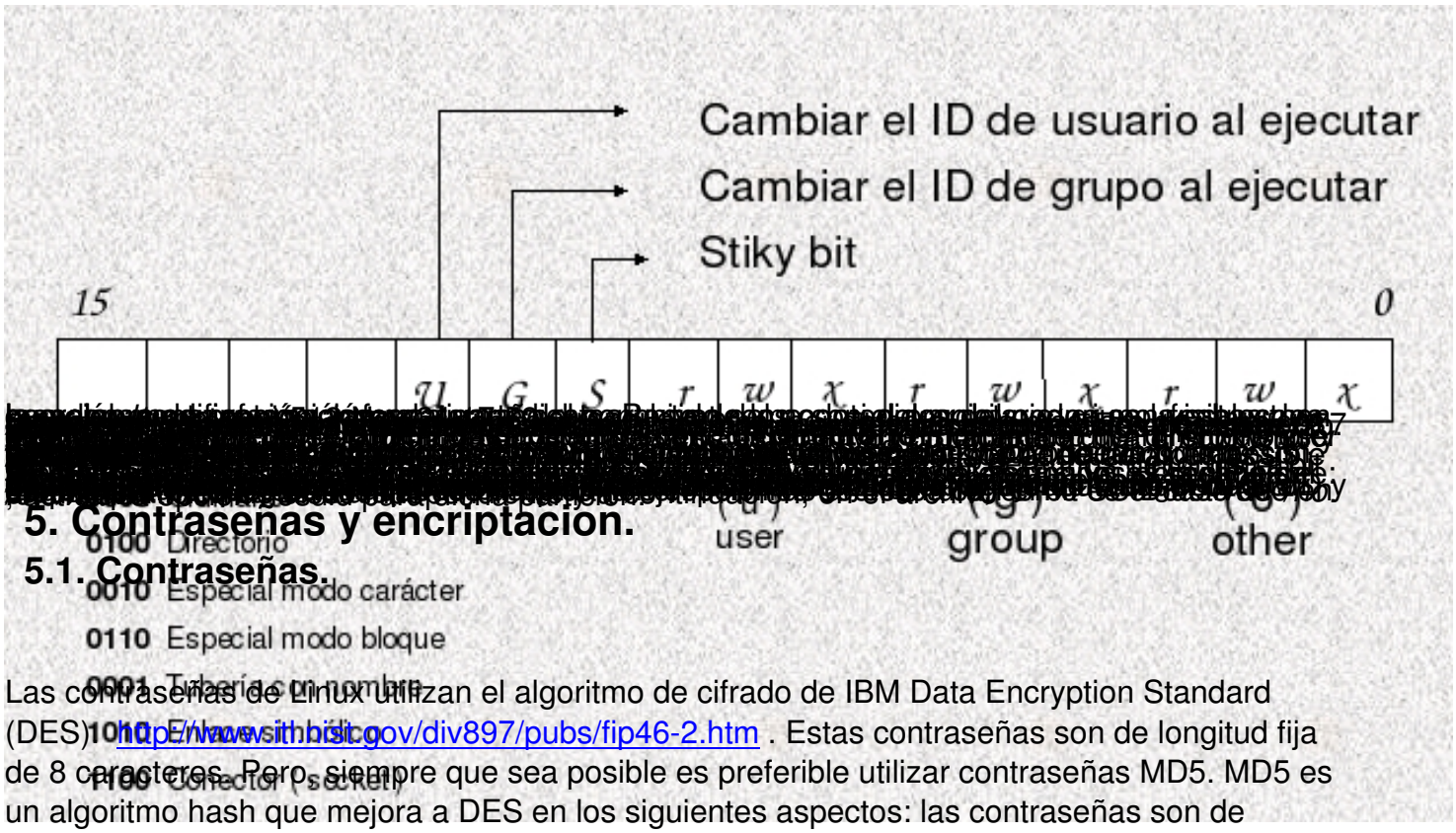
ls:S:once:/shutdown -r now

ls:S:once:/bin/sh -c `exec login`

Todas ellas impiden, de formas diversas, la entrada al sistema en modo *single root* (S).

4. Protección de los archivos.

La palabra de protección contiene 16 bits cuyo contenido se interpreta según se muestra en la figura siguiente:



5. Contraseñas y encriptación.

5.1. Contraseñas.

Las contraseñas de Linux utilizan el algoritmo de cifrado de IBM Data Encryption Standard (DES) http://www.itl.nist.gov/div897/pubs/fip46-2.htm . Estas contraseñas son de longitud fija de 8 caracteres. Pero, siempre que sea posible es preferible utilizar contraseñas MD5. MD5 es un algoritmo hash que mejora a DES en los siguientes aspectos: las contraseñas son de longitud infinita, permiten al inclusión de signos de puntuación y otros caracteres, y además, es exportable fuera de Estados Unidos, ya que no ha sido desarrollado por su gobierno.

Se almacenan, en principio, en el archivo ***/etc/passwd*** que debe tener permisos **644**. Por seguridad y, dado que es necesario mantener la lectura sobre este archivo, se utilizan las shadows passwords y, en este caso, las contraseñas se almacenan en el archivo */etc/shadow*
con permisos 600, es decir sin lectura para el grupo y los otros. De esta forma se impide que los usuarios normales puedan ver las contraseñas cifradas. El propietario de ambos archivos debe ser root.

Es conveniente cambiar periódicamente las contraseñas, tanto los usuarios como el administrador root. La utilidad ***chage*** cambia la fecha de caducidad de las contraseñas. La orden:

chage -M 30 usuario

obliga a usuario cambiar su contraseña cada 30 días. Y la orden:

chage -W 5 usuario

avisa a usuario que su contraseña va a expirar en 5 dias.

También es conveniente periódicamente ejecutar la orden ***pwconv*** para asegurarnos de que todas las contraseñas tienen shadow. En ocasiones, y debido a manipulaciones anómalas, hay contraseñas que se quedan en
/etc/passwd
en vez de en /etc/shadow. Con
pwconv
sincronizamos ambos archivos.

La integridad del archivo */etc/passwd* se realiza con la orden *pwck*, que comprueba que el formato del archivo es correcto y los datos de cada uno de sus campos son válidos.

Aunque más adelante se habla del sistema de logs, hay que indicar que las modificaciones o intentos de modificaciones de las contraseñas quedan registradas en el archivo ***/var/log/messages*** .
Utilizando el comando
less
y los filtros adecuados se pueden controlar los mensajes correspondientes a estos cambios (intentos) de contraseña.

Como normas para la formación de una contraseña se puede decir que es conveniente:

1.

No utilizar nuestro nombre, apellidos o apodo.

2.

No utilizar nombres de parientes o amigos.

3.

No seleccionar una contraseña en la que se repita la misma letra o dígito.

4.

Utilizar una contraseña con 8 o más dígitos.

5.

Utilizar mezclas de letras mayúsculas y minúsculas.

6.

Utilizar algún carácter no alfabético, como signos de puntuación.

7.

Utilizar contraseñas fáciles de memorizar, y no escribirlas en ningún sitio.

8.

Utilizar contraseñas que se puedan escribir rápidamente, por si alguien nos vigila.

Aunque la incorporación de las shadows paswords introduce un nivel de seguridad en el sistema, es necesario que los usuarios sean rigurosos al asignarse contraseña y no utilicen la más sencilla y fácil de recordar.

El administrador debe, también, periódicamente ejecutar un comprobador de passwords, como **Crack (**
http://www.crypticide.org/users/alecm/
) o John the Ripper (

http://openwall.com/john
)
, para detectar passwords descifrables y forzar su cambio. A continuación hay que añadir a la lista de palabras utilizada por
Crack
las contraseñas encontradas de los usuarios. Se pueden encontrar más programas de este tipo en:
http://packetstorm.security-guide.de,
junto con un gran número de diccionarios y listas de palabras.

Otra posibilidad es que la comprobación de contraseñas (utilizando una lista de palabras y una serie de reglas) se haga antes de actualizar la base de datos de contraseñas (**comprobación proactiva de las contraseñas)**
. Para ello existen herramientas como:

-

passwd+:hace un log de todas las sesiones, errores, usuarios que han cambiado su contraseña, reglas que no cumplían las contraseñas y si ha tenido éxito o no el cambio de contraseña. En la dirección: **ftp://ftp.dartmouth.edu/pub/security.**

-

npasswd : sustituto del comando passwd del sistema, más completo y eficiente. En la dirección: **http://www.utexas.edu/cc/unix/software/npasswd .**

Otro tipo de soluciones relativas a la seguridad de las contraseñas sería:

1.

Control de acceso biométrico: autentican a los usuarios en función de sus huellas dactilares, patrones de iris o retina, voz, olor corporal, etc. Es eficiente, pero caro.

2.

Contraseña de un solo uso: se basa en que la contraseña no viaja por la red, con lo cual

desaparece el peligro si alguien esta haciendo un seguimiento a la red y pueda capturarla. El servidor envía un número al cliente, y éste utiliza este número para generar un valor secreto que se devuelve.

5.2. Encriptación PGP (Pretty Good Privacy).

La criptografía tradicional utiliza una sola clave para encriptar y desencriptar. Por lo que la clave debe ser conocida por las dos partes y transferida de modo seguro. Sin embargo, la criptografía de clave publica, como la utilizada por PGP, utiliza una clave para encriptar y una clave para desencriptar. Funciona como la caja fuerte de un banco, necesita dos llaves para abrirla: una pública y una privada. La pública se publica, la privada no, y la combinación pública-privada es única.

Cada usuario, entonces, tiene dos claves. La clave privada solo la conoce el dueño de la clave. La clave pública es conocida por otros usuarios en otras máquinas.

De estas dos claves, la publica y la privada, solo viaja la publica y esta disponible para que cualquiera haga la encriptación. La clave privada no viaja y es mantenida por el usuario para desencriptar los mensajes encriptados con la clave publica correcta. Por tanto l a clave pública cifra y la privada descifra.

Por ejemplo, tenemos dos hosts conectados en red y en ambos se puede utilizar el servicio *ssh* , por tanto cada host tendrá su clave pública y su clave privada. Suponemos que el host2 se conecta al host1 (solicitando datos) y, envía su clave pública. El host1 comprueba la clave pública del host2 y la utiliza junto con su clave privada para encriptar los datos.

El host1 envía los datos encriptados al host2, que busca la clave pública del host1. El host2 utiliza la clave pública del host1 con su clave privada para desencriptar los datos.

De esta forma, los datos encriptados con la clave pública del host2 y la clave privada del host1, solo pueden ser desencriptados con la clave privada del host2 y la pública del host1. Si alguien intercepta los datos no podrá desencriptar sin las claves privadas.

En este momento, y en la distribución Linux GNU Debian o Red-Hat, se utilizan claves PGP para la firma de paquetes fuente (.deb) e ISOs y MD5 para los archivos.

Mas información sobre claves PGP en http://www.pgp.org.

5.3. Módulos de autenticación PAM.

En las últimas distribuciones Linux se han integrado los módulos de autenticación PAM (Pluggable Authentication Modules) desarrollados inicialmente por Sun. PAM permite decidir el método de autenticación que se requiere para cada servicio o en cada caso. Cada método tiene sus módulos que son los que manejan cada tipo de petición. Es decir, para cada método de autenticación, como Kerberos, LDAP, etc, se han desarrollado los módulos correspondientes.

Existen en Linux gran cantidad de módulos PAM disponibles, como por ejemplo el módulo *pam_cracklib.so* que puede ser utilizado por la orden *passwd* para hacer una comprobación contra la biblioteca *pam_cracklib* y determinar si la contraseña elegida por el usuario es débil. O también desactivar el uso en todo el sistema de archivos *.rhosts* en los home de los usuarios. Para ello habría que utilizar el modulo *pam_rhosts_auth.so* .

Más información en: http://www.kernel.org/pub/linux/libs/pam/modules.html.

6. Seguridad del root. Restricción de acceso como root.

En ocasiones el causante de los 'desastres' en el sistema es el propio administrador. Es importante que el root tenga su propia cuenta de usuario y que se conecte siempre como usuario normal. Cuando tenga que realizar tareas de administración puede pasar a root ejecutando el comando *su*.

De igual forma es interesante no incluir en la variable PATH el camino actual y asi forzar a que se introduzca delante del ejecutable los caracteres ./ o bien lanzarlo con sh.

Es importante que la password del root no viaje por la red en texto plano. Debe hacerlo de forma cifrada. Evitar el uso de ordenes -r- (*rlogin, rcp, rsh*,...) y utilizar las ordenes *ssh* o *scp*, en las que la contraseña viaja encriptada.

Se puede utilizar la orden *sudo* para permitir a ciertos usuarios realizar tareas de administración. Para ello hay que especificar en el archivo de configuración
/etc/sudoers
que usuarios están permitidos y que acciones pueden llevar a cabo. El sistema mantiene un registro de todas las actividades realizadas por esos usuarios autorizados.

En principio y como medida de seguridad, solo se debe poder acceder a la cuenta root desde la consola. Como ya se ha comentado, si se necesita hacer un acceso remoto a la cuenta root, entrar primero con cuenta de usuario y luego ejecutar el comando ***su*** para pasar a root. Estos intentos quedan registrados en el archivo de logs
/var/log/messages
. De esta forma un posible atacante
tendría que conocer el nombre de un usuario del sistema, conocer su clave y también conocer la clave del
root,
y añadiría
dificultades para obtener privilegios remotos en el sistema.

No utilizar las ordenes *rlogin/rsh/rcp* como *root*. Pueden ser objeto de diversos tipos de ataques y es peligroso ejecutarlas como *root*.
No crear nunca un archivo
.rhosts
para
root
.

En el archivo ***/etc/securetty*** se especifican aquellas terminales (ttyn | vc/n) desde las que se

puede conectar el root como tal. Se puede limitar la conexión de root, como tal, desde las terminales que se deseen. Si el root debe conectarse desde un lugar diferente de la consola y esta limitado desde */etc/securetty*, debera conectarse como usuario y luego ejecutar la orden
su
.

7. Integridad del sistema de archivos.

La suma de comprobación o *checksum* es una cadena de texto generada por un algoritmo matemático que permite determinar si dos archivos son idénticos. Si se modifica un solo bit la suma es diferente. Comparando el valor de la suma de comprobación de un archivo acabado de descargar con el valor que aparece en el sitio del cual se ha bajado, se puede determinar si los dos archivos son idénticos.

Existen varios tipos de sumas de comprobación, pero de ellos quizás MD5 sea el sistema de suma de comprobación mas importante.

MD5 toma como entrada un mensaje de longitud cualquiera y genera como salida una síntesis de mensaje de 256 bits (de los de la entrada). La probabilidad de que dos archivos generen la misma síntesis es mínima. La siguiente orden es un ejemplo de utilización de MD5:

```
# md5sum /etc/passwd
```

```
e86c7496cc9302459dc5b1d8e26ab2a8 /etc/passwd
```

Si se edita el archivo y se introduce alguna modificación al volver a ejecutar MD5 se comprueba que son diferentes las sumas comprobación generadas antes y después del cambio.

```
# md5sum /etc/passwd
```

8d12e06e0b0fe758b791d64a8138f0ea /etc/passwd

Tripware fue la primera herramienta de verificación de integridad del sistema de archivos. Linux lanza la versión libre de GPL, que es la 2.3.x y se puede conseguir en: **http://sourceforge.net/projects/tripwire**
.
Utiliza muchos algoritmos de suma de comprobación, pero no todos están disponibles en todas las versiones. El mas utilizado es MD5.

Tripwire ejecuta varios *cheksums* de todos los binarios importantes y archivos de configuración y los compara con una base de datos con valores de referencia aceptados como buenos. Así se detecta cualquier cambio en los archivos.

Se debe instalar inmediatamente después de instalar Linux y u na vez que *Tripwire* se ha configurado, es conveniente planificar con
crontab
su ejecución periódicamente, junto con otras tareas de administración, para comprobar si algo ha cambiado. Es preferible ejecutarlo desde un medio externo para, de esa forma, estar seguro de que
Tripwire
no se ha modificado y su base de datos tampoco.

8. Seguridad en el entorno gráfico X.

El sistema X Window es utilizado para crear un entorno gráfico de ventanas en los sistemas Linux. Las X utilizan los puertos TCP del 6000 al 6003 y permite la ejecución remota de aplicaciones. Si un intruso puede acceder a nuestra pantalla gráfica puede apropiarse de las contraseñas cuando se escriban y más ...

Para impedir esto las X tienen varios mecanismos de control de acceso. El mas sencillo consiste en utilizar la orden *xhost* para indicar a que hosts se van a permitir acceder a nuestra pantalla gráfica. Utilizando la orden *xhost* de la forma siguiente:

```
# xhost +IP
```

cualquier usuario desde esa IP tiene el acceso permitido. Si se escribe:

```
# xhost +
```

cualquier host tiene acceso.

Por ejemplo, la máquina B quiere ejecutar la aplicación *xeyes* utilizando los recursos gráficos de la máquina A. En la máquina A hay que ejecutar:

```
$ xhost +IP_maquinaB
```

A continuación hacer una conexión vía ssh a la máquina B. Ya en la máquina B ejecutar:

```
$ export DISPLAY=IP_maquinaA:0
```

Lanzar el entorno de escritorio preferido o la aplicación xeyes:

```
$ startkde
```

Si se utiliza un gestor de pantalla, como *gdm*, existe el método de control de acceso **Mit-Magic -Cookie, que es mejor**
. El método consiste en generar una clave cada vez que se arranca el servidor X, y sólo

permitir el acceso a los clientes que comuniquen esa clave. Se genera una cookie de 128 bits y se almacena en el archivo ~/
.Xauthority
.

Para habilitar este método de acceso hay que editar ***/etc/X11/xinit/xserverrc*** y añadir la opción ***-auth $HOME/.Xauthority***
en la línea que arranca el servidor X:

```
exec /usr/bin/X11/X -dpi 100 -nolisten tcp -auth $HOME/.Xauthority
```

Así cada vez que un usuario arranca el servidor X se guarda la Magic-Cookie en ***~/.Xauthority***
.

Para acceder como root al display del usuario1 hay que hacer que los clientes X comuniquen la misma Magic-Cookie que ha creado el arranque del servidor X. Eso se hace con el comando ***xauth***
.
En el ejemplo se ejecuta
xeyes
como root en el escritorio del usuario1 (display 2):

```
# xauth merge ~usuario1/.Xauthority

# export DISPLAY=:2.0

# xeyes &
```

Este método no necesita el puerto tcp 6000 abierto.

Otra forma de ejecutar sesiones remotas es habilitando en XDMCP las peticiones indirectas.

En cualquier caso y como norma de seguridad es necesario no conectarse nunca al servidor X como root.

9. Logs del sistema.

El logging es cualquier procedimiento por el que un sistema operativo o aplicación graba eventos mientras ocurren y los guarda para su estudio posterior.

Los archivos de log de un servidor deben ser propiedad del usuario y grupo root y no tener ningún permiso para los otros. Además, por seguridad, se les debería poner el flag de solo añadir:

chattr +a nombre_archivo

Linux registra muchas operaciones y eventos importantes del sistema en archivos de texto, para su consulta. Están ubicados en el directorio ***/var/log/***. Entre ellos están:

-

/var/log/lastlog*:** almacena información sobre el último login hecho por los usuarios. Ejecutando la orden ***lastlog se ve el contenido del archivo. Si se quiere observar a un usuario en concreto hay que utilizar la opción u.

-

/var/log/wtmp*:** almacena la lista de todos los usuarios que han hecho login y logout desde que se creó el archivo. El comando ***last vuelca esta información. A ***last*** **se le puede indicar**
la cuenta de usuario que se quiere controlar o la terminal tty.

-

***/var/run/utmp*:** registra las entradas de los usuarios que todavía están conectados al sistema. Cada vez que un usuario se desconecta se borra la entrada correspondiente en utmp. El contenido de este archivo es utilizado por la orden *who.*

-

***/var/log/btmp*: contiene** todos los intentos fallidos de conexión de los usuarios del sistema. Su contenido se visualiza con la orden *lastb*.

-

***/var/log/messages*:** almacena (en el orden en que se presentan) los mensajes del kernel y del sistema. Estos son manipulados por los demonios *syslogd* y *klogd*.

syslogd* es el** servicio de login para programas y aplicaciones (no para conexiones y desconexiones de usuarios). Guarda el nombre del programa, el tipo de servicio, prioridad, etc. Su archivo de configuración es ***/etc/syslog.conf. En él se establece qué eventos se van a registrar y en qué archivos. La estructura de la línea del archivo de configuración es:

opcion.nivel_de_registro destino_registro

Opción puede ser uno de los siguientes valores:

-

auth: servicio de seguridad que sigue la pista de cualquier acción de un usuario que requiera un nombre de usuario y una contraseña para hacer login.

-

cron: sigue los mensajes del sistema cron.

-

kern: sigue los mensajes del kernel.

-

lpr: sigue los mensajes del sistema de impresión.

-

mail: sigue los mensajes del sistema de correo.

Ejemplo: kern.* /dev/console envía todos los mensajes del kernel a la consola.

Nivel de registro indica que las aplicaciones generan entradas en el registro con un cierto nivel. *Syslogd*, en función de ese nivel, puede aceptarla o rechazarla. Este campo no es necesario.

-

alert: problemas serios que requieren una atención inmediata.

-

emerg: el sistema no funciona

-

crit: condiciones críticas.

-

err: errores típicos de stderr.

-

debug: mensajes con información sobre depuración.

-

info: mensajes de información.

-

warning: avisos estándar

Ejemplo: kern.err /var/log/messages

Destino_registro indica cual va a ser el destino de los mensajes:

-

Un archivo: /ruta/archivo

-

Un terminal: /dev/ttyx

-

Una máquina remota:@nombre_máquina

-

Uno o varios usuarios:usuario1, usuario2,...

-

*mensajes a todos los usuarios conectados

Si se quiere monitorizar los mensajes del sistema en una consola, por ejemplo la 12, se debe editar el archivo
/etc/syslog.conf
y se añade la linea:

. (TAB) /dev/tty12

Salir del editor guardando los cambios y reiniciar el servicio *syslog*:

/etc/rc.d/init.d/syslog restart

Hay programas que mantienen y manejan por sí mismos los logins. Por ejemplo, *bash* mantiene su propio histórico de comandos. Las variables que se pueden configurar y que son utilizadas para hacer su propio log son:

1.

HISTFILE: nombre del archivo de historial, por defecto ~nombre_usuario/.bash_history.

2.

HISTFILESIZE: nº máximo de comandos que puede contener el archivo.

3.

HISTSIZE : nº de comandos que recuerda (con las teclas de cursor).

Estas variables se pueden configurar a nivel general, para todos los usuarios, en ***/etc/profile*****.**
De forma p
ersonalizada en el ~/
.bash_profile
de cada usuario.

10. Seguridad de red.

El tema de la seguridad de la red se ha convertido en un punto básico y fundamental, ya que puede comprometer de forma seria el sistema. Cada vez los usuarios se conectan mas a Internet y las posibilidades de intrusiones no deseadas o recepción de todo tipo de virus, aumentan de forma alarmante.

Hay una serie de puntos a tener en cuenta en la configuración de la red, tales como:

-

Ignorar las peticiones ICMP_ECHO dirigidas a direcciones de broadcast. Para ello hay que comprobar que el archivo */proc/sys/net/ipv4/icmp_echo_ignore_broadcasts* tenga un valor distinto de 0.

-

Si la máquina no hace de router (no enruta paquetes) conviene que el archivo */proc/sys/net/ipv4/ip_forward* est
e a 0. Si por el contrario la máquina debe hacer reenvío de paquetes deberá tener un 1:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

-

Activar el soporte para SYN Cookies y de esa forma evitar ataques de denegación de servicio por envío de multitud de tramas con el bit SYN activado, y de esa forma, saturar el sistema. Para ello hay que ejecutar:

```
# echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

Los ataques mas usuales a máquinas conectadas a la red suelen ser:

Spoofing.

IP-Spoofing es un ataque en el que el posible atacante intenta acceder a nuestra máquina haciéndose pasar por 'alguien' conocido. Se basa en la posibilidad de producir paquetes TCP/IP con una dirección IP destino que no es la nuestra sino de una máquina que mantiene una 'relación de confianza' con el objetivo. Es un ataque difícil de detectar y evitar.

Sniffing.

El sniffing es un tipo de ataque en el que se busca recoger información para poder, a continuación, analizar el trafico de red y obtener, por ejemplo, logins y passwords de usuarios.

El clásico ataque sniffing es el que se produce en una red ethernet, con topología en bus, en la que los paquetes de red se difunden a lo largo del bus. El sniffer en este caso sera una tarjeta de red puesta en modo promiscuo que recibirá todos los paquetes, vayan o no para ella.

Una forma de evitar este tipo de ataques es utilizar siempre comunicaciones encriptadas. De esta forma aunque se intercepte el tráfico, se dificulta o imposibilita la obtención de información.

Analizadores de paquetes o sniffers conocidos son: *tcpdump, hunt, ethereal*...

Denegación de servicio (DoS).

Los ataques por denegación de servicio intentan que la máquina tenga un funcionamiento anormal, pudiendo incluso dejarla sin poder prestar los servicios. No comprometen la información almacenada en el sistema, pero si su funcionamiento en condiciones óptimas.

Como ejemplo de este tipo de ataque tenemos la inundación de SYN (ya comentada) y la inundación de ping que consiste en el envío masivo de paquetes ICMP que, en función del ancho de banda del atacante, puede impedir que la máquina atacada envíe nada a la red, y quede inutilizada.

Algunos ataques de denegación de servicio pueden evitarse instalando un cortafuegos con el fin de evitar que se pueda acceder a servicios no esenciales de la máquina, o también deshabilitando totalmente servicios no necesarios, como ECHO.

Existen herramientas que ayudan a mantener unos niveles de seguridad aceptables en la red. Por ejemplo las ordenes *lsof* y *nmap*.

lsof muestra los procesos asociados a un puerto determinado. La opcion -i es de las mas interesantes ya que muestra la lista con los puertos de red que están escuchando y el programa que esta detrás de ese puerto.

nmap es una herramienta que hace todo tipo de escaneados de puertos sobre la dirección o lista de direcciones que se le pasen. Suele ser utilizado por los administradores para identificar los servicios que se están ejecutando en el propio sistema. La orden siguiente indica qué servicios se están ejecutando en puertos TCP en la propia máquina:

```
# nmap -sT -O localhost
```

11. Conclusión.

A lo largo del artículo se ha hecho un descripción de algunos de los muchos mecanismos

básicos que el sistema operativo Linux incorpora teniendo como objetivo la seguridad del sistema. Lógicamente habría que describir mecanismos específicos para la seguridad de los diferentes servicios activos en el sistema, pero que, a criterio del autor excederían el objetivo implícito en el titulo del artículo. Quedaría, por tanto, una segunda parte en la que se detallarían todos estos aspectos avanzados de la seguridad en sistemas Linux.

12. Bibliografía.

'Linux: máxima seguridad' Prentice Hall Anónimo 2000

'Hackers en Linux' Osborne MacGraw-Hill Brian Hatch y otros 2001

'Seguridad en UNIX y redes' Antonio Villalon 2002

'Administración de Sistema Linux' Prentice-Hall M. Carling y otros 2000

'Manual de administración Linux' Osborne MacGraw-Hill S. Shah 2001

13. Enlaces de interés sobre seguridad en Linux.

http://hackerx.netspain.com/index2.htm

http://www.blackbrains.org/

http://www.iec.csic.es/criptomicon/

http://members.ea