Grado en Ingeniería Informática

Departamento de Ingeniería Informática

Seguridad en los Sistemas Informáticos

Práctica 3.3: Auditorías, Sudo, Shadow-password y Rootkits

Autores:
Juan Boubeta Puig,
Antonia Estero Botaro,
Antonio García Domínguez
y Jesús Rosa Bilbao

Fecha:
21 de octubre de 2019

Índice

1. Objetivo

El objetivo de esta práctica es conocer y utilizar herramientas y utilidades de auditorías, sudo, shadow-password y rootkits.

2. Auditorías

En esta sección, se describe cómo llevar a cabo la rotación de ficheros de auditoría así como la revisión de ficheros de registro en GNU/Linux.

2.1. Rotación de los ficheros de auditoría

Estos ficheros requieren una atención regular puesto que pueden llegar a ser muy grandes, y la información antigua no es tan valiosa como la reciente. La orden *logrotate* ayuda a automatizar el proceso de compresión y archivado de los ficheros de auditoría para que estos no se hagan excesivamente grandes. Los datos antiguos se pueden guardar en otro medio, como CD's, para que no ocupen espacio en el disco duro. El comportamiento de esta orden se puede controlar mediante su fichero de configuración `/etc/logrotate.conf` y mediante los ficheros individuales incluidos en `/etc/logrotate.d/`.

A continuación se muestra este fichero:

```
/var/log/cups/*log {
  daily
  missingok
  rotate 7
  sharedscripts
  postrotate
    if [ -e /var/run/cups/cupsd.pid ]; then
      invoke-rc.d --quiet cups force-reload > /dev/null
      sleep 10
    fi
  endscript
  compress
  notifempty
  create 640 root lpadmin
}
```

Una posible configuración de *logrotate* puede ser la siguiente. Ejecutamos *logrotate* diariamente. Una vez por semana, *logrotate* copia con otro nombre los ficheros de registro actuales. Se guardan 4 semanas de mensajes, de forma que cuando se guarda un nuevo fichero de mensajes semanal, se borra el más antiguo (rotación de registros). La configuración de *logrotate* permite guardar los ficheros comprimidos, enviarlos por correo electrónico a alguien, o rotarlos cuando alcanzan un cierto tamaño.

A continuación podemos ver un fichero `logrotate.conf`:

```
# see "man logrotate" for details
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# uncomment this if you want your log files compressed
#compress

# packages drop log rotation information into this directory
include /etc/logrotate.d

# no packages own wtmp, or btmp -- we'll rotate them here
/var/log/wtmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

/var/log/btmp {
    missingok
    monthly
    create 0660 root utmp
    rotate 1
}

# system-specific logs may be configured here
```

2.2. Revisión de los ficheros de registro

Los programas que se ejecutan en el sistema Linux están constantemente añadiendo información a sus correspondientes ficheros de registro. Esta es la razón por la que no es demasiado conveniente ver los ficheros completos, aunque sería posible. La forma más fácil de revisar un fichero de registro es buscando lo que nos interese mediante la orden `grep`. También puede ser adecuado ver los mensajes más recientes mediante la orden *tail*. Veamos algunos ejemplos:

```
$ grep "FAILED" ruta
```

Con esta línea buscamos las líneas de la ruta que hemos especificado donde aparece la palabra «*FAILED*».

```
$ tail -fn 20 ruta
```

Esta orden nos mostrará las últimas 20 líneas del fichero que se encuentra en la ruta que hemos especificado y mediante la opción `-f` le indicamos que vaya mostrando las nuevas líneas a medida que estas se van añadiendo al fichero.

2.3. Ejercicios

- ¿Qué línea deberíamos introducir en el fichero `logrotate.conf` para cambiar la configuración de forma que los ficheros de registro roten todos los días y para grabarlos durante 7 días? (puede consultarse la página del manual de `logrotate`).
- ¿Por qué razón querríamos rotar los ficheros de registro más a menudo y grabar ficheros de registro más antiguos?
- ¿Con qué orden podemos lanzar manualmente `logrotate` para comprobar si está correctamente configurado?
- Si queremos que el sistema realice todo automático, ¿qué se añadiría al `crontab` para que la salida vaya a un fichero —denominado *salida.txt*— localizado en un USB —denominado *ssi*— y, además, se lleve a cabo a las 3 de la mañana (3 a.m.)?
- Busca y describe brevemente algunas herramientas, al menos dos, que estén instaladas de forma nativa en la distribución Kali Linux y que puedan ser utilizadas para realizar auditorías [1].

3. La utilidad sudo

A continuación, se describe la utilidad sudo en sistemas Linux.

3.1. Introducción

Hay ciertas actividades de un sistema Linux que solo puede llevar a cabo el **administrador del sistema**, utilizando para ello el identificador de usuario **root**. El programa *sudo* proporciona un método para que usuarios diferentes de **root** puedan realizar ciertas acciones que en principio están restringidas a este, es decir, nos permite asignar privilegios a cualquier cuenta de usuario para que este pueda ejecutar ciertos programas específicos.

Algunos ejemplos de la utilidad que puede tener *sudo* son:

- El administrador del sistema puede realizar ciertas tareas de administración sin necesidad de cambiar su identificador a **root**.
- Los usuarios pueden montar USBs o CD-ROMs en sistemas en los que no tienen acceso como administradores.
- Los usuarios pueden acceder a las herramientas de configuración de ciertos programas.

El programa *sudo* utiliza el fichero de configuración `/etc/sudoers` (en Kali Linux este fichero no existe ya que somos usuarios root por defecto) para determinar qué usuarios pueden realizar ciertas tareas. La página del manual SUDOERS(5) describe la sintaxis de este fichero, que es bastante compleja.

El fichero contiene dos tipos de entradas:

- Alias.
- Especificaciones de privilegios de usuarios, sirven para especificar qué órdenes pueden ejecutar los usuarios y en qué máquinas.

Mediante la definición de alias es posible que una sola palabra represente:

- Un conjunto de usuarios.
- Un conjunto de máquinas.
- Un conjunto de programas.

Para definir un alias se utilizará:

```
Tipo_Alias Nombre = item1, item2, item3 : Nombre = item4, item5
```

donde TIPO_ALIAS puede ser «*User_Alias*», «*Host_Alias*», o «*Cmnd_Alias*». NOMBRE es el nombre del alias y es una cadena que puede contener letras mayúsculas , dígitos y el carácter '_'. Un «*Nombre*» debe comenzar por una letra mayúscula.

Un ejemplo de definición de alias podría ser:

```
Host_Alias AULA = aula01, aula02, aula03, aula04, ...
```

Las líneas de especificación de privilegios de usuarios tienen el formato:

```
usuario_maquina = lista_de_ordenes
```

Por tanto, si queremos que todos los usuarios del grupo **users** puedan montar un USB o un CD-ROM en cualquier máquina de las especificadas en el alias AULA, pondremos:

```
% users AULA = /sbin/mount /media/usb, /sbin/mount /media/cdrom
```

Para que esta línea de `/etc/sudoers` funcione correctamente es necesario que se haya indicado en el fichero `/etc/fstab` dicho punto de montaje para el USB y el CD-ROM.

Una forma rápida de especificar a todos los usuarios es mediante la etiqueta «*ALL*».

Por ejemplo, la línea siguiente permitiría a todos los usuarios hacer las mismas acciones anteriores desde cualquier máquina especificada en el alias AULA:

```
ALL AULA = /sbin/mount /media/usb, /sbin/mount /media/cdrom
```

Por omisión, cuando un usuario quiera ejecutar una de las órdenes que tiene permitidas en `/etc/sudoers` se le pedirá su contraseña, a no ser que se indique lo contrario mediante la etiqueta NOPASSWD.

Para ejecutar una de estas órdenes el usuario debe darla precedida de la orden *sudo*.

Para editar el fichero `/etc/sudoers` es necesario utilizar la orden *visudo*, este programa bloquea el fichero para que ningún otro usuario lo pueda editar simultáneamente y hace comprobaciones de la sintaxis del fichero.

Puede encontrar más información acerca de la orden *sudo* y del fichero `sudoers` en las páginas correspondientes del manual en línea.

3.2. Ejercicios

- Si nos encontráramos en otra distribución de Linux que no fuera Kali Linux, ¿qué líneas deberíamos añadir al fichero `/etc/sudoers` para conseguir que todos los usuarios que se conecten desde cualquier máquina del aula puedan dar la orden `halt` para parar el sistema sin necesidad de introducir su contraseña? Supongamos que las máquinas del aula tienen como nombres: *aula01*, *aula02*, *aula03*...

4. Shadow-password

En esta sección, se presenta la gestión de contraseñas en sistemas Linux.

4.1. Introducción

El método clásico que utilizaban los sistemas UNIX para almacenar las contraseñas de los usuarios daba lugar a una serie de problemas. Al ser el fichero `/etc/passwd` de lectura pública, cualquier usuario del sistema podía emplear mecanismos de fuerza bruta para encontrar contraseñas de los usuarios. Por otro lado, un intruso que tuviera acceso al sistema también podía hacerse con este fichero y emplear la misma técnica anterior, asegurándose el conocimiento de algunas contraseñas, para así poder acceder al sistema posteriormente. Todo esto condujo a la implantación de un método más seguro para almacenar las contraseñas, que se conoce como *shadow passwords*.

Cuando se utiliza el oscurecimiento de contraseñas, el formato del fichero `/etc/passwd` queda como se muestra a continuación; es decir, en el segundo campo aparece una «x» en vez de la contraseña codificada.

```
antonia:x:501:100:Antonia Estero Botaro:/home/antonia:/bin/bash
```

Además de este tendremos el fichero `/etc/shadow` que es el que almacena la contraseña codificada, utilizando el siguiente formato:

```
antonia:3s5RxKpTg7b0Q:12078:2:180:7:7:12265:
```

Los dos primeros campos se corresponden con el nombre de usuario y su contraseña codificada, respectivamente. El resto de campos de este fichero corresponden a información que permite implementar otro mecanismo para proteger las contraseñas de los usuarios, el «envejecimiento de contraseñas».

La idea básica de este mecanismo es proteger las contraseñas de los usuarios dándoles un período de vida máximo, es decir, las contraseñas de los usuarios solo son válidas durante un cierto tiempo, pasado el cual expirarán y deberán ser cambiadas.

¿De qué nos protege este mecanismo? La idea es la siguiente: si un intruso ha conseguido nuestra contraseña por la razón que sea, y esta es siempre la misma (no expira), tendrá asegurado el acceso al sistema durante tiempo indefinido. Sin embargo, si la contraseña expira, cuando la cambiemos dejará de tener acceso al sistema.

Los campos que almacena `/etc/shadow` relacionados con el envejecimiento son los siguientes:

- Cuándo se cambió la contraseña por última vez, en forma de los días transcurridos desde el 1 de enero de 1970 hasta ese momento.
- Días que han de transcurrir antes de que el usuario pueda volver a cambiar su contraseña.

- Días tras los cuales se ha de cambiar la contraseña.
- Días durante los que el usuario será avisado de que su contraseña va a expirar antes de que esta lo haga.
- Días que la cuenta estará habilitada tras la expiración de la contraseña.
- Días desde el 1 de enero de 1970 hasta que la cuenta se deshabilite.
- Campo reservado.

Cuando un usuario cambia su contraseña, se le puede impedir cambiarla durante un cierto tiempo; esto tiene como objetivo que el usuario no restaure inmediatamente la contraseña antigua después de haberla cambiado. Pasado este período el usuario podrá volver a cambiar su contraseña de forma voluntaria. Si el número máximo de días en los que el usuario no puede cambiar su contraseña es mayor que el número de días tras los cuales es obligatorio el cambio, el usuario no podrá cambiarla nunca, y la cuenta quedará bloqueada después del período de gracia que se da una vez expirada la contraseña.

Aunque **root** tiene acceso para modificar los ficheros `/etc/passwd` y `/etc/shadow`, no debería editarlos utilizando un editor de texto. La razón de esto es la seguridad del sistema, ya que en la edición de estos ficheros se podría cometer un error que la comprometiera.

Aún así, si se necesita editar estos ficheros porque están corruptos, se debería utilizar la orden *vipw*, ya que esta bloquea el fichero `/etc/passwd` antes de lanzar el editor *vi* o aquel que tengamos establecido en la variable del *shell* EDITOR. Esto previene los conflictos que pudieran aparecer si varios administradores decidieran editar el fichero. La orden *vigr* es similar a la anterior y sirve para editar el fichero `/etc/group`. Una vez editado el fichero de grupos con `vigr` podemos utilizar la orden *grpck* para verificar que todas las líneas tienen el número correcto de campos, un nombre de grupo único, y una lista válida de miembros.

Si lo que se desea es editar los ficheros `/etc/shadow` o `/etc/gshadow` podemos utilizar las órdenes anteriores pero con la opción `-s`.

Todas las opciones que se contemplan en el fichero `/etc/shadow` pueden ser controladas cuando creamos una cuenta de usuario con *useradd* o cuando cambiamos su contraseña con *passwd*.

La orden *passwd* proporciona una serie de opciones que permiten controlar los aspectos comentados anteriormente (véase Tabla 1).

Opción	Significado
`-l`	Bloquea la cuenta de un usuario.
`-u`	Desbloquea una cuenta que ha sido bloqueada anteriormente con la opción `-l`. La contraseña de la cuenta no varía.
`-n`	Establece el número mínimo de días que el usuario debe esperar antes de cambiar su contraseña.
`-x`	Establece el número máximo de días durante los que va a ser válida la contraseña actual.
`-w`	Establece el número de días durante los cuales el usuario va a ser avisado (al entrar en su cuenta) de que debe cambiar la contraseña para evitar que la cuenta se bloquee.
`-i`	Establece el número de días en que la cuenta estará habilitada después de que la contraseña haya expirado. Pasado este tiempo la cuenta se bloqueará.

Tabla 1: Opciones de la orden *passwd*.

4.2. Ejercicios

- ¿Se puede utilizar cualquier editor de texto para editar los ficheros `/etc/shadow` `/etc/group` y `/etc/sudoers`? ¿Qué editores deben utilizarse? Justifica tu respuesta.
- ¿Cuáles son las ventajas de realizar este tipo de tareas usando Kali Linux? ¿Y las desventajas?

5. Rootkits

A continuación, se describe la utilidad de los rootkits.

5.1. Introducción

Aunque el mantenimiento de los ficheros de registro y las herramientas de análisis pueden ayudar a capturar intrusos potenciales, no son técnicas infalibles. A veces un intruso experto puede llegar a tener un acceso suficiente al sistema y los ficheros de registro no indicar nada acerca del problema. Para tener evidencias de este tipo de ataque más sofisticado, se necesita seguir la pista del estado de ficheros importantes del sistema por si sufren cambios inesperados.

Por ejemplo, suponga que un intruso ha reemplazado la utilidad *ls* con una nueva versión que falla al listar cualquier fichero que comience con un determinado código.

Programa	Alteración
crontab	Oculta ciertas tareas planificadas de su salida.
du	No incluye el tamaño de ciertos ficheros ocultos cuando muestra la información sobre el uso del disco.
find	No lista ciertos ficheros ocultos.
ifconfig	No muestra información de red que podría revelar la actividad del intruso.
inetd	Permite el acceso automático a ciertos puertos de red.
killall	No mata ciertos procesos que utiliza el intruso para mantener el acceso.
login	Permite login remoto sin contraseña para el usuario **root**.
ls	No muestra ciertos ficheros ocultos.
ps	No muestra ciertos procesos creados por el intruso.

Tabla 2: Posibles objetivos para un *rootkit*.

Esto permitiría al intruso almacenar en el disco duro ficheros de configuración que podría utilizar para irrumpir en el sistema, sin que fuésemos capaces de verlos.

5.2. Búsqueda de rootkits

Una vez que un intruso ha tenido acceso como **root** a tu sistema, quiere mantenerlo. Normalmente los intrusos introducen este tipo de programas utilizando un *rootkit*, una colección de programas y *scripts* diseñado para permitir al intruso tener acceso de forma continuada, incluso si se descubre la entrada inicial. Es decir, el *rootkit* le proporciona varios métodos de acceso, de forma que si se elimina uno de ellos le queden más para poder acceder. Por ejemplo, el *rootkit* denominado *lrk4*, incluye los programas modificados que se muestran en la Tabla 2.

5.2.1. Medidas preventivas

Una buena forma de prepararse para un ataque de *rootkit* sería la siguiente:

- Hacer una copia de las utilidades críticas del sistema tales como: *ls*, *ps*, *login*, *inetd* y *find*, colocándolas en un USB o CD. Si sospecha de su sistema, puede usar estas utilidades para explorar el sistema, en vez de las del disco duro que han podido ser modificadas.
- También es conveniente considerar la creación de copias de estas utilidades que estén enlazadas estáticamente. Es decir, que sean programas autocontenidos que no dependan de otros componentes del sistema (bibliotecas compartidas)

para funcionar. La razón de este procedimiento es que algunos *rootkits* pueden modificar las bibliotecas compartidas. Si estas han sido modificadas, cualquier programa que las utilice puede no funcionar de forma adecuada. Por ejemplo, suponga que las órdenes *ls* y *find* operan solicitando una serie de ficheros de un componente de una biblioteca compartida. Si esta ha sido alterada, las órdenes *ls* y *find* no devolverán resultados exactos, incluso si los programas no han sido alterados.

5.2.2. Medidas de detección

Según el sistema operativo y su versión utilizada, el conjunto de herramientas cambia considerablemente. En este apartado se mencionarán algunas de las más conocidas para distribuciones de GNU/Linux.

En general, el uso de un disco de arranque de una distribución GNU/Linux o un conjunto mínimo de utilidades es una buena forma de preparar un análisis de emergencia sin necesidad de recompilar software.

5.2.3. chkrootkit

El paquete *chkrootkit* permite comprobar la existencia de *rootkits* en el sistema. Este paquete incluye un *script* que funciona como un antivirus, y puede informar de la presencia de un *rootkit* en nuestro sistema, aunque no puede eliminarlo. Examina los ficheros binarios del sistema para detectar unos 30 *rootkits* diferentes. Este paquete se puede obtener en `http://rpmfind.net/`. También puede ser muy útil visitar `www.chkrootkit.org`, bajarse el paquete, y revisar algunos de los enlaces para ver cómo los intrusos pueden explotar *rootkits* para mantener su acceso no autorizado al sistema.

5.2.4. Gestor de paquetes RPM

Se puede utilizar la orden *rpm* para verificar la integridad de los ficheros de un paquete.

```
$ rpm -V paquete
```

Esta orden puede comprobar diferentes aspectos de los ficheros en el paquete original y los ficheros instalados en el sistema. Los aspectos que compara son:

- Tamaño del fichero.
- Permisos y tipo de fichero.

- Suma MD5.
- Números de dispositivo mayor y menor.
- Propietario.
- Grupo.
- Fecha y hora de modificación.

Cualquier cambio en algunos de estos aspectos del fichero es comunicado, aunque también nos podríamos plantear que el programa *rpm* podría haber sido modificado por el *rootkit*.

5.2.5. Medidas de recuperación

Si descubre un *rootkit* en su sistema los pasos a dar serían los siguientes:

- Si es posible, desconecte el servidor de la red hasta que se haya solucionado el problema.
- Haga una copia de seguridad del sistema completo, incluyendo los ficheros del sistema operativo y los ficheros de datos. Estos podrían ser revisados posteriormente para seguir la pista al intruso y perseguirlo si es posible.
- Reconstruya el sistema, actualizando los paquetes que se hayan dañado, o reinstalando el sistema operativo completo si fuera necesario.

5.3. Utilización de comprobadores de integridad

Aunque la verificación del sistema buscando un *rootkit* es una buena idea, si sospecha que alguien ha comprometido la seguridad del sistema, una forma más amplia de abordar el problema es comprobar la integridad de los ficheros del sistema. Utilidades tales como *md5sum* y *gpg*, además de la opción `-checksig` de la utilidad *rpm*, nos sirven para verificar la integridad.

5.4. Ejercicios

- Encuentra ejemplos de herramientas *antirootkits* para diferentes sistemas operativos. Para cada una de ellas indica su nombre, una breve descripción, los sistemas operativos soportados y, al menos, una referencia web para ampliar esta información.
- Encuentra utilidades de comprobación de integridad e indica sus características más relevantes junto con su URL de descarga.

Referencias

[1] Pablo González Pérez, Germán Sánchez Garcés y Jose Miguel Soriano de la Cámara: *Pentesting con Kali 2.0.* 0xWORD, 1ª edición, 2015, ISBN 978-84-608-3207-2.