Módulo 3.1

Redes y Seguridad
Gr. Ing. Informática
J.L. Vázquez Poletti

Objetivo

En este módulo aprenderemos a gestionar usuarios y grupos de un sistema Linux, así como sus permisos.

ATENCIÓN: Este módulo está pensado para ser realizado en 2 sesiones. Debido al borrado de datos una vez que se apaga la máquina virtual, será necesario volver a crear los usuarios y grupos en la segunda sesión.

Punto de partida

Como recién incorporados a la plantilla de administradores de sistemas de ENCOM (una empresa dedicada al desarrollo de software) en la última oleada de contrataciones, nos tocará dar de alta en una de las terminales del Centro de proceso de datos a las nuevas incorporaciones, así como crear grupos al respecto y manejar sus permisos.

Por otro lado, se han traído los archivos de contraseñas de una máquina que levanta las sospechas del Director de la División. Tocará comprobar si dichas contraseñas son débiles, incurriendo en un evidente peligro para la empresa.

Creación de usuarios

El comando usado para crear usuarios es **adduser**. Se debe ejecutar como **root** a través del comando **sudo**.

Se pide dar de alta a los siguientes usuarios (se suministra el nombre real), asignarles contraseñas distintas, introduciendo lo que se desee en el resto de información:

- kflynn: Kevin Flynn
- abradley: Alan Bradley
- edillinger: Ed Dillinger
- lbaines: Lora Baines

1. Verificar que las entradas correspondientes se han añadido a los ficheros **/etc/passwd** y **/etc/shadow**.
2. Desde la cuenta de **root** cambiar a cualquiera de los usuarios creados usando el comando **su**.
3. Desde la cuenta **ubuntu** cambiar al mismo usuario.

¿Qué diferencia se aprecia en el método descrito en el paso 2 y el 3?

Máquinas Virtuales

- RyS-ENCOM (ubuntu:reverse)

Más información

John the Ripper: http://www.openwall.com/john/

Eiciel: http://rofi.roger-ferrer.org/eiciel/

Creación de grupos de usuarios

El comando usado para crear grupos de usuarios es **addgroup**. Se pide crear los siguientes grupos:

- employees
- developers
- assistants

1. Verificar que los grupos se han creado en **/etc/group**.
2. Crear un nuevo usuario llamado **sflynn** con las siguientes características:
   - Nombre real: Sam Flynn
   - Grupo: employees

   CONSEJO: Para borrar el usuario se usará el comando **deluser**.

3. Cambiar al nuevo usuario y verificar con el comando **id** que efectivamente pertenece al grupo.

¿Por qué no hay un grupo llamado **sflynn** en **/etc/group**?

Modificación de usuarios y grupos

Uno de los aspectos que se puede modificar es la contraseña. Esto se realiza con el comando **passwd**. El comando **chage** establece los detalles de expiración de la contraseña.

El comando **usermod** permite mucho más nivel de modificación.

1. Modificar las contraseñas de dos de los usuarios creados con **passwd**. La contraseña de uno de ellos será igual al nombre de usuario.
2. Modificar las contraseñas del resto de usuarios creados con **usermod**. La contraseña de uno de ellos será igual al nombre de usuario.
3. Asignar en los siguientes grupos a los usuarios correspondientes:
   - employees: todos
   - developers: kflynn, abradley, edillinger
   - assistants: edillinger, lbaines
4. Verificar los cambios realizados tanto desde los usuarios (comandos **id** y **groups**) como en el fichero **/etc/group**.
5. Bloquear la cuenta **sflynn**.
6. Establecer la expiración de la contraseña de **kflynn** a mañana.

¿En qué archivo se puede verificar que la cuenta **sflynn** ha sido bloqueada? ¿Y la expiración de la contraseña de **kflynn**?
¿Qué símbolos/campos se suelen emplear y para qué situaciones?

Verificación de contraseñas débiles con John the Ripper

John the Ripper (JTR) es un programa de criptografía muy popular, con licencia GNU, que aplica tanto fuerza bruta como diccionario para descifrar las contraseñas de multitud de sistemas operativos.

En el directorio **/home/ubuntu/Quarantine** están el **passwd** y el **shadow** de una máquina de ENCOM sospechosa de albergar un grave agujero de seguridad en forma de contraseñas débiles.

1. Copiar el directorio **Quarantine** a uno nuevo llamado **Analysis**.
2. Para obtener el archivo con el que trabajar, se deberán consolidar **passwd** y **shadow** (dependiendo del sistema operativo). Esto se realizará con el comando **unshadow** y volcar el resultado a un fichero llamado **passwords**.

   PISTA: Usar redirecciones de línea de comandos.

3. Ejecutar **john** sobre el archivo **passwords**.
   - Interrumpir el proceso y pedir a **john** que muestre las contraseñas ya averiguadas.
   - Reanudar el proceso durante 5 minutos pulsando entre medias la tecla **enter**.

¿**John the Ripper** comienza el proceso desde el principio?
¿Cuáles son las debilidades encontradas en las contraseñas?

4. Probar ahora con las contraseñas de los usuarios creados recientemente **copiando los archivos con los que se trabajará en un directorio aparte y ejecutando john como root.**

¿Qué fallo da **John the Ripper**?
¿A qué se debe este fallo? **PISTA:** mirar el formato de la contraseña.

Verificación de contraseñas débiles con John the Ripper (versión mejorada)

Como la versión de **John the Ripper** provista por la distribución de Linux no sirve para los ficheros de contraseñas que se quieren analizar, se ha descargado una mejorada por la comunidad.

Ésta se encuentra en el directorio **/home/ubuntu/Zuse/**.

1. Descomprimir el fichero **john-1.7.9-jumbo-6.tar.gz**.
2. Entrar en el subdirectorio **src** y ejecutar **make clean linux-x86-native**.
3. El nuevo ejecutable **john** se encuentra en el subdirectorio **run**. Verificar que funciona perfectamente pasándole el argumento **--test**.

   IMPORTANTE: Si se ejecuta "**john**" sin nada delante, se invocará la

versión (no mejorada) instalada en el sistema.

4. Ejecutar el nuevo **john** sobre el fichero objetivo pasándole como parámetro **--format=crypt**.

Gestión de permisos de archivos y directorios

En esta parte se trabajará en el directorio **/tmp**. Los permisos se cambian con el comando **chmod**. El propietario se cambia con los comandos **chown** y **chgrp**.

1. Como usuario **ubuntu** crear un directorio llamado **Trace**. Verificar los permisos.
2. Dentro del directorio, crear un archivo de texto llamado **data** con el nombre del alumno. Verificar los permisos.
3. Dejar los permisos de lectura del archivo **data** sólo para el usuario propietario.
4. Leer el archivo **data** como los usuarios **edillinger** y **root**.

¿El resultado es el mismo? ¿Por qué?

5. Crear un archivo de texto llamado **script.pl** con el siguiente contenido:

```
#!/usr/bin/perl

 open FILE,"data" or die $!;

$data=<FILE>;

close(FILE);

 chomp($data);

 print "$data is the new Space Paranoids Champion!\n";
```

6. Probar a ejecutar el archivo.

¿Qué es lo que falla? Solucionar el problema.

7. Cambiar el grupo propietario de **script.pl** a **employees**. Verificar los permisos.
8. Ejecutar **script.pl** como el usuario **edillinger**.

¿Cuál es el resultado? ¿Por qué?
Solucionar el problema para que **edillinger** pueda ejecutar exitosamente **script.pl**
¿Qué usuarios pueden cambiar los propietarios y permisos de los archivos?

9. Cambiar el propietario del directorio **Trace** tanto al usuario como al grupo **edillinger**.
10. Dejar los permisos del directorio exclusivamente para el usuario propietario.
11. Con otro usuario, intentar acceder al directorio.

12. Cambiar el grupo propietario del directorio a **developers** y darle permisos de lectura.
13. Como **abradley**, listar el contenido de **Trace** pero **sin acceder a él**.
14. Añadir el permiso de escritura del directorio para el grupo **developers**.
15. Como **abradley**, intentar crear un nuevo archivo de texto con el apellido del alumno.
16. Añadir el permiso de ejecución del directorio para el grupo **developers** y probar a crear el archivo de texto de nuevo.

¿Cuál es la traducción de los permisos de los ficheros (rwx) a los directorios?

El comando **chmod** permite también asignar, entre otras cosas, el bit de SUID (u+/-s) y GUID (g+/-s).

1. Cambiar el usuario y grupo propietarios de **script.pl** a **root**.
2. Añadir las siguientes líneas al archivo **script.pl**:

```
$ENV{"PATH"} = "/usr/bin";

system ("tail /etc/shadow");

system (id);
```

3. Ejecutar **script.pl** como **edillinger**.

   CONSEJO: revisar las opciones del comando **sudo**.

4. Asignar el bit SUID a **script.pl**. Revisar los permisos del archivo.
5. Volver a ejecutar **script.pl** como **edillinger**.

¿Qué diferencias se aprecian?

¿Qué implicaciones para la seguridad tienen los bits SUID y GUID?

El comando **umask** permite establecer los permisos por defecto de los archivos y directorios que se crearán. Al igual que **chmod**, este comando tiene dos modos de introducir los parámetros: octal (complemento, permisos que no se establecerán) y texto (ejemplo: u=rwx,g=rwx,o=).

1. Verificar los permisos por defecto actuales. Crear un fichero (con el comando **touch**) y verificar la correspondencia con la máscara.
2. Cambiar la máscara a 0007. Crear un nuevo fichero y verificar los permisos.
3. Cambiar la máscara a 0002. Crear un nuevo fichero y verificar los permisos.
4. Cambiar la máscara a 0077. Crear un nuevo fichero y verificar los permisos.
5. Volver a la máscara original (0022) **pero empleando el texto**.

¿En qué situaciones son mejores cada máscara de las vistas anteriormente?

Gestión de permisos de ejecución con sudo

Se puede definir qué usuario puede ejecutar qué y como qué usuario gracias a **sudo**. Para ello, se tiene que editar el archivo **/etc/sudoers**. Este archivo puede ser editado con una a través del comando **visudo** (en la máquina de las prácticas carga el editor **nano**).

1. Quitar el bit SUID de **script.pl**. Comprobar que el resultado de su ejecución ha cambiado.
2. Ejecutar como **edillinger**: **sudo /tmp/Trace/script.pl** (**./script.pl** si se está en el mismo directorio).
3. En el archivo **/etc/sudoers**:
   - Agregar a **edillinger** al alias ENCOM:

     ```
     # User alias specification

     User_Alias ENCOM = edillinger
     ```

   - Añadir el permiso de ejecución como **root** a **script.pl**:

     ```
     ENCOM ALL = /tmp/Trace/script.pl
     ```

4. Volver a ejecutar **script.pl** como **edillinger**.
5. Dar permisos a **abradley** para ejecutar **script.pl** como **root**.

Poner algún ejemplo de cómo **sudo** puede ayudar en las tareas de mantenimiento del sistema.

Gestión de ACL

La ACL de un archivo o directorio se puede gestionar mediante los comandos **getfacl** y **setfacl**. Además, se cuenta con un entorno gráfico llamado **Eiciel** (invocable con el comando **eiciel**).

1. Por línea de comandos, verificar la ACL de **script.pl**.
2. Abrir **script.pl** en **eiciel**.
3. Quitar todos los permisos a **otros**.
4. Dar permiso sólo de ejecución a **edillinger** y **abradley**.
5. Dar permiso sólo de lectura al grupo **assistants**.
6. Verificar todos los cambios anteriores.

¿Qué diferencias tiene la gestión de ACL respecto a la vista con **sudo**?