Lista de control de acceso:

Una Lista de Control de Acceso o ACL (del inglés, Access Control List) es un concepto de seguridad informática usado para fomentar la separación de privilegios. Es una forma de determinar los permisos de acceso apropiados a un determinado objeto, dependiendo de ciertos aspectos del proceso que hace el pedido.

Las ACLs permiten controlar el flujo del tráfico en equipos de redes, tales como routers y switches. Su principal objetivo es filtrar tráfico, permitiendo o denegando el tráfico de red de acuerdo a alguna condición. Sin embargo, también tienen usos adicionales, como por ejemplo, distinguir "tráfico interesante" (tráfico suficientemente importante como para activar o mantener una conexión) en ISDN.
ACLs en redes de computadoras:

En redes de computadoras, ACL se refiere a una lista de reglas que detallan puertos de servicio o nombres de dominios (de redes) que están disponibles en una terminal u otro dispositivo de capa de red, cada uno de ellos con una lista de terminales y/o redes que tienen permiso para usar el servicio. Tanto servidores individuales como routers pueden tener ACLs de redes. Las listas de acceso de control pueden configurarse generalmente para controlar tráfico entrante y saliente y en este contexto son similares a un cortafuegos.

Existen dos tipos de ACLs:

- ACL estándar, donde solo tenemos que especificar una dirección de origen.
- ACL extendida, en cuya sintaxis aparece el protocolo y una dirección de origen y de destino.

El sistema de permisos de archivos utilizado en Linux y conocido comúnmente como **UGO (Propietario, Grupo, Otros),** algunas veces no resulta ser muy eficaz, dado que, con este esquema no se puede lograr que más de un usuario o más de un grupo, a la vez, puedan tener el mismo privilegio sobre un mismo archivo o directorio. Para solucionar este problema, se crearon las Listas de Control de Acceso (abreviado ACL), mediante las cuales, a través de comandos simples de la Shell, se pueden dar permisos sobre diversos archivos y / o directorios, con todo su contenido, a diferentes grupos de usuarios o a usuarios particulares.
En la presente actividad, el aprendiz implementa ACL’s en Linux para compartir información entre los diferentes usuarios y grupos del sistema, acorde con las políticas de la compañía.

1. Habilitar el uso de listas de control de acceso en la partición actual de Linux.
2. Crear los grupos y usuarios, mediante comandos de Shell, tal como se observa en la figura 1.

3. Crear los archivos y directorios mostrados en la figura 1, **deshabilitando** siempre los permisos de lectura, escritura y ejecución para **otros** usuarios.
4. Crear las ACL's necesarias para permitir el acceso a los archivos y directorios, de la manera indicada en la figura 1, con los comandos apropiados de la Shell de Linux.

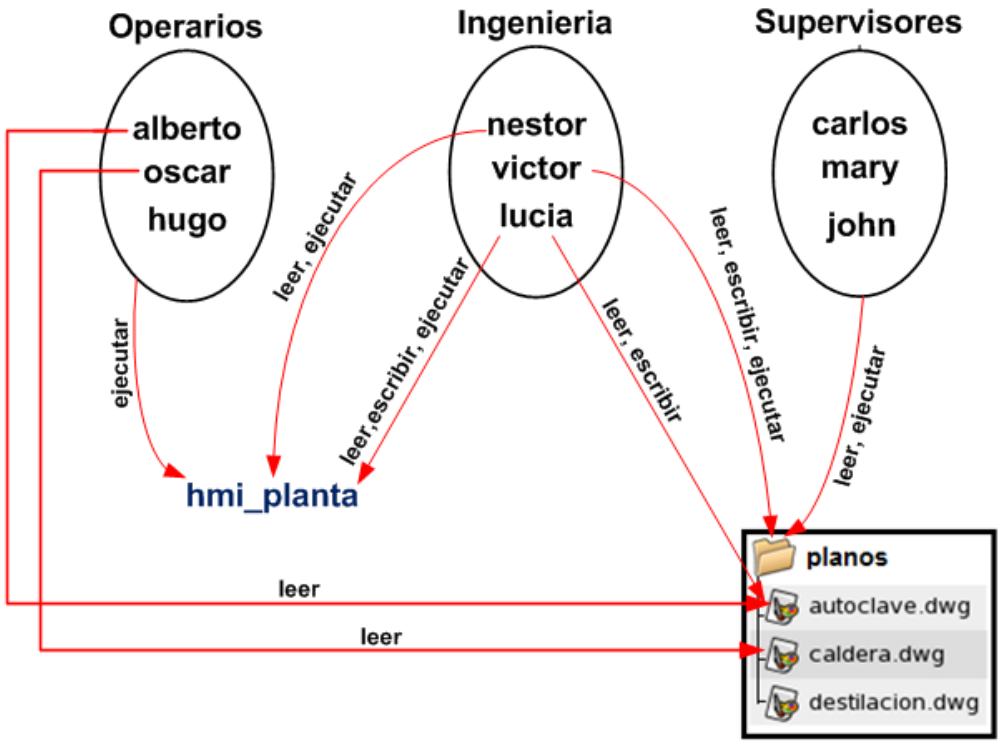

Figura 1. Esquema de permisos de archivos y directorios mediante ACL's en Linux.

5. Efectuar un **login,** desde diferentes ventanas de Shell, con diferentes cuentas de usuario, para probar la efectividad en la configuración de las ACL's.
6. Investigue, con sus compañeros de grupo, qué utilidad tienen los bits SetUID y SetGID de un archivo y cómo se pueden cambiar estos valores en Linux.
7. Presente un informe en formato PDF, el cual recoja las evidencias de todo el proceso realizado y entréguelo al instructor.

MATERIAL DE APOYO PARA LA ACTIVIDAD DE LISTAS DE CONTROL DE ACCESO (ACL) EN LINUX

1. Aplicar una lista de control de acceso a un usuario sobre un archivo.
Sintaxis general:

setfacl -m u:*usuario*:rwx *filename*

Donde **rwx** corresponde a los permisos de lectura, escritura y ejecución que se le quieren dar al ***usuario*** sobre el archivo ***filename.***

2. Aplicar una lista de control de acceso a un directorio:
Sintaxis general:

setfacl -m d:u:*usuario*:rwx *dirname*

Donde **rwx** corresponde a los permisos de lectura, escritura y ejecución que se le quieren dar al ***usuario*** sobre el directorio ***dirname.***

3. Aplicar una lista de control de acceso a un directorio y su contenido:
Sintaxis general:

setfacl -R -m u:*usuario*:rwx *dirname/**

Donde **rwx** corresponde a los permisos de lectura, escritura y ejecución que se le quieren dar al ***usuario*** sobre el contenido del directorio ***dirname.***

4. Consultar las ACL's de un archivo:

getfacl *filename*

Donde ***filename*** es el nombre del archivo.

5. Eliminar una ACL de un usuario sobre un archivo:

setfacl -x u:*usuario filename*

Donde ***usuario*** es el propietario de la ACL sobre el archivo ***filename***

6. Aplicar una lista de control de acceso a un grupo sobre un archivo.
Sintaxis general:

setfacl -m g:*grupo*:rwx *filename*

Donde **rwx** corresponde a los permisos de lectura, escritura y ejecución que se le quieren dar al ***grupo*** sobre el archivo ***filename.***

PARA USUARIOS AVANZADOS QUE QUIERAN SEGUIR INVESTIGANDO

Las listas de control de acceso (ACLs, //Access Control Lists//) proveen de un nivel adicional de seguridad a los ficheros extendiendo el clásico esquema de permisos en Unix: mientras que con estos últimos sólo podemos especificar permisos para los tres grupos de usuarios habituales (propietario, grupo y resto), las ACLs van a permitir asignar permisos a usuarios o grupos concretos; por ejemplo, se pueden otorgar ciertos permisos a dos usuarios sobre unos ficheros sin necesidad de incluirlos en el mismo grupo. Este mecanismo está disponible en la mayoría de Unices (Solaris, AIX, HP-UX...), mientras que en otros que no lo proporcionan por defecto, como Linux, puede instalarse como un //software// adicional. A pesar de las agresivas campañas de //marketing// de alguna empresa, que justamente presumía de ofrecer este modelo de protección en sus //sistemas operativos// frente al `arcaico' esquema utilizado en Unix, las listas de control de acceso existen en Unix desde hace más de diez años ([Com88]).

Los ejemplos que vamos a utilizar aquí (órdenes, resultados...) se han realizado sobre Solaris; la idea es la misma en el resto de Unices, aunque pueden cambiar las estructuras de las listas. Para obtener una excelente visión de las ACLs es recomendable consultar [Fri95], y por supuesto la documentación de los diferentes clones de Unix para detalles concretos de cada manejo e implementación.

La primera pregunta que nos debemos hacer sobre las listas de control de acceso es obvia: >cómo las vemos? Si habitualmente queremos saber si a un usuario se le permite cierto tipo de acceso sobre un fichero no tenemos más que hacer un listado largo:

```
anita:~# ls -l /usr/local/sbin/sshd -rwx
1 root bin 2616160 Apr 28 1997 /usr/local/sbin/sshd
anita:~#
```

Viendo el resultado, directamente sabemos que el fichero ##sshd## puede ser ejecutado, modificado y leído por el administrador, pero por nadie más; sin embargo, no conocemos el estado de la lista de control de acceso asociada al archivo. Para ver esta lista, en Solaris se ha de utilizar la orden ##getfacl##:

```
anita:/# getfacl /usr/local/sbin/sshd

# file: /usr/local/sbin/sshd
# owner: root
# group: bin
user::rwx
group::
#effective:


mask:

other:

anita:/#
```

Acabamos de visualizar una lista de control de acceso de Solaris; en primer lugar se nos

indica el nombre del fichero, su propietario y su grupo, todos precedidos por ##`#'##. Lo que vemos a continuación es la propia lista de control: los campos ##user##, ##group## y ##other## son básicamente la interpretación que ##getfacl## hace de los permisos del fichero (si nos fijamos, coincide con el resultado del ##ls -l##). El campo ##mask## es muy similar al ##umask## clásico: define los permisos máximos que un usuario (a excepción del propietario) o grupo puede tener sobre el fichero. Finalmente, el campo ##effective## nos dice, para cada usuario (excepto el propietario) o grupo el efecto que la máscara tiene sobre los permisos: es justamente el campo que tenemos que analizar si queremos ver quién puede acceder al archivo y de qué forma.

Sin embargo, hasta ahora no hemos observado nada nuevo; podemos fijarnos que la estructura de la lista de control de acceso otorga los mismos permisos que las ternas clásicas. Esto es algo normal en todos los Unix: si no indicamos lo contrario, al crear un fichero se le asocia una ACL que coincide con los permisos que ese archivo tiene en el sistema (cada archivo tendrá una lista asociada, igual que tiene unos permisos); de esta forma, el resultado anterior no es más que la visión que ##getfacl## tiene de los bits //rwx// del fichero ([Gal96c]).

Lo interesante de cara a la protección de ficheros es extender los permisos clásicos del archivo, modificando su lista asociada. Esto lo podemos conseguir con la orden ##setfacl##:

```
anita:~# setfacl -m user:toni:r-x /usr/local/sbin/sshd
anita:~# getfacl /usr/local/sbin/sshd

# file: /usr/local/sbin/sshd
# owner: root
# group: bin
user::rwx
user:toni:r-x #effective:

group::
#effective:

mask:

other:

anita:~#
```

Como vemos, acabamos de modificar la lista de control de acceso del archivo para asignarle a ##toni## permiso de ejecución y lectura sobre el mismo. La orden ##setfacl## se utiliza principalmente de tres formas: o bien añadimos entradas a la ACL, mediante la opción ##-m## seguida de las entradas que deseemos añadir separadas por comas (lo que hemos hecho en este caso, aunque no se han utilizado comas porque sólo hemos añadido una entrada), o bien utilizamos el parámetro ##-s## para reemplazar la ACL completa (hemos de indicar todas las entradas, separadas también por comas), o bien borramos entradas de la lista con la opción ##-d## (de sintaxis similar a ##-m##). Cada entrada de la ACL tiene el siguiente formato:

##tipo:UIDGID:permisos##

El tipo indica a quién aplicar los permisos (por ejemplo, ##user## para el propietario del archivo, o ##mask## para la máscara), el UID indica el usuario al que queremos asociar la

entrada (como hemos visto, se puede utilizar también el //login//, y el GID hace lo mismo con el grupo (de la misma forma, se puede especificar su nombre simbólico). Finalmente, el campo de permisos hace referencia a los permisos a asignar, y puede ser especificado mediante símbolos //rwx-// o de forma octal.

Acabamos de indicar que el usuario //toni// tenga permiso de lectura y ejecución en el archivo; no obstante, si ahora este usuario intenta acceder al fichero en estos modos obtendrá un error:

```
anita:/usr/local/sbin$ id
uid=100(toni) gid=10(staff)
anita:/usr/local/sbin$ ./sshd
bash: ./sshd: Permission denied
anita:/usr/local/sbin$
```

> Qué ha sucedido? Nada anormal, simplemente está actuando la máscara sobre sus permisos (antes hemos dicho que debemos fijarnos en el campo ##effective##, y aquí podemos comprobar que no se ha modificado). Para solucionar esto hemos de modificar el campo ##mask##:

```
anita:~# setfacl -m mask:r-x /usr/local/sbin/sshd
anita:~#
```

Si ahora //toni// intenta acceder al fichero para leerlo o ejecutarlo, ya se le va a permitir:

```
anita:/usr/local/sbin$ id
uid=100(toni) gid=10(staff)
anita:/usr/local/sbin$ ./sshd
/etc/sshd_config: No such file or directory
...
```

Aunque obtenga un error, este error ya no depende de la protección de los ficheros sino de la configuración del programa: el administrador obtendría el mismo error. No obstante, sí que hay diferencias entre una ejecución de //toni// y otra del //root//, pero también son impuestas por el resto del sistema operativo Unix: //toni// no podría utilizar recursos a los que no le está permitido el acceso, como puertos bien conocidos, otros ficheros, o procesos que no le pertenezcan. Hay que recordar que aunque un usuario ejecute un archivo perteneciente al //root//, si el fichero no está setuidado los privilegios del usuario no cambian. Sucede lo mismo que pasaría si el usuario tuviera permiso de ejecución normal sobre el fichero, pero éste realizara tareas privilegiadas: podría ejecutarlo, pero obtendría error al intentar violar la protección del sistema operativo.

En Solaris, para indicar que una lista de control de acceso otorga permisos no reflejados en los bits //rwx// se situa un símbolo ##`+'## a la derecha de los permisos en un listado largo:

```
anita:~# ls -l /usr/local/sbin/sshd
-rwx
```

+ 1 root bin 2616160 Apr 28 1997 /usr/local/sbin/sshd

anita:~#

Otra característica que tiene Solaris es la capacidad de leer las entradas de una lista de control de acceso desde un fichero en lugar de indicarlas en la línea de órdenes, mediante la opción ##-f## de ##setfacl##; el formato de este fichero es justamente el resultado de ##getfacl##, lo que nos permite copiar ACLs entre archivos de una forma muy cómoda:

```
anita:~# getfacl /usr/local/sbin/sshd >/tmp/fichero
anita:~# setfacl -f /tmp/fichero /usr/local/sbin/demonio
anita:~# getfacl /usr/local/sbin/demonio

# file: /usr/local/sbin/demonio
# owner: root
# group: other
user::rwx
user:toni:r-x #effective:r-x
group::
#effective:

mask:r-x
other:

anita:~#
```

Esto es equivalente a utilizar una tubería entre las dos órdenes, lo que produciría el mismo resultado:

```
anita:~# getfacl /usr/local/sbin/sshd | setfacl -f - /usr/local/sbin/demonio
```

Antes de finalizar este apartado dedicado a las listas de control de acceso, quizás sea conveniente comentar el principal problema de estos mecanismos. Está claro que las ACLs son de gran ayuda para el administrador de sistemas Unix, tanto para incrementar la seguridad como para facilitar ciertas tareas; sin embargo, es fácil darse cuenta de que se pueden convertir en algo también de gran ayuda, pero para un atacante que desee situar puertas traseras en las máquinas. Imaginemos simplemente que un usuario autorizado de nuestro sistema aprovecha el último //bug// de ##sendmail## (realmente nunca hay un `último') para conseguir privilegios de administrador en una máquina; cuando se ha convertido en //root// modifica la lista de control de acceso asociada a ##/etc/shadow## y crea una nueva entrada que le da un permiso total a su //login// sobre este archivo. Una vez hecho esto, borra todo el rastro y corre a avisarnos del nuevo problema de ##sendmail##, problema que rápidamente solucionamos, le damos las gracias y nos

olvidamos del asunto. >Nos olvidamos del asunto? Tenemos un usuario que, aunque los bits //rwx// no lo indiquen, puede modificar a su gusto un archivo crucial para nuestra seguridad. Contra esto, poco podemos hacer; simplemente comprobar frecuentemente los listados de todos los ficheros importantes (ahora entendemos por qué aparece el símbolo ##`+'## junto a las ternas de permisos), y si encontramos que un fichero tiene una lista de control que otorga permisos no reflejados en los bits //rwx//, analizar dicha lista mediante ##getfacl## y verificar que todo es correcto. Es muy recomendable programar un par de //shellscripts// simples, que automaticen estos procesos y nos informen en caso de que algo sospechoso se detecte.