Control De Acceso En Los Sistemas Computacionales

Que Es El Control De Acceso

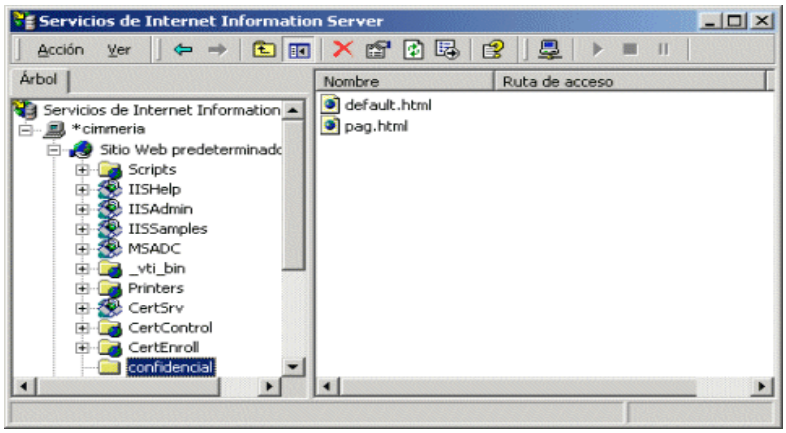
El control de acceso constituye una poderosa herramienta para proteger la entrada a un web completo o sólo a ciertos directorios concretos e incluso a ficheros o programas individuales. Este control consta generalmente de dos pasos:

- En primer lugar, la **autenticación**, que identifica al usuario o a la máquina que trata de acceder a los recursos, protegidos o no.
- En segundo lugar, procede la cesión de derechos, es decir, la **autorización**, que dota al usuario de privilegios para poder efectuar ciertas operaciones con los datos protegidos, tales como leerlos, modificarlos, crearlos, etc.

Por defecto, todas las páginas y servicios del servidor web se pueden acceder anónimamente, es decir, sin necesidad de identificarse ante el servidor y sin ningún tipo de restricción. En máquinas NT, el usuario anónimo pertenece al grupo Invitados y tiene asignada la cuenta IUSR_nombremáquina, donde nombremáquina toma el valor del nombre del servidor: para una máquina llamada Mordor, la cuenta de acceso anónimo a Internet sería IUSR_MORDOR. Esta cuenta anónima debe tener permiso para conectarse localmente. En Linux, en cambio, no es necesario crear una cuenta en la máquina para los usuarios anónimos.

Análogamente, toda la información que viaja por las redes de comunicaciones lo hace en claro, de manera que puede ser fácilmente interceptada por un atacante. De ahí la necesidad de proteger los datos mientras se encuentran en tránsito por medio de un canal cifrado, para lo que se utiliza normalmente SSL, como se describirá más adelante.

Los diversos métodos de control de acceso presentados en este curso se han particularizado para dos servidores ampliamente difundidos en Internet: IIS 5.0 corriendo bajo Windows 2000, y servidores para Linux tipo Apache, como Stronghold 2.4. Aunque en otros servidores el proceso no será idéntico, sí es cierto que resultará muy parecido. Se puede consultar una completa comparativa de servidores web en webcompare.internet.com.



Consola de administración para internet Information Services 5.0.

Control de accseso interno

Palabras Claves (Passwords)

Generalmente se utilizan para realizar la autenticación del usuario y sirven para proteger los datos y aplicaciones. Los controles implementados a través de la utilización de palabras clave resultan de muy bajo costo. Sin embargo cuando el usuario se ve en la necesidad de utilizar varias palabras clave para acceder a diversos sistemas encuentra dificultoso recordarlas y probablemente las escriba o elija palabras fácilmente deducibles, con lo que se ve disminuida la utilidad de esta técnica.

Se podrá, por años, seguir creando sistemas altamente seguros, pero en última instancia cada uno de ellos se romperá por este eslabón: la elección de passwords débiles.

Es mi deseo que después de la lectura del presente quede la idea útil de usar passwords seguras ya que aquí radican entre el 90% y 99% de los problemas de seguridad planteados.

- **Sincronización de passwords:** consiste en permitir que un usuario acceda con la misma password a diferentes sistemas interrelacionados y, su actualización automática en todos ellos en caso de ser modificada. Podría pensarse que esta es una característica negativa para la seguridad de un sistema, ya que una vez descubierta la clave de un usuario, se podría tener acceso a los múltiples sistemas a los que tiene acceso dicho usuario. Sin embargo, estudios hechos muestran que las personas normalmente suelen manejar una sola password para todos los sitios a los que tengan acceso, y que si se los fuerza a elegir diferentes passwords tienden a guardarlas escritas para no olvidarlas, lo cual significa un riesgo aún mayor. Para implementar la sincronización de passwords entre sistemas es necesario que todos ellos tengan un alto nivel de seguridad.
- **Caducidad y control:** este mecanismo controla cuándo pueden y/o deben cambiar sus passwords los usuarios. Se define el período mínimo que debe pasar para que los usuarios puedan cambiar sus passwords, y un período máximo que puede transcurrir para que éstas caduquen.

Encriptación

La información encriptada solamente puede ser desencriptada por quienes posean la clave apropiada. La encriptación puede proveer de una potente medida de control de acceso. Este tema será abordado con profundidad en el Capítulo sobre Protección del presente.

Listas de Control de Accesos

Se refiere a un registro donde se encuentran los nombres de los usuarios que obtuvieron el permiso de acceso a un determinado recurso del sistema, así como la modalidad de acceso permitido. Este tipo de listas varían considerablemente en su capacidad y flexibilidad.

Límites sobre la Interfase de Usuario

Esto límites, generalmente, son utilizados en conjunto con las listas de control de accesos y restringen a los usuarios a funciones específicas. Básicamente pueden ser de tres tipos: menús, vistas sobre la base de datos y límites físicos sobre la interfase de usuario. Por ejemplo los cajeros automáticos donde el usuario sólo puede ejecutar ciertas funciones presionando teclas específicas.

Etiquetas de Seguridad

Consiste en designaciones otorgadas a los recursos (como por ejemplo un archivo) que pueden utilizarse para varios propósitos como control de accesos, especificación de medidas de protección, etc. Estas etiquetas no son modificables.

Control de acceso externo

Dispositivos de Control de Puertos

Estos dispositivos autorizan el acceso a un puerto determinado y pueden estar físicamente separados o incluidos en otro dispositivo de comunicaciones, como por ejemplo un módem.

Firewalls o Puertas de Seguridad

Permiten bloquear o filtrar el acceso entre dos redes, usualmente una privada y otra externa (por ejemplo Internet). Los firewalls permiten que los usuarios internos se conecten a la red exterior al mismo tiempo que previenen la intromisión de atacantes o virus a los sistemas de la organización. Este tema será abordado con posterioridad.

Acceso de Personal Contratado o Consultores

Debido a que este tipo de personal en general presta servicios temporarios, debe ponerse especial consideración en la política y administración de sus perfiles de acceso.

Accesos Públicos

Para los sistemas de información consultados por el público en general, o los utilizados para distribuir o recibir información computarizada (mediante, por ejemplo, la distribución y recepción de formularios en soporte magnético, o la consulta y recepción de información a través del correo electrónico) deben tenerse en cuenta medidas especiales de seguridad, ya que se incrementa el riesgo y se dificulta su administración.

Debe considerarse para estos casos de sistemas públicos, que un ataque externo o interno puede acarrear un impacto negativo en la imagen de la organización.

Listas 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 enrutadores y conmutadores. 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 RDSI.

Una de las características que se echan en falta en los sistemas Linux actuales, especialmente en entornos corporativos, es la posibilidad de especificar los permisos de acceso a los ficheros con mayor granularidad. Los sistemas Linux siguen el modelo de permisos Unix tradicional segementando el tipo de acceso en tres categorías:

- El propietario del fichero (User)
- El grupo al que pertenece el fichero (Group)
- El resto de usuarios del sistema que no están en ninguna de las dos categorías anteriores (Other).

también conocido como modelo UGO (User, Group, Other).

Sin embargo, estas tres categorías se revelan insuficientes en una gran cantidad de situaciones, donde desearíamos poder especificar permisos diferenciados para varios usuarios o grupos determinados.

Aquí es donde entran en juego los permisos basados en Listas de Control de Acceso, más conocidos como ACLs. En este sistema de permisos los ficheros no tienen un juego fijo de permisos (como en el modelo tradicional, que tiene tres permisos y sólo tres), sino que los permisos del fichero son en realidad una lista de Entradas de Control de Acceso (o ACEs).

Cada una de estas ACEs contiene un par (usuario/grupo, permiso) que indica un tipo de acceso determinado para un usuario o grupo, y el conjunto de todas ellas forman la ACL que marca el tipo de acceso permitido en un fichero.

Este sistema es el utilizado entre otros, por los sistemas de ficheros NTFS (de Windows NT y sucesores), el sistema UFS de Solaris y el sistema HFS de HP-UX.

ACLs en Linux

Hay un dato que se suele desconocer sin embargo y es que el sistema de ficheros ext2, desde su diseño original, previó la inclusión de este tipo de sistemas de control de acceso y estaban incluídos los enganches (*hooks*) necesarios para su implementación posterior, de forma 100% transparente y compatible hacia atrás.

Pues bien, desde hace varios años existen varios proyectos para incorporar los sistemas basados en ACL en los diferentes sistemas de ficheros soportados por Linux, y especialmente en ext2 (por ser el tipo de sistema de ficheros más extendido hasta el momento en los sistemas Linux).

Uno de ellos es el proyecto RSBAC, cuyos objetivos son mucho más ambiciosos (realmente mucho más, su objetivo es conseguir un sistema Linux con un nivel de seguridad equivalente al nivel B1 del antiguo Libro Naranja -TCSEC-), pero que incorpora también una implementación 100% operativa de ACLs para ext2.

Otro de ellos es el proyecto Linux Extended Attributes and Access Control Lists, que originalmente tenía como objetivo incorporar el sistema de ACLs al sistema de ficheros ext2 (y posteriormente ext3 cuando este apareció). Posteriormente, al ser el sistema de ACLs elegido por el equipo de Samba para su implementación de ACLs sobre ext2 (para poder ofertar recursos compartidos via SMB con ACLs al igual que los sistemas Windows NT y posteriores) ha sido el candidato oficial de ACLs en ext2 para su inclusión definitiva en el kernel. De hecho, desde la versión 2.5.23 forma parte del kernel estándar.

Para ello se ha hecho un esfuerzo de coordinación y estandarización bastante grande con los desarrolladores de otros sistemas de ficheros como XFS y JFS (principalmente, aunque no sólo con estos) que también soportaban ACLs desde su origen. La idea era ofertar una capa abstracta común en el VFS (*Virtual File System*) de forma que las aplicaciones y el resto del sistema operativo trabajasen por igual, y con la misma API, con cualquiera de los sistemas de ficheros que soportasen ACLs (ext2/ext3, XFS, JFS, ReiserFS, etc.).

El resultado: ya están disponibles los sistemas de ficheros con ACLs en el núcleo estándar, en su versión de desarrollo. Sin embargo, no es necesario esperar hasta la estabilización del actual núcleo de desarrollo para disfrutar de las ventajas de las ACLs. Todos los sistemas de ficheros mencionados arriba están disponibles bien como parte del kernel estándar, bien como parches, para las versiones estables del núcleo. Y son parches con calidad de producción, con lo cual son perfectamente utilizables en entornos cuya estabilidad sea una condición indispensable.

En el caso del sistema de ficheros ext2/ext3, que son el objetivo del proyecto Linux Extended Attributes and Access Control Lists, las ACLs son incluso transparentes para aquellos núcleos que no lleven incorporados los parches necesarios, de forma que si accidentalmente se arranca el sistema con un núcleo sin soporte para ACLs no ocurre absolutamente nada, salvo obviamente que no disponemos de las características avanzadas de las ACLs y sólo tenemos a nuestra disposición el modelo de permisos tradicional.

A condición de que tengamos un ejecutable de e2fsck que soporte ACLs, incluso si el núcleo no lo soporta, podemos ejecutar e2fsck sobre el sistema de ficheros de forma segura. En caso de tener un e2fsck sin soporte de ACLs, el único problema que tendremos en este caso es la pérdida de las ACLs, pero nunca la pérdida de datos.

Las versiones de e2fsprogs (el paquete donde se incluyen las utilidades de ext2) a partir de la version 1.28 ya incorporan soporte de serie para Atributos Extendidos (Extended Attributes o EAs), que es la característica del sistema de ficheros necesaria para poder implementar las ACLs. En el sitio del propio proyecto se pueden encontrar parches para algunas versiones anteriores de e2fsprogs, en caso de ser necesario.

Cómo incorporar ACLs a nuestro sistema Linux

¿Qué debemos hacer por tanto para disfrutar de ACLs en nuestros sistemas de ficheros ext2/ext3 ya mismo? Lo que sigue es un resumen de las instrucciones dadas en el propio sitio del proyecto, donde se indica paso a paso todo el proceso a seguir. Voy a presuponer en este caso que nuestro sistema no dispone de sistemas de ficheros con EAs y ACLs, con lo cual me voy a saltar los pasos de copia de respaldo de las ACLs actuales (y la limpieza del sistema).

1. Lo primero es obtener una copia de e2fsprogs que soporte EAs y ACLs. La version 1.28 o posterior servirá. En caso contrario, hay que obtener los parches mencionados arriba y las fuentes de nuestra version actual de e2fsprogs (seguramente nuestra distribución tendrá disponibles las fuentes en su repositorio habitual) y construir de nuevo los ejecutables e instalarlos en el sistema. Alternativamente podemos optar por instalar directamente una versión 1.28 o posterior directamente, sin parchear la actualmente usada en nuestra distribución.
2. A continuación debemos compilar un kernel con soporte para EAs y ACLs. Para ello debemos descargar los parches correspondientes a nuestro núcleo y aplicarlos de la siguiente forma (el directorio linux/ indica la ubicación donde tenemos desempaquetadas las fuentes del núcleo):
3. $ cd linux/
4. $ zcat ../linux-a.b.c-xattr-x.y.z.diff.gz | patch -p1
5. $ zcat ../linux-a.b.c-acl-x.y.z.diff.gz | patch -p1

   Una vez parcheado es necesario incluir en la compilación las siguientes opciones al menos (disponibles en la categoria File Systems):

   CONFIG_FS_POSIX_ACL=y (POSIX Access Control Lists)

CONFIG_EXT3_FS_XATTR=y (Ext3 extended attributes)
CONFIG_EXT3_FS_POSIX_ACL=y (Ext3 POSIX Access Control Lists)
CONFIG_EXT2_FS_XATTR=y (Ext2 extended attributes)
CONFIG_EXT2_FS_POSIX_ACL=y (Ext2 POSIX Access Control Lists)

Los nombres pueden variar ligeramente de una versión del kernel a otra. Las opciones xxx_XATTR son para activar los Atributos Extendidos, y las opciones xxx_POSIX_ACL para activar las ACLs. Los valores xxx_EXT2_xxx son para sistemas de ficheros ext2 y los valores xxx_EXT3_xxx para ext3.

Existen otras dos opciones más (al menos en el kernel 2.4.19, que es el que estoy usando para escribir esto):

I. Ext2 extended attribute block sharing (IG_EXT2_FS_XATTR_SHARING): que permite el uso de un mismo bloque común para almacenar los atributos extendidos de varios nodos-i, en el caso de que dichos atributos sean indénticos en todos los nodos-i.
II. Ext2 extended user attributes (CONFIG_EXT2_FS_XATTR_USER): que permite a los procesos de usuario almacenar atributos extendidos adicionales en los nodos-i. Por ejemplo para almacenar cualquier tipo de información adicional que dichos procesos deseen, como el juego de caracteres usuado en el fichero (o cualquier otra cosa que se nos ocurra).

Por supuesto, los dos valores anteriores existen de idéntica forma para el sistema de ficherox ext3.

6. El siguiente paso es construir las utilidades que sirven para gestionar los AEs y las ACLs de los ficheros: getfattr, setfattr, getfacl, setfacl, etc. Tenemos que descargar el paquete attr que es necesario para poder construir despues el paquete acl, que incluye las utilidades de gestión de las ACLs en sí (el paquete attr incluye además algunas utilidades de gestión de atributos extendidos).

   Es conveniente revisar primero si existen versiones ya empaquetas para nuestra distribución de Linux (tanto Debian GNU/Linux como RedHat, entre otras, las tienen), y que sea una versión de las mismas que sea compatible con la revisión del parche aplicada al núcleo que acabamos de construir

   En caso de no tenerlas en nuestra distribución, no ser compatibles o simplemente querer tener la última versión disponible compilada por nosotros mismos, estos son los pasos a seguir (necesitaremos el entorno de desarrollo autoconf y todas la utilidades de compilación/desarrollo habituales, más algún retoque manual):

   I. Desempaquetar las utilidades de gestión de Atributos Extendidos (attr-2.0.10.src.tar.gz en el momento de escribir esto):
   II. tar xzvf attr-2.0.10.src.tar.gz
   III. cd attr-2.0.10

A continuación tenemos que compilar las herramientas en sí. Si nuestra distribución es RedHat o Debian GNU/Linux, las utilidades vienen con los ficheros de especificación y control necesarios para crear paquetes nativos (ver el fichero doc/INSTALL).

En caso contrario, debemos compilar todo desde cero con las siguientes instrucciones (en el directorio raíz de los ficheros extraídos):

```
autoconf
./configure
make
su root
make install install-lib install-dev
```

Este último método dejará todos los ejecutables más las bibliotecas de funciones de manejo de AEs bajo /usr/local. En general el fichero doc/INSTALL da las indicaciones necesarias para obtener los ejecutables en cualquiera de los casos (incluídas algunas indicaciones para deshabilitar el código de depuración, etc).

IV. Desempaquetar las utilidades de gestión de ACLs (acl-2.0.18.src.tar.gz en el momento de escribir esto):

V. `tar xzvf acl-2.0.18.src.tar.gz`

VI. `cd acl-2.0.18`

A continuación tenemos que compilar las herramientas en sí. Como en el caso anterior, si nuestra distribución tiene ya compiladas estas utilidades y con compatibles con la versión de los parches incluídos en el núcleo, lo más sencillo es usar los paquetes nativos de la distribución.

En caso contrario, debemos compilar todo desde cero con las siguientes instrucciones (en el directorio raíz de los ficheros extraídos):

```
autoconf
./configure
make
su root
make install install-lib install-dev
```

Este último método dejará todos los ejecutables más las bibliotecas de funciones de manejo de ACLs bajo /usr/local. Como en el caso anterior, el

fichero doc/INSTALL da las indicaciones necesarias para obtener los ejecutables en cualquiera de los casos.

VII. Por último debemos obtener una versión del paquete fileutils parcheado para soportar EAs y ACLs. De lo contrario no podremos copiar los ficheros con sus AEs y ACLs, sacar en los listados la indicación de que hay acls adicionales a los permisos tradicionales (ya que se siguen manteniendo los permisos tradicionales), etc.

    En el propio sitio del proyecto se puede encontrar el parche para algunas versiones del paquete fileutils. En general esta es la parte más complicada del proceso, al ser un paquete completamente externo al sistema de ficheros y las utilidades ext2/ext3. Los problemas se presentarán si la versión de fileutils que vamos a utilizar no se corresponden exactamente a las indicadas en los parches, ya que tendremos que integrar parte de los cambios a mano, y eso siempre es más complicado.

Una vez compilado e instalado todo lo mencionado en los puntos anteriores ya tenemos todos los elementos necesarios para poder disfrutar de los EAs y las ACLs en nuestros sistemas de ficheros ext2/ext3. Sólo nos queda un último paso, indicar al núcleo que en un determinado sistema de ficheros deseamos usar ACLs (y EAs).

Para ello debemos editar el fichero /etc/fstab y añadir una opción adicional a la de aquellos sistemas de ficheros a los que queremos activar las ACLs : *acl*. También podemos añadir una opción para indicar explícitamente que no queremos usar las ACLs en un sistema de ficheros, aun si dicho sistema de ficheros contiene ACLs: *noacl*.

Existe un juego de opciones adicional para activar o desactivar el uso de los AEs de usuario (si hemos optado por compilarlos en nuestro núcleo): *user_xattr* y *nouser_xattr*. Por cierto, que los valores por defecto para todos los sistemas de ficheros ext2/ext3 en caso de no especificar nada son *noacl* y *nouser_xattr*

Una vez hecho lo anterior, sólo nos resta arrancar con el nuevo núcleo que acabamos de compilar y *¡voilá!*, ya tenemos nuestras ACLs disponibles y listas para usar.

Uso de las ACLs

Una vez que hemos compilado el kernel con soporte para ACLs y que hemos compilado las herramientas necesarias para poder trabajar con ellas (además de parchear aquellas utilidades de gestión de ficheros y sistemas de ficheros para que reconozcan las ACLs y las respeten y sepan interpretarlas), podemos ya dedicarnos a la gestión de las propias ACLs en si.

Para ello disponemos de dos utilidades principalmente:

- getfacl: que nos permite consultar las ACLs de un fichero dado.

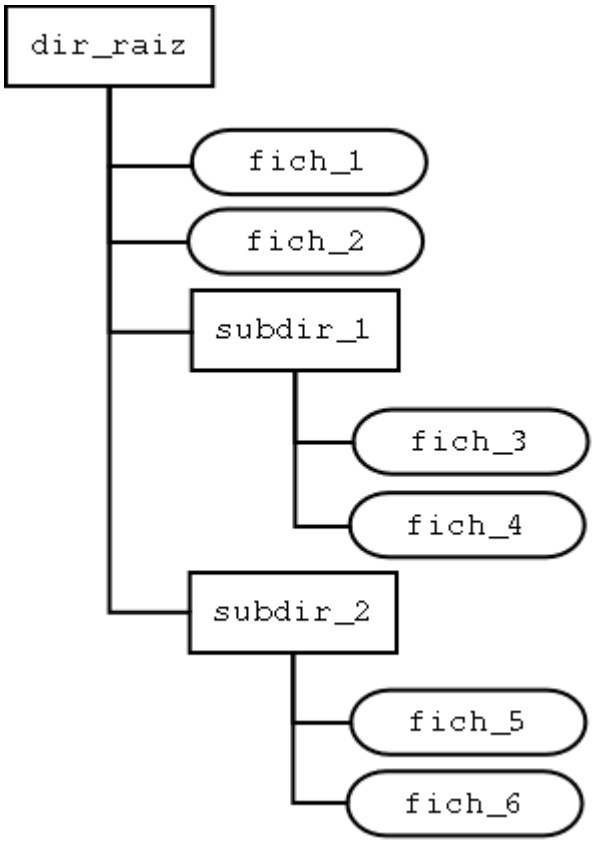
- setfacl: que nos permite modificar las ACLs de un fichero dado.

Ambas utilidades se hayan perfectamente documentadas en sus respectivas páginas del manual (setfacl(1) y getfacl(1)), y podemos encontrar una pequeña introducción en la página del manual de acl(5), vamos a ver aquí algunos ejemplos sencillos que ilustrarán el uso básico de estas utilidades.

Creación de una ACL básica.

Existen casos en los cuales el tradicional juego de pemisos UGO (User, Group, Others) no es suficiente. Casos en los que desearíamos que más de un usuario o más de un grupo pudiese tener acceso a un fichero, pero con permisos diferenciados. Con el modelo UGO esto no es posible, puesto que sólo tenemos sitio para los permisos de un único Usuario o un único Grupo. Todo lo demás cae en el Other (el resto). Con las ACLs esto es sencillo.

Vamos a presuponer que tenemos un directorio que contiene una serie de ficheros y dos subdirectorios (a su vez con ficheros) como se indica en la siguiente figura:



Tenemos asimismo los siguiente tipos de usuarios diferenciados:

1. Los usuarios del grupo de sistemas de información, que deben tener acceso completo a todos los directorios y ficheros, para su mantenimiento. Llamaremos a este grupo *sistemas*.
2. Los usuarios del grupo de desarrollo (llamaremos a este grupo *desarrollo*), que deben tener acceso de lectura y escritura (y ejecución en su caso) en el primer subdirectorio y todos sus ficheros y subdirectorios. En este directorio es donde se desarrollan las nuevas versiones del software que usa el tercer grupo.

   Asi mismo debe tener acceso de lectura/escritura al segundo de los subdirectorios para implantar las nuevas versiones estables del software.

3. Los usuarios de explotación. Este grupo debe tener acceso en modo lectura (y eventualmente ejecucion) sobre los ficheros de la aplicacion. Llamaremos a este grupo *explotacion*.
4. El resto de usuarios del sistema no deben tener ningún tipo de acceso a ninguno de los ficheros o subdirectorios.

Con las condiciones anteriores es imposible usar el modelo de permisos tradicional UGO, puesto que tenemos tres grupos de usuarios diferentes, sin contar el resto de usuarios del sistema. Eso significa que con un sólo grupo de usuarios con permisos asignados por directorio o fichero no es posible acomodar los permisos para los tres grupos.

Utilizando ACLs es fácil resolver el problema, puesto que podemos asignar un juego de permisos diferente para cada grupo de usuarios. Para ello debemos crear las ACLs necesarias para cada grupo de usuarios y directorio o fichero.

1. Tenemos que dar permisos completos al grupo de sistemas sobre todos los ficheros y directorios implicados. Para ello usamos setfacl de la siguiente forma:
2. setfacl -b -k -R dir_raiz
3. setfacl -R -m g:sistemas:rw

   Vayamos por partes con la sintaxis de setfacl:

   - En el primer caso usamos la opcion -b para borrar la ACL que ya pudiera tener el directorio raíz. Usamos también la opción -k para borrar la ACL por defecto que pudiera tener el directorio raíz (más sobre esto después), y por último usamos la opción -R para aplicar los cambios de forma recursiva a todo lo que cuelga del directorio raíz. Con esto conseguimos tener todo limpio y listo para empezar.

     Teóricamente la primere vez no hace falta limpiar la ACL puesto que aún no hemos asignado ninguna ACE, pero de paso aprovechamos para mostrar como se hace :)

- En el segundo caso indicamos de nuevo que queremos aplicar de forma recursiva los cambios (opción -R) pero esta vez le decimos que queremos modificar (-m) la ACL del objeto en cuestión. En este caso es necesario además indicar el valor de la ACE que deseamos añadir o modificar. *setfacl* distingue entre asignar una ACL (opción -s) en la cual se eliminan las ACEs existentes y se añade la ACE especificada en la orden, y modificar una ACL (opción -m) en la cual podemos modificar o añadir una ACE.

  En este caso queremos añadir una nueva ACE. Para ello debemos escribimos la ACE que nos interesa:

  g:sistemas:rw

  Todas las ACEs tienen tres componentes separadas por ':' (en el caso de las operaciones de borrado de ACEs el tercer componente es opcional). El primero de los componentes indica si se trata de un ACE de usuario (valor u) o de grupo (valor g). Incluso es posible asignar ACEs al grupo de usuarios *resto* (other), pero esto no suele ser habitual, así que omitiremos la sintaxis (se pueden encontrar más detalles en la página del manual de setfacl(1)).

  El segundo de los componentes es el nombre de usuario o grupo al que se le aplica la ACE. Se puede dar el nombre simbólico o el valor numérico del uid o gid correspondiente. El tercer componente es el valor del permiso asociado a esta ACE, y puede ser una combinación cualquiera de las letras r, w, x y -, o un valor numérico octal (como en chmod).

  Por tanto, en nuestro caso tenemos que es una ACE de grupo (g), que se aplica al grupo de sistemas y que le estamo dando los permisos de lectura y escritura (rw).

4. A continuacion tenemos que dar permisos al grupo de desarrollo tanto en el directorio subdir_1 y todo su contenido, como en el directorio subdir_2 y todo su contenido. Sin embargo no tenemos que dar acceso a los ficheros que cuelgan directamente de dir_raiz. Para ello usamos setfacl con la sintaxis explicada en el punto anterior:
5. setfacl -R -m g:desarrollo:rw dir_raiz/subdir_1
6. setfacl -R -m g:desarrollo:rw dir_raiz/subdir_2

7. Por último tenemos que dar permisos al grupo de explotación en el directorio subdir_2 y todo su contenido:
8. setfacl -R -m g:explotacion:rx dir_raiz/subdir_2

Lo normal en este punto es comprobar cuales son los permisos reales que tienen cada uno de los ficheros y directorios existentes, para ver si efectivamente se ajustan a los requisitos planteados.

Para ello podemos usar getfacl para ver cuales son las ACL de cada uno de los directorios o ficheros implicados. La sintaxis es sencilla:

```
getfacl fichero ...
```

Si lo usamos para ver el resultado de las órdenes anteriores tenemos:

```
# getfacl dir_raiz
# file: dir_raiz
# owner: root
# group: root
user::rwx
group::r-x
group:sistemas:rw-
mask::rwx
other::r-x
```

El listado de permisos que obtenemos al ejecutar getfacl se compone de entradas de tipo user y group, además de las entradas para mask y other. En el caso de las entradas user y group, tendremos siempre una entrada para el propietario del fichero y el grupo del fichero (son las líneas que no indican un usuario o grupo concreto) y tantas líneas adicionales como ACEs de ese tipo hayamos asginado al fichero o directorio. La entrada other es la entrada del permiso tradicional *other* del modelo UGO.

La entrada mask es especial. Esa entrada, que se puede manipular con setfacl permite especificar el máximo de permisos que se pueden asignar en dicho fichero con las ACEs de usuario y grupo.

Control de acceso obligatorio

En seguridad de la computadora, **control de acceso obligatorio** (**MAC**) refiere a un tipo de control de acceso por cuál el sistema obliga el operativo la capacidad de a *tema* o *iniciador* para tener acceso o realizar generalmente a una cierta clase de operación en *objeto* o *blanco*. En la práctica, un tema es generalmente un proceso o un hilo de rosca; los objetos son construcciones tales como archivos, directorios, puertos de TCP/UDP, segmentos compartidos de la memoria, etc. Los temas y los objetos cada uno tienen un sistema de cualidades de la seguridad. Siempre que un tema procure tener acceso a un objeto, una regla de la autorización hecha cumplir por el núcleo del sistema operativo

examina estas cualidades de la seguridad y decide a si el acceso puede ocurrir. Cualquier operación por cualquier tema en cualquier objeto será probada contra el sistema de autorización gobierna (aka *política*) para determinarse si se permite la operación.

Con control de acceso obligatorio, esta política de la seguridad centralmente es controlada por un administrador de la política de la seguridad; los usuarios no tienen la capacidad de eliminar el acceso de la política y, por ejemplo, de la concesión a los archivos que serían de otra manera restrictos. Por el contrario, **control de acceso discrecional** (DAC), que también gobierna la capacidad de temas de tener acceso a objetos, no prohibe a usuarios la capacidad de tomar decisiones de política y/o de asignar cualidades de la seguridad. (El sistema tradicional del Unix de usuarios, de grupos, y de permisos del rwx es un ejemplo de DAC.) los sistemas MAC-permitidos permiten que los administradores de la política pongan políticas organización-anchas de la seguridad en ejecución. Desemejante con de DAC, los usuarios no pueden eliminar o modificar esta política, accidentalmente o intencionalmente. Esto permite que los administradores de la seguridad definan una política central que esté garantizada (en principio) para ser hecha cumplir para todos los usuarios.

Históricamente y tradicionalmente, el MAC se ha asociado de cerca a **de niveles múltiples asegure (MLS)** sistemas. Criterios confiados en de la evaluación del sistema informático[1], el trabajo seminal sobre el tema que se refiere a menudo como el "libro anaranjado", define el MAC como "los medios de restringir el acceso a los objetos basados en la sensibilidad (según lo representado por una etiqueta) de la información contenida en los objetos y la autorización formal (*es decir*., separación) de los temas para tener acceso a la información de tal sensibilidad ". Las puestas en práctica tempranas del MAC tales como HPUX BLS, Harris CS/SX, y SGI confiaban en que IRIX todos fueron centrados en MLS.

Más recientemente, con el advenimiento de puestas en práctica por ejemplo SELinux (incorporado en los núcleos de Linux después de 2.6), el MAC ha comenzado a convertirse en más corriente principal y se está desarrollando fuera del lugar de MLS. Estas puestas en práctica más recientes del MAC han reconocido que la definición anaranjada estrecha del libro, enfocada como estaba en MLS, no es suficiente para el uso general.[2] Estas puestas en práctica proporcionan más profundidad y flexibilidad que puestas en práctica anterior MLS-enfocadas,[3] permitir que (por ejemplo) los administradores se centren en ediciones tales como ataques y malware de la red sin el rigor o los apremios de los sistemas de MLS.