T E M A 4

Contenido

TEMA 4

SEGURIDAD Y PROTECCIÓN

4.1. Introducción

Los términos "seguridad" y "protección" se suelen considerar como sinónimos, pero se puede hacer una distinción entre ambos. La seguridad comprende toda la problemática relacionada con hacer que los ficheros no puedan ser leídos o escritos por ninguna persona no autorizada, e incluye aspectos técnicos, administrativos, legales y políticos. Por otro lado, la protección se podría definir como el conjunto de mecanimos específicos del sistema operativo que tienen como finalidad el proporcionar seguridad.

4.2. Seguridad

La seguridad se aplica tanto a pérdida de datos como a intrusos que intentan hacer lo que no deben. Algunas de las causas comunes de pérdida de información son las siguientes:

- Desastres tales como incendios, terremotos, riadas, etc.
- Errores hardware o software.
- Errores humanos.

La mayoría de estos problemas se pueden solucionar realizando copias de seguridad y almacenando las cintas en lugares físicamente lejanos a los datos originales.

Existen dos tipos de intrusos: intrusos pasivos que únicamente quieren leer información no autorizada e intrusos activos que quieren realizar modificaciones no permitidas. Los intrusos se pueden clasificar en las siguientes categorías:

- Usuarios no técnicos motivados por la curiosidad.
- Personal cualificado (estudiantes, programadores, técnicos, ...), que consideran un reto personal violar la seguridad de un sistema.
- Intento de obtener beneficios económicos.
- Espionaje comercial o militar.

Obviamente, el esfuerzo dedicado a la seguridad depende del tipo de intrusión que se pretenda prevenir.

Podemos definir tres requisitos necesarios para la seguridad dentro de un sistema informático:

- Privacidad: la información debe ser accesible para lectura únicamente por las partes autorizadas. Este tipo de acceso incluye la impresión, tanto por pantalla como impresora, e incluso la revelación de la existencia de un determinado objeto o suceso.
- Integridad: los elementos relacionados con la seguridad únicamente pueden ser modificados por las partes autorizadas. Modificación incluye escritura, creación, cambio de estado y borrado.
- Disponibilidad: los elementos relacionados con la seguridad deben ser estar disponibles únicamente a las partes autorizadas.

4.2.1. Tipos de Amenazas

Los tipos de amenazas a la seguridad de un sistema informático los podemos caracterizar teniendo en cuenta como esta información es suministrada por el sistema. En general, hay un flujo de información de una fuente a un destino:

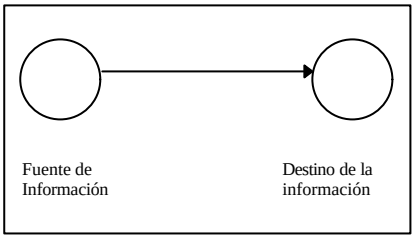

Figura 1.- Flujo Normal.

Teniendo esto en cuenta, podemos señalar cuatro categorías de amenazas:

- Interrupción: Un elemento del sistema es destruido o se hace inservible. Es una amenaza a la **disponibilidad**. Ejemplos son la destrucción de algún elemento hardware (discos, líneas de comunicación, etc.) y la desactivación del sistema de gestión de ficheros.

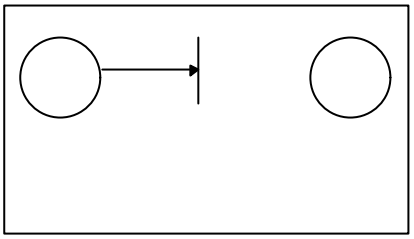

Figura 2.- Interrupción.

- Intercepción: Una parte no autorizada obtiene acceso a un elemento relacionado con la seguridad. Es una amenaza a la **privacidad**. La parte no autorizada puede ser una persona, un programa o un computador. Ejemplos son la copia ilícita de programas y la visualización de ficheros que han de permanecer ocultos.

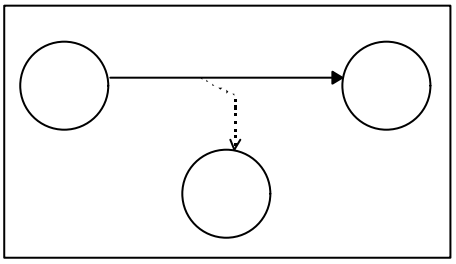

Figura 3.- Intercepción.

- Modificación: Una parte no autorizada no sólo obtiene acceso sino que puede modificar un elemento relacionado con la seguridad. Es una amenaza a la **integridad**. Ejemplos son la alteración del contenido de un fichero y modificar un programa para que funcione de forma diferente.

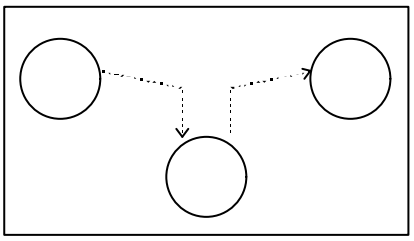

Figura 4.- Modificación.

- Fabricación: Una parte no autorizada inserta nuevos elementos en el sistema. Es una amenaza a la **integridad**. Ejemplos son adición de registros a un fichero y la inclusión de mensajes espúreos en una red.

Figura 5.- Fabricación.

.

La mayoría de los métodos de autenticación se basan en identificar algo que el usuario tiene o conoce. El mecanismo más común de autenticación consiste en que todo usuario ha de introducir una , que es solicitada por el programa de conexión cuando el usuario introduce su nombre. El inconveniente de este método es que las contraseñas pueden ser fácilmente averiguables si el usuario utiliza su nombre, dirección, o similar como contraseña. Otra forma de averiguar una contraseña consiste en probar todas las combinaciones de letras, números y símbolos de puntuación hasta adivinar la contraseña.

Existen variantes como que el sistema le añada a cada contraseña un número aleatorio, o asignarle a cada usuario un libro con una secuencia de contraseñas, de forma que cada vez que se conecta tiene que introducir la palabra de paso siguiente.

Otros tipos de mecanismos de autenticación se pueden basar en objetos que únicamente cada usuario puede tener, como tarjetas con banda magnética, al estilo de los cajeros automáticos. Otra

posibilidad es medir o comprobar ciertas características que están indisolublemente unidas a cada persona, como la voz, escritura, huellas dactilares, etc.

En instalaciones en las que la seguridad es prioritaria, estas medidas se pueden complementar con restricciones de acceso a la habitación en la que se encuentran los terminales, asignar a cada usuario un terminal concreto, establecer un horario concreto de trabajo, etc.

4.2.5. Amenazas de Origen Software

Uno de las los tipos más sofisticados de amenazas tienen su origen en programas que explotan las debilidades de los sistemas. Estos programas se dividen en dos grupos: aquellos que necesitan un programa anfitrión y aquellos que son independientes. Los primeros son trozos de programas que no pueden existir de forma autónoma, mientras que los segundos son programas completos que pueden ser planificados y ejecutados por el sistema operativo. También hay que distinguir entre aquellos programas que no se replican y los que lo hacen. Estos últimos son programas o trozos de programas que cuando se ejecutan pueden generar una o más copias de ellos mismos, que serán posteriormente activadas en la

Podemos distinguir seis tipos de amenazas de origen software:

- **Bomba Lógica**: Es un código incrustado en un programa que comprueba si ciertas condiciones se cumplen, en cuyo caso ejecuta alguna acción no autorizada. Estas condiciones pueden ser la existencia de ciertos ficheros, una fecha particular, la ejecución de una aplicación concreta, etc. Una vez que la bomba explota, puede alterar o eliminar datos, parar el sistema, etc. Un ejemplo de uso de bomba lógica es el caso de un programador que vende un programa a una empresa. Si transcurrido un cierto tiempo la empresa no ha pagado, el programador revela la existencia de la bomba lógica con el fin de obtener su dinero.
- **Puerta Falsa (*Trapdoor*)**: Es un punto de entrada secreto en un programa, de forma que alguien que conozca la existencia de dicha puerta puede obtener permisos de acceso sin tener que pasar por los mecanismos normales de autentificación. La puerta falsa es un código que reconoce alguna secuencia de entrada especial o se dispara si es ejecutado por cierto usuario o por la ocurrencia de una secuencia determinada de sucesos.
- **Caballo de Troya (*Trojan Horse*)**: Es una rutina oculta en un programa de utilidad. Cuando el programa se ejecuta, se ejecuta la rutina y ésta realiza acciones no autorizadas y perniciosas. Estos programas permiten realizar de forma indirecta acciones que no puede realizar de forma directa. Por ejemplo, un programa caballo de troya puede ser un editor que cuando es ejecutado modifica los permisos de los ficheros que edita de forma que éstos puedan ser accedidos por cualquier usuario. El autor del programa suele inducir a su utilización colocándolo en un directorio común y dándole un nombre de forma que aparente ser un programa de utilidad.
- **Virus**: Es código introducido en un programa que puede infectar otros programas mediante la copia de sí mismo en dichos programas. Además de propagarse, un virus realiza alguna función no permitida.
- **Bacteria**: Programa que consume recursos del sistema replicándose asimismo, pero no daña explícitamente ningún fichero. Se suele reproducir exponencialmente, por lo que puede acaparar recursos como CPU, memoria y disco.
- **Gusano (*Worm*)**: Es un programa que usa las redes de computadores para pasar de unos sistemas a otros. Una vez que llega a un sistema, el gusano se puede comportar como un virus o una bacteria, puede implantar programas caballo de troya, o puede realizar acciones no autorizadas. Para replicarse, los gusanos emplean algunos programas que proporcionan servicios de red, como correo electrónico, ejecución remota de programas y conexión a sistemas remotos.

4.3. Protección

La aparición de la multiprogramación introdujo en los computadores la posibilidad de compartir recursos entre varios usuarios. Entre los recursos compartidos están la CPU, la memoria, los dispositivos de entrada/salida, los programas y los datos. El hecho de poder compartir recursos es lo que introdujo la necesidad de protección.

Existen varios motivos por lo que la protección es necesaria. El más obvio es la necesidad de prevenir intentos de violación de restricciones de acceso por parte de un usuario. El más importante es, sin embargo, la necesidad de asegurar que cada proceso use los recursos del sistema de forma consistente de acuerdo con las políticas establecidas para el uso de esos recursos. Este requisito es fundamental para que un sistema sea fiable.

El papel de la protección en un sistema informático es proporcionar un conjunto de mecanismos que permita llevar a cabo las políticas que gobiernan el uso de los recursos. Estas políticas pueden ser fijas, establecidas en el diseño del sistema operativo, pueden ser formuladas por el administrador del sistema, o incluso puede ser establecidas por los usuarios.

Un principio importante es la separación entre política y mecanismo. Los mecanismos determinan hacer cierta cosa, mientras que las políticas deciden es lo que hay que hacer. En el ámbito de la protección, la política define qué datos han de ser protegidos y el mecanismo la forma de llevar a cabo la política. Esta separación favorece la flexibilidad. Debido a que las políticas pueden cambiar con el tiempo o con el lugar, es deseable que el sistema operativo ofrezca un conjunto general de mecanismos por medio de los cuales se puedan establecer políticas diferentes.

4.3.1. Dominios de Protección

Un computador contiene un conjunto de objetos que han de ser protegidos. Estos objetos pueden ser tanto hardware (CPUs, segmentos de memoria, discos, impresoras, etc.) como software (procesos, ficheros, semáforos, etc.). Cada objeto tiene un identificador único en el sistema y un conjunto de operaciones que le pueden ser aplicadas. Las operaciones posibles dependen del objeto. Por ejemplo, los segmentos de memoria pueden ser leídos o escritos, mientras que un fichero puede ser creado, abierto, leído, escrito, cerrado y borrado. El mecanismo de protección ha de prohibir a los procesos el acceso a aquellos objetos

Para estudiar los diferentes mecanismos de protección se introduce el concepto de **dominio de protección**. Un dominio de protección es un conjunto de pares objeto-derechos, de forma que cada par especifica un objeto y algún subconjunto de operaciones que se pueden realizar sobre dicho objeto. Los

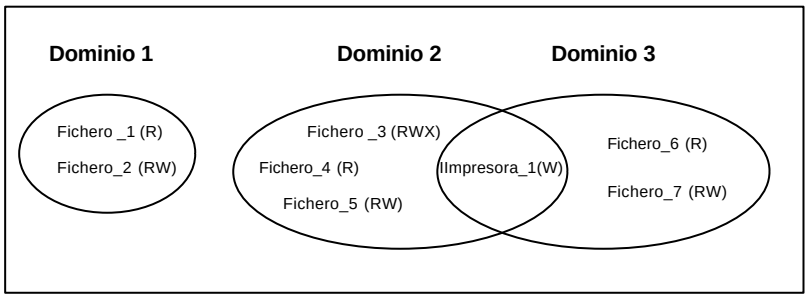

Figura 6.- Ejemplos de dominios de protección.

En cada instante de tiempo, cada proceso se ejecuta dentro de un dominio concreto. Este dominio puede cambiar durante la vida del proceso.

Un dominio puede ser varias cosas:

- Cada usuario puede ser un dominio, en cuyo caso el conjunto de objetos que puede ser accedido depende de la identidad del usuario. Los cambios de dominio ocurren cuando se cambia de usuario.
- Cada proceso puede ser un dominio, en cuyo caso el conjunto de objetos que puede ser accedido depende de la identidad del proceso. Los cambios de dominio se producen cuando un proceso envía un mensaje a otro y espera la respuesta.
- Cada procedimiento puede ser un dominio, en cuyo caso el conjunto de objetos que puede ser accedido se corresponde con las variables locales definidas dentro del procedimiento. Los cambios de dominio se producen cuando se realiza una llamada a procedimiento.

En UNIX el dominio de un proceso lo determina su *uid* y su *gid*. Dado un par de estos dos identificadores, es posible obtener un lista de todos los objetos que puede utilizar, y si pueden ser accedidos para lectura, escritura y/o ejecución. Además, en UNIX cada proceso tiene dos partes, la parte de usuario y la parte de núcleo. Cuando un proceso invoca una llamada al sistema pasa a ejecutarse en modo núcleo. Este cambio de modo de ejecución da lugar a un cambio de dominio. Cuando un proceso UNIX realiza una llamada *exec()* sobre un fichero que tiene uno de los bits *SETUID* o *SETGID* a uno adquiere un nuevo *uid* o *gid* efectivo. Esto también provoca un cambio de dominio.

El sistema operativo debe de conocer en todo momento los objetos que pertenecen a cada dominio. Una forma de llevar este control es tener una **matriz de acceso**, en la que las filas son los dominios y las columnas los objetos. Cada elemento de la matriz muestra los derechos de un dominio sobre un objeto.

Objeto / Dominio	File 1	File 2	File 3	File 4	File 5	File 6	Printer 1	Printer 2
D_1	R	R W						
D_2			R	R W X	R W		W	
D_3						R W E	W	W

Figura 7.- Ejemplo de matriz de accesos.

Los cambios de dominio se pueden incluir fácilmente en la matriz de protección considerando que cada dominio es a su vez un objeto sobre el que existe una operación que permite entrar en él.

Objeto / Dominio	File 1	File 2	File 3	File 4	File 5	File 6	Printer 1	Printer 2	Domain 1	Domain 2	Domain 3
D_1	R	R W								Enter	
D_2			R	R W X	R W		W				
D_3						R W E	W	W			

Figura 8.- Matriz de acceso con dominios.

En la práctica no es viable almacenar la matriz de protección porque es grande y tiene muchos huecos (*sparse matrix*). La mayoría de los dominios no tiene acceso a la mayoría de los objetos, por lo que almacenar una gran matriz casi vacía desperdicia mucho espacio en disco. Dos soluciones prácticas consisten en almacenar la matriz, bien por filas, bien por columnas, y almacenar únicamente los elementos no vacíos. En el primer caso se habla de capacidades, y en el segundo, de listas de control de acceso.

4.3.2. Listas de Control de Acceso

Cuando la matriz se almacena por columnas, a cada objeto se le asocia una lista ordenada que contiene todos los dominios que pueden acceder al objeto y la forma de hacerlo. Esta lista se denomina **lista de control de acceso** (*control access list* o ACL).

En UNIX esta lista está compuesta por tres grupos de tres bits, especificando cada grupo los permisos para el propietario, grupo y resto de usuarios, y los tres bits indican si existe permiso de lectura, escritura y/o ejecución. Sin embargo, un esquema basado en listas de control de acceso puede ser más general:

```
Fichero 0: (sod4, *, RWX)
Fichero 1: (antonio, profesores, RWX)
Fichero 2: (sod2,*,RW-), (*, practicas, R--), (root,system, RWX)
Fichero 3: (*, practicas, R--)
```

El propietario de un objeto puede modificar la lista de control de acceso del mismo. El único problema es que los cambios pueden no afectar a los usuarios que ya estén usando el objeto.

4.3.3. Capacidades

La otra forma de almacenar la matriz de protección es por filas. Con este método, cada proceso tiene una lista de los objetos a los que puede acceder así como una indicación de las operaciones que están permitidas sobre cada uno de ellos. Esta lista se denomina **lista de capacidades** (listas_C o *C_lists*), y cada elemento de la lista se llama capacidad.

Cada objeto está representado por una capacidad. Cuando un proceso quiere ejecutar una operación sobre un objeto, ha de especificar la capacidad del objeto como parámetro. La simple posesión de la capacidad permite que se conceda el acceso (siempre que éste esté permitido sobre el objeto).

Cada capacidad tiene un campo que indica el tipo del objeto, un campo de derechos de acceso, y un campo que identifica al objeto.

	Tipo	Permisos	Objeto
0	Fichero	R--	Puntero al fichero_3
1	Fichero	RWX	Puntero al fichero_4
2	Fichero	RW-	Puntero al fichero_5
3	Impresora	-W-	Puntero a la impresora_1

Figura 9.- Lista de capacidades para el dominio 2 de la Figura 7.

Las listas de capacidades se asocian a un dominio, pero los procesos que se ejecutan en ese dominio no deben de acceder a la lista de forma directa. Por el contrario, cada capacidad es un objeto protegido que debe ser gestionado por el sistema operativo y únicamente puede ser accedido de forma indirecta por parte de los procesos. Generalmente, suelen ser referenciadas por su posición dentro de la lista de capacidades. Las listas de capacidades son también objetos, por lo que pueden ser apuntadas desde otras listas, permitiendo el compartir subdominios.

Para proporcionar protección, las capacidades se han de distinguir de otro tipos de objetos. Esta

- Cada objeto tiene asociado una etiqueta que indica si se trata de un objeto o de una capacidad. Las etiquetas no deben ser accesibles directamente por los programas de aplicación. Una solución hardware consiste en añadir a cada palabra de memoria un bit adicional que indica si la palabra de memoria contiene una capacidad. Este bit sólo se puede ser modificado por el sistema operativo. Si se añaden más bits, el hardware podrá conocer los tipos de los objetos (si son enteros, reales,

- Dividir el espacio de direcciones de un programa en dos zonas. La primera es accesible al programa y contiene las instrucciones y los datos. La segunda contiene la lista de capacidades y es accesible únicamente por el sistema operativo.
- Mantener las capacidades en el espacio de usuario, pero encriptadas con una clave secreta desconocida para el usuario. Este método es empleado por el sistema operativo Amoeba.

acceso relativos a cada objeto, las capacidades tienen permisos genéricos que son aplicables a todos los objetos. Algunos de estos permisos son:

- **Copiar capacidad**: crear una nueva capacidad para el mismo objeto.
- **Copiar objeto**: Crear un objeto duplicado con una nueva capacidad.

- **Eliminar capacidad**: borrar de forma permanente una entrada de la lista_C, pero dejando inalterado al objeto.
- **Eliminar objeto**: eliminar de forma permanente un objeto y su capacidad.

Los sistemas basados en capacidades se suelen organizar como una colección de módulos, siendo cada módulo un gestor para cada tipo de objeto. Por ejemplo, las operaciones sobre ficheros se envían al gestor de ficheros, mientras que las que tiene que ver con memoria se envían al gestor de memoria. Cada operación va acompañada de la capacidad correspondiente.

Un problema que tiene este esquema basado en módulos gestores es que éstos son, a su vez, programas ordinarios. Pongámonos en el caso de un gestor de ficheros. Las capacidades asociadas a ficheros permiten a los programas las operaciones típicas sobre éstos (apertura, lectura, etc.). Sin embargo, el gestor de ficheros ha de ser capaz de acceder a la representación interna de los ficheros (en UNIX, por ejemplo, habría que acceder a los nodos índice). Por lo tanto, es esencial que los módulos gestores tengan más atribuciones de las que las capacidades permiten a los procesos ordinarios. Este problema se resuelve en el sistema Hydra mediante una técnica llamada amplificación de derechos, que consiste en que a los módulos gestores se les asigna una plantilla de permisos que les permiten obtener más derechos sobre los objetos de los que les permiten las capacidades.

Un problema que se presentan en los sistemas basados en capacidades es que es difícil revocar los accesos a un objeto, ya que ello supone localizar todas las capacidades que hacen referencia al objeto, y éstas se encuentran en las listas de capacidades, que se encuentran repartidas por todo el disco. Una solución a este problema consiste en que una capacidad no apunte al objeto en sí, sino a un objeto indirecto que apunta al objeto. De esta forma, si se quiere prohibir los accesos, basta con romper el enlace entre el objeto indirecto y el objeto. Otra posibilidad consiste en asociar a cada objeto un número aleatorio, que también está presente en la capacidad. Cuando se presenta una capacidad al solicitar un acceso, los dos números son comparados y el acceso se permite si la comparación ha tenido éxito. La forma de revocar los accesos consiste en que el propietario del objeto puede solicitar que el número aleatorio del objeto sea cambiado, lo que automáticamente invalida el resto de las capacidades.

4.3.4. Caso de Estudio: Windows NT

El sistema operativo Windows NT explota conceptos de orientación a objeto para proporcionar un sistema flexible y potente de control de accesos.

Windows NT tiene un esquema uniforme para el control de accesos, que se aplica tanto a los procesos como hebras, ficheros, semáforos, ventanas y otros objetos del sistema. El control de acceso se lleva a cabo a través de dos entidades: un token de acceso, que está asociado a cada proceso, y un descriptor de seguridad, que está asociado a todo objeto.

Cuando un usuario se conecta a un sistema Windows NT, usa un esquema nombre/contraseña para autentificar al usuario. Si la conexión es aceptada, se crea un proceso para ese usuario y se le asocia un token de acceso.

El token de acceso presenta la siguiente estructura:

- Identificador de seguridad (*Security ID* o SID): identifica a un usuario. Se corresponde, normalmente, con el nombre que el usuario introduce al conectarse.
- Identificadores de grupo (*Group SIDs*): lista de grupos a los que pertenece el usuario. Cada grupo tiene su propio SID de grupo. El acceso a un objeto se puede definir en función del SID de grupo,
- Privilegios: lista de servicios del sistema que el usuario puede solicitar. La mayoría de los usuarios no tienen privilegios.
- Propietario por defecto: normalmente, el propietario de un proceso es el del usuario lo crea.
- Lista de control de accesos por defecto: lista que indica las protecciones aplicadas inicialmente a los objetos que crea el proceso. El usuario puede, posteriormente, alterar la lista de control de acceso para cualquier objeto suyo o del grupo al que pertenezca.

El token de acceso tiene dos funciones:

a) Mantener toda la información de seguridad unida para permitir que la validación de los accesos se realice con rapidez. Cuando un proceso asociado con un usuario intenta realizar

un acceso, el subsistema de seguridad puede usar el token asociado con ese proceso para determinar los privilegios de acceso del usuario.

b) Permite que cada proceso pueda modificar su características de seguridad hasta cierto punto, sin que ello afecte a otros procesos del usuario.

Normalmente, el token de acceso tiene los privilegios denegados. Si un proceso necesita realizar una operación privilegiada debe solicitar el privilegio adecuado para realizar el acceso. Si toda la información de seguridad de un usuario se almacenase en un mismo lugar, permitir un privilegio para un proceso lo habilitaría para el resto de los procesos del usuario, lo que no es deseable.

Cada objeto que puede ser accedido por un proceso tiene asociado un descriptor de seguridad, que presenta la siguiente estructura:

1. Lista de control de acceso del sistema (*System Acces Control List* o SACL): Especifica las operaciones sobre el objeto que deberían de generar mensajes de auditoría. Las aplicaciones nodeben tener privilegio para leer o escribir la lista de control de acceso del sistema, ya que, de lo contrario, un proceso no autorizado podría leerla y aprender qué operaciones no debe realizar para evitar la generación de mensajes de auditoría, o podría escribirla, de forma que operaciones no
2. Propietario: El propietario del objeto puede realizar, generalmente, cualquier acción sobre el descriptor de seguridad. Puede ser un individuo o un grupo.
3. Lista de control de acceso discrecional (*Discretionary Access Control List* o DACL): Determina qué usuarios y qué grupos pueden acceder a este objeto y con qué operaciones. Consiste en una lista de entradas de control de acceso (*Access Control Entries* o ACE).
4. Indicadores: Define el tipo y contenidos del descriptor de seguridad.

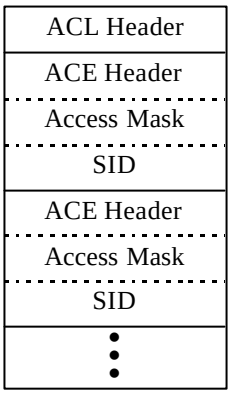

Figura 10.- *Lista de Control de Acceso en Windows NT.*

Cada lista de control de acceso consiste en una cabecera y en un número variable de ACEs (Figura 10). Cada entrada especifica un SID individual o de grupo y una máscara de acceso que define los derechos que son permitidos a este SID. Cuando un proceso intenta acceder a un objeto, el gestor de objetos de Windows NT lee los SID individual y de grupo del token de acceso y realiza una búsqueda en la DACL del objeto. Si se encuentra una ACE cuyo SID coincida con uno de los del token de acceso, entonces el proceso tiene los derechos de acceso especificados en la máscara de acceso de la ACE.

La máscara de acceso tiene 32 bits (Figura 11). Los 16 bits menos significativos especifican derechos de acceso que se aplican a un tipo particular de objeto.

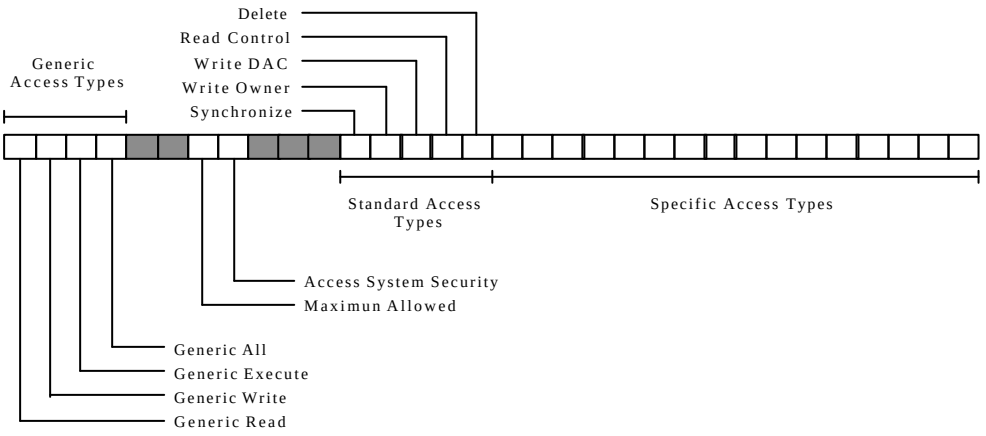

Figura 11.- *Máscara de Acceso en Windows NT.*

Los 16 bits más significativos contienen bits que se aplican a todos los tipos de objetos. Cinco de

1. *Synchronize*: Da permiso para sincronizar la ejecución con algún suceso asociado al objeto.
2. *Write_owner*: Permite modificar el propietario del objeto.
3. *Write_DAC*: Permite modificar la DACL o, lo que es lo mismo, la protección del objeto.
4. *Read_control*: Permite preguntar por el propietario y los campos de la DACL del descriptor de seguridad del fichero.
5. *Delete*: Permite borrar el objeto.

Los cuatro bits más significativos se conocen como tipos de acceso genéricos, y constituyen el mecanismo para establecer tipos de acceso específicos sobre varios tipos de objeto:

1. *Generic_all*: Permite todos los accesos.
2. *Generic_execute*: Permite la ejecución si es ejecutable.
3. *Generic_write*: Permite el acceso de escritura.
4. *Generic_read*: Permite el acceso de sólo lectura.

El resto son bits especiales:

1. *Access_System_Security*: Permite modificar los controles sobre el objeto.
2. *Maximum_Allowed*: Modifica el algoritmo de búsqueda de un SID sobre la DACL.

Una característica importante del esquema de seguridad de Windows NT es que las aplicaciones pueden hacer uso del mismo para objetos definidos por el usuario.

4.4. Referencias

- “Modern Operating Systems”. A.S. Tanenbaum. Prentice-Hall. 1992.
- "Operating Systems Concepts". Silberschatz, Peterson. Addison-Wesley, 1994.
- “Operating Systems, Second Edition”. Stallings, W. Prentice-Hall. 1995.