UNIVERSIDADE DA CORUÑA

Facultade de Informática

Seguridad en Sistemas Operativos

Seguridad en Sistemas de Información

Gracia Fernández López
Fecha: *6 de mayo de 2007*

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Definición de Sistema Operativo

Todos sabemos, aunque sea de manera intuitiva, lo que es un sistema operativo. Se podría entender como la capa intermedia que nos permite interactuar con el ordenador y sobre la que se ejecutan las aplicaciones.

1.2. Funciones Basicas de un Sistema Operativo

Podemos englobar las funcionalidades de un Sistema Operativo en los siguientes grupos básicos:

- **Gestion del Procesador y la Memoria:** el SO tiene que asegurar que cada proceso obtiene una parte del tiempo del procesador, y que el procesador es usado eficientemente. Además define los métodos por los cuales el operativo asigna la memoria a los procesos.
- **Gestion de los Sistemas de Almacenamiento:** el SO define como son almacenados los datos de una manera fiable.
- **Gestion de la Entrada/Salida:** el SO debe ser capaz de gestionar como interactúan los componentes hardware, e interactuar con las aplicaciones y el usuario.
- **Gestion de la Red:** Aunque puede considerarse parte de la entrada y salida, su importancia le hace tomar entidad propia. El Sistema operativo debe permitir la comunicación en red de las distintas aplicaciones.

1.3. Principios fundamentales de la seguridad informática

Como hemos visto en la introducción de la asignatura, la seguridad informática se puede dividir en varios principios a cumplir:

Integridad Requiere que no se la información se altere de forma no autorizada.

Confidencialidad Requiere que la información sea accesible únicamente por las entidades autorizadas.

Control de Acceso Requiere que la información sólo pueda ser accedida por las entidades autorizadas.

Autenticidad Capacidad de verificar la autenticidad de quién accede a los recursos y de los recursos en sí mismos.

No repudio Capacidad de probar la participación de las partes en el acceso a los recursos.

Disponibilidad Requiere que los recursos del sistema están disponibles para las entidades autorizadas cuando los necesiten.

Ahora que hemos visto las funcionalidades básicas de los sistemas operativos y los principios de la seguridad, veremos cómo los sistemas operativos cumplen dichos principios en cada una de sus funcionalidades.

Capítulo 2

Gestión del Procesador y la Memoria

2.1. Protección de Memoria

En un sistema multiproceso podemos distinguir dos tipos de protecciones de memoria:

- La protección de los procesos o tareas entre ellos.
- La protección del propio sistema operativo (kernel) de los procesos.

2.1.1. Protección entre Procesos

Los procesos manejan direcciones virtuales, las cuales son mapeadas mediante una controladora hardware (MMU - memory management unit) a direcciones físicas de la memoria. Cada proceso tiene un espacio de direcciones virtuales diferente, y, cuando se realiza un cambio de contexto, se cambia la tabla de páginas que mapea las direcciones virtuales a direcciones físicas. Esto, a nivel de seguridad, da lugar a que un proceso no pueda acceder a la zona de memoria de otro, ya que cada uno va a tener un espacio de direcciones virtuales diferentes.

2.1.2. Protección del kernel

El kernel no se puede considerar un proceso aislado con su propio espacio de memoria, ya que se debe permitir la ejecución de determinadas partes del kernel por parte de cada proceso, manteniendo siempre el espacio de memoria del proceso. Una solución sería que el proceso ejecutara directamente el

núcleo, pero dada su naturaleza y que tiene acceso a instrucciones privilegiadas, debe protegerse su espacio de memoria y establecer unos puntos de entrada determinados.

Para realizar esta protección se debe recurrir a un servicio hardware, que son los niveles de privilegio de ejecución. Habitualmente existen dos niveles de ejecución: Nivel de kernel y nivel de usuario. El nivel de kernel tiene ciertos privilegios, como la ejecución de instrucciones privilegiadas, acceso a todos los registros, acceso a la tabla de traducciones, etc.
Cuando un proceso intenta acceder a una nueva porción de memoria (segmento o página) se comprueba su nivel de privilegio. Si el nivel de esta zona de memoria es suficiente (usuario) se permite la ejecución, en otro caso (es una porción del kernel) se lanza una excepción.
Las direcciones donde se encuentra el kernel poseen un nivel de ejecución de kernel, de forma que sólo se podrá ejecutar este código con el nivel de kernel. Sólo el propio kernel puede ejecutar el segmento del kernel.

Normalmente parte del kernel se encuentra aislado con la incorporación de una tabla de traducción de direcciones específica para el kernel. Esta tabla no se alterara en los cambios de contexto y es común para todos los procesos, activándose en el momento en que el control del sistema pasa al kernel.

Para realizar el cambio al modo kernel, existen unos puntos de entrada definidos, a los que se accede mediante una instrucción de llamada al sistema.

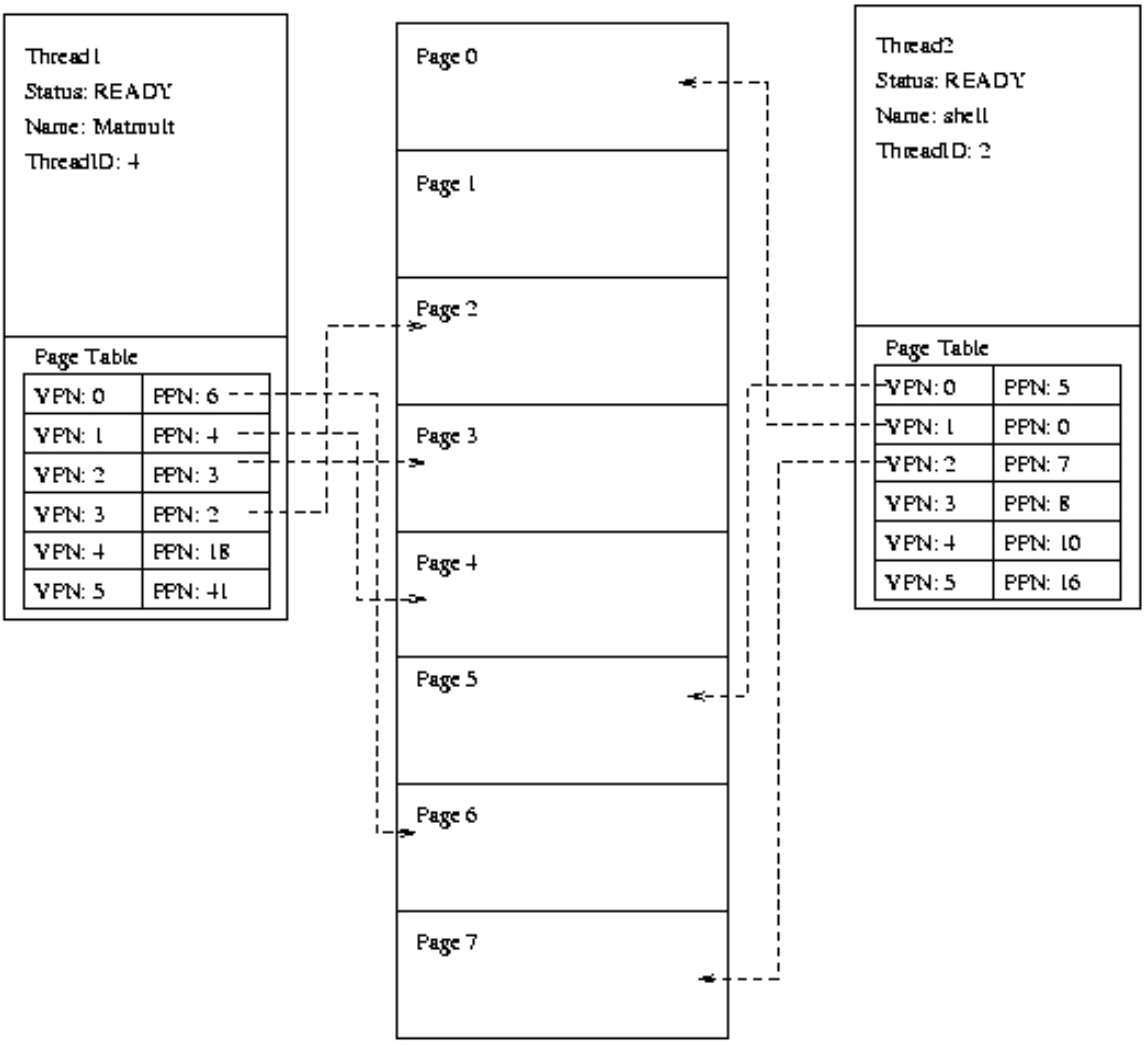

Figura 2.1: Mapeado de direcciones de memoria

2.2. Buffer Overflow

2.2.1. Estructura de un proceso

Antes de ver Buffer Overflow tenemos que saber que un proceso tiene una estructura formada por cuatro partes diferenciadas:

- *Texto:* este segmento contiene principalmente el código de ejecución, es decir, una serie de instrucciones para la ejecución del programa.
- *Datos:* zona de memoria que contiene las variables globales. Su tamaño es determinado en tiempo de compilación.
- *Heap:* contiene variables dinámicas (tamaño asignado con malloc).
- *Stack:* buffer con estructura LIFO. Está compuesta por frames, creadas por cada llamada a función dentro del código. Cada frame tiene a su vez estructura, y en ella se guardan las variables locales y argumentos de la función, la dirección de retorno y la referencia al frame anterior.

La colocación de todos estos componentes en memoria varía dependiendo del sistema operativo concreto, pero los elementos básicos del proceso y los componentes principales del stack descritos están presentes en todos ellos, independientemente de su organización en la memoria.

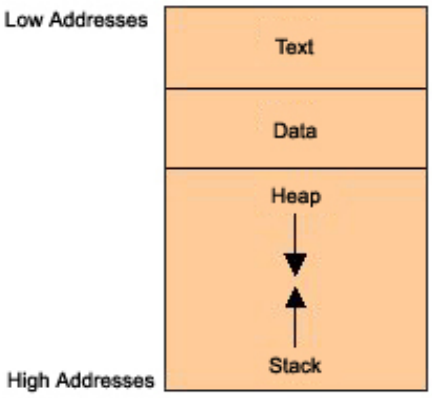

Figura 2.2: Estructura de Proceso

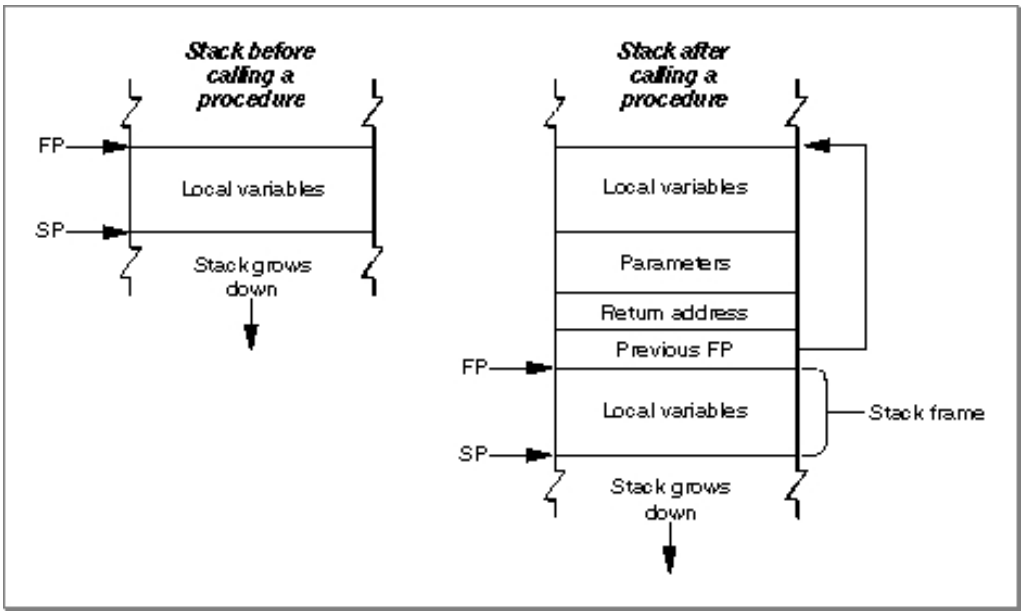

Figura 2.3: Ejemplo de estructura del stack (680x0 stack frame)

2.2.2. Definición de Buffer Overflow

Buffer overflow o *buffer overrun* es probablemente la vulnerabilidad de seguridad software más famosa. Es una condición anómala en la cual un proceso permite asignar datos de un tamaño mayor al reservado. El resultado es que esos datos extra sobreescriben las zonas de memoria contiguas. Pueden consistir en entradas especialmente diseñadas para ejecutar código malicioso. Buffer overflow es la base de muchos exploits.
Las partes del proceso en las que se puede salir de los márgenes de las variables son el stack y el heap, por lo que la vulnerabilidad de buffer overflow consiste, básicamente, en que el código reciba los datos de entrada en una variable local.

2.2.3. Buffer Overflow sobre el Stack

El buffer overflow sobre el stack de proceso o *Stack Overflow* consiste en que un exploit machaca el buffer de entrada con un shellcode y modificar la variable de retorno de una función del programa para que apunte a dicho shellcode. Ésto se puede hacer debido a que la dirección de retorno de una función es uno de los campos de su frame en el stack, por lo que sólo hay que sobreescribirla para que apunte al propio stack en la zona donde está el shellcode.

El exploit está compuesto por una ristra de NOPs, que permiten sólo tener que calcular la dirección donde estará el código del shellcode en la pila de una forma aproximada, proporcionándonos un margen de error, el código del shellcode, y la dirección de retorno calculada a partir de la dirección base del stack también repetida N veces, de nuevo para permitirnos un margen de error.

Stack overflow es una técnica de hacking bien conocida y existen muchos métodos para evitarla.

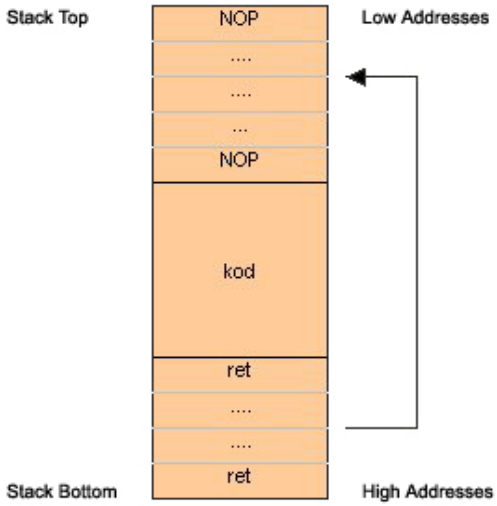

Figura 2.4: Estructura de un exploit en Stack Overflow

2.2.4. Buffer Overflow sobre el Heap

El buffer overflow ocurrido en el heap recibe el nombre de *Heap Overflow*. La memoria en el heap es asignada dinámicamente por la aplicación en tiempo de ejecución. El objetivo del ataque consiste en sobreescribir alguna variable importante almacenada en el heap.

Este tipo de ataque es más complejo que el del stack porque depende de la implementación concreta del sistema de asignación de memoria (malloc). Pero al mismo tiempo es más peligroso, ya que es menos conocido que el ataque sobre el stack y es más difícil de detectar y evitar.

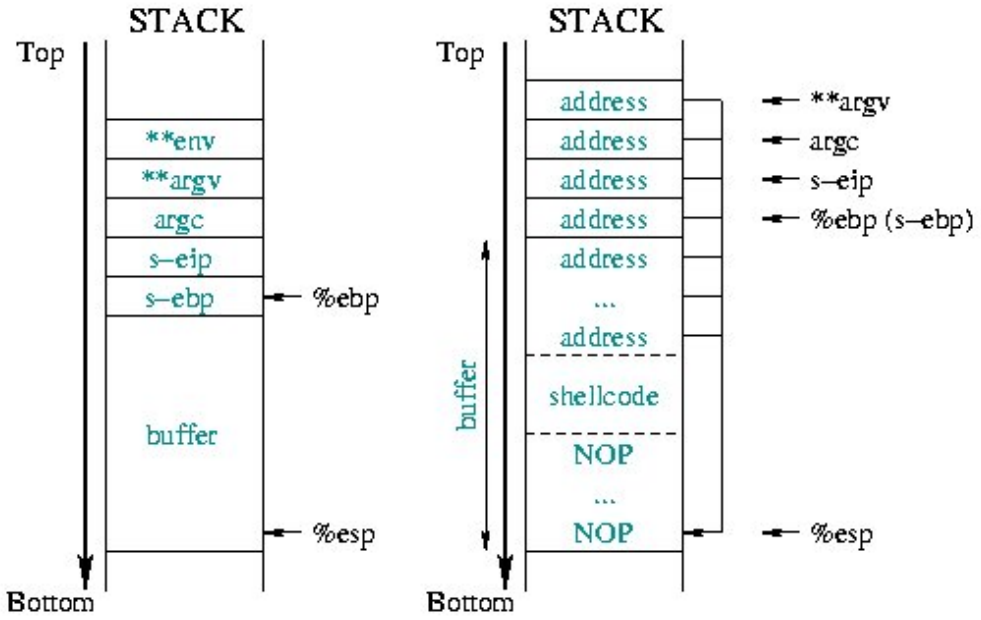

Figura 2.5: Proceso de Stack Overflow - Antes y después de introducir el exploit

2.2.5. Protección contra Buffer Overflow

Existen una serie de técnicas para evitar este tipo de ataques:

- *Protección Stack-smashing:* Básicamente, consiste en modificar la estructura del frame del stack añadiendo una marca ("canario") que, cuando es modificada, permite saber que se ha producido desbordamiento. Esta técnica no es propia del sistema operativo, sino que es una protección del compilador. Implementaciones:
  - StackGuard: incluído a partir de gcc 4.1 en las distribuciones Linux.
  - Propolice: Parche estándar de gcc 3.x para OpenBSD, Trusted Debian, DragonFly BSD, Hardened Linuxfromscratch e IPCop Linux. Está también disponible en Debian y Gentoo, pero por defecto está desactivado.
- *Protección del espacio ejecutable:* consiste en marcar las regiones de memoria donde residen el stack y el heap como no ejecutables. Si se intenta ejecutar código en esas regiones lanzará una excepción. Esta técnica a menudo hace uso del bit hardware NX ("No eXecute") o

XD ("eXecute Disabled"). FreeBSD, Mac OS X, Linux kernel 2.6.8+, Windows XP SP2+, Windows Server SP1+, entre otros, ofrecen soporte para ese bit en Intel. Además, OpenBSD ofrece W^X, que permite esta protección en CPUs sin bit NX.

- *Address space layout randomization (ASLR):* consiste en colocar las partes de un proceso de forma aleatoria en el espacio de direcciones. Ésto hace que el stack no comienze siempre en la misma dirección de memoria, lo que, como vimos en la definición de buffer overflow, era fundamental para calcular la dirección de retorno del shellcode. Está implementado en Linux kernel 2.6+, Windows Vista y OpenBSD.

- *Deep packet inspection (DPI):* puede detectar, en el perímetro de la red, intentos remotos de ataques por buffer overflow usando firmas de ataques y heurísticas. Es capaz de bloquear paquetes que tengan la firma de un ataque conocido o cuando detectan series largas de NOPs. Este método no es muy efectivo ya que sólo previene de ataque conocidos y, además, existen muchas formas de codificar las NOPs. Los hackers han empezado a usar shellcodes alfanuméricos, metamórficos y automodificables para evadir su detección por parte de los sistemas heurísticos. Esta técnica tampoco es implementada por los sistemas operativos, la realizan los sistemas de detección de intrusos.

Capítulo 3

Sistemas de Almacenamiento

3.1. RAID

3.1.1. Definición

El acrónimo RAID (*Redundant Array of Inexpensive Disks*) hace referencia a un sistema de almacenamiento que usa múltiples discos duros entre los que distribuye o replica los datos. Está formado por dos o más discos y una controladora, la cual gestiona la repartición de datos entre el mencionado conjunto de discos.
En el nivel más simple, RAID combina múltiples discos en una sola unidad lógica, de manera que para el sistema operativo es como si sólo hubiera un disco.

3.1.2. Ventajas

- Mayor integridad: Las soluciones RAID emplean dos técnicas para aumentar la fiabilidad: la redundancia de datos y la información de paridad.
- Tolerancia a fallos: RAID protege contra la pérdida de datos y proporciona recuperación de datos en tiempo real.
- Mayor throughput (rendimiento): RAID permite a varias unidades trabajar en paralelo, lo que aumenta el rendimiento del sistema.
- Alta Disponibilidad: RAID aumenta el tiempo de funcionamiento y la disponibilidad de la red.

3.1.3. Niveles RAID

RAID 0 (disk striping) la información se distribuye en bloques entre los diferentes discos. Es el único nivel de RAID que no duplica la información, por lo tanto no se desperdicia capacidad de almacenamiento pero no ofrece tolerancia a fallos. Se utiliza normalmente para incrementar el rendimiento. Se requieren mínimo dos discos. Útil para configuraciones en las que montar muchos discos es un proceso muy costoso en tiempo (por ejemplo servidores NFS de solo lectura) o cuando esté limitado el número de discos (por ejemplo, hasta Windows 2000 estaba limitado a 24 el número de unidades lógicas (letras)).

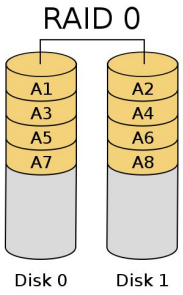

Figura 3.1: RAID 0

RAID 1 (mirroring o duplicación) consiste en asociar a cada disco primario del RAID un segundo disco *espejo*, en el que se duplica la información. Presenta tolerancia a fallos, ya que si el disco primario falla el espejo continúa trabajando. Al igual que RAID 0, el rendimiento en la lectura es mayor, ya que al estar los datos distribuídos en los discos, se pueden hacer varios accesos simultáneos. En escritura se pierden prestaciones, al tener que escribir la misma información simultáneamente en dos discos. Por ello, en ocasiones, se utiliza la duplicación de controladoras del disco además de la duplicación de los discos. Este sistema resulta caro, ya que requiere instalar en el RAID el doble de la capacidad requerida. Por tanto, sólo se utiliza en sistemas que requieran una alta disponibilidad y tolerancia a fallos.

RAID 2 Acceso paralelo con discos especializados. Redundancia a través del código Hamming. Divide los datos a nivel de bits en lugar de a nivel de bloques y usa un código de Hamming para la corrección de errores. Los discos son sincronizados por la controladora para funcionar al unísono. Teóricamente, un RAID 2 necesitaría 39 discos en un

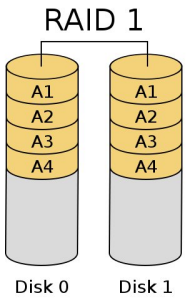

Figura 3.2: RAID 1

sistema informático moderno: 32 se usarían para almacenar los bits individuales que forman cada palabra y 7 se usarían para la corrección de errores. Actualmente no se usa.

RAID 3 (parallel data access) Un RAID 3 usa división a nivel de bytes con un disco de paridad dedicado. Aplicando un determinado algoritmo se genera el byte de paridad, que se escribe en el disco de paridad. La recuperación de datos se consigue calculando el OR exclusivo (XOR) de la información registrada en los otros discos. RAID 3 ofrece altas tasas de transferencia, alta fiabilidad y alta disponibilidad, a un coste muy inferior que un Mirroring (RAID 1). Sin embargo, su rendimiento de transacción es pobre porque sólo puede atender una petición de entrada salida a la vez (se accede a todos los discos al mismo tiempo). Por tanto, RAID 3 es adecuado para sistemas de un sólo usuario con aplicaciones que contengan grandes registros (por ejemplo, un servidor de base de datos mono-usuario).

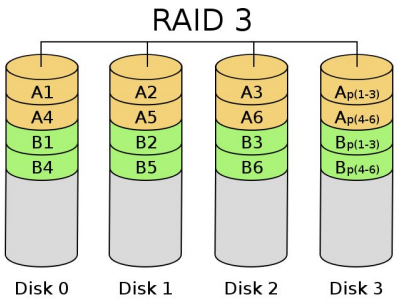

Figura 3.3: RAID 3

RAID 4 Un RAID 4 usa división a nivel de bloques con un disco de paridad dedicado. El RAID 4 es parecido al RAID 3 excepto en que divide a nivel de bloques en lugar de a nivel de bytes. Esto permite que cada miembro del conjunto funcione independientemente cuando se solicita un único bloque. Por tanto, un conjunto RAID 4 puede servir varias peticiones de lectura simultáneamente. En principio también sería posible servir varias peticiones de escritura simultáneamente, pero al estar toda la información de paridad en un solo disco, éste se convertiría en el cuello de botella del sistema. Debido a su organización interna, este RAID está especialmente indicado para el almacenamiento de ficheros de gran tamaño, lo cual lo hace ideal para aplicaciones gráficas donde se requiera, además, fiabilidad de los datos

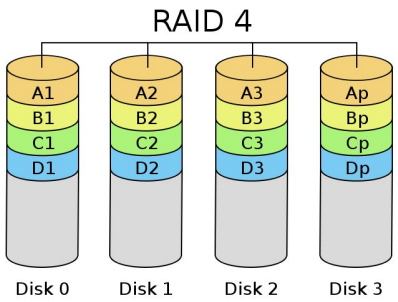

Figura 3.4: RAID 4

RAID 5 (independent data access) Similar al RAID 4 con la diferencia de que la información de paridad se reparte en los discos de forma rotatoria, eliminando las limitaciones de escritura de RAID 4. Así, todas las operaciones de lectura y escritura pueden superponerse. RAID 5 es el nivel RAID con tolerancia a fallos que ofrece la mejor relación rendimiento-coste. Este nivel RAID es recomendable para aplicaciones que trabajen con ficheros pequeños pero con un gran número de transacciones E/S, como es el caso de las bases de datos relacionales o las aplicaciones de gestión.

RAID 6 (independent data access) Similar al RAID 5, pero incluye un segundo esquema de paridad distribuído por los distintos discos y por tanto ofrece tolerancia extremadamente alta a los fallos y a las caídas de disco, ofreciendo dos niveles de redundancia. Hay pocos ejemplos comerciales en la actualidad, ya que su coste de implementación es mayor al de otros niveles RAID, debido a que las controladoras que

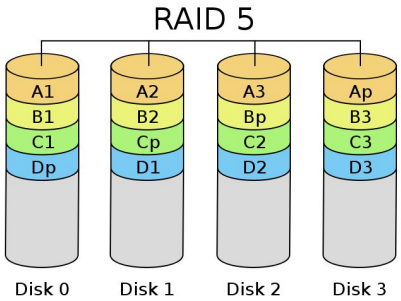

Figura 3.5: RAID 5

soportan esta doble paridad son más complejas y caras que las de otros niveles RAID. Así pues, comercialmente no se implementa.

Además de los niveles de RAID básicos vistos, existen otros como:

- RAID 5E y 6E: variantes de RAID 5 y 6 que incluyen discos de reserva, que pueden estar conectados y preparados (*hot spare*) o en espera (*standby spare*).
- RAID 7: incluye un sistema operativo incrustado de tiempo real como controlador, haciendo las operaciones de caché a través de un bus de alta velocidad y otras características de un ordenador sencillo.
- RAID-Z: Es un nuevo sistema RAID desarrollado por Solaris similar al RAID-5, pero que soluciona el problema de "write hole"que presentan los RAID con paridad. Write hole consiste en que si se produce una caída del sistema mientras hay escrituras activas, la paridad y los datos pueden quedar inconsistentes. RAID-Z no presenta esta vulnerabilidad debido al hecho de que en él, a diferencia de los demás RAIDs, el tamaño de bloque es variable. Dicho tamaño es determinado por el tamaño de los datos a escribir (cada escritura a disco supone una escritura de un bloque completo). Por encima del RAID-Z se utiliza el sistema de ficheros ZFS, con características especiales para este tipo de RAIDs. RAID-Z tiene claras ventajas sobre los sistemas tradicionales, y está siendo incorporado en la mayoría de los sistemas operativos.
- Niveles de RAID anidados:
  - RAID 0+1: Un espejo de divisiones.
  - RAID 10: Una división de espejos.
  - RAID 30: Una división de niveles RAID con paridad dedicada.
  - RAID 100: Una división de una división de espejos.

3.1.4. Implementaciones

La distribución de datos en varios discos puede ser gestionada por hardware dedicado o por software. Además, existen sistemas RAID híbridos basados en software y hardware específico.
Una implementación de RAID basada en hardware requiere al menos una controladora RAID específica, ya sea como una tarjeta de expansión independiente o integrada en la placa base, que gestione la administración de los discos y efectúe los cálculos de paridad. Esta opción suele ofrecer un mejor rendimiento y hace que el soporte por parte del sistema operativo sea más sencillo (de hecho, puede ser totalmente transparente para éste). Las implementaciones basadas en hardware suelen soportar sustitución en caliente (*hot swapping*), permitiendo que los discos que fallen puedan reemplazarse sin necesidad de detener el sistema.
Con la implementación por software, el sistema operativo gestiona los discos del conjunto de discos. Esta solución supone un coste menor pero también un menor rendimiento, ya que todos los cálculos se realizan en la CPU, equiparando una parte considerable del tiempo del procesador.

3.1.5. Soluciones proporcionadas por los Sistemas Operativos

Aquí veremos algunos ejemplos de implementaciones de RAID proporcionadas por los sistemas operativos:

MS Windows Podemos crear un sistema RAID por software a través del uso de *discos dinámicos* en Windows 2000 o XP (Pro/Server). Con los discos dinámicos, podemos crear sistemas RAID0, RAID1 y RAID5.

Linux El device driver md (multiple device) crea dispositivos virtuales a partir de uno o más dispositivos físicos. Actualmente, Linux soporta RAID0, RAID1, RAID4, RAID5, RAID6 y RAID10. En Linux la composición de RAID se hace a nivel de partición y los dispositivos /dev/mdx que representan dispositivos raid. En /proc/mdstat se lista todos los dispositivos md activos e información acerca de éstos. Se usa el comando mdadm para manejar dichos dispositivos MD.

3.2. Sistemas de ficheros con journaling

Para mejorar el rendimiento de las operaciones de E/S, los datos del disco son temporalmente almacenados en la memoria RAM a través del page-cache y buffer-cache. Los problemas surgen si hay un corte de suministro eléctrico antes que los datos modificados en la memoria (dirty buffers) sean grabados a disco. Se generaría una inconsistencia en el estado global del sistema de ficheros.

3.2.1. Definición

Un **sistema con journaling** es un sistema de ficheros tolerante a fallos en el cual la integridad de los datos está asegurada porque las modificaciones de la meta-información de los ficheros son primero grabadas en un registro cronológico (log o journal) antes que los bloques originales sean modificados. En el caso de un fallo del sistema, un sistema con journaling asegura que la consistencia del sistema de ficheros es recuperada. El método más común es el de grabar previamente cualquier modificación de la meta-información en un área especial del disco, el sistema realmente grabará los datos una vez que la actualización de los registros haya sido completada. A la hora de recuperar la consistencia después de un fallo, el módulo de recuperación analizará el registro y sólo repetirá las operaciones incompletas en aquellos ficheros inconsistentes, es decir, que la operación registrada no se haya llevado a cabo finalmente.

3.2.2. Implementaciones

- Linux: Ext3 (extensión de Ext2 con un módulo de transacciones y otro de registro), ReiserFS, XFS y JFS.
- Windows: NTFS (New Technology File System) desde Windows 2000.
- SUN Solaris: UFS Logging (desde Solaris 7).
- Mac OS X: HFS+.

3.3. Backup

Hacer una copia de seguridad o copia de respaldo se refiere hacer copias adicionales de la información para poder restaurar el sistema a un estado operacional anterior (copias de seguridad del sistema) o para recuperar datos que se han perdido o que están corruptos (backup de datos).

3.3.1. Mecanismos de backup incluídos por los SO

Algunos de los mecanismos de backup proporcionados por los sistemas operativos son:

- Instantáneas (*snapshots*) de sistemas de ficheros: Las instantáneas permiten a un usuario crear imágenes de uno o más sistemas de ficheros dados, y tratarlas como un fichero. Se marcan todos los bloques de la instantánea, de manera que cuando se vaya a modificar o borrar alguno de ellos, se realiza una copia del bloque, manteniendo la instantánea inalterable. Esta funcionalidad es muy adecuada para copias de seguridad del sistema, pudiendo volver a un estado anterior. Muchos operativos ofrecen esta funcionalidad (FreeBSD 5.0+, Windows XP+, Linux, ...)
- Windows XP: Ntbackup. La herramienta de backup de Windows permite, de una forma fácil y eficaz, realizar y restaurar copias de seguridad. Las copias de seguridad se pueden realizar de forma puntual o programarlas para que se realizen en un intervalo de tiempo determinado. Permite hacer un backup normal, copia, diferencial, incremental o diario (ver tabla).

Tipo	**Archivos**	**Marca copiados**
Normal	Todos	Sí
Copia	Todos	No
Diferencial	Los archivos que no han sido marcados anteriorment	No
Incremental	Los archivos que no han sido marcados o han sido modificados	Sí
Diaria	Los archivos que han cambiado ese día	No

Cuadro 3.1: Modos de backup con Ntbackup

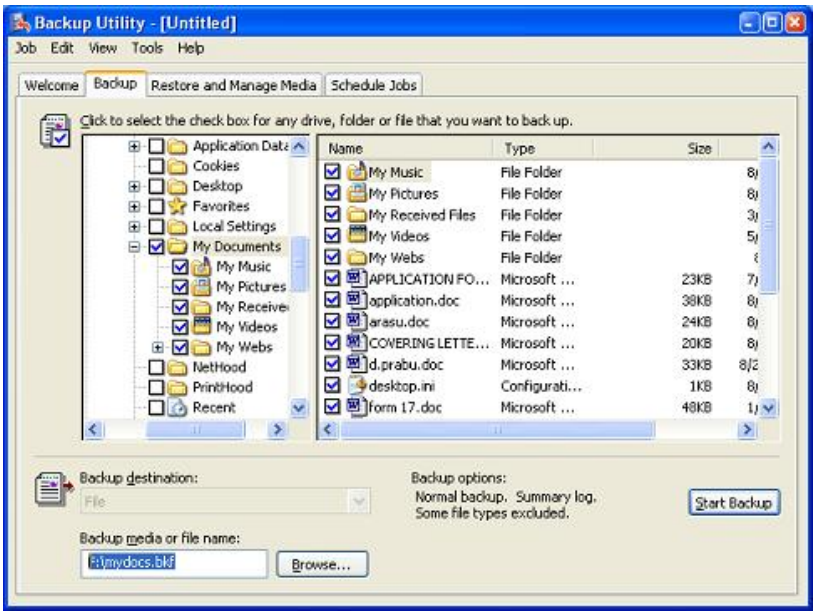

Figura 3.6: Ntbackup de Windows XP

- Unix: Hay una serie de comandos que te facilitan la realización de un backup:

  dd comando de copia de datos a muy bajo nivel. Se suele utilizar para hacer copias exactas de discos. Por ejemplo, para hacer backup del disco duro /dev/hda en otro disco /dev/hdb sería:

  ```
  $ dd if=/dev/hda of=/dev/hdb
  ```

  dump/rdump/ufsdump realiza el backup del sistema de ficheros. Por ejemplo, el comando dump para hacer backup del sistema de ficheros la unidad de cinta /dev/nst0 sería:

  ```
  $ dump -0ua -f /dev/nst0 /
  ```

  restore recupera las copias de seguridad realizadas con dump. Por ejemplo, podemos iniciar la consola de recuperación para recuperar el backup del anterior ejemplo con el comando:

  ```
  $ restore -i -f /dev/nst0
  ```

3.4. Control de acceso

El control de acceso define a qué objetos puede acceder cada sujeto. Por *objeto* entendemos cualquier entidad que contiene información, y puede ser físico o abstracto. Los *sujetos* acceden a los objetos, y pueden ser usuarios, procesos, programas u otras entidades.

3.4.1. Matriz de Control de Acceso

La Matriz de Control de Acceso o Matriz de acceso es un modelo abstracto formal de seguridad que caracteriza los permisos de cada sujeto con respecto a todos los objetos en el sistema. Las filas de la matriz de acceso representan dominios y las columnas objetos. La entrada *acceso(i,j)* define el conjunto de operaciones que un sujeto en el dominio D_i puede invocar con el objeto O_j. Hay que tener en cuenta que la Matriz de Control de Acceso es sólo un modelo teórico de permisos. Una implementación literal de este array bidimensional tendría excesivos requerimientos de memoria.

objeto / dominio	F_1	F_2	F_3
D_1	ejecutar	-	escribir
D_2	ejecutar	leer	ejecutar
D_3	ejecutar	-	-

Cuadro 3.2: Ejemplo de matriz de acceso

3.4.2. Políticas de Control de Acceso

- *Control de Acceso Discrecional (Discretionary Access Control DAC):* es una política determinada por el propietario de un objeto. El propietario es el que decide quién puede acceder al objeto y qué privilegios tiene.
- *Control de Acceso Obligatorio (Mandatory Access Control MAC):* ésta es una política de acceso determinada por el sistema, no por el propietario.
- *Control de Acceso Basado en Roles (Role-Based Access Control RBAC):* esta política permite que los privilegios sean asignados a roles arbitrarios. Estos roles se pueden asignar después a usuarios reales.

3.4.3. Control de Acceso Discrecional (DAC)

Es un tipo de control de acceso definido por el TCSEC (Trusted Computer System Evaluation Criteria, estándar del DoD) que consiste en que el creador de los objetos es el que determina el acceso a dichos objetos.
Se basa en dos conceptos fundamentales:

- Propietario: todo objeto en el sistema tiene un propietario. En la mayoría de los sistemas DAC, el propietario inicial es el que ha creado o causado el objeto.
- Derechos de acceso y permisos: un propietario puede asignar a otros sujetos recursos.

Se implementa mediante:

- **Bits de permisos:** Se especifica a un usuario como el propietario de un archivo y cada archivo o directorio se afilia a un grupo. Se otorga permiso de sólo lectura, escritura o ejecución a un grupo. En un momento dado un usuario pertenece a un grupo y adquiere los derechos de ese grupo.
- **Sistema de Contraseñas:** el propietario asigna a cada archivo una contraseña.
- **Lista de Capacidades:** cada objeto tiene un solo propietario, el cual otorga o cancela privilegios a los demás sujetos sobre dicho objeto. Cada usuario tiene una lista de capacidades que contiene los nombres de los objetos a los que tiene acceso y sus permisos correspondientes. Dicha lista es mantenida por el sistema operativo, y los usuarios no pueden acceder a ella directamente.
- **Lista de Control de Acceso (ACL):** los archivos y directorios tienen conjuntos de permisos configurados para el propietario del archivo, el grupo asociado con el archivo y todos los otros usuarios del sistema. Sin embargo, estos permisos tienen sus limitaciones. Por ejemplo, no se pueden configurar diferentes permisos para usuarios diferentes. Una lista de control de acceso es un conjunto de entradas de usuario, grupo y modo asociado a un archivo que especifica los permisos de acceso para todas combinaciones posibles de identificación de usuario o identificación de grupo.

Permisos

- En Unix la seguridad de los diferentes objetos del sistema vienen determinado por usuario y el grupo. Internamente, estas entidades el sistema las representa a través de dos números (credenciales), el uid - user id, para el usuario - y el gid - group id, para el grupo -, que representa a un conjunto de usuarios. Cada usuario pertenece a un grupo primario y puede pertenecer a varios grupos secundarios.
  Los permisos que pueden existir sobre los objectos del sistema de ficheros son:
  - Permiso de lectura (r).
  - Permiso de ecritura (w).
  - Permiso de ejecucion (x).
  - Permiso setuid: Un fichero con este permiso puesto, cuando lo ejecuta cualquier usuario, en vez de ejecutarse con dicho permiso, lo hace con el del propietario del fichero.
  - Permiso setgid: Igual que setuid, pero para los grupos.
- Windows Vista proporciona los siguientes tipos de permisos:
  - Control total: otorga todos los permisos disponibles.
  - Modificar: otorga permisos para modificar archivos o carpetas.
  - Lectura y ejecución.
  - Lectura.
  - Escritura.
  - Permisos especiales: dentro de los permisos especiales encontramos los permisos necesarios para sincronizar un archivo, cambiar permisos y cambiar el propietario de un archivo.

  Los grupos más comunes en Windows Vista son:
  - Administradores.
  - Usuarios.
  - Invitados.
  - Todos: grupo que reúne todas las cuentas asociadas al sistema.

  Existen además otros grupos que corresponden a procesos internos u aplicaciones de Windows Vista o de terceros como los Usuarios del

Administrador del sistema o S_IUSERS que corresponden a los usuarios del servidor Web si está instalado.
Entre los usuarios habituales de Windows Vista se encuentran:

- Administrador.
- Invitado: cuenta que se crea por defecto con permisos muy limitados
- Usuarios: su nivel de privilegios variará en función del tipo de cuenta que les hayamos asociado al crearlos.

Listas de Control de Acceso - ACLs

En este sistema de permisos los ficheros no tienen un juego fijo de permisos (como en el modelo tradicional de Unix, que consta sólo de 3 permisos), 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.

- Linux (ext2/ext3/JFS/FS/ReiserFS): con estos sistemas de ficheros podemos crear ACLs en Linux una vez añadido su soporte al kernel. Proporciona los siguientes comandos básicos:

  getfacl nos permite consultar las ACLs de un fichero dado.

  setfacl nos permite modificar las ACLs de un fichero dado.

  Ejemplo de una ACL en Linux:

  ```
  user::rw-
  user:lisa:rw-          #effective:r--
  group::r--\\
  group:toolies:rw-      #effective:r--
  mask::r--\\
  other::r--\\
  ```

- Windows (NTFS): Windows (a partir de Windows NT) utiliza descriptores de seguridad (security descriptors SDs) para llevar a cabo el control de acceso. Un descriptor de seguridad es una estructura que contiene toda la información sobre el control de acceso de un objeto específico. Está compuesto por:

  - El propietario del objeto.

- El grupo primario del objeto (raramente usado).
- Una lista de control de acceso discrecional (Discretionary access control list DACL): lista de control de acceso controlada por el propietario del objeto, en la que especifica que permisos otorga sobre dicho objeto.
- Una lista de control de acceso del sistema (System access control list SACL): lista de control de acceso en la que se especifica qué intentos de acceder al objeto son auditados en el log de eventos de seguridad.
- Información de control.

Ejemplo de lista de control de acceso en Windows:

```
e:\              administrators and systems:     full control
e:\users         help desk:                      change
e:\users\user1   user1:                          change
e:\users\user2   user2:                          change
```

El conjunto de permisos resultantes para el fichero e:\users\user1 sería:

```
administrators:          full control
system:                  full control
help desk:               change
user1:                   change
```

- MAC OS X: Incluye ACLs desde la versión 10.4.

fsaclctl -p / -e : -e activa el uso de ACLs y -d lo desactiva.

ls -le : la opción e de ls muestra los ACLs.

chmod +a : para añadir o eliminar ACEs en el ACL correspondiente.

Por ejemplo:

```
$ chmod +a "tony allow delete" foo
$ chmod +a "admin allow delete" foo
$ ls -lde foo
drwxr-xr-x + 2 apl apl 68 Jul 19 18:32 foo
  0: group:admin allow delete
  1: user:tony allow delete
$ chmod +a# 1 "admin deny delete" foo
```

3.4.4. Control de Acceso Obligatorio (MAC)

MAC es un tipo de control de acceso definido por el TCSEC que consiste en restringir el acceso a los objetos en función de la "sensibilidad" (representada por una etiqueta) de la información contenida en dichos objetos. Su característica más importante es que es la política de seguridad del sistema la única que determina el acceso a los objetos, retirándole este privilegio al creador de los objetos. MAC es usada en sistemas multinivel que procesan datos altamente confidenciales, como información clasificada gubernamental o militar. Un sistema multinivel es aquel que maneja múltiples niveles de clasificación entre sujetos y objetos. Conceptos básicos:

- Etiquetas: en un sistema basado en MAC, todos los sujetos y objetos deben tener una etiqueta asignada. Para acceder a un objeto, el sujeto debe tener un valor de etiqueta igual o mayor que dicho objeto.
- Importación y exportación de datos: controlar dicho flujo de información de y para otros sistemas es una función crítica de los sistemas basados en MAC. Deben asegurar que las etiquetas son mantenidas de forma adecuada.

Implementaciones de MAC:

- *SELinux (Security-Enhanced Linux)*: proyecto de la NSA (National Security Agency) que ha añadido la arquitectura MAC al kernel de Linux utilizando Linux Security Modules (LSM) del kernel 2.6.
- *AppArmor:* Implementación de SUSE Linux que también se ha basado en LSM.
- *TrustedBSD:* proyecto incluído a partir de FreeBSD v5 que implementa MAC.
- *Trusted Solaris:* es un sistema operativo de seguridad evaluada basado en Solaris de Sun, que implementa el modelo MAC.
- *Mandatory Intregrity Control:* Windows Vista introduce esta implementación del modelo MAC.
- *Mac OS X 10.5 Leopard*: Apple anunciaba incluir MAC en esta versión de su operativo, pero el anuncio ha sido retirado.

3.4.5. Control de Acceso Basado en Roles (RBAC)

Los permisos para realizar ciertas operaciones son asignados a roles específicos. Los usuarios son asignados a roles particulares, a través de los cuales adquieren permisos para realizar funciones particulares.
RBAC se diferencia de las ACLs usadas por los sistemas discrecionales en que asigna permisos a operaciones específicas con significado en la organización, en lugar de a objetos de datos de bajo nivel. Por ejemplo, una lista de control de acceso puede ser usada para permitir o denegar el acceso a un ficheros, pero no se le podría decir en qué formas puede ser modificado dicho fichero. Además, el uso de roles como medio para asignar privilegios a usuarios simplifica en gran medida el manejo y creación de usuarios.

El uso de RBAC para manejar privilegios de usuario está muy extendido. Sistemas como Microsoft Active Directory, SELinux, FreeBSD, Solaris, Oracle DBMS, PostgreSQL 8.1, SAP R/3 implementan alguna forma de RBAC.

3.5. Cifrado

3.5.1. Sistemas de Ficheros de propósito general con encriptación

El cifrado a nivel de Sistema de Ficheros (*Filesystem-level encryption*), también conocido como cifrado de carpeta, es una forma de cifrado del disco donde los ficheros y directorios individuales son cifrados por el propio sistema de ficheros, en contrate con el cifrado completo del disco, en el que se cifra la partición o disco entera.
La principal desventaja de estos sistemas es que no cifran los metadatos (estructura de directorio, nombres de ficheros, tamaños o fechas de modificación).

Ejemplos de sistemas de ficheros con cifrado:

- EFS (Encrypting File System): extensión de NTFS a partir de Windows XP.

3.5.2. Sistema de Ficheros criptográficos

Los Sistemas de Ficheros Criptográficos son FS que están específicamente diseñados para el cifrado y la seguridad. Normalmente cifran todos los datos que contienen, incluyendo metadatos. En lugar de implementar un formato

de disco y su propia asignación de bloques, suelen residir sobre otro sistema de ficheros, por ejemplo, en un directorio.

Ejemplos de Sistemas de Ficheros criptográficos:

- Linux:
  - CryptoFS
  - eCryptfs
  - EncFS
  - EVFS
  - Magikfs
  - PhoneBookFS
  - Rubberhose filesystem
  - StegFS
- FreeBSD: cifrado de disco basado en GEOM (GEOM Based Disk Encryption, gbde), que cifra sistemas de ficheros completos de forma transparente. Ni un solo texto en limpio llega a tocar el disco duro.
- Windows Vista: BitLocker Drive Encryption, que proporciona cifrado de todo el volumen del sistema, incluídos los ficheros del sistema Windows y el fichero de hibernación.

Capítulo 4

Gestión de la Entrada Salida

4.1. Sistemas de Autenticación

Autenticación es el proceso por el cual una entidad se identifica en el sistema obteniendo unas credenciales (usuario, grupo, ...), las cuales determinan los permisos de acceso a recursos (control de acceso visto en el tema anterior).

El sistema operativo tiene en este punto dos misiones muy ligadas y que se confunden fácilmente: La gestión de las credenciales y la autenticación. Para implementar ambas cosas se crea el concepto de cuenta de usuario. Una cuenta de usuario normalmente incluye la siguiente información básica:

- *login* o *nombre de usuario:* identificador único de un usuario en el sistema.
- *identificador de usuario (UID):* credencial de usuario. Número único dentro del sistema que se emplea en la autorización.
- *identificadores de grupo (GID):* credenciales de grupos. Identificadores de grupos a los que pertenece este usuario.
- información adicional del usuario: nombre, teléfono, ...
- información adicional del mecanismo de autenticación: passwords, claves, tiempo de validez, ...
- información adicional del sistema operativo: directorio home, ...

Los sistemas operativos presentan un API de abstracción que abstrae el sistema de autenticación usado a las aplicaciones:

- **Unix: PAM (Pluggable Authentication Modules)** PAM es, básicamente, un mecanismo flexible para la autenticación de usuarios. PAM permite el desarrollo de programas independientes del mecanismo de autenticación a utilizar. Así es posible que un programa que aproveche las facilidades ofrecidas por PAM sea capaz de utilizar desde el sencillo /etc/passwd hasta dispositivos hardware como lectores de huella digital, pasando por servidores LDAP o sistemas de gestión de bases de datos. Además, permite al administrador del sistema construir políticas diferentes de autenticación para cada servicio. PAM fue desarrollado en 1996 por Sun Microsystems, y actualmente tiene soporte en AIX, HP-UX, Solaris, Linux, FreeBSD, Mac OS X y NetBSD.

- **Windows: LSA Authentication** La LSA (Local Security Authority) es el subsistema de seguridad responsable de todos los servicios de autenticación y autorización de usuarios en una máquina local. También es usada para procesar peticiones de autenticación hechas a través del protocolo Kerberos o el protocolo NTLM en Active Directory.

- **Windows Vista:** Windows Vista introduce una mejora de seguridad llamada *User Account Control (UAC)*, que consiste, básicamente, en que el usuario administrador sólo realiza como tal las tareas administrativas, realizando como un usuario normal todas las demás. El problema parece ser que esta mejora resulta bastante irritante para el usuario, al que le pide el password de administrador casi por cada cosa que hace, por lo que muchos usuarios están optando por desactivar esta opción.

Dependiendo de quién realice la autenticación, distinguimos entre los siguientes tipos:

- *Autenticación local:* es el propio sistema operativo el que autoriza el acceso a la máquina local a un usuario determinado.

- *Autenticación centralizada:* el sistema operativo delega en un sistema de directorio, que es el encargado de autenticar a los usuarios en el sistema. En este tipo de autenticación existen dos protocolos fundamentales:

  - Kerberos
  - LDAP

Kerberos

Kerberos es el protocolo de seguridad principal para la autenticación dentro de un dominio. El protocolo Kerberos comprueba la identidad del usuario

que solicita la autenticación y la del servidor que proporciona la autenticación solicitada. Esta comprobación doble se denomina también autenticación mutua.

El mecanismo de autenticación de Kerberos emite tickets para tener acceso a los servicios de red. Estos tickets contienen datos cifrados, que incluyen una contraseña cifrada para confirmar la identidad del usuario al servicio solicitado. Aparte de la escritura de una contraseña o las credenciales de tarjeta inteligente, todo el proceso de autenticación pasa desapercibido para el usuario.
El servicio principal de Kerberos es el Centro de distribución de claves (KDC, Key Distribution Center), que está formado por dos partes lógicas diferenciadas: un Servidor de Autenticación (Authentication Server AS) y un Servidor de Concesión de Tickets (Ticket Granting Server TGS).
El proceso de autenticación de Kerberos V5 funciona de la siguiente manera:

1. El usuario de un sistema de cliente se autentica en el AS mediante una contraseña o tarjeta inteligente (AS_REQ).

2. El KDC emite al cliente un ticket especial que concede tickets (AS_REP). El sistema cliente utiliza este ticket para tener acceso al servicio de concesión de tickets (TGS) (TGS_REQ).

3. El TGS emite a continuación un ticket de servicio para el cliente (TGS_REP).

4. El cliente presenta este ticket de servicio al servicio de red solicitado. El ticket de servicio prueba la identidad del usuario al servicio y la identidad del servicio al usuario (AP_REQ y AP_REP).

LDAP ((Lightweight Directory Access Protocol)

LDAP es un protocolo a nivel de aplicación para consultar y modificar servicios de directorio.

El protocolo proporciona una serie de mensajes para acceder a directorios LDAP (consultar, modificar, eliminar, ...). Estos directorios siguen el modelo X.500:

- Un directorio es un árbol de entradas de directorio.
- Una estrada está formada por un conjunto de atributos.
- Un atributo consta de un nombre y uno o más valores.

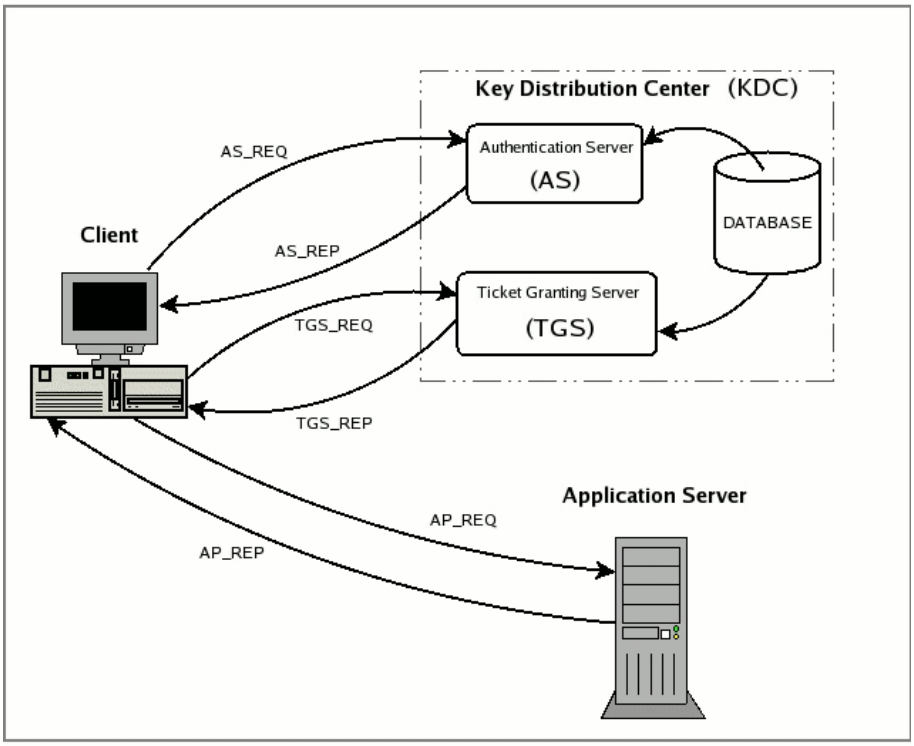

Figura 4.1: Funcionamiento básico de Kerberos

- Cada entrada tiene un nombre distinguido (Distinguished Name DN), que la identifica.
- El DN de una entrada está formado por su nombre distinguido relativo (Relative Distinguished Name RDN), que es el valor de alguno de los atributos de la entrada, seguido del DN de la entrada padre.

Un cliente puede realizar las siguientes peticiones al servidor LDAP:

- Start TLS: habilita Transport Layer Security (TLS).
- Bind: autentica al cliente en el servidor y fija la versión de LDAP.
- Search: consulta entradas de directorio.
- Compare: consulta si una entrada contiene un determinado valor de atributo.
- Add: añade una entrada.
- Delete.
- Modify.
- Modify DN: mueve o cambia de nombre una entrada.
- Abandon: aborta una petición anterior.
- Extended Operation: operación genérica usada para definir otras operaciones.
- Unbind: cierra la conexión (ojo: no es la inversa de Bind)

Servicios de Directorio (Directory Services DS)

Un servicio de directorio es una aplicación software que almacena y organiza información acerca de los usuarios y recursos de una red, controlando el acceso de los usuarios a dichos recursos.
Los servicios de directorio más conocidos son:

- *OpenLDAP*
- *Active Directory*

Active Directory

Active Directory es el servicio de directorio proporcionado por Microsoft. Utiliza distintos protocolos (principalmente protocolo LDAP, DNS, DHCP, kerberos...). Su funcionalidad principal es proveer un servicio de autenticación y autorización centralizado para redes con Windows. Su estructura jerárquica permite mantener una serie de objetos relacionados con componentes de una red, como usuarios, grupos de usuarios, permisos y asignación de recursos y políticas de acceso.

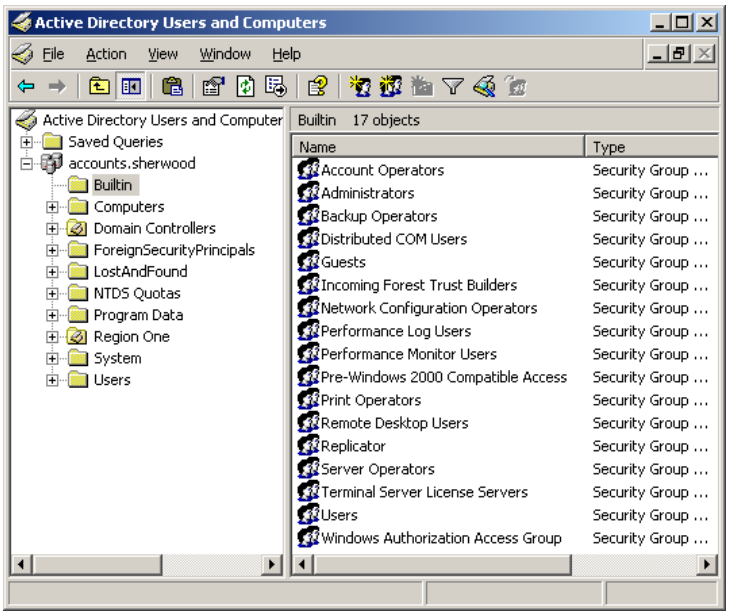

Figura 4.2: Active Directory

Capítulo 5

Gestión de Redes

En esta funcionalidad del sistema operativo, la seguridad se implementa mediante firewall, cifrado (PGP), sistemas de detección de intrusos, ...
En este tema no profundizaré, ya que ya está explicado de forma detallada en otros trabajos de la asignatura.

Capítulo 6

Conclusiones

La seguridad es un aspecto fundamental en el ámbito de los sistemas operativos. En este trabajo hemos visto algunas de las funcionalidades básicas con las que implementan la seguridad, pero hay muchas otras y variantes o alternativas de las vistas aquí.
Todos los sistemas operativos actuales tienen la seguridad como uno de sus principales objetivos, y por ello todos incluyen un sin fin de mecanismos para conseguirla.
Por tanto, la comparativa entre sistemas operativos no es muy llamativa, dando todos soporte a las funciones básicas de seguridad.

Bibliografía

[1] *Permisos, usuarios y grupos en windows vista.* http://www.pcdecasa.net/guias/uac-windows-vista.asp.

[2] P. B. G. Abraham Silberschatz, *Sistemas operativos* (Addison Wesley, 1999).

[3] J. Bonwick, *RAID-5WriteHole.* http://boink.pbwiki.com/RAID-5WriteHole.

[4] D. Brown, *Windows software raid guide.* http://www.techimo.com/articles/index.pl?photo=149.

[5] S. de los Santos, *Microsoft windows y el control de acceso, al desnudo.* http://www.hispasec.com/unaaldia/2660.

[6] T. FAQ, *How do i backup unix?* http://www.tech-faq.com/backup-unix.shtml.

[7] D. Ferraiolo and R. Kuhn, *Role-based access controls.* http://hissa.ncsl.nist.gov/rbac/paper/rbac1.html.

[8] A. R. Galdo, *Cómo configurar raid por software en linux.* http://bulma.net/body.phtml?nIdNoticia=1863http://bulma.net/body.phtml?nIdNoticia=1863.

[9] R. G. Granada, *Sistemas de ficheros con journaling en linux.* http://mnm.uib.es/gallir/.

[10] D. HP, *Control de acceso de los sistemas de confianza.* http://docs.hp.com/es/5187-2217/ch07s10.html.

[11] C. P. i Estany, *Cifrando un sistema de ficheros.* http://bulma.net/body.phtml?nIdNoticia=1970.

[12] M. P. M. M. P. M. Imobach González Sosa, *Pluggable authentication modules (pam).*
http://sopa.dis.ulpgc.es/ii-aso/portal_aso/leclinux/seguridad/pam/pam_doc.pdf.

[13] R. B. Julio Gómez, *Seguridad en sistemas operativos windows y linux* (Ra-Ma, 2006).

[14] Kriptopolis, *Sistemas de ficheros cifrados con debian gnu/linux.*
http://www.kriptopolis.org/sistemas-de-ficheros-cifrados-con-debian-gnu-linu

[15] J. M. T. Llop, *Memoria virtual.*
http://multingles.net/docs/memoria_virtual.htm.

[16] L. Manpages, *Manual de usuario del comando md(4).*
http://www.die.net/doc/linux/man/man4/md.4.htmlhttp://www.die.net/doc/linux/man/man4/md.4.htmlhttp://www.die.net/doc/linux/man/man4/md.4.html.

[17] B. F. F. Manual, *acl(5).*
http://acl.bestbits.at/cgi-man/acl.5.

[18] N. Martenet, *Stack and heap overflow.*
http://diuf.unifr.ch/tns/teaching/SecurityThreads/Martenet_StackandHeapOverflow.pdf.

[19] A. P. Martín, *Niveles de RAID.*
http://sindominio.net/~apm/articulos/raid/niveles.html.

[20] mdsn, *Windows authentication.*
http://msdn2.microsoft.com/en-us/library/aa374735.aspx.

[21] OpenSolaris, *What is zfs?*
http://www.opensolaris.org/os/community/zfs/whatis/.

[22] T. Rhodes, *Instantáneas ("snapshots") de sistemas de ficheros. manual de FreeBSD.*
http://www.freebsd.org.mx/handbook/snapshots.html.

[23] F. Ricciardi, *Kerberos authentication protocol.*
http://www.zeroshell.net/eng/kerberos/Kerberos-operation/.

[24] F. A. D. Rodríguez, *Sistemas raid.*
http://www.monografias.com/trabajos6/sira/sira.shtml.

[25] S. SM DATA, *Descripción de los niveles RAID*.
http://www.smdata.com/NivelesRAID.htm.

[26] M. R. M. Sotelo, *Seguridad informática. capítulo 4 control de acceso*.
http://www.mygnet.com/articulos/seguridad/763/.

[27] M. TechNet, *Conceptos de protocolo de autenticación*.
http://www.microsoft.com/technet/prodtechnol/windowsserver2003/es/library/ServerHelp/39486813-c18b-44af-8ad6-b832d1a9385f.mspx.

[28] Wikipedia, *Access control matrix*.
http://en.wikipedia.org/wiki/Access_Control_Matrix.

[29] ———, *Buffer overflow*.
http://en.wikipedia.org/wiki/Buffer_overflow.

[30] ———, *Directory service*.
http://en.wikipedia.org/wiki/Directory_service.

[31] ———, *Filesystem-level encryption*.
http://en.wikipedia.org/wiki/Filesystem-level_encryption.

[32] ———, *Journaling*.
http://es.wikipedia.org/wiki/Journaling.

[33] ———, *Lightweight directory access protocol (ldap)*.
http://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol.

[34] ———, *List of cryptographic file systems*.
http://en.wikipedia.org/wiki/List_of_cryptographic_file_systems.

[35] ———, *Sistemas raid*.
http://es.wikipedia.org/wiki/RAID.

[36] ———, *Stack-smashing protection*.
http://en.wikipedia.org/wiki/Stack-smashing_protection.