TEMA 3: SEGURIDAD Y PROPTECCIÓN

INTRODUCCIÓN

La ***protección*** tiene carácter interno, ya que es una tarea encargada al sistema operstivo. Y la ***seguridad*** tiene un carácter más general, en el que se incluyen, ademas de la protección, otros aspectos como la politica de copias de seguridad, las autorizaciones de acceso.

SEGURIDAD

REQUISITOS DE SEGURIDAD

En el caso de la seguridad de ordenadores y redes, se definen los siguientes requisitos de seguridad:

- **Confidencialidad**: exige que la información pueda ser leída solo por usuarios autorizados.
- **Integridad**: exige que los elementos puedan ser modificados solo por usuarios autorizados.
- **Disponibilidad**: exige que los elementos estén disponibles solo para usuarios autorizados.

TIPOS DE AMENAZAS

Las amenazas se caracterizan mejor contemplando la función que del sistema como suministrador de información. Estas son cuatro categorías generales de amenazas:

- **Interrupción**: se destruye un elemento del sistema, o se hace inaccesible o inútil (amenaza a la disponibilidad).
- **Intercepción**: una parte no autorizada consigue leer (amenaza a la confidencialidad).
- **Modificación**: una parte no autorizada no solo consigue acceder, sino que también es capaz de falsificar un elemento (amenaza a la integridad).
- **Invención**: una parte no autorizada inserta objetos falsos (amenaza a la integridad).

AMENAZAS SEGÚN LOS ELEMENTOS DE UN SISTEMA DE COMPUTACIÓN

Los elementos de un sistema de computación pueden clasificarse en cuatro categorías: hardware, software, datos y líneas de comunicaciones (redes).

ATAQUES GÉNERICOS A LA SEGURIDAD

La forma más habitual de probar la seguridad de un sistema es contratar a un grupo de expertos para ver si son capaces de penetrar en él de alguna forma. A este grupo se le conoce como *equipo tigre* o *equipo de penetración*.

1. Reserve espacio de memoria, disco o cinta, solo léalo.
2. Intente llamadas al sistema inválidas, o bien llamadas válidas con parámetros inválidos.
3. En determinados sistemas, oprimir DEL, CTRL+C o CTRL+PAUSA, el programa de verificación de contraseñas quedará invalido y el acceso se considerará exitoso.
4. Engañe al usuario escribiendo un programa que muestre el mensaje login.
5. Busque manuales que digan «no lleve a cabo X» e intente tantas variaciones de X como sea posible.
6. Convenza al administrador del sistema para que evite ciertas verificaciones.
7. Encontrar al personal de administración del centro de cálculo y engañarlo o sobornarlo.

ATAQUES ESPECÍFICOS A LA SEGURIDAD

- **Bombas lógicas**: "*estallan*"; en muchas ocasiones, son creadas por el propio desarrollador para protegerse de un posible despido.
- **Puertas traseras (*backdoors*)**: son programas que muchas veces realizan una tarea correcta, pero a través de los cuales es posible el acceso al sistema.
- **Desbordamiento de buffer**: consiste en aprovechar fallos de programación sobrescribiendo la pila de una función para que, cuando la función termine, no regrese al punto desde el que se la invocó, sino a un código cuidadosamente preparado.
- **Caballos de Troya**: sustituyen una orden interna por otra con el mismo nombre, pero que realiza labores ilegales.
- **Virus y gusanos**: programas cuya principal característica es su habilidad para extenderse dentro de un mismo ordenador o entre diferentes ordenadores por medio de sistemas de almacenamiento.
- **Spyware**: programas que se instalan sin que el usuario sea consciente de ello ni lo autorice; se ejecutan en segundo plano para recopilar información.
- **Rootkits**: son programas que han corrompido el sistema de tal forma que no pueden ser fácilmente detectados y permiten a un atacante externo acceder con privilegios.

PRINCIPIOS DE DISEÑO PARA LA SEGURIDAD

Saltzer y Schroeder (1975) crearon una guía para el diseño de sistemas seguros:

1. El diseño de un sistema debe ser público.
2. El estado predefinido debe ser el de «no acceso».
3. Se debe verificar la autorización actual.
4. Los procesos deben tener el mínimo privilegio posible que les permita seguir realizando su trabajo.
5. El mecanismo de protección debe ser simple, uniforme e integrado hasta las capas más bajas del sistema.
6. El esquema elegido debe ser psicológicamente aceptable. Si los usuarios sienten que la protección de sus ficheros implica trabajo o es molesta, simplemente no los protegerán.

PROTECCIÓN

El **hardware** proporciona tres mecanismos que ayudan al sistema operativo a proteger:

- **Hardware de direccionamiento de memoria**: asegura que un proceso únicamente se puede ejecutar dentro de su espacio de direcciones.
- **Cronómetro**: garantiza que ningún proceso podrá obtener el control de la CPU sin renunciar al mismo en algún momento.
- **Modo dual**: asegura que solo el sistema operativo puede realizar las operaciones privilegiadas (E/S).

POLÍTICA Y MECANISMO

La política indica qué operaciones será posible realizar sobre qué elementos y el mecanismo especifica cómo el sistema hará cumplir la política.

Las políticas determinan qué se va a proteger. Unas políticas vendrán determinadas por el propio diseño del sistema, otras serán especificadas por los administradores, mientras que algunas serán definidas por los usuarios.

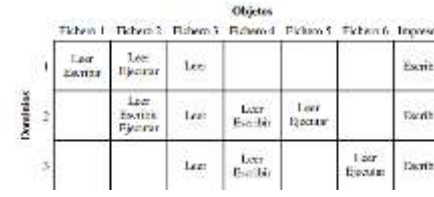
DOMINIOS DE PROTECCIÓN

Teniendo en cuenta los objetos existentes en un sistema y los procesos que pueden acceder a ellos, es necesario disponer de un mecanismo que impida que los procesos accedan a aquellos objetos sobre los que no tienen ningún permiso. Este mecanismo también debe posibilitar que los procesos puedan acceder a los objetos sobre los que tienen permitido el acceso, pero solo para realizar aquel subconjunto de operaciones que les hayan sido autorizadas. Este mecanismo se llama ***dominio de protección***. Un dominio es un *conjunto de parejas* (objeto, derechos).

Cada proceso existente en el sistema se tiene que ejecutar en, al menos, uno de los dominios de protección que haya definidos.

MATRIZ DE ACCESO

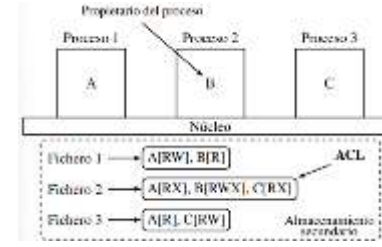
Un aspecto importante a tener en cuenta es la forma en la que el sistema lleva un registro de las parejas (objeto, derechos) que pertenecen a un dominio dado. Teóricamente es una enorme matriz en la que las filas corresponden a los dominios y las columnas a los objetos (matriz de protección o matriz de acceso). Cada celda contendrá los derechos correspondientes a un objeto en un dominio.



En la práctica, rara vez se almacena tal cual la matriz de protección que acabamos de describir; como la mayor parte de los dominios solo tiene acceso a un número reducido de objetos, la matriz estará casi vacía en relación con su tamaño. No obstante, existen dos métodos que guardan solo los elementos no vacíos de la matriz. Estos métodos son las listas de control de acceso y las listas de posibilidades.

LISTAS DE CONTROL DE ACCESO

En las listas de control de acceso, la matriz de protección se almacena por columnas; se asocia a cada objeto una lista con todos los dominios que pueden acceder al objeto y los derechos concretos de cada dominio para dicho acceso. Además de los típicos permisos sobre ficheros, un sistema puede definir permisos adicionales para realizar otro tipo de operaciones (copiar, borrar, ordenar, etc).



Lo más razonable es que cada ACL se almacene en disco junto con su fichero. Sin embargo, las ACL están protegidas por el núcleo del sistema operativo, por lo que no pueden ser directamente manipuladas por los procesos de usuario.

Si los usuarios se pueden agrupar, las listas de control de acceso también se pueden aplicar a grupos. Y un aspecto a tener en cuenta es que si la ACL de un fichero solo se comprueba cuando se abre este.

POSIBILIDADES O CAPACIDADES

Las ***listas de posibilidades*** o ***listas de capacidades*** almacenan una matriz de protección por filas, se trataría de asociar a cada dominio una lista de los objetos a los cuales puede acceder junto con una indicación de las operaciones permitidas sobre cada uno. Ya que son los procesos, los que realmente acceden a los objetos, cada proceso tendrá su propia lista, en la que cada elemento recibe el nombre de capacidad o posibilidad.

Generalmente, cada *posibilidad* (entrada de una lista) contiene el tipo de objeto al que hace referencia la entrada, los derechos y un apuntador al propio objeto.

Aunque las listas de posibilidades se asocian a procesos, deben ser protegidas. Una forma de conseguir esto es manteniendo la lista de posibilidades en el núcleo del sistema operativo.

Además de los derechos dependientes del objeto, las posibilidades suelen tener también derechos genéricos, como:

- **Copiar posibilidad**: crea una nueva posibilidad para el mismo objeto.
- **Eliminar posibilidad**: elimina un dato de la lista.
- **Destruir objeto**: elimina de forma permanente un objeto y una posibilidad.

La primera vez que un proceso accede a un fichero, el sistema operativo consulta la ACL asociada al fichero para saber si el proceso tiene permisos para realizar la operación que desea. Si no tiene permiso, se produce un error. En caso contrario, se crea una posibilidad para dicho fichero en la lista de posibilidades y se devuelve la posición ocupada por la posibilidad.

COMPARACIÓN

Para la protección de un equipo se necesita comparar si los usuarios o procesos tienen acceso a la información o sistema operativo o al proceso, etc…

CANCELACIÓN Y REVOCACIÓN

En un sistema de protección dinámico es necesario a veces cancelar derechos de acceso a objetos que son compartidos por usuarios. ***La cancelación*** o ***la revocación*** se demora, no tiene efecto inmediato.

Con las listas de control de acceso, la *cancelación* es bastante sencilla. Dado un objeto, se busca en su ACL los derechos que serán cancelados y simplemente se eliminan. La cancelación es inmediata y puede ser general o selectiva, total o parcial, o permanente o temporal.

Las posibilidades, sin embargo, presentan un problema de cancelación más difícil. Dado que las posibilidades de un objeto están distribuidas por el sistema, es necesario localizarlas antes de poder cancelarlas. Al sistema le lleva tiempo, ya que pueden estar almacenadas en listas de posibilidades de muchos procesos. Aunque existen varias técnicas que permiten implementar la cancelación de posibilidades.

Una primera técnica consiste en hacer que cada posibilidad apunte hacia un objeto indirecto. Si el objeto indirecto apunta hacia el objeto real, el sistema siempre puede romper esta conexión, invalidando así todas las posibilidades.

Otra técnica es asignar a cada objeto un número aleatorio y almacenarlo también en la posibilidad. De esta forma, cuando se intenta acceder a un objeto a través de su posibilidad, se comparan los dos números y se permite la operación si ambos números coinciden.

Como vemos, ninguna de estas dos técnicas permite ni una revocación selectiva ni parcial.

AUTENTIFICAIÓN DE USUSARIOS

La protección, tal como la hemos descrito, es estrictamente un problema interno del sistema operativo. La seguridad, por otra parte, no solo requiere un adecuado sistema de protección, sino también tener en cuenta el entorno externo donde opera el sistema. Los problemas de seguridad son esencialmente problemas administrativos, no del sistema operativo. El principal problema de seguridad para los sistemas operativos es el de la validación. El problema de la identificación de los usuarios cuando se conectan al sistema se conoce como autenticación de usuarios.

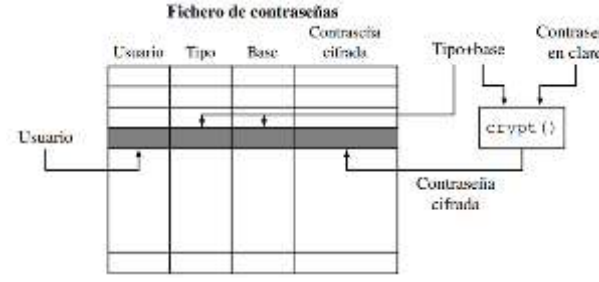
El método más general para realizar la autenticación de usuarios consiste en usar un identificador (ID o usuario) más una contraseña. La contraseña permite autenticar el ID del individuo que se conecta

al sistema. Por su parte, el ID introduce seguridad en dos sentidos; si el usuario está autorizado para acceder al sistema y los privilegios acordados con el usuario.

CONTRASEÑAS EN LINUX

A la hora de almacenar una nueva contraseña para un usuario, se pasan como parámetros a una rutina de cifrado conocida como crypt() la propia la contraseña, un valor «base» y el algoritmo de cifrado a utilizar. Esta rutina devuelve la contraseña cifrada, la cual se almacena en el fichero de contraseñas en la línea asociada al ID del usuario. En concreto, el formato del campo que guarda la contraseña es: $tipo$base$contraseña_cifrada, donde el tipo indica el algoritmo empleado. El valor de la base normalmente está relacionado con el momento en que se asigna la contraseña a un usuario y sirve para impedir que contraseñas iguales se cifren de la misma manera.

Cuando un usuario intenta conectarse a un sistema Unix, debe proporcionar un ID y una contraseña. El sistema operativo utiliza el ID como índice en el fichero de contraseñas para recuperar el tipo, la base y la contraseña cifrada. El tipo y la base, junto con la contraseña introducida por el usuario, se pasan como parámetros a la función crypt(), cuyo resultado se compara con la contraseña cifrada almacenada en el fichero de contraseñas. Si el resultado obtenido es igual al valor almacenado, la contraseña es aceptada. En caso contrario, se produce un error de autenticación.



ESTRATEGIAS DE ELECCIÓN DE CONTRASEÑAS

Las funciones ***hash*** utilizadas para el cifrado de contraseñas están diseñadas de tal forma que no existe la correspondiente función inversa. Además, el diseño de estas funciones evita los ataques por adivinación.

Por desgracia, algunos usuarios, cuando pueden elegir su propia contraseña, eligen una fácilmente adivinable. La solución a este problema es hacer que el sistema rechace cualquier contraseña que no cumpla unos criterios como la longitud que contengan unos mínimos números, símbolos, etc…

Existen varias técnicas básicas para elegir contraseñas:

En la ***instrucción del usuario***, se explica a los usuarios la importancia de seleccionar contraseñas difíciles de adivinar y se les enseña cómo escoger contraseñas seguras. También se recomiendan algunos buenos hábitos. En la actualidad, una buena contraseña está formada por letras mayúsculas y minúsculas, números y signos de puntuación.

En la ***inspección proactiva de contraseñas***, un usuario puede elegir su propia contraseña, pero, si en el momento de la selección el sistema estima que la contraseña es fácilmente adivinable, la rechaza.

Las dos técnicas no son incompatibles, es más, se complementan perfectamente.

PROTECCIÓN EN UNIX

DOMINIOS EN UNIX

En Unix, cada usuario se identifica mediante un número llamado ***identificador de usuario*** o ***UID***. Este número se asocia al login del usuario a través de la entrada correspondiente del fichero /etc/passwd. Además, existen ***grupos de usuarios***, también se identifican mediante un número llamado ***identificador de grupo*** o ***GID***. El fichero /etc/group es el que asocia el nombre del grupo con su número; también almacena la lista de usuarios pertenecientes a cada grupo.

Un aspecto a tener en cuenta es que puede haber varios usuarios definidos con el mismo UID, o grupos con el mismo GID. Por tanto, para el sistema operativo se tratará siempre del mismo usuario o del mismo grupo.

Dos procesos con el mismo par (UID, GID) tendrán acceso al mismo conjunto de objetos por pertenecer al mismo dominio, mientras que procesos con diferentes valores (UID, GID) tendrán acceso a un conjunto distinto de objetos.

Entre todos los UID, el 0 tiene todos los privilegios y está asociado al usuario con nombre root.

LISTAS DE CONTROL DE ACCESO RESTRINGIDAS

Cada fichero y cada dispositivo pertenece a un usuario y a un grupo. Estos ficheros tienen asociadas listas de control de acceso que se califican como restringidas por estar limitadas a tres conjuntos de usuarios (propietario, grupo y otros).

Cada conjunto de usuarios tiene asociados tres posibles permisos: lectura («r»), escritura («w») y ejecución («x»). Estos permisos se suelen mostrar mediante una cadena de 9 letras, 3 para cada conjunto.

Además de estas ***ACL restringidas*** también implementan listas de control de acceso completas para aquellos sistemas de ficheros que las admiten. Gracias a estas listas, es posible especificar con gran detalle los usuarios y grupos concretos que podrán acceder a un fichero y con qué permisos.

Todo lo que un proceso puede hacer sobre un determinado fichero depende del UID y GID a los que pertenece el proceso del UID y GID a los que pertenece el fichero, y de la ACL restringida del fichero. A la hora de comprobar el acceso, se llevan a cabo los siguientes pasos:

1. Si el UID del proceso coincide con el UID del fichero, se consultan los permisos del fichero asociados al usuario propietario del fichero.
2. Si lo anterior no se cumple, entonces se comprueba si el GID del proceso coincide con el GID del fichero. Si coinciden, entonces se consultan los permisos del fichero asociados al grupo propietario del fichero.
3. Si lo anterior tampoco se cumple, entonces el proceso podrá hacer lo que indiquen los permisos asociados al resto de usuarios.

DOMINIO DE UN PROCESO Y CAMBIO DE DOMINIO

Por omisión, cada proceso se ejecuta siempre con el **dominio (UID, GID)** del usuario que lo crea. No obstante, de forma excepcional, los ejecutables pueden poseer en sus atributos los bits ***SETUID*** y ***SETGID***, que pueden cambiar el dominio final al que pertenece un proceso. En Unix, cuando se crea un proceso con `fork()`, el nuevo proceso hereda del padre su dominio. Sin embargo, si un proceso cualquiera realiza la llamada al sistema `execve()` y el ejecutable que se pasa como parámetro tiene activo el bit SETUID, entonces el proceso cambiará su UID cambiando así de dominio de forma efectiva. De igual forma le ocurre al bit SETGID.

Un posible cambio de dominio producido por los bits SETUID y SETGID hace que, en realidad, un proceso tenga dos parejas de identificadores: el usuario/grupo real (UID, GID) y el usuario/grupo efectivo (EUID, EGID).

La orden `passwd` permite a un usuario cambiar su contraseña, pero no la de los demás usuarios. Para saber qué tipo de usuario está cambiando una contraseña, `passwd` utiliza la función `getuid()` que devuelve el UID real. De esta manera, sabe qué contraseñas puede cambiar. Si la función devuelve 0, entonces el usuario root puede cambiar la contraseña de cualquier usuario. En cambio, si la función devuelve un valor distinto de 0, entonces el programa lo está ejecutando un usuario normal que solo podrá cambiar su contraseña.

COMBINACIÓN DE LISTAS DE CONTROL DE ACCESO Y DE POSIBILIDADES

Unix implementa un ***sistema híbrido*** entre *listas de control de acceso* y *listas de posibilidades*. Los ficheros tienen asociada una ACL restringida. Cuando un proceso quiere abrir un fichero, se comprueba la ACL para ver si tiene acceso. En caso afirmativo, se crea una entrada en la tabla de ficheros abiertos y se devuelve un índice a la misma denominado descriptor del fichero. La entrada contiene los permisos para ese proceso sobre ese fichero. De este modo, cuando el proceso quiera realizar una operación sobre el fichero, solo tendrá que pasar el descriptor del fichero como parámetro para tener acceso al mismo. Cuando se cierra el fichero, se elimina la entrada de la tabla.

Aún debe verificarse el derecho en cada acceso, y que la entrada de la tabla de ficheros tiene una posibilidad solo para las operaciones permitidas. Un problema de esta combinación de listas de control de acceso y listas de posibilidades es que la ACL de un fichero solo se comprueba al abrirlo. Esto incumple el principio de diseño que establece que la autorización se tiene que comprobar continuamente. No obstante, este tipo de seguridad un poco más laxa permite a cambio mejorar el rendimiento al trabajar con ficheros.