AVANCE 11/06/2025

Núñez Anrubio Jonathan Mahonri & Peña Razo Christian Yael

# 6.1 El ambiente de seguridad

**Conceptos clave:**

* **Seguridad**: Problema general que abarca todos los aspectos (técnicos, administrativos, legales, políticos) necesarios para evitar que personas no autorizadas lean, modifiquen o destruyan datos.
* **Mecanismos de protección**: Conjunto de técnicas y herramientas específicas del sistema operativo (como son las listas de control de acceso, autenticación, cifrado, aislamiento de procesos) que implementan la seguridad.

Claramente son conceptos muy distintos, cumpliendo con funciones específicas y en donde no siempre existe un límite claro.

Utilizaremos el término seguridad para hacer referencia al problema general, y mecanismos de protección para referirnos a los mecanismos específicos del sistema operativo. Se hablará sobre la seguridad para ver cuál es la naturaleza del problema, más adelante analizaremos los mecanismos de protección y los modelos disponibles para ayudar a obtener seguridad, estableciendo soluciones.

**Principales facetas de la seguridad:**

**Amenazas**

Dentro de este marco general, las amenazas se presentan como cualquier circunstancia que pueda comprometer la información. Pueden manifestarse sin modificar directamente los datos, por ejemplo, cuando se interceptan comunicaciones, o bien mediante acciones activas que alteran o eliminan información valiosa. Tanto usuarios internos, con permisos legítimos, como agentes externos pueden aprovechar vulnerabilidades en la red, en las aplicaciones o en dispositivos de almacenamiento extraíble para lograr sus objetivos, que pueden ir desde el espionaje y el sabotaje hasta la obtención de ventajas económicas o políticas.

**Intrusiones**

Quienes llevan a cabo estas intrusiones, llamados intrusos, adoptan perfiles muy diversos según sus motivaciones. Hay quienes impulsados por la curiosidad técnica exploran sistemas sin intenciones claramente maliciosas, mientras que organizaciones criminales se enfocan en el lucro a través del robo de identidad o fraude. Los hackers actúan movidos por causas ideológicas o políticas, y en ocasiones incluso gobiernos emplean técnicas de espionaje cibernético. Para acceder al sistema, estos actores combinan métodos como la ingeniería social, la explotación de vulnerabilidades en el software o ataques de fuerza bruta, y pueden llegar a comprometer tanto el nivel de usuario como el núcleo del sistema mediante rootkits y modificaciones de controladores.

**Pérdida de datos**

No obstante, no todo peligro proviene de la malicia; la pérdida accidental de datos representa otra cara de la inseguridad. Los errores humanos al borrar archivos o configurar sistemas, las fallas de hardware en discos o memorias, los errores de software y los desastres naturales pueden ocasionar daños irreversibles o interrupciones prolongadas. Frente a esto, las organizaciones suelen implementar copias de seguridad periódicas (tanto locales como remotas), emplear sistemas de almacenamiento redundante, utilizar sistemas de archivos con registro de transacciones que facilitan la recuperación tras cortes de energía, y diseñar planes de recuperación ante desastres que permitan restablecer las operaciones con la mayor rapidez posible.

Este enfoque de primero entender amenazas, luego intrusos y pérdidas accidentales, y finalmente estudiar las soluciones nos permitirá tener una visión del ambiente de seguridad antes de meternos en la implementación de mecanismos específicos.

# 6.1.1 Seguridad en los sistemas operativos

La seguridad en los sistemas operativos parte de la necesidad creciente de proteger la información valiosa, tanto en entornos corporativos como en equipos domésticos, pues cada vez se almacenan diseños técnicos, estudios comerciales, datos financieros o recuerdos personales en formato digital. Esta protección se asemejaba al resguardo físico de un edificio: guardias en la puerta, oficinas con llave y controles de acceso a archivos, sin embargo, cuando trasladamos esa analogía al mundo de las computadoras, las amenazas toman formas específicas, virus, gusanos y “bugware” explotable, y los mecanismos de protección se integran en el propio sistema operativo.

En las décadas de los 60 y 70, en los grandes mainframes y minicomputadoras multiusuario, la seguridad consistía sobre todo en impedir que un usuario viera o alterara los datos de otro, se desarrollaron modelos formales y mecanismos sencillos que, al aplicarse a sistemas relativamente pequeños y estáticos, funcionaban casi sin fallos. Con la llegada de la computadora personal y su conexión masiva a Internet, ese escenario cambió por completo: lo que antes era un único punto de falla aislado se convirtió en millones de dispositivos interconectados y expuestos a ataques externos.

El crecimiento desmesurado de la complejidad del software (con kernels que llegan a tener millones de líneas de código y aplicaciones de gran tamaño) ha multiplicado las vulnerabilidades. Aunque en teoría podemos demostrar la corrección de un modelo de seguridad, en la práctica bastan unos pocos errores en el código para que un atacante, o incluso un programa malicioso, ejecute acciones prohibidas, esto realiza un desequilibrio entre los modelos formales y la realidad del “bugware inflado” sitúa a la industria ante un dilema: mantener sistemas sencillos y seguros o satisfacer la demanda con nuevas funcionalidades, arriesgando la integridad del sistema.

La noción de “sistemas confiables” demuestra que es técnicamente posible diseñar plataformas seguras, como lo evidenció MULTICS en los años sesenta, sin embargo, la adopción de estas soluciones tropieza con dos barreras: los usuarios y las organizaciones suelen preferir sistemas más versátiles, aunque menos seguros, y los desarrolladores añaden características a costa de aumentar la complejidad y el número de errores. En la práctica, la simplicidad es la única garantía real de seguridad, pero la presión del mercado y la competencia lleva a sacrificarla.

Finalmente, todo sistema que aspire a ser seguro debe autenticar a sus usuarios antes de permitirles cualquier operación. La autenticación ha evolucionado desde no existir en los primeros ordenadores hasta convertirse en un requisito en sistemas personales y corporativos, así como en servicios remotos de banca y comercio electrónico. Los métodos de autenticación, basados en algo que el usuario sabe, constituyen la primera línea de defensa para garantizar que solo las personas autorizadas accedan a los recursos del sistema.

# 6.1.2 Control de acceso a los recursos

En los **sistemas operativos modernos**, la matriz completa de accesos resulta impracticable: su tamaño crece rápidamente y la mayor parte de sus celdas suelen estar vacías. Para resolverlo, se almacena la información por columnas, de manera que cada objeto (un archivo, un directorio o cualquier recurso) lleva consigo una lista ordenada de aquellos sujetos (usuarios o grupos) a los que se les concede algún tipo de permiso. A esta estructura se la denomina **Lista de Control de Acceso**, o ACL, y constituye la forma más extendida de encarnar la matriz de protección en un espacio compacto.

Cada entrada en la ACL asocia un identificador de sujeto (generalmente un par UID, y opcionalmente un GID) con un conjunto de privilegios: lectura, escritura, ejecución y otros más específicos según el tipo de objeto (por ejemplo, adjuntar mensajes en una bandeja de correo o destruir un objeto genérico). A diferencia de la lógica a nivel de proceso, los permisos se evalúan por propietario: basta con que un proceso pertenezca al usuario o grupo indicado para heredar los derechos allí definidos. Cuando un solicitante trata de **acceder al recurso**, el sistema recorre la ACL en busca de la primera coincidencia y aplica los privilegios listados; si no aparece ninguna, la petición se rechaza.

El uso de grupos introduce un nivel de abstracción similar al de los roles. Un usuario puede pertenecer simultáneamente a varios grupos, y las ACL pueden contener entradas tanto de usuarios individuales como de identificadores de grupo. En algunos esquemas se exige que el proceso especifique el grupo activo al iniciar sesión, mientras que en otros basta con que pertenezca a cualquiera de ellos para que se apliquen los permisos correspondientes. Para casos especiales, se recurre al comodín “\*”, que puede otorgar derechos universales o bien bloquear de forma selectiva a un sujeto concreto si aparece en la primera posición de la lista.

La **flexibilidad de las ACL** facilita también la revocación de permisos: basta con editar la lista y cambiar o eliminar la entrada correspondiente. No obstante, hay una sutileza operativa importante: los cambios solo afectan a las nuevas llamadas de apertura de archivo. Si un descriptor ya está abierto, sus privilegios persisten hasta que se cierre, de modo que un usuario podría seguir trabajando con un recurso aun cuando su acceso haya sido revocado posteriormente. Este detalle, junto con el orden de las entradas y la potencial complejidad de las listas, sobre todo en entornos con muchos grupos y permisos específicos, revela que, a pesar de su eficiencia espacial, las ACL requieren un diseño cuidadoso y unas políticas claras para evitar comportamientos inesperados.

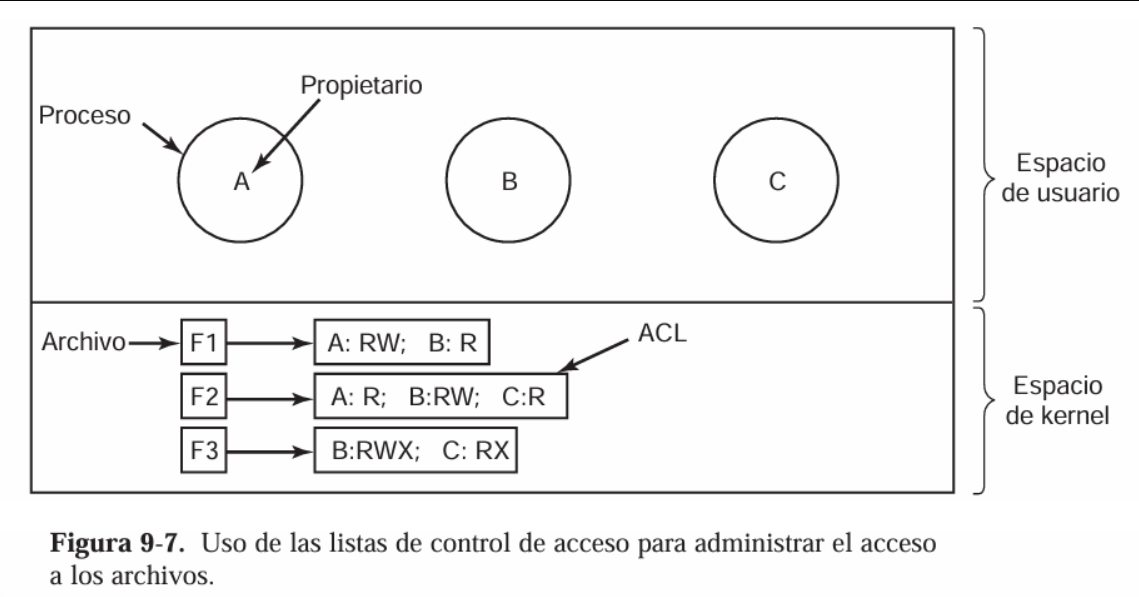
# 6.1.3 Implementación de matrices de acceso

Cuando intentamos hacer práctica la matriz de protección enseguida nos topamos con dos problemas fundamentales: la inmensa mayoría de las celdas estarán vacías y el tamaño crece con cada usuario y cada recurso nuevo. Para aprovechar la dispersión de la matriz (“sparseness”) y no malgastar espacio ni tiempo de búsqueda, se han ideado dos estrategias opuestas pero complementarias: una almacena la matriz por columnas (las ACL, que ya vimos en 6.1.2) y la otra por filas, lo que da lugar a las llamadas **listas de capacidades**.

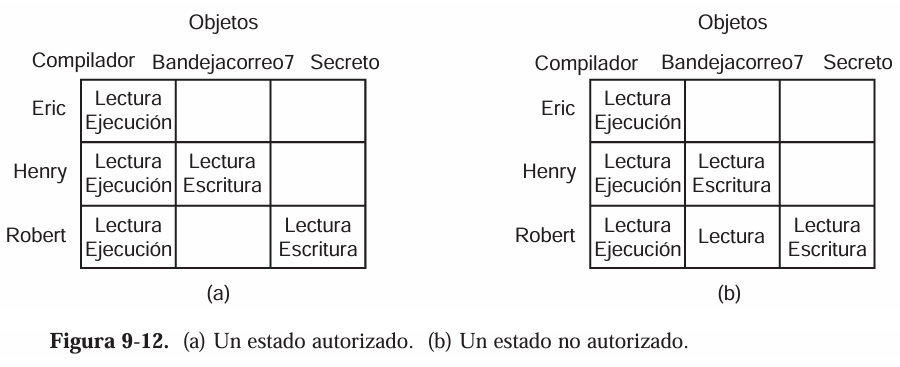
En la aproximación por filas, al dominio o propietario (un usuario, un grupo o un rol) se le asocia una lista de **capacidades**: cada entrada identifica un objeto junto con el conjunto de permisos concedidos (lectura, escritura, ejecución, etc.). Esa “capacidad” suele representarse como un token inalterable o un descriptor cerrado por el sistema, de modo que ningún proceso pueda fabricar o modificar su propia lista. Cuando un programa quiere acceder a un recurso, “presenta” la capacidad correspondiente en vez de que el sistema tenga que recorrer una ACL. Esto convierte la comprobación en una simple verificación de validez del descriptor, lo que resulta muy rápido y localiza el coste de la operación en el dominio del solicitante.

Sin embargo, las listas de capacidades plantean su propio desafío: la **revocación**. Si quiero retirar derechos de un usuario concreto, ¿cómo invalido la capacidad que ya tiene en su poder? A diferencia de las ACL, donde basta con editar la lista asociada al objeto, aquí no existe un punto único de control. Para atajar esto, se recurren a mecanismos adicionales, como llevar un registro centralizado de todas las capacidades emitidas para cada objeto (y comprobar en cada acceso si no ha sido revocada), o implementar “martes de revocación” periódicos que comparan listas y cancelan tokens obsoletos.

En la práctica, muchos sistemas operativos y lenguajes de ejecución combinan ambas ideas: usan ACL para recursos de gran nivel (sistemas de archivos, dispositivos) y listas de capacidades para objetos más dinámicos (canales de comunicación, entradas de servicio). De este modo logran un equilibrio entre la **granularidad y la facilidad de revocación** de las ACL y la **velocidad y descentralización** de las capacidades. Cualquiera que sea el método elegido, comprender que la matriz completa es solo un modelo conceptual y que su implementación exige transformar filas o columnas en estructuras manejables es la clave para diseñar sistemas de protección escalables y eficientes.



# 6.1.4 Modelos formales de seguridad



Los modelos formales de seguridad parten de la idea de que la matriz de protección—esa tabla que asocia dominios (usuarios o roles) con permisos sobre objetos—es en realidad una estructura dinámica, sometida constantemente a la creación y destrucción de objetos y dominios, así como a la concesión o revocación de derechos. Harrison y sus colaboradores identificaron en 1976 un conjunto de seis operaciones primitivas sobre esta matriz—crear y borrar objetos, crear y borrar dominios, y añadir o quitar permisos—que sirven de cimiento para cualquier sistema de protección. En la práctica, esas operaciones no se invocan de manera directa, sino que se agrupan en comandos de más alto nivel (por ejemplo, “crear archivo” o “conceder permiso de lectura a todos los usuarios”), que realizan comprobaciones previas y actualizan la matriz de forma controlada.

Es importante distinguir entre lo que la matriz hace cumplir (el mecanismo) y la política deseada (la directiva de autorización). Un sistema puede permitir, por ejemplo, que Henry acceda a cierto buzón de correo y que Robert modifique un documento secreto, y esa sería la directiva autorizada. Sin embargo, si un intruso lograra invocar comandos que alteren la matriz más allá de lo previsto, podría alcanzar un estado en el que adquiera permisos indebidos sin que el sistema lo detecte, exactamente como en el ejemplo de la figura 9-12(b), donde Robert gana acceso a un recurso que no le corresponde.

Esa observación da pie al conocido **problema de seguridad** o de “safety”: partiendo de un estado inicial autorizado y de un conjunto de comandos de protección, ¿se puede garantizar que nunca se llegará a un estado no autorizado? Lamentablemente, Harrison et al. demostraron que, en el caso más general, este problema es indecidible: no existe un algoritmo que, dado cualquier sistema de protección arbitrario, asegure por adelantado que la seguridad no podrá violarse. No obstante, cuando el conjunto de operaciones y la estructura de dominios y objetos están acotados y bien definidos, sí es posible analizar exhaustivamente todas las transiciones y probar que el sistema es seguro en la práctica. Para profundizar en estas demostraciones y sus matices, conviene revisar trabajos posteriores como los de Landwehr (1981), que exploran condiciones bajo las cuales la seguridad puede certificarse de forma efectiva.

# Bibliografía:

## 6.1 El ambiente de seguridad

*Tanenbaum, A. S., & Van Steen, M. (2009). Sistemas operativos modernos (3.ª ed., A. V. R. Elizondo, Trad.; J. R. R. Sánchez & A. J. Govea, Revisión técnica). Pearson Educación.*

* *9.1ELENTORNO DE SEGURIDAD 613*
* *9.1.1Amenazas 613*
* *9.1.2 Intrusos 615*
* *9.1.3 Pérdida accidental de datos 616*

## 6.1.1 Seguridad en los sistemas operativos

*Tanenbaum, A. S., & Van Steen, M. (2009). Sistemas operativos modernos (3.ª ed., A. V. R. Elizondo, Trad.; J. R. R. Sánchez & A. J. Govea, Revisión técnica). Pearson Educación.*

* *9 SEGURIDAD 611*
* *9.3.4 Sistemas confiables 630*
* *9.4 AUTENTICACIÓN 641*

## 6.1.2 Control de acceso a los recursos

*Tanenbaum, A. S., & Van Steen, M. (2009). Sistemas operativos modernos (3.ª ed., A. V. R. Elizondo, Trad.; J. R. R. Sánchez & A. J. Govea, Revisión técnica). Pearson Educación.*

* *9.3.2 Listas de control de acceso 624*

## 6.1.3 Implementación de matrices de acceso

*Tanenbaum, A. S., & Van Steen, M. (2009). Sistemas operativos modernos (3.ª ed., A. V. R. Elizondo, Trad.; J. R. R. Sánchez & A. J. Govea, Revisión técnica). Pearson Educación.*

* *9.3.2 Listas de control de acceso 624*

## 6.1.4 Modelos formales de seguridad

*Tanenbaum, A. S., & Van Steen, M. (2009). Sistemas operativos modernos (3.ª ed., A. V. R. Elizondo, Trad.; J. R. R. Sánchez & A. J. Govea, Revisión técnica). Pearson Educación.*

* *9.3.6 Modelos formales de los sistemas seguros 632*