Resumen Teórica 2 : Control de Acceso

Tomás F. Melli

August 2025

Índice

1 Introducción

El control de acceso es un pilar fundamental en la seguridad de la información, ya que define **quién puede acceder a qué recursos y bajo qué condiciones**. Su objetivo principal es proteger los datos y sistemas de accesos no autorizados, garantizando la confidencialidad, integridad y disponibilidad de la información. En términos generales, el control de acceso se ocupa de:

- **Identificación**: Determinar la identidad del usuario que intenta acceder al sistema (por ejemplo, mediante un nombre de usuario).
- **Autenticación**: Verificar que el usuario sea realmente quien dice ser (contraseñas, tokens, biometría, etc.).
- **Autorización**: Decidir qué recursos y operaciones puede realizar el usuario una vez autenticado, según políticas definidas.
- **Auditoría / Monitoreo**: Registrar y supervisar los accesos para detectar incidentes y cumplir con regulaciones.

Cuando hablamos de control de acceso, nuestro objetivo es proteger recursos y datos del sistema frente a accesos no autorizados. Para entender cómo esto se aplica, necesitamos definir el **estado del sistema**:

- **Estado de un sistema**: Es el conjunto de todos los valores actuales de memoria, almacenamiento, registros del procesador y otros componentes del sistema. En otras palabras, describe la "fotografía" completa del sistema en un momento dado.
- **Estado de protección del sistema**: Es un subconjunto del estado del sistema, enfocado exclusivamente en los aspectos relacionados con la seguridad y protección. Incluye, por ejemplo: Qué usuarios están autenticados.Qué permisos tiene cada usuario. Qué objetos (archivos, procesos, dispositivos) están protegidos y cómo.

1.1 Sujetos y Objetos

Para decidir quién puede hacer qué sobre qué, necesitamos definir claramente quiénes actúan (sujetos) y qué se protege (objetos).

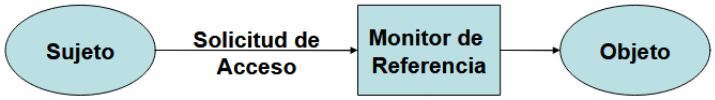
- **Sujetos**: Son las entidades que realizan acciones sobre el sistema y, por lo tanto, pueden modificar su estado. Incluye:
  - Usuarios: personas que acceden al sistema.
  - Grupos o roles: conjuntos de usuarios con permisos similares.
  - Procesos: programas en ejecución que actúan en nombre de un usuario.
- **Objetos**: Son las entidades que deben ser protegidas, porque su modificación afecta el estado de protección del sistema.Incluye:
  - Memoria, archivos, directorios.
  - Datos en dispositivos de almacenamiento.
  - Programas y dispositivos de hardware.

1.2 Permisos y Solicitudes de Acceso

Los sujetos son quienes solicitan operaciones sobre los recursos (**solicitudes de acceso**). Los objetos son los recursos sobre los que los sujetos quieren actuar. Si esas acciones son autorizadas por el sistema, las llamamos **permisos**.

1.3 Monitor de Referencia (MR)

En la gestión de la seguridad de la información, no basta con definir quiénes son los sujetos, qué objetos deben protegerse y qué permisos tienen. También necesitamos un mecanismo que haga cumplir estas reglas en tiempo real. Acá es donde entra en juego el **Monitor de Referencia**. Un componente central del sistema encargado de mediar cada operación solicitada por un sujeto sobre un objeto. Su función es garantizar que todas las acciones respeten la política de control de acceso, evitando que se comprometa el estado de protección del sistema. Cada solicitud de acceso pasa obligatoriamente por el MR, que verifica permisos, autoriza o deniega la operación, y mantiene el sistema seguro.

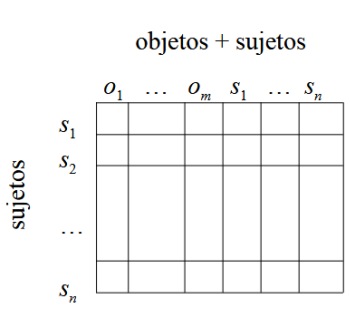


Debe cumplir con ciertas propiedades :

- **Intermediación obligatoria** : Todo acceso a objetos protegidos debe pasar necesariamente por el MR. Esto asegura que ninguna operación pueda evadir las políticas de control de acceso. Sin esta propiedad, un sujeto podría interactuar directamente con un objeto, comprometiendo la seguridad.
- **Aislamiento** : El MR y sus estructuras de datos deben ser incorruptibles. No deben existir modificaciones no autorizadas que alteren su funcionamiento o sus decisiones. Garantiza que los sujetos no puedan manipular el mecanismo que controla el acceso, manteniendo la integridad del sistema.
- **Verificabilidad** : Se debe poder demostrar que el MR funciona correctamente, es decir, que siempre aplica la política de control de acceso de manera consistente y segura. Esto es clave para la confianza y auditoría: permite certificar que el sistema protege los objetos como se espera.

2 Matriz de Control de Accesos (ACM)

Hasta ahora vimos que el Monitor de Referencia (MR) es el mecanismo que medía todas las solicitudes de acceso de los sujetos sobre los objetos, verificando permisos y manteniendo seguro el estado de protección del sistema. Para que el MR pueda cumplir correctamente su función, necesita un modelo que defina de manera clara y precisa qué acciones están permitidas para cada sujeto sobre cada objeto. Ahí es donde entra la Matriz de Control de Accesos. La ACM es un modelo conceptual que representa el estado de protección del sistema en forma de tabla o matriz. Cada fila corresponde a un sujeto (usuario, proceso o rol), cada columna a un objeto (archivo, programa, dispositivo), y las celdas indican los permisos que ese sujeto tiene sobre ese objeto.Así, la ACM funciona como una referencia para el MR, permitiéndole decidir de manera consistente y verificable si una solicitud de acceso se aprueba o se deniega.



- Sujetos $S = \{ s_1,\ldots,s_n \}$
- Objetos $O = \{ o_1,\ldots,o_m \}$
- Permisos $R = \{ r_1,\ldots,r_k \}$
- Entradas $A[s_i, o_j] \subseteq R$

$A[s_i, o_j] = \{ r_x, \ldots, r_y \}$
Es decir el sujeto s_i tiene permisos $r_x, \ldots, r_y$ sobre el objeto o_j

2.1 Ejemplos

Ejemplo 1

Sujetos: proc1, proc2
Objetos: arch1, arch2
Permisos: r = read, w = write, x = execute, a = append, o = own

	arch1	arch2	proc1	proc2
proc1	rwo	r	rwxo	w
proc2	a	ro	r	rwxo

Cada celda indica los permisos que ese proceso tiene sobre el archivo correspondiente. Por ejemplo, proc1 puede leer, escribir y ejecutar arch1, mientras que proc2 tiene permisos distintos.

Ejemplo 2

Sujetos: alice, bob
Objetos: memo.doc, demo.exe, backup.pl
Permisos: r = read, w = write, x = execute

	memo.doc	demo.exe	backup.pl
alice	− − −	x	rx
bob	rw	x	rxw

Cada celda muestra los permisos de lectura, escritura y ejecución que cada usuario tiene sobre cada archivo. Si un permiso no aparece, significa que el usuario no puede realizar esa acción.

Ejemplo 3

Sujetos: inc_ctr, dec_ctr, manage
Objeto: counter
Permisos: + = incrementar, - = decrementar, call = llamar a la función

	counter	inc_ctr	dec_ctr	manage
inc_ctr	+			
dec_ctr	-			
manage		call	call	call

Acá la matriz indica qué funciones pueden modificar o interactuar con la variable counter y con otras funciones. Por ejemplo, inc_ctr puede incrementar counter, mientras que manage puede llamar a otras funciones.

2.2 Implementaciones

Dos formas de implementar la matriz de control de accesos son:

2.2.1 Lista de Control de Accesos (ACL - Access Control List)

Cada objeto mantiene una lista que indica qué sujetos tienen qué permisos sobre él. Consiste en almacenar la matriz de control de accesos por columnas. La lista está asociada al objeto. Permite ver rápidamente quién puede hacer qué sobre un objeto específico. Es fácil revocar todos sus accesos, reemplazando su ACL por una vacía. Es fácil darlo de baja, borrando su ACL.
Surgue el siguiente problema : ¿Cómo chequear a que puede acceder un sujeto? La verificación de los permisos de un sujeto requiere revisar todas las ACL de los objetos relevantes. Si el número de objetos a chequear es chico, bárbaro, el tema es si son muchos.

2.2.2 Lista de Capacidades (Capabilities List)

Cada sujeto mantiene una lista que indica sobre qué objetos puede actuar y con qué permisos. Consiste en almacenar la matriz de control de accesos por filas. La lista está asociada al sujeto. Permite ver rápidamente todos los objetos a los que un sujeto tiene acceso. Es fácil revocar todos sus accesos, reemplazando su lista de capacidades por una vacía. Es fácil darlo de baja, borrando su lista de capacidades.
Surge el siguiente problema : ¿Cómo chequear quien puede acceder a un objeto? Cada lista está organizada por sujeto, por lo que verificar quién puede acceder a un objeto requiere recorrer todas las listas de todos los sujetos.

3 Técnicas de control de acceso

3.1 Introducción

Ya estuvimos hablando un poco sobre sujetos y objetos del sistema y cómo se definen y verifican los permisos. Ahora, queremos distinguir dos áreas bien definidas :

3.1.1 Control de acceso al sistema

El propósito del control de acceso al sistema es permitir el acceso sólo a aquellos usuarios autorizados (identidades verificadas). Para lograr esto último, el mecanismo es doble :

- **Identificación** : el usuario le indica al sistema quién es.
- **Autenticación** : el sistema verifica la identidad del usuario.

3.1.2 Control de acceso los recursos

El objetivo es establecer cuáles usuarios pueden acceder a los distintos recursos que provee el sistema y de qué manera pueden hacerlo. O sea, determinar el nivel de permisos y privilegios de cada usuario sobre la información, aplicaciones o servicios disponibles. Para lograrlo, sigue **dos** reglas :

- Los recursos deben ser accedidos sólo por los usuarios autorizados. Es decir, los que tienen permisos explícitos.
- Las acciones sobre los recursos debes ser las que están permitidas para esos usuarios,

3.2 Clasificación

Las técnicas de control de acceso generalmente se clasifican en

3.3 Control de acceso Discrecional (DAC)

El control de acceso discrecional (Discretionary Access Control - DAC) es una política de control de acceso **determinada por el dueño de un recurso** (un archivo, una impresora). Es discrecional porque **el dueño del recurso decide de manera arbitraria a quien le permite acceder al mismo y con que permisos** (derechos de acceso a usuarios o grupos de usuarios). En el sistema cada objeto debe tener un dueño. La política de acceso es determinada por el dueño del recurso. En teoría un objeto sin dueño no se encuentra protegido. Normalmente el dueño de un recurso es el usuario que lo crea.
Los permisos son derechos de acceso definidos como el par ordenado (S_i, O_j) donde S (subject) y O (object). Esto quiere decir que el sujeto S_i puede acceder al objeto O_j de acuerdo con los derechos (read, write, execute, etc.) otorgados.

Ventajas

Lo que tiene de bueno esta técnica de control de acceso es que es flexible y adaptable a muchos sistemas y aplicaciones. Ampliamente usado, especialmente en ambientes comerciales e industriales.

Problema

Discrecionalidad para la cesión de derechos. Es decir que como los usuarios propietarios pueden otorgar permisos a otros usuarios, existe el riesgo de que se den más permisos de los necesarios (principio de privilegios mínimos incumplido). O también que alguien con permisos pueda transmitirlos a terceros sin control. Y como consecuencia se facilite la propagación de accesos indebidos, aumentando las posibilidades de fuga de información.

3.3.1 Permisos básicos en UNIX

En sistemas basados en UNIX tradicionales, se implementa principalmente un modelo DAC. Donde cada **archivo o directorio** tiene :

- Propietario (user)
- Grupo (group)
- Permisos para otros (others)

Los permisos son de tipo :

- Read (**r**)
- Write (**w**)
- Execute (**x**)

Cada permiso tiene un valor numérico octal para representar las combinaciones. Existen varios tipos de archivos :

- **-** : archivo regular
- **d** : directory
- **l** : link
- **c** : character device
- **b** : block device
- **p** : pipe

- **s** : socket (archivo para comunicación entre procesos (IPC))

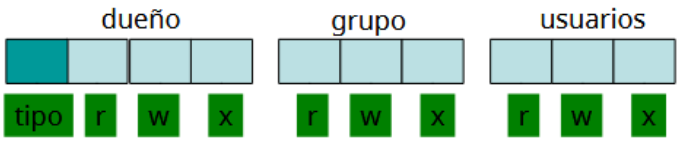
Si por ejemplo ejecutamos `ls -l` vamos a ver algo así :

```
crw-rw----  1 root tty       4,   1 ago 26 13:15 tty1
brw-rw----  1 root disk      8,   0 ago 26 13:15 sda
drwxr-xr-x  2 root root     4096 ago 26 13:15 pts
lrwxrwxrwx  1 root root        3 ago 26 13:15 cdrom -> sr0
prw-------  1 user user        0 ago 26 13:15 mypipe
srwxr-xr-x  1 user user        0 ago 26 13:15 mysocket
```

Donde lo que vemos es :

`[Tipo+Permisos] [Links] [Propietario] [Grupo] [Tamaño / mayor, menor] [Fecha] [Nombre]`

Donde para la primer parte distinguimos :



Permisos especiales

SETUID y **SETGID** son permisos de acceso que pueden asignarse a archivos o directorios en un sistema operativo basado en Unix. Se utilizan principalmente para permitir a los usuarios del sistema ejecutar binarios con privilegios elevados temporalmente para realizar una tarea específica.
Si un archivo tiene activado el bit "Setuid" se identifica con una "**s**" en un listado de la siguiente forma:

```
-rwsr- x r x  1 root shadow 27920 ago 15 22:45 usr/bin/passwd
```

Otros permisos en Linux

- **Chattr** (change attributes) : agrega atributos que van más allá de los permisos básicos rwx y permiten definir restricciones adicionales de cómo manipular un archivo :

  `chattr [opción] [archivo]`

  Donde las opciones son :

  - **+** : agregar
  - **-** : sacar
  - **=** : establece sólo el que pongas y elimina el resto.

  y los atributos que van dentro de esa opción son:

  - **i** : el archivo no puede ser modificado, borrado, renombrado ni linkeado.
  - **a (append only)** : solo se puede abrir en modo append (agregar datos al final).
  - **e** : archivo usa extent.
  - **A** : no actualiza el tiempo de último acceso (atime).
  - **S** : asegura que cada escritura se guarde inmediatamente en disco (synchronous).
  - **d** : el archivo no se guarda en backups al ejecutar `dump`

  Ejemplo

```
# Hacer inmutable un archivo
sudo chattr +i documento.txt

# Permitir solo escritura por append (no se puede truncar)
sudo chattr +a log.txt

# Quitar el atributo inmutable
sudo chattr -i documento.txt
```

  Si queremos ver los atributos, ejecutamos `lsattr` (list attributes)

```
----i--------e-- documento.txt
```

- **POSIX ACLs** (getfacl, setfacl) y **NFSv4 ACLs**. Flexibilizan las ACLs standard, posibilitando dar permisos a usuarios específicos, a más de un grupo, etc. Puede requerir montar el filesystem con la opción acl. Las POSIX ACLs son utilizadas en sistemas Unix tradicionales y los comandos principales son
  - getfacl [archivo] : muestra la ACL de un archivo o directorio.
  - setfacl : establece o modifica la ACL de un archivo o directorio.

  Ejemplo

```
# Dar permisos de lectura/escritura a usuario 'juan' sobre archivo.txt
setfacl -m u:juan:rw archivo.txt

# Dar permisos de lectura a grupo 'devs'
setfacl -m g:devs:r archivo.txt

# Ver ACL del archivo
getfacl archivo.txt
```

  Para el caso de NFSv4 ACLs son usadas en sistemas de archivos que soportan NFSv4, como servidores de red.

3.3.2 Control de acceso en Windows

Componentes de seguridad Windows

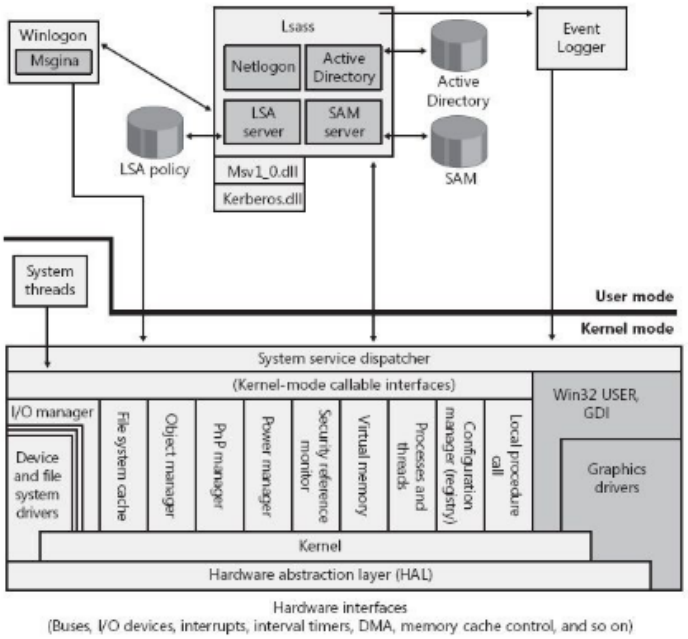


Figure 8-1 Windows security components

No voy a listar todos, pero sí los que el profe mencionó en la clase, ambos utilizan DAC como modelo de control de acceso.

- **Active dirctory** : Es un servicio de directorio utilizado en redes Windows (sobre todo en entornos empresariales) para administrar usuarios, grupos, equipos y políticas de seguridad. Su función principal es **centralizar la autenticación y autorización de los usuarios y permitir definir grupos de usuarios y políticas de acceso a recursos de la red.** Para funcionar hace lo siguiente, cuando un usuario intenta acceder a un recurso, el sistema consulta el AD para verificar si el usuario está autorizado. Se basa en listas de control de acceso (ACLs) para cada objeto, que indican qué usuarios o grupos tienen permisos específicos.

- **SAM** : Es una base de datos local que almacena las cuentas de usuario y los hashes de las contraseñas en computadoras Windows individuales que no son parte de un dominio. La función principal es administrar las cuentas locales del sistema y manejar autenticación local y definir los permisos básicos de los usuarios sobre archivos y recursos del sistema.

En los sistemas operativos Windows modernos (NT4 en adelante), este utiliza identificadores de seguridad (**SID**) que representan de manera única a usuarios, grupos y cuentas del sistema. Cada recurso del sistema (archivos, carpetas, impresoras, etc.) está asociado a una **DACL** (Discretionary Access Control List), que es una lista de entradas llamadas **ACEs** (Access Control Entries). Estas ACEs especifican los permisos permitidos o denegados para cada SID, proporcionando un control discrecional y granular sobre los accesos. Ejemplo de ACE

```
ace(
  denied,
  aceType(['ACCESS_DENIED_ACE_TYPE']),
  aceFlags,
  aceMask(['READ_CONTROL', 'SYNCHRONIZE']),
  sid('S-1-1-0')
).
```

Los componentes típicos de una ACE son :

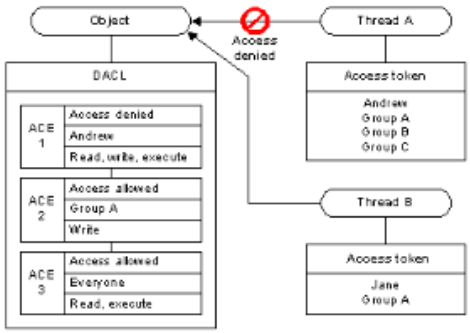
- **aceType** : de dos tipos, `ACCESS_ALLOWED_ACE_TYPE o ACCESS_DENIED_ACE_TYPE`
- **aceFlags** : opciones adicionales como *herencia, notificación de auditoría*
- **aceMask** : permisos concretos sobre el objeto como `READ_CONTROL (leer la info de seguridad del objeto) , SYNCHRONIZE (permitir sinco con eventos del objeto)`
- **sid** : el SID al que se aplica la ACE.

La idea es entonces

1. Cada objeto tiene su DACL.
2. La DACL contiene varias ACEs, que permiten o niegan permisos a SIDs específicos.
3. Cuando un usuario intenta acceder al objeto, Windows:
   - Obtiene el SID del usuario.
   - Recorre las ACEs de la DACL para ese SID.
   - Aplica los permisos permitidos y denegados.
4. El resultado final determina si el acceso se concede o se bloquea.

Ejemplo

Cuando varios threads intentan acceder a un recurso cómo Windows evalúa las ACEs :



La situación es, tenemos un objeto (archivo, carpeta, etc.) con una DACL que contiene varias ACEs:

- ACE 1 : deniega acceso a algunos usuarios.
- ACE 2 : permite acceso de escritura a otros.
- ACE 3 : permite acceso de lectura y ejecución a otros.

Dos hilos distintos intentan acceder al objeto: Thread A y Thread B.

1. Thread A : El sistema lee la DACL en orden. Encuentra ACE 1 que deniega el acceso. Como una ACE de denegación tiene prioridad, el sistema bloquea el acceso inmediatamente. ACE 2 y ACE 3 no se revisan, porque la negación ya se aplicó. Lo que sucede es que la primera coincidencia de denegación detiene la evaluación. Esto es importante para garantizar que las reglas de seguridad restrictivas se apliquen de inmediato.
2. Thread B : ACE 1 no se aplica al usuario de este hilo (por ejemplo, porque no pertenece al SID afectado). Entonces el sistema sigue evaluando la DACL donde, ACE 2 permite acceso de escritura entonces se concede ese permiso. ACE 3 permite lectura y ejecución, entonces también se concede.. Como resultado, obtiene permisos combinados: lectura, escritura y ejecución según las ACEs que aplican.

Windows evalúa las ACEs de arriba hacia abajo en la DACL. Una ACE de denegación detiene la evaluación si aplica al usuario. Si una ACE no aplica al SID del usuario, se continúa con la siguiente ACE.

3.4 Control de acceso Mandatorio (MAC)

El control de acceso mandatorio (Mandatory Access Control – MAC) es una **política de control de accesodeterminada por el sistema** no por el dueño de un recurso. Clasifica a sujetos y objetos en niveles de seguridad. Se utiliza en sistemas multinivel que procesan información altamente sensible. Un **sistema multinivel** es un sistema de computadoras que maneja múltiples niveles de clasificación entre sujetos y objetos, como por ejemplo información gubernamental o militar. Es decir, este sistema tiene políticas de seguridad obligatorias que clasifican tanto a los sujetos (usuarios o procesos) como a los objetos (archivos, carpetas, bases de datos) en diferentes niveles de seguridad. Cada sujeto tiene un **nivel de autorización** y cada objeto un **nivel de clasificación** (confidencial, secreto, ...)

3.5 Control de acceso basado en Roles (RBAC)

RBAC (Role Based Access Control) surge a fines de la década del 80 y toma un fuerte impulso en los 90. Combina aspectos de DAC y de MAC, pero con una visión más orientada a la estructura organizacional. Básicamente consiste en la creación de roles para los trabajos o funciones que se realizan en la organización. Los miembros del staff se asignan a roles y a través de estos roles adquieren permisos para ejecutar funciones del sistema. Los sujetos acceden a los objetos en base a las actividades que (los sujetos) llevan a cabo en el sistema. Es decir, considerando los **roles** (conjunto de acciones y responsabilidades asociadas con una actividad en particular) que ocupan en el sistema.

Implementación

Se necesitan mecanismos para:

- Identificar los roles en un sistema y asignar los sujetos a los roles definidos.
- Establecer los permisos de acceso a los objetos para cada rol.
- Establecer permisos a los sujetos para que puedan adoptar roles.

Características

- Administración de autorizaciones : la asignación de permisos a usuarios tiene **dos partes** :

  1. Asociar usuarios a roles.
  2. Asignar permisos para objetos a roles.

  Si un usuario cambia de tareas, sólo basta con cambiarle el rol. (Más sencillo que DAC o MAC).

- Jerarquía de roles : los roles también poseen relaciones de jerarquía. Pueden heredar privilegios y permisos de otros roles de menor jerarquía, simplificando la administración de las autorizaciones.
- Menor privilegio : permite implementar la política del menor privilegio posible (dar a cada usuario, proceso o programa únicamente los permisos que necesita para realizar su tarea, y nada más.Esto reduce la superficie de ataque y la posibilidad de errores accidentales o maliciosos.). Si una tarea no va a ser ejecutada por un usuario, entonces su rol no tendrá permisos para hacerla, de esta manera se minimizan riesgos de daños.
- Separación de responsabilidades : la división de resposabilidades se base en el principio de que ningún usuario tenga suficientes privilegios para usar el sistema en su propio beneficio. Por ejemplo, una persona encargada de autorizar un pago no debe ser el beneficiario de ese pago. Se puede establecer de dos maneras :

  1. **Estáticamente** : definiendo los roles excluyentes para un mismo usuario.
  2. **Dinámicamente** : realizando el control al momento de acceso.

Desventaja

Dificultad de establecer los roles y definir sus alcances.

Estándares de RBAC

- INCITS 359-2012 : American National Standard for Role-Based Access Control, define RBAC de forma estructurada, los conceptos, componentes y reglas que debe cumplir un sistema RBAC.
- eXtensible Acces Control Markup Language (XACML) : Un lenguaje basado en XML para describir políticas de control de acceso.
- XACML RBAC Profile : Es una extensión o perfil de XACML específicamente para implementar RBAC.

Algunas implementaciones

- Sun Solaris : Sistema operativo tipo Unix. Permite asignar roles administrativos con privilegios específicos.
- Microsoft Active Directory : Servicio de directorio de Windows para gestionar usuarios, grupos y recursos de red. Se crean roles y grupos de seguridad.
- Base de datos Oracle : Sistema de gestión de bases de datos relacional (RDBMS). Define roles que agrupan privilegios (lectura, escritura, administración).
- Base de Datos MS Sql Server : RDBMS de Microsoft. Roles predefinidos o personalizados con permisos sobre bases de datos, tablas, vistas y procedimientos.
- Aplicaciones de negocio : Software empresarial como ERP, CRM o sistemas financieros. Los roles determinan qué módulos, funcionalidades o datos puede ver o modificar un usuario.
- Servicios de nube : Plataformas como AWS, Azure o Google Cloud. Se usan roles y políticas de IAM (Identity and Access Management).
- Kubernetes : Plataforma para orquestar contenedores. Roles (Role y ClusterRole) definen permisos sobre recursos del clúster. Los usuarios o cuentas de servicio se asignan a roles mediante RoleBindings o ClusterRoleBindings.

3.6 Attribute-Based Access Control (ABAC)

Es un método de control de acceso donde las solicitudes de sujetos para realizar operaciones en objetos se otorgan o niegan en función de los atributos asignados del sujeto, los atributos asignados del objeto, las condiciones ambientales y un conjunto de políticas que se especifican en términos de esos atributos y condiciones.

Funcionamiento

Cuando un sujeto (usuario o proceso) intenta acceder a un objeto (archivo, servicio, recurso), el sistema evalúa:

1. Atributos del sujeto (quién solicita acceso) : nombre, cargo, departamento, seniority, nivel de autorización.
2. Atributos del objeto (sobre qué se solicita acceso) : tipo de recurso, clasificación de información, propietario, fecha de creación.
3. Atributos del ambiente (condiciones contextuales) : hora del día, ubicación geográfica, nivel de riesgo, estado del sistema.
4. Políticas de acceso : Son reglas que combinan los atributos anteriores para permitir o denegar el acceso. Ejemplo: “Solo los gerentes del departamento de finanzas pueden modificar archivos clasificados como confidenciales durante el horario laboral dentro de la oficina”.

3.7 Relationship-Based Access Control (ReBAC)

Zanzibar es el sistema de autorización centralizado y distribuido globalmente de Google. Funciona como un servicio unificado para gestionar permisos a gran escala a través de las diversas aplicaciones de Google, como Google Drive, YouTube y Google Cloud.
A diferencia de los modelos tradicionales basados en roles (RBAC), Zanzibar modela los permisos como relaciones entre objetos. Por ejemplo, en lugar de asignar un rol de ”editor” a un usuario, define una tupla de relación: (usuario, ”editor”, documento). Este enfoque ofrece una flexibilidad mucho mayor para definir políticas de acceso complejas y granulares.

Tupla de relación en Google Zanzibar

```
(usuario, relación, objeto)

(Alice, editor, Documento123)
```

Esta relación nos dice que Alice tiene permisos de edición sobre Documento123.