Conceptos de seguridad Windows.

La seguridad de Windows permite controlar el acceso a los pipes con nombre. Se lleva a cabo a través de lo que llamamos *Acces-Control Model.*

# Access-Control Model.

Básicamente se fundamenta en dos partes:

* *Access Token:* contiene información acerca de un usuario autenticado.
* *Security Descriptors*, contiene información sobre un *securable object* (ficheros, directorios, recursos compartidos, etc).

Al logearse un usuario, si la autenticación tiene éxito el sistema le asigna un token y todos los procesos ejecutados en esa sesión del usuario tienen una copia de ese token.

El token contiene el SID (Identificador de seguridad que identifica la cuenta y los grupos a los que pertenece). El token también contiene una lista de privilegios que tiene el usuario o sus grupos. El sistema usará este token para saber cuándo un proceso tiene acceso a un *securable object*, o cuándo puede realizar determinada acción del sistema que requiera de ciertos privilegios.

Cuando se crea un *securable object* el sistema le asigna un *security descriptor* que contiene la información de seguridad especificada por su creador, o información de seguridad por defecto si no se ha especificado nada.

El *Security Descriptor*  contiene la identificación del propietario del *securable object* y adicionalmente también puede contener las siguientes listas de control de acceso (ACL).

* *Discretionary Access control list (DACL)*: identifica los usuarios y cuyo acceso está permitido o denegado para el objeto en cuestión.
* *System Access Control List (SACL)*: controla la manera en la que el sistema audita los intentos de acceso al objeto.

En general, podemos decir que una ACL contiene un conjunto de reglas. Cada una especifica una cuenta de usuario, un grupo o una sesión y los accesos que le son adjudicados, denegados o auditados.

# Access Token

El token de acceso es un objeto que describe el contexto de seguridad de un proceso o hilo. Incluye la identidad y los privilegios que tiene la cuenta asociada con ese proceso o hilo.

Cuando el usuario inicia sesión, comprueba la información de seguridad (login y password) en la base de datos de seguridad y, si la comprobación tiene éxito, se crea un token de acceso. Cada proceso ejecutado por ese usuario tiene una copia de dicho token de acceso, de tal manera, que cuando el usuario accede a un *securable object* o intenta realizar una tarea del sistema, se utiliza dicho token para saber si el usuario está o no autorizado para realizar la acción.

El contenido del token es el siguiente

* El SID de la cuenta de usuario.
* El SID de los grupos de los que el usuario es miembro.
* Un SID que identifica el logon de la sesión actual.
* La lista de privilegios tanto del usuario, como de los grupos de los que es miembro
* SID de propietario.
* El SID del grupo primario.
* El DACL por defecto que utiliza el sistema a cuando el usuario crea un objeto *securable* sin especificar un descriptor de seguridad.
* El origen del Token.
* Si el token es de nivel primario o de nivel de *impersonation*
* Una lista opcional de SID restringidos.
* Niveles de impersonation actuales.
* Otros datos.

Todos los procesos tienen un token primario que describe el contexto de seguridad de la cuenta asociada, que es el que usa el sistema por defecto. Pero un hilo puede suplantar (impersonate) una cuenta de cliente, lo que daría acceso a dicho hilo interactuar con los *securable objects* autorizados para dicho cliente. Por esta razón un hilo que suplanta a un cliente, tiene los dos tokens el primario y el *impersonation token.*

# Funcionamiento del AccessCheck

Cuando el hilo intenta acceder a un *securable object*, el sistema autoriza o deniega el acceso. Si el objeto no tiene DACL, automáticamente el sistema lo autoriza, si no mira las entradas (ACE) del DACL y las aplica. Dichas ACEs especifican los derechos autorizados o denegados para un *trustee* que puede ser una cuenta de usuario, de grupo, o un logon de sesión.

## DACL.

El sistema compara los *trustees* identificados en el token de acceso del thread, con las ACEs del DACL del objeto de sesión. Y el resultado puede ser alguno de los siguientes:

* Una denegación explícita a alguno de los trustees del token, deniega el acceso.
* Una autorización a un *trustee* autoriza explícitamente los derechos requeridos.
* Si todas las entradas han sido evaluadas y no se ha encontrado ninguna regla que autorice explícitamente un acceso, éste se deniega.

El orden de las ACEs en la DACL es importante, porque si se encuentra una autorización o una denegación explícita, el sistema deja de evaluar el resto de las reglas.

El orden recomendado sería el siguiente:

The following steps describe the preferred order:

1. Todos los ACE explícitos, irán antes que los heredados.
2. Dentro de las ACE explícitas, las denegaciones irán antes que las autorizaciones.
3. Las ACES heredades, irán por orden de herencia, esto es, de más específico a más genérico.
4. Para el mismo nivel de herencia, las denegaciones irán antes que las autorizaciones.

# Seguridad en los pipes.

Podemos especificar un *security descriptor* para un pipe cuando lo creamos. Este descriptor controlaría el acceso tanto al cliente como al servidor que utiliza el pipe. Si no indicamos ningún *security descriptor* se asignará uno por defecto que básicamente otorga privilegios de control total a la cuenta *Local System*, a los administradores y al *creator owner.* También otorga permisos de lectura a los miembros del Grupo *Everyone (Todos)* y a la cuenta Anonymous.

Existe la posibilidad de recuperar y establecer el descriptor de seguridad de un pipe con nombre desde código.

Cuando un determinado hilo abre un Pipe, el sistema lleva a cabo la comprobación antes de recuperar el Handle del mismo. En esta comprobación se compara los derechos de acceso solicitados contra la DACL del descriptro del pipe. Además de los derechos de acceso el sistema tiene que otorgar el permiso FILE\_CREATE\_PIPE\_INSTANCE.

El handle retornado cuando se crea el Pipe siempre tiene el acceso de SYNCHRONIZE, además de GENERIC\_READ, GENERIC\_WRITE o ambos, dependiendo del modo del pipe (in/out)

FILE\_GENERIC\_READ de un pipe con nombre combina los derechos de lectura desde el pipe, lectura de los atributos del pipe, lectura de los atributos extendidos y lectura de la DACL del pipe.

FILE\_GENERIC\_WRITE de un pipe con nombre combina los derechos de escritura en el pipe, concatenación (append) al pipe, modificación de los atributos del pipe, modificación de los atributos extendidos y modificación de la DACL del pipe.

Para poder leer o escribir la SACL hay que solicitar el permiso ACCESS\_SYSTEM\_SECURITY.

Para evitar que los usuarios de una sesión diferente (por escritorio remoto), puedan acceder al pipe, hay que filtrar los derechos por SID en la DACL del pipe.

# Impersonation

Es la capacidad de los hilos de ejecutarse en un contexto de seguridad diferente del que tiene el proceso propietario del hilo.

Los niveles de impersonation son:

* SecurityAnonymous: El servidor no puede identificar ni “*impersonar”* el contexto de seguridad del cliente.
* SecurityIdentification: El servidor puede obtener la identidad y privilegios del cliente, pero no puede “*impersonar”* el contexto de seguridad del cliente.
* SecurityImpersonation: El servidor puede “*impersonar”* el contexto de seguridad del cliente en el contexto de usuario local
* SecurityDelegation: El servidor puede “*Impersonar”* el contexto de seguridad del cliente en sistemas remotos.

# Controlando la seguridad por código

## La clase pipe security.

Especifica los accesos de seguridad para un pipe y la forma en la que va a ser auditado.

Podemos, por tanto, agregar nuevas reglas de auditoría y nuevas reglas de seguridad (SACLs y DACL).

Al construir un pipe con nombre le agregaríamos una instancia de PipeSecurity.

## La clase AccessRule:

Representa una regla de control de acceso. A la hora de crearla se identifica la entidad a la que afectan los permisos, los permisos otorgados o denegados y si la regla es para conceder o denegar los permisos especificados.

### Ejemplo:

PipeSecurity ps = new PipeSecurity();

ps.AddAccessRule(new PipeAccessRule("pelado", PipeAccessRights.ReadWrite, System.Security.AccessControl.AccessControlType.Allow));