Windows Access Control List (ACL) 1

What do we have in this session?

1. Introduction
2. Access Control
3. Access Control Model
4. Access Control Components
5. Access Tokens
6. Access Rights for Access-Token Objects
7. Security Descriptors
8. Securable Objects

Abilities that supposed to be acquired in this session:

1. Able to understand the Windows access control.
2. Able to understand Windows Access Token, Securable Object and Security Descriptor.

Introduction

In this Module and that follows, we will try to learn how the security is implemented in Windows Operating Systems (OS). Access Control is one of the important and fundamental topics in Windows SDK Platform from Security category. We will start with the access control model used by Windows OSes and then dig deeper the details of every component in the model. On the way we will also be introduced with functions that available for manipulation and interaction with various objects of the Windows OS in the security aspect. The working program examples were presented starting on the third Module of this series. The program examples used were low level programming using C (Mix of the Microsoft C and standard C), still no Graphical User Interface (GUI) here, hoping that the student will understand better and faster. This also will provide the students with a very good practice on the aspects of how C programming is used in the specific implementation. All the required information for learning the security aspect of the Windows OSes has been included in this compilation note, in order to focus and avoid a lot of cross references that encountered in MSDN documentation. It is prepared by lazy teachers for Joe average and lazy students.

Access Control

It has been mentioned in MSDN documentation that at the beginning, Windows OSes followed (already) an obsolete Class C2 standard, formally known as Trusted Computer System

Evaluation Criteria (TCSEC) which superseded by Common Criteria and the ISO version is ISO 15408 Common Criteria for Information Technology Security Evaluation (Part 1, 2 and 3). Access control refers to security features that control **who can access which resources in the operating system**. Applications call access control functions to set who can access specific resources or control access to resources provided by the applications.

Access Control Model

The access control model enables you **to control the ability of a process to access securable objects or to perform various system administration tasks**. A process is a security context under which an application runs. Typically, the security context is **associated with a user**, so all applications running under a given process take on the permissions and privileges of the owning user.

Access Control Components

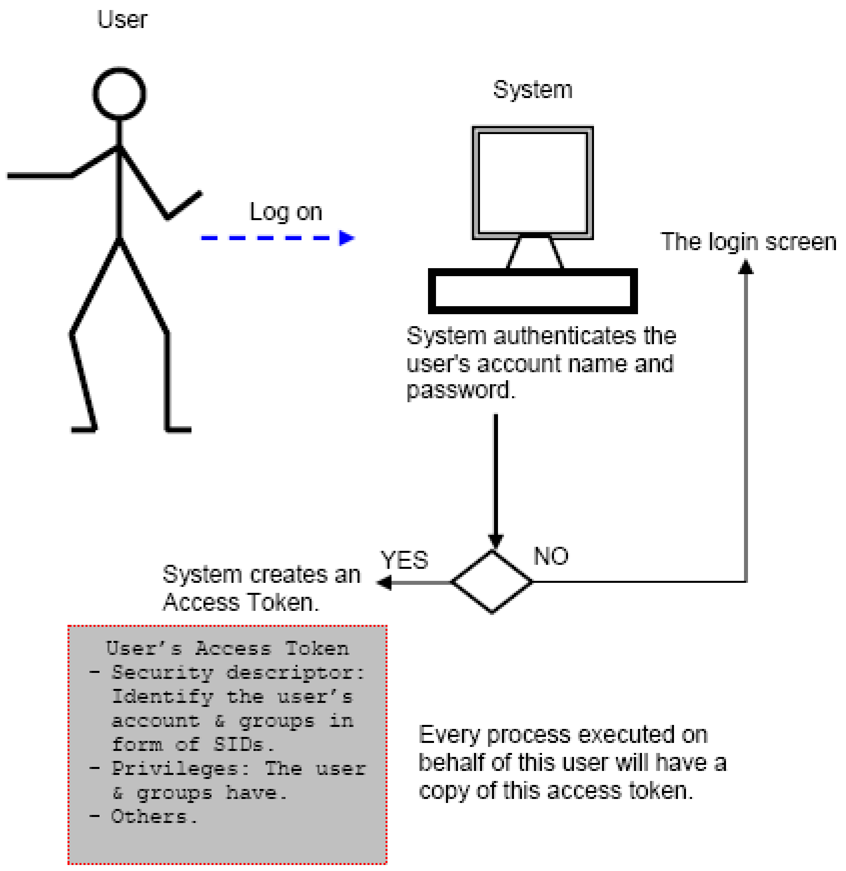
There are two basic components of the access control model:

1. **Access tokens**, which contain information about a **logged-on user**.
2. **Security descriptors**, which contain the security information that protects a securable object.

When a user logs on, the system authenticates the user's account name and password. If the logon is successful, the system creates an access token. Every process executed on behalf of this user will have a copy of this access token. The access token contains security identifiers (SID) that identify the user's account and any group accounts to which the user belongs. The token also contains a list of the privileges held by the user or the user's groups. The system uses this token to identify the associated user when a process tries to access a securable object or perform a system administration task that requires privileges.
When a securable object is created, the system assigns it a security descriptor that contains security information specified by its creator, or default security information if none is specified. Applications can use functions to retrieve and set the security information for an existing object.
A security descriptor identifies the object's owner and can also contain the following access control lists (ACLs):

1. A **discretionary access control list** (DACL) that identifies the users and groups allowed or denied access to the object.
2. A **system access control list** (SACL) that controls how the system audits attempts to access the object.

An ACL contains a list of access control entries (ACEs). Each ACE specifies a set of access rights and contains a security identifier that identifies a trustee for whom the rights are allowed, denied, or audited. A trustee can be a user account, group account, or logon session. A logon session begins whenever a user logs on to a computer. All processes in a logon session have the same primary access token. The access token contains information about the security context of the logon session, including the user's SID, the logon identifier, and the logon SID. The following illustration shows what will happen when a user log on to a system. Keep in mind that the user is nothing because user's credential such as his/her username and password was created in the system. User's credential just another Windows object.


User
System
Log on
The login screen
System authenticates the user's account name and password.
YES
NO
System creates an Access Token.
User's Access Token
- Security descriptor: Identify the user's account & groups in form of SIDs.
- Privileges: The user & groups have.
- Others.
Every process executed on behalf of this user will have a copy of this access token.

An access token is an object that describes the security context of a process or thread. The information in a token includes the identity and privileges of the user account associated with the process or thread. When a user logs on, the system verifies the user's password by comparing it with information stored in a security database. If the password is authenticated, the system produces an access token. Every process executed on behalf of this user has a copy of this access token.

The system uses an access token to identify user when a thread interacts with a securable object or tries to perform a system task that requires privileges. Access tokens contain the following information:

1. The SID for the user's account.
2. SIDs for the groups of which the user is a member.
3. A logon SID that identifies the current logon session.
4. A list of the privileges held by either the user or the user's groups.
5. An owner SID.
6. The SID for the primary group.
7. The default DACL that the system uses when the user creates a securable object without specifying a security descriptor.
8. The source of the access token.
9. Whether the token is a primary or impersonation token.
10. An optional list of restricting SIDs.
11. Current impersonation levels.
12. Other statistics.

A primary token is an **access token that is typically created only by the Windows kernel**. It may be assigned to a process to represent the default security information for that process. The **impersonation token** is an access token that has been created to capture the security information of a **client process**, allowing a server to "impersonate" the client process in security operations. Every process has a primary token that describes the security context of the user account associated with the process. By default, the system uses the primary token when a thread of the process interacts with a securable object. Moreover, a thread can impersonate a client account. Impersonation allows the thread to interact with securable objects using the client's security context. A thread that is impersonating a client has both a primary token and an impersonation token. You can use the OpenProcessToken() function to retrieve a handle to the primary token of a process.

Function	Description
AdjustTokenGroups()	Changes the group information in an access token.
AdjustTokenPrivileges()	Enables or disables the privileges in an access token. It does not grant new privileges or revoke existing ones.
CheckTokenMembership()	Determines whether a specified SID is enabled in a specified access token.
CreateRestrictedToken()	Creates a new token that is a restricted version of an existing token. The restricted token can have disabled SIDs, deleted privileges, and a list of restricted SIDs.

DuplicateToken()	Creates a new impersonation token that duplicates an existing token.
DuplicateTokenEx()	Creates a new primary token or impersonation token that duplicates an existing token.
GetTokenInformation()	Retrieves information about a token.
IsTokenRestricted()	Determines whether a token has a list of restricting SIDs.
OpenProcessToken()	Retrieves a handle to the primary access token for a process.
OpenThreadToken()	Retrieves a handle to the impersonation access token for a thread.
SetThreadToken()	Assigns or removes an impersonation token for a thread.
SetTokenInformation()	Changes a token's owner, primary group, or default DACL.

Table 1

Use the OpenThreadToken() function to retrieve a handle to the impersonation token of a thread. You can use the following functions to manipulate access tokens.
The access token functions use the following structures to describe the components of an access token.

Structure	Description
TOKEN_CONTROL	Information that identifies an access token.
TOKEN_DEFAULT_DACL	The default DACL that the system uses in the security descriptors of new objects created by a thread.
TOKEN_GROUPS	Specifies the SIDs and attributes of the group SIDs in an access token.
TOKEN_OWNER	The default owner SID for the security descriptors of new objects.
TOKEN_PRIMARY_GROUP	The default primary group SID for the security descriptors of new objects.
TOKEN_PRIVILEGES	The privileges associated with an access token. Also determines whether the privileges are enabled.
TOKEN_SOURCE	The source of an access token.
TOKEN_STATISTICS	Statistics associated with an access token.
TOKEN_USER	The SID of the user associated with an access token.

Table 2

The access token functions use the following enumeration types.

Enumeration type	Specifies
TOKEN_INFORMATION_CLASS	Identifies the type of information being set or retrieved from an access token.
TOKEN_TYPE	Identifies an access token as a primary or impersonation token.

Table 3

Access Rights for Access-Token Objects

An application cannot change the access control list of an object **unless the application has the rights to do so**. These rights are controlled by a security descriptor in the access token for the object. To get or set the security descriptor for an access token, call the GetKernelObjectSecurity() and SetKernelObjectSecurity() functions. When you call the OpenProcessToken() or OpenThreadToken() function to get a handle to an access token, the system checks the requested access rights against the DACL in the token's security descriptor. The following are valid access rights for access-token objects:

1. The DELETE, READ_CONTROL, WRITE_DAC, and WRITE_OWNER standard access rights. Access tokens do not support the SYNCHRONIZE standard access right.
2. The ACCESS_SYSTEM_SECURITY right to get or set the SACL in the object's security descriptor.

The specific access rights for access tokens, which are listed in the following table.

Value	Meaning
TOKEN_ADJUST_DEFAULT	Required to change the default owner, primary group, or DACL of an access token.
TOKEN_ADJUST_GROUPS	Required to adjust the attributes of the groups in an access token.
TOKEN_ADJUST_PRIVILEGES	Required to enable or disable the privileges in an access token.
TOKEN_ADJUST_SESSIONID	Required to adjust the session ID of an access token. The SE_TCB_NAME privilege is required.
TOKEN_ASSIGN_PRIMARY	Required to attach a primary token to a process. The SE_ASSIGNPRIMARYTOKEN_NAME privilege is also required to accomplish this task.
TOKEN_DUPLICATE	Required to duplicate an access token.
TOKEN_EXECUTE	Combines STANDARD_RIGHTS_EXECUTE and

	TOKEN_IMPERSONATE.
TOKEN_IMPERSONATE	Required to attach an impersonation access token to a process.
TOKEN_QUERY	Required to query an access token.
TOKEN_QUERY_SOURCE	Required to query the source of an access token.
TOKEN_READ	Combines STANDARD_RIGHTS_READ and TOKEN_QUERY.
TOKEN_WRITE	Combines STANDARD_RIGHTS_WRITE, TOKEN_ADJUST_PRIVILEGES, TOKEN_ADJUST_GROUPS, and TOKEN_ADJUST_DEFAULT.
TOKEN_ALL_ACCESS	Combines all possible access rights for a token.

Table 4

Security Descriptors

A security descriptor contains the security information associated with a securable object. A security descriptor consists of a SECURITY_DESCRIPTOR structure and its associated security information. A security descriptor can include the following security information:

1. SIDs for the owner and primary group of an object.
2. A DACL that specifies the access rights allowed or denied to particular users or groups.
3. A SACL that specifies the types of access attempts that generate audit records for the object.
4. A set of control bits that qualify the meaning of a security descriptor or its individual members.

The Windows API provides functions for setting and retrieving the security information in an object's security descriptor. In addition, there are functions for creating and initializing a security descriptor for a new object. Applications working with security descriptors on Active Directory objects can use the Windows security functions or the security interfaces provided by the Active Directory Service Interfaces (ADSI).

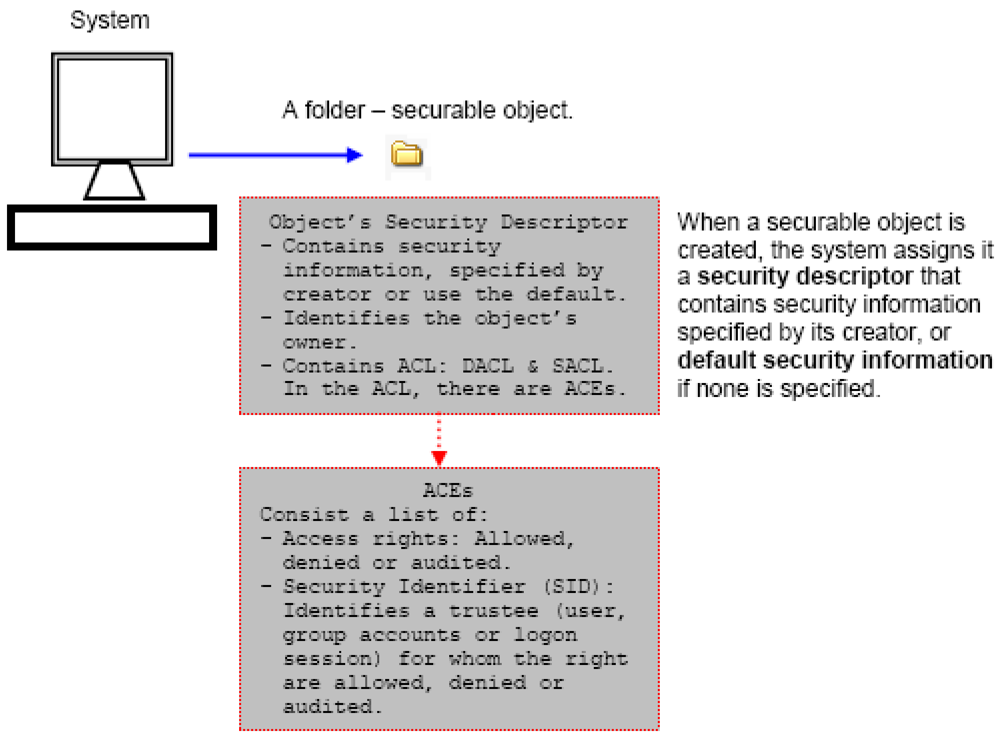
Securable Objects

A securable object is **an object that can have a security descriptor**. All named Windows objects are securable. Some unnamed objects, such as process and thread objects, can have security descriptors too. For most securable objects, you can specify an object's security descriptor in the function call that creates the object. For example, you can specify a security descriptor in the CreateFile() and CreateProcess() functions. In addition, the Windows security functions enable you to get and set the security information for securable objects created on operating systems other than Windows. The Windows security functions also provide support for using security descriptors with private, application-defined objects. Each type of securable object defines its own set of specific access rights and its own mapping of generic access rights. The following table shows the functions that can be used to manipulate the security information for some common securable objects.

Securable Object type	Security Descriptor Functions
Files or directories on an NTFS file system.	GetNamedSecurityInfo(), SetNamedSecurityInfo(), GetSecurityInfo(), SetSecurityInfo()
Named pipes, Anonymous pipes.	GetSecurityInfo(), SetSecurityInfo()
Processes, Threads.	GetSecurityInfo(), SetSecurityInfo()
File-mapping objects.	GetNamedSecurityInfo(), SetNamedSecurityInfo(), GetSecurityInfo(), SetSecurityInfo()
Access tokens.	SetKernelObjectSecurity(), GetKernelObjectSecurity()
Window-management objects (window stations and desktops).	GetSecurityInfo(), SetSecurityInfo()
Registry keys.	GetNamedSecurityInfo(), SetNamedSecurityInfo(), GetSecurityInfo(), SetSecurityInfo()
Windows services.	GetNamedSecurityInfo(), SetNamedSecurityInfo(), GetSecurityInfo(), SetSecurityInfo()
Local or remote printers.	GetNamedSecurityInfo(), SetNamedSecurityInfo(), GetSecurityInfo(), SetSecurityInfo()
Network shares.	GetNamedSecurityInfo(), SetNamedSecurityInfo(), GetSecurityInfo(), SetSecurityInfo()
Interprocess synchronization objects (events, mutexes, semaphores, and waitable timers).	GetNamedSecurityInfo(), SetNamedSecurityInfo(), GetSecurityInfo(), SetSecurityInfo()
Job objects.	GetNamedSecurityInfo(), SetNamedSecurityInfo(), GetSecurityInfo(), SetSecurityInfo()
Directory service objects.	These objects are handled by Active Directory Objects.

Table 5

The following figure shows a simple the relationship between securable object (a folder) and security descriptor.


System
A folder – securable object.
Object's Security Descriptor
- Contains security information, specified by creator or use the default.
- Identifies the object's owner.
- Contains ACL: DACL & SACL. In the ACL, there are ACEs.
ACEs
Consist a list of:
- Access rights: Allowed, denied or audited.
- Security Identifier (SID): Identifies a trustee (user, group accounts or logon session) for whom the right are allowed, denied or audited.
When a securable object is created, the system assigns it a **security descriptor** that contains security information specified by its creator, or **default security information** if none is specified.