Permissions and ACLs: Section 1 Transcript

Introduction

1/2

Welcome to the Permissions and ACLs module.

This module will discuss access control list and entries, granting and denying access to objects, defining permissions, and how inherited and explicit permissions are applied to copy and move procedures. You will also be given the opportunity to utilize command line tools to edit permissions and ACLs.

Throughout this module, you'll be presented with opportunities to assess and apply what you've learned.

At the end of this module, you will be able to:

- Describe Access Control Lists and Entries,
- Define NTFS permissions,
- Describe inherited and explicit permissions, and
- Demonstrate the ability to utilize command line tools to edit permissions and ACLs.

Bypass Exam Introduction

2/2

If you are already familiar with the subject matter presented in this module, you can choose to take a Bypass Exam to skip this module.

The Bypass Exam option provides a single opportunity to successfully demonstrate your competence with the material presented within the module. If you pass, you'll receive credit for completing the module, unlocking the content within, and you will be free to proceed to the next module. If you do not pass, you will need to successfully complete the module, including all exercises and the Module Exam, to receive credit.

Click the Next Section button to continue.

Permissions and ACLs: Section 2 Transcript

Access control list (ACLs)

1/7

Your computer is set up with a list of objects that you either are or are not allowed to access. An Access Control List (ACL) is the list of permissions attached to each object. An ACL specifies which user or system processes are granted access to objects, as well as what operations are allowed on given objects. Each entry in a typical ACL also specifies a subject and an operation.

For example, if a file has an ACL that contains the following (Alice: read, write; Bob: read), this means that Alice has permission to read and write the file and Bob only has permission to read it.

Type of ACLs

2/7

ACLs are contained within security descriptors which are data structures of security information for objects that can be identified by a unique name such as files, registry keys, processes, and application-defined.

There are two main types of ACLs contained within the security descriptors:

- Discretionary Access Control List (DACL) - Identifies the security principals that are allowed or denied access and the level of access being allowed or denied.
- System Access Control List (SACL) - Controls how object access will be audited.

An ACL is made up of a header and zero or more Access Control Entry (ACE) structures. ACEs may be explicitly applied to an object or inherited from a parent object. The order of ACEs in an ACL is important, with access-denied ACEs appearing higher in the order than ACEs that grant access.

Note that the ACL also refers to rules that are applied to port numbers or IP Addresses that are available on a host, each with a list of hosts and/or networks permitted to use the service.

DACLs

3/7

In a DACL, each ACE contains a security identifier (SID) and an access mask (and a set of flags, explained shortly).

Four types of ACEs can appear in a DACL: access allowed, access denied, allowed-object, and denied object.

- The access allowed ACE grants access to a user.
- The access denied ACE denies the access rights specified in the access mask.

The difference between allowed object and access allowed, and between denied-object and access denied, is that the object types are used only within the Active Directory.

ACEs

ACEs have a Globally Unique Identifier (GUID) field that indicates that the ACE applies only to particular objects or sub-objects (those that have GUID identifiers). Another optional GUID indicates what type of child object will inherit the ACE when a child is created within an Active Directory container that has the ACE applied to it.

The accumulation of access rights granted by individual ACEs forms the set of access rights that are granted by an ACL.

- If no DACL is present (a null DACL) in a security descriptor, everyone has full access to the object.
- If the DACL is empty (that is, it has 0 ACEs), no user has access to the object.
- The ACEs used in DACLs also have a set of flags that control and specify characteristics of the ACE related to inheritance.
- Object namespaces (which protects named objects from unauthorized access) have container objects and leaf objects, which are objects that have no child objects.
- A container can hold other container objects and leaf objects, which are its child objects.

Examples of containers are directories in the file system namespace and keys in the registry namespace.

Certain flags in an ACE control how the ACE propagates to child objects of the container associated with the ACE.

User Accounts

Any person who uses a computer or network service, namely the user, has a user account and is identified by a username.

A user account is a collection of information that tells Windows which files and folders you can access, what changes you can make to the computer, and your personal preferences, such as your desktop background or screen saver. User accounts let you share a computer with several people, while having your own files and settings. Each person accesses his or her user account with a username and password.

There are different types of user accounts:

- Local user accounts,
- Domain user accounts, and
- Built-in user accounts.

Let's take a closer look at these. Click each of the accounts to learn more.

Local user accounts - A local user account is created in a local security database and gives the user access to log onto the local computer. When an account is created, it exists only in the local security database on that computer and enables users to log on and access resources on a specific computer. The local user account resides in the Security Accounts Manager (SAM).

Domain user accounts - Domain user accounts enable users to log on to the domain to gain access to network resources. These accounts reside in an Active Directory. In a network environment, users need to access resources located anywhere on the network. To access these resources, you need to use a domain user account.

Built-in user accounts - Built-in user accounts enable users to perform administrative tasks or gain temporary access to network resources. They reside in SAM (local built-in user accounts) or reside in an Active Directory (domain built-in user accounts).

Groups

6/7

Access permission that is assigned at one time to a collection of user accounts is called a group. Rights and permissions are granted to the entire group versus to each user account. Once access has been provided to a group, appropriate users can be added. Members of a group can also be members of multiple groups.

To create a group, you can use the default or built in groups that Windows provides, or you can create a new group.

Section Completed

7/7

Permissions and ACLs: Section 3 Transcript

Permissions 1/12

Permissions define the type of access a user or group has to an object. An object is defined as an entity such as a file, folder, shared folder, or printer. The type of permissions that you can assign to a user depends on the type of object.

Permissions that are assigned to a user for one of the objects are called object permissions. You can assign permissions for objects in the Active Directory or on a local computer.

You can also assign permissions to a group of users instead of to individual users. Using groups in this way eases the task of managing permissions on objects.

For example, you might have access to a document on a shared folder on a network. Even though you can read the document, you might not have permission to make changes to it. System administrators and people with administrator accounts on computers can assign permissions to individual users or groups.

NTFS Permissions 2/12

There are three primary file types: file, folder, and share.

File and folder permissions define the file or folder access types (such as read, modify, create, delete, and so on) that you grant to a user, group, or computer.

- If you are a member of a security group that is associated with a file or folder, you have some ability to manage the permissions on that object.
- For those objects you own, you have full control.
- You can manage different types of objects through methods such as Active Directory, Group Policy, and access control lists.

Most file systems have methods to assign permissions or access rights to specific users and groups of users. These systems control the ability of the users to view or make changes to the contents of the file system. Permissions determine whether you can access an object and what you can do with it.

NTFS File Permissions 3/12

Permissions are assigned to a file from the Security tab of the Properties dialog box for the file. You can also view the current permissions to the file on the same tab.

The security tab consists of two sections: name and permissions.

The name section displays a list of existing users or groups who have permissions to the file.

The permissions section displays a list of permissions that you can grant or deny to the user or group. You normally select the permissions that you want to grant. In some cases, it may be easier to specify the permissions you want to deny. File permissions include read, write, read and execute, modify, and full control. Folder permissions include read, write, list folder contents, read and execute, modify, and full control.

For example: Although you allow everyone access to a file, you may need to restrict users who connect to the resource using the Guest account. To do this, you deny permissions to the Guest account.

NTFS shared permissions

When multiple users have access to the same folder, the folder must be shared.

Let's take a look at the characteristics of shared folder permissions:

- To provide multiple users with access to the same folder, you must share the folder.
- After a folder is shared, users can access all of the files and subfolders within the shared folder if they have been granted permission.
- You can only share folders, not individual files. If multiple users need access to the same file, enclose the file in a folder and then share the folder.

Shared folders are usually placed on a file server, but can also be located on any computer on the network. You can store files in shared folders according to categories or functions. For example: You can place shared data files in one shared folder and shared application files in another.

Let's take a look at some of the characteristics of shared folders:

- A shared folder appears in Windows Explorer with an icon of a hand holding the folder.
- Permissions are assigned to the entire folder only, not to individual files or subfolders within the shared folder.
- When a folder is shared, the Full Control permission is assigned to the Everyone group as the default permission.
- When a user is added to a shared folder, the user receives the Read permission by default.
- When a shared folder is copied, the original shared folder is still shared, but the copy is not shared.
- When a shared folder is moved to another location, the folder is no longer shared.
- You can control the level of access to a shared folder by assigning permissions to it.
- Shared folder permissions are assigned to users and groups from the Permissions dialog box of the shared folder. You can view existing shared folder permissions in the dialog box as well.

Multiple NTFS permissions

Let's take a look at the characteristics of multiple NTFS permissions.

NTFS permissions are cumulative. Users can have permissions to both their individual user account and to another group. In this case, multiple permissions have been granted to the user.

For example: If a user has the read permission for a folder and is a member of a group with the write permissions for the same folder, then the user has both read and write permissions for that folder.

File Permissions are separate from folder permissions. NTFS file permissions can take priority over, or override, NTFS folder permissions.

For example: A user with the write permissions for a file will be able to make changes to the file even if he or she has only read permissions for the folder containing the file.

Deny permission. The deny permission overrides other permissions. This is an exception to the cumulative rule. Granting the deny permission to the user account or group denies access even if the user has permission to access the file or folder as a member of a group; denying permission to the user blocks any other permission that the user has.

Note there is a difference between a user not having access, and specifically denying a user access by adding a deny entry to the ACL for the file or folder. Even though a user may not be explicitly granted access to a file, that user may still have access by virtue of a group membership.

So, not granting access does not guarantee that access was not granted.

NTFS Permissions Inheritance

When you grant NTFS permissions to give access to a folder, you grant permissions for the folder, for any existing files and subfolders, and for any new files and subfolders that are created in the folder. This is called inheritance.

If you want folders or files to have different permissions than their parent folder, you can prevent permissions inheritance by removing the inherited permissions and retain only the permissions that were explicitly granted.

When you do this,

- The subfolder for which you prevent permission inheritance from its parent folder now becomes a new parent folder, and
- The subfolders and files that are contained within this new parent folder inherit the permissions granted for its parent folder.

One important takeaway is that access control entries with explicit permissions take precedence over inherited permissions, even DENY which would otherwise take precedence if both permissions were explicit.

Copying and moving files and folders

When you copy or move a file or folder to a different NTFS partition, the file or folder inherits permissions from the destination folder.

When you copy or move a file or folder within an NTFS partition, the file or folder inherits permissions from the new parent folder.

When moving a file or folder that has explicitly assigned permissions, the permissions are retained in addition to the newly inherited permissions, only when the MOVE operation on the file or folder occurs within the same partition.

Principle of Least Privilege (PoLP) 8/12

It's good practice for system administrators to use accounts with restrictive permissions when performing routine, non-administrative tasks, such as running applications and viewing web-based material. In other words, administrators should use accounts that only have the privileges required to complete the task, and no others. This practice is known as "principle of least privilege."

When administrative permissions are needed, an administrator can use an account which is assigned a more broad range of permissions. One way of accomplishing this is to:

- Use the switch user function to invoke the Windows login screen
- Login to the higher-permissioned account
- Complete the administrative tasks
- Logout of the higher-permissioned account, and finally
- Log back into the system via the "regular" account.

Understandably, this process can become a bit cumbersome.

RunAs 9/12

"RunAs" is a command that allows administrators to attain those same administrative privileges in a much more efficient manner. More specifically, if a user knows the login name and credentials for an account with escalated permissions, they can use the Runas command to run specific commands and programs with different permissions than what their current login provides, without having to logout of their current account.

Using the RunAs command 10/12

In addition, there may be situations where a user will have access to a domain network environment although they do not have an account on that domain. In such instances, the user can leverage the RunAs command to run local applications or commands as a domain user on a non-domain machine.

RunAs command parameters 11/12

There are several useful parameters associated with the RunAs command which can be used to manipulate and control how users are authenticated and the environment in which the applications and commands are run. Displayed are a few examples of the most commonly used parameters.

Additional information pertaining to the RunAs command can be found by using the "/?" parameter.

Knowledge Check Introduction 12/12

It is time for a Knowledge Check. This Knowledge Check will not be scored, but may indicate areas that you need to review prior to the Module Exam.

Permissions and ACLs: Section 4 Transcript

Summary

You have completed the Permissions and ACLs module.

This module discussed access control list and entries, granting and denying access to objects, defining permissions, how inherited and explicit permissions are applied to copy and move procedures, and utilized command line tools to edit permissions and ACLs.

You should now be able to:

- Describe Access Control Lists and Entries,
- Define NTFS permissions,
- Describe inherited and explicit permissions, and
- Demonstrate the ability to utilize command line tools to edit permissions and ACLs.

To receive credit and advance to the next module, you must achieve a passing score on the Module Exam.

Click the Next Section button to begin the Module Exam.