

NTFS Security Considerations

by: Mohammad Heidari, 06/21/2005

<http://www.securitydocs.com/library/3396>

Abstract

The New Technology File System was introduced with Windows NT to address security problems. With NTFS, files, directories, and windows can each have their own security. This allows a great deal of flexibility in setting up a network. Microsoft recommends that all network shared be established using the NTFS file system.

This paper has its origins in two events - a spate of papers that compare FAT and NTFS and a personal attempt to describe the security points of NTFS.

Genesis

New Technology File System (NTFS)

The FAT family of file systems, including FAT12, FAT16 and FAT32, has been the file systems underlying Microsoft operating systems in the early 1980s. On the whole, it does an adequate job of managing the files on a typical PC, especially for machines that are not subjected to demanding use. For home PC users, and for the typical PC in a small business, FAT is generally "good enough". To state my conclusions at the outset: FAT may be an admirable file system for typical PCs but it is not adequate for using in somewhere that security and reliability are important.

However, while FAT is acceptable in many cases, it is also a very old, limited and relatively simplistic file system. It has low level of security, capacity and reliability features that are necessary for advanced users, and especially, servers and workstations in a corporate networking environment. For this reason, Microsoft created the New Technology File System, or NTFS. The goals behind NTFS were to provide a flexible, adaptable, high-security and high-reliability file system, to help the position of Windows NT as a "serious" operating system for business and corporate users.

In the early 1990s, Microsoft set out to create a high-quality, high-performance reliable and secure operating system. Microsoft's operating systems were MS-DOS and Windows 3.x, neither of which had the power or features needed for Microsoft to take on UNIX or other "serious" operating systems. One of the biggest weaknesses of MS-DOS and Windows 3.x was that they relied on the FAT file system. FAT provided few of the features needed for data storage and management in a high-end, networked, corporate environment. To avoid crippling Windows NT, Microsoft had to create for it a new file system that was not based on FAT. The result was the New Technology File System or NTFS.

NTFS was designed to meet a number of specific goals. Some of these advantages are:

- **Reliability:** One important characteristic of a "serious" file system is that it must be able to recover from problems without data loss resulting. NTFS implements specific features to allow important transactions to be completed as an integral whole, to avoid data loss, and to improve fault tolerance.
- **Security and Access Control:** A major weakness of the FAT file system is that it includes no built-in facilities for controlling access to folders or files on a hard disk. Without this control, it is nearly impossible to implement applications and networks that require security and the ability to manage who can read or write various data.
- **Breaking Size Barriers:** In the early 1990s, FAT was limited to the FAT16 version of the file system, which only allowed partitions up to 4 Giga Bytes in size. NTFS was designed to allow very large partition sizes, in anticipation of growing hard disk capacities, as well as the use of RAID arrays.
- **Storage Capability:** When NTFS was developed, at around that time most PCs used FAT16, which results in significant disk space due to slack. NTFS avoids this problem by using a very different method of allocating space to files than FAT does.
- **Long File Names:** NTFS allows file names to be up to 255 characters, instead of the 8+3 character limitation of conventional FAT.

- **Networking:** Although some of the NT features that allow networking are not strictly related to the file system, though some are, some of the most important features of networking are based on NTFS. When Windows NT was developed, businesses were just beginning to understand the importance of networking, and Windows NT was given some facilities to enable networking on a larger scale.

There are also some disadvantages associated with NTFS, that Microsoft has made changes to it, for many different reasons. These include correcting problems with the file system, adding support for newer hardware, and enabling new operating system features. The biggest change to NTFS came with the introduction to the market of Windows 2000. NTFS changes enable some of the most important features of that operating system. The two versions of NTFS that are commonly used on PCs are NTFS 1.1 which is also called NTFS 4.0 (since it is most commonly used with Windows NT version 4) and the newest NTFS version, NTFS 5.0, which is an integral part of Windows 2000.

General Concepts of NTFS Security

Security under Windows NT and 2000 in general is one of the most important aspects of these operating systems. Security, including controlling access to the system and its various resources, is a subject that gets a lot of attention in any NT or 2000 system. Managing security issues such as user accounts and groups is a big part of the job of any Windows NT or 2000 system administrator. Security in NTFS is oriented around the key concept of assigning rights to specific users or groups of users. Consider a network with a Windows NT or windows 2000 Server, consisting of a server and various client machines. Any user who sits down at one of these client machines can connect to the server computer, but he or she must log in to the server in order to access any of its resources, including NTFS volumes it contains. In fact, the same applies to someone who uses the server machine directly, again, assuming it has been correctly configured. The administrator of the server sets up user accounts for everyone who will use the network. Someone who does not have a user account on the network may be allowed to use a guest account, but the rights assigned to such an account are generally quite minimal, for obvious reasons. If someone does not have even the guest account password, she or he can do nothing on the server. The access rights for files and directories on NTFS volumes are assigned based on these same user or group accounts.

Another key aspect of NTFS security is NTFS permissions .For instance suppose that a small company has 20 employees with a server that contains a variety of data. There may be a folder on the D: drive on this server called "D:Budget", which contains budgeting information for the company. This is sensitive data, which is only supposed to be accessible to the President and Vice-President of the company, and their Administrative Assistant. Under NTFS, this is easy to set up by assigning specific permissions to that folder for only those persons' accounts. All others in the company can be easily blocked from the folder entirely.

Permission

One of the most important advantages that you obtain when you choosing the NTFS file system is accurate control of the operation that are implemented on various data within the file system. FAT was designed in the era of single-user PCs, and contains virtually no built-in security or access management features. NTFS offers a secure environment and flexible control over what can be accessed by which users, to allow for many different users and groups of users to be transferred data together and everyone only can access to the appropriate data.

NTFS security and permission are linked to features and aspects of the operating system. A full discussion of Windows NT or Windows 2000 domains, directory services, groups, login procedures and so on is far beyond the scope of our coverage of NTFS. Therefore, I am attempting to limit myself to a description of how security works within NTFS itself.

Permission Groups

At the following I decide to illustrate the tables of Permissions Groups in the standard form for Windows NT and Windows 2000.Windows NT provides a set of six individual permissions for controlling access to files and folders. Windows 2000 refines

these individual permissions even further, into a set of over a dozen different permission components.

Standard Permission Groups for Windows NT

Standard Permission Group	Object Types Affected	Permission Types Granted (Applies Only To Appropriate Object Types)						Description
		Read (R)	Write (W)	Execute (X)	Delete (D)	Change Permissions (P)	Take Ownership (O)	
No Access	Folders or Files							Denies all access to the file or folder. The user can see the name of the object, but cannot do anything with it.
List	Folders Only	✓		✓				Users can see the list of files in the folder and traverse subfolders, but cannot view or execute files.
Read	Folders or Files	✓		✓				Users can read files and folders, execute files and traverse folders, but cannot change anything.
Add	Folders Only		✓	✓				Users can add files or subfolders to the folder, and can traverse subfolders, but cannot read or execute files.
Add & Read	Folders Only	✓	✓	✓				Users can add files or subfolders to the folder, and can read and execute files in the folder as well.
Change	Folders or Files	✓	✓	✓	✓			The user can read, write, execute or delete the file, or if applied to a folder, the files and subfolders within the folder. Note that this does <i>not</i> grant access to delete the folder itself. The user also cannot change permissions on the file or folder, or take ownership of it.
Full Control	Folders or Files	✓	✓	✓	✓	✓	✓	All permissions are granted. This also includes the special permission "Delete Subfolders and Files", which can only be given through the "Full Control" group under Windows NT.

Standard Permission Groups for Windows 2000

Permission Components (Windows 2000 and Windows NT 4.0 SCM)	Standard Permission Groups (Windows 2000 and Windows NT 4.0 SCM)					
	Read	Write	List Folder Contents	Read and Execute	Modify	Full Control
Traverse Folder / Execute File			✓	✓	✓	✓
List Folder / Read Data	✓		✓	✓	✓	✓
Read Attributes	✓		✓	✓	✓	✓
Read Extended Attributes	✓		✓	✓	✓	✓
Create Files / Write Data		✓			✓	✓
Create Folders / Append Data		✓			✓	✓
Write Attributes		✓			✓	✓
Write Extended Attributes		✓			✓	✓
Delete Subfolders and Files						✓
Delete					✓	✓
Read Permissions	✓	✓	✓	✓	✓	✓
Change Permissions						✓
Take Ownership						✓

Under the more advanced Windows 2000 scheme, there are 13 different permission components, which are collected into six different standard groups.

"List Folder Contents" and "Read and Execute" have the same permission components, which is a bit confusing. The differences between them have to do with how NTFS handles inheritance of these permissions. "List Folder Contents" is only used for folders and is not inherited by files within the folder. "Read and execute" applies to folders and files and is inherited by both. Also, 14th permission component, "Synchronize", is a member of all of the groups above.

You may notice, in looking at this table, which the "No Access" group is missing under the Windows 2000 scheme. In Windows NT, all permission groups except "No Access" provide "positive access"--saying, in effect, "you are allowed" to do something. "No Access" is the only one that says "you are not allowed" to do something. It really says "you cannot do anything". This inflexibility was corrected under Windows 2000 by giving users the ability to allow or disallow any permission group or individual permission. Under this setup, "No Access" simply isn't required.

I have tried to organize the rest of the material around the issues of: **Object ownership, Permission inheritance, auditing.**

Object Ownership

Every object within the NTFS volume has an owner, which is a user identified by the object as being the one who controls it. By default, the user who creates a file or folder becomes its owner. The significance of ownership is that the owner of a file or folder always has the ability to assign permissions for that object. The owner can decide what permissions should be applied to the object, and the others how can access to the files and folder. In the other words permissions work in combination with another key NTFS security concept that is called Ownership.

The two special permissions that are associated with ownership and permission assignment are "Change Permissions" (P) and "Take Ownership" (O). If a user is granted the "Change Permissions" permission, the user can change the permission settings for the object even if he or she does not own it. If a user has "Take Ownership" permission, the user has the ability to take over ownership of the resource.

Deciding how to assign permissions to various files and folders is an important system administration task. Very careful thought needs to go into how user groups are created and permissions assigned to various objects. One common mistake that many administrators make is misusing the "No Access" permission group under Windows NT. If used incorrectly, this can lock everyone out of large areas of an NTFS volume. Problems can also occur if different users take ownership of files or change permissions when they should not--this is in fact the reason for the distinction between the "Full Control" permission group, and the slightly more restricted groups "Change" or "Modify". One should be very careful when granting "Full Control" permission. Note that by default, members of the "Administrators" user group can always take ownership of any file and folder or change permissions on any file or folder. This allows administrators to fix permission problems if they occur.

I am not going to delve into the details on exactly how permissions are set. However, I think it's important to highlight that the - Windows NT and 2000 - permissions models work differently.

Under Windows NT, there is only one kind of permission assignment possible. Generally speaking, you can only "allow" users to do things. For example, you can allow someone read permission on a folder. By not granting the write permission as well, the system infers that the person may not write the folder. However, there is no way to explicit say "no write permission on this folder for this user". The difference is important, because it has implications in multiple-level hierarchies of folders. The only way to disallow access to something in the Windows NT NTFS security model is to use the "No Access" permission group. Windows 2000 greatly improved the control you have in assigning NTFS permissions by creating two explicit settings for each permission and permission group: allow and deny. Using these setting types, it is easy to specifically allow read access and deny write access to a file or folder.

Permission Inheritance

Before Windows 2000, we face with static Permission inheritance, but after that the Permission inheritance divided into two separate parts: Static Permission inheritance and dynamic one. But what is the permission inheritance indeed?

When you are using Windows NT and create a new subfolder or file within a folder, the new object is given a default set of permissions by copying the current set of permissions associated with the object's parent folder. This is called permission *inheritance*, or sometimes, *propagation*. Under NT's inheritance model, this only happens once, at the time the object is created. For this reason, conventional inheritance under NT is also called static permission inheritance, to distinguish it from the dynamic inheritance used by Windows 2000.

Static inheritance creates serious difficulties for administrators who need to manage large directory structures. Imagine a large tree structure of folders. Under static inheritance, after any subfolder is created, its permissions are no longer linked to those of the parent object. This makes it easy for any grouping or "branch" of the tree to have its permissions changed after the fact. Problems are particularly occur if the "Full Control" permission group has been used, as this means users are free to play around with the permissions on parts of the directory structure. Also, the static inheritance makes it very difficult to add new

permissions to an existing structure. Suppose you create a new user group and want to give everyone in that group access to an existing set of directories: how do you do it? To address these concerns, Windows NT provides a special feature when permissions are being assigned. If you select the "Replace Permissions on Subdirectories" and "Replace Permissions on Existing Files" options when changing the permissions of a folder, NT will reset the permissions of all child objects to match those of the parent. So if you add a new user group and want to give it access to the existing structure, you can use these options to force NT to propagate the new permissions down the directory tree, giving the new user group access to every folder and file.

As mentioned earlier the static permission inheritance method used by Windows NT on NTFS volumes addresses some of the concerns involved in managing large directory structures, but also has some very serious weaknesses. It does not allow an administrator to easily customize the permissions of branches of a directory tree while also allowing the administrator to assign new permissions to an entire existing structure. To correct some of the problems with the static permission inheritance system, Microsoft replaced it with a dynamic permission inheritance system in Windows 2000.

In the dynamic permission inheritance when you create a subfolder or file in a Windows 2000 folder, the child object inherits the parent's permissions, but remains linked to the parent. Furthermore, the parent's permissions are stored separately from any permission that is manually set on the child object. This dynamic linking method solves the two biggest problems with the static inheritance model. First, any changes to the parent folder are automatically inherited by the child objects. Second, any changes that were made to the child object are not destroyed by this automatic propagation. Under dynamic inheritance, an administrator or user is able to manage a hierarchical tree of permissions that matches the hierarchical tree of directories. Since each child inherits permissions from its parent, when you set up a hierarchy of three or more levels of folders, the objects deep within the structure will inherit permissions from their parent, "grandparent", "great grand-parent" and so on. This is called recursion.

Auditing

The biggest part of NTFS file system security revolves around controlling access to different types of objects. Obviously, it is quite important to deal with security in the present: managing what users are doing and ensuring that access is correct for various files and folders. However, there's another important aspect to security that also deserves attention: keeping records of the past. There are many situations where it is essential for system administrators to be able to not only manage what security happenings are occurring immediately, but what they have been in recent days as well. To allow administrators and managers this capability, NTFS includes a feature called auditing. When auditing is enabled, the system can be set to keep track of certain events. When any of these events occur, the system will make an entry in a special auditing log file that can be read by administrators or others with the appropriate permission level. Each entry will indicate the type of event, the date and time that it occurred, which user triggered the event, and other relevant information. Auditing within NTFS is really just a small part of the various auditing features offered by the Windows NT and Windows 2000 operating systems. These tools allow administrators to keep track of everything from logins, to the use of printers, to system errors. Within NTFS, auditable events are generally accesses of various types, roughly corresponding to the different types of permissions. Auditing can be selected for files and for folders, and can be selected for individual objects or hierarchies of folders, just like permissions can.