NTFSSecurity Tutorial 2 - Managing NTFS Inheritance and Using Privileges

12/05/2014 • 7 minutes to read

In this article

Summary

In my previous post, NTFSSecurity Tutorial 1 - Getting, adding and removing permissions, I talked about NTFS inheritance. Inheritance is a fundamental feature of NTFS to keep permissions consistent and easy to manage.

However, there are some scenarios where you want to disable inheritance on folders or find out where it has been disabled. This post explains how this can be achieved by using the NTFSSecurity module.

Installation

You can download NTFSSecurity on the TechNet Script Gallery: https://gallery.technet.microsoft.com/scriptcenter/1abd77a5-9c0b-4a2b-acef-90dbb2b84e85. Please unblock the file before extracting it. More information about installing PowerShell modules can be found here: Hey, Scripting Guy! How Can I Install Windows PowerShell Modules on Multiple Users' Computers?

Managing NTFS Inheritance

Determine Inheritance Settings

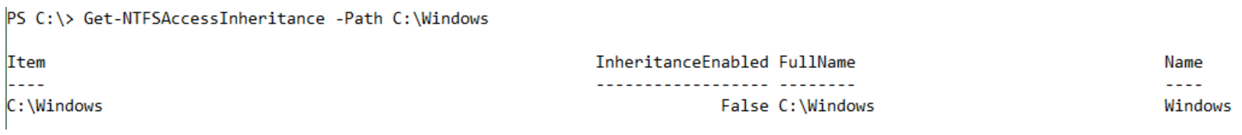
To determine if a file or folder inherits from its parent, use the **Get-NTFSAccessInheritance** cmdlet (there is also a **Get-NTFSAuditInheritance** cmdlet).

There are two ways to specify the file or folder: You can use the Path parameter or pipe the file or folder object to **Get-NTFSAccessInheritance**:

Get-NTFSAccessInheritance -Path C:\Windows

Get-Item C:\Windows | Get-Inheritance

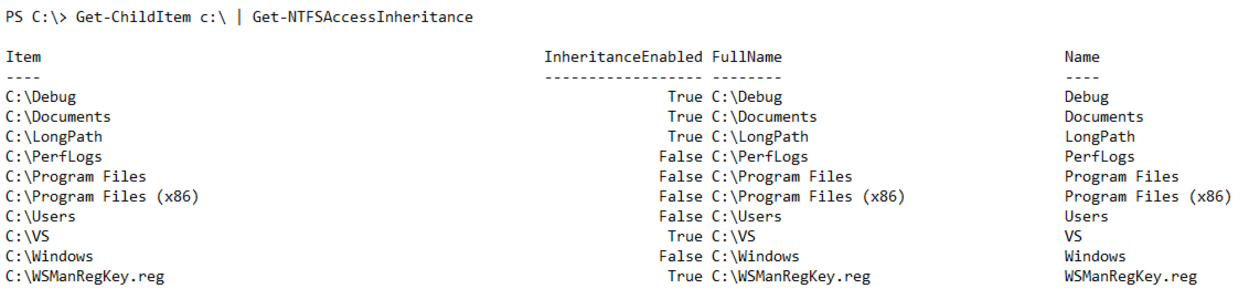
The output might looks like this:



```
PS C:\> Get-NTFSAccessInheritance -Path C:\Windows

Item                                                InheritanceEnabled FullName                                 Name
----                                                ------------------ --------                                 ----
C:\Windows                                                       False C:\Windows                               Windows
```

You can easily get the information for a bunch of items by using the built-in Get-ChildItem cmdlet, and then pipe all the items to Get-NTFSAccessInheritance:

Get-ChildItem c:\ | Get-NTFSAccessInheritance



```
PS C:\> Get-ChildItem c:\ | Get-NTFSAccessInheritance

Item                                                         InheritanceEnabled FullName                                         Name
----                                                         ------------------ --------                                         ----
C:\Debug                                                                   True C:\Debug                                         Debug
C:\Documents                                                               True C:\Documents                                     Documents
C:\LongPath                                                                True C:\LongPath                                      LongPath
C:\PerfLogs                                                               False C:\PerfLogs                                      PerfLogs
C:\Program Files                                                          False C:\Program Files                                 Program Files
C:\Program Files (x86)                                                    False C:\Program Files (x86)                           Program Files (x86)
C:\Users                                                                  False C:\Users                                         Users
C:\VS                                                                      True C:\VS                                            VS
C:\Windows                                                                False C:\Windows                                       Windows
C:\WSManRegKey.reg                                                         True C:\WSManRegKey.reg                               WSManRegKey.reg
```

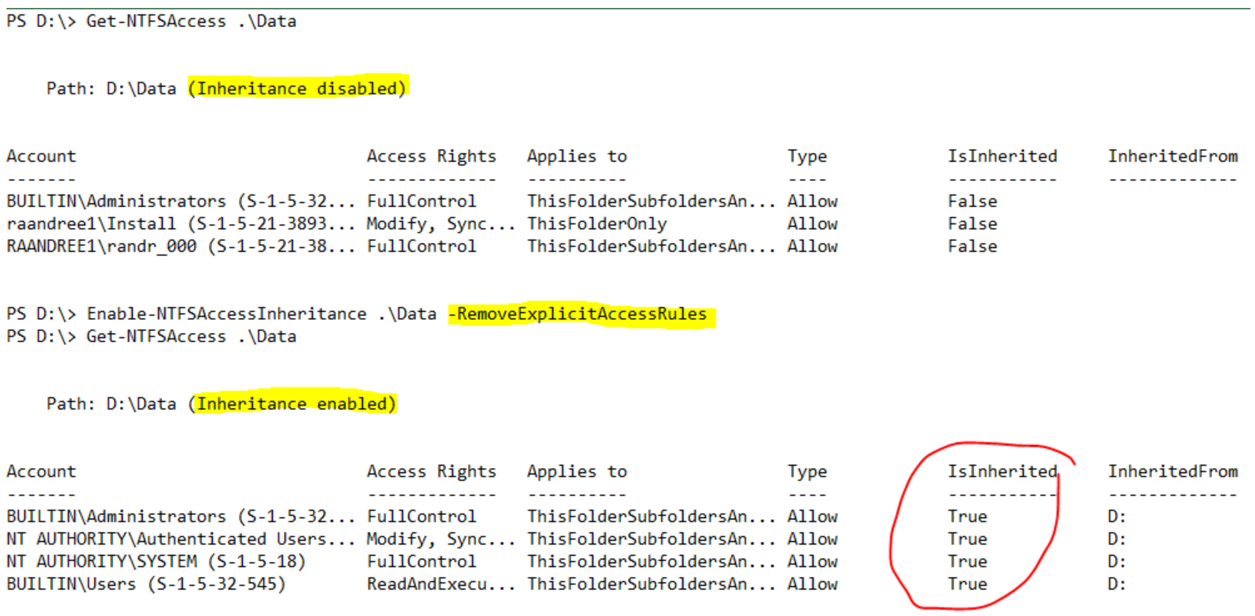
Enabling Inheritance

The cmdlet for enabling the inheritance on an object is similar to reading the inheritance information:

Enable-NTFSAccessInheritance -Path .\Data

By default, the cmdlet does not return any output unless you use the **PassThru** switch.

The cmdlet provides one additional switch parameter: **RemoveExplicitAccessRules**. When specified, all explicitly assigned access control entries (ACEs) will be removed after enabling the inheritance. This is like setting the folder back to default permissions.



```
PS D:\> Get-NTFSAccess .\Data


    Path: D:\Data (Inheritance disabled)


Account                             Access Rights    Applies to                Type            IsInherited     InheritedFrom
-------                             -------------    ----------                ----            -----------     -------------
BUILTIN\Administrators (S-1-5-32... FullControl      ThisFolderSubfoldersAn... Allow           False
raandree1\Install (S-1-5-21-3893... Modify, Sync...  ThisFolderOnly            Allow           False
RAANDREE1\randr_000 (S-1-5-21-38... FullControl      ThisFolderSubfoldersAn... Allow           False


PS D:\> Enable-NTFSAccessInheritance .\Data -RemoveExplicitAccessRules
PS D:\> Get-NTFSAccess .\Data


    Path: D:\Data (Inheritance enabled)


Account                             Access Rights    Applies to                Type            IsInherited     InheritedFrom
-------                             -------------    ----------                ----            -----------     -------------
BUILTIN\Administrators (S-1-5-32... FullControl      ThisFolderSubfoldersAn... Allow           True            D:
NT AUTHORITY\Authenticated Users... Modify, Sync...  ThisFolderSubfoldersAn... Allow           True            D:
NT AUTHORITY\SYSTEM (S-1-5-18)      FullControl      ThisFolderSubfoldersAn... Allow           True            D:
BUILTIN\Users (S-1-5-32-545)        ReadAndExecu...  ThisFolderSubfoldersAn... Allow           True            D:
```

The previous output shows that there were only explicitly assigned ACEs in the Data folder. These were removed after enabling the inheritance and there are only inherited ACEs remaining.

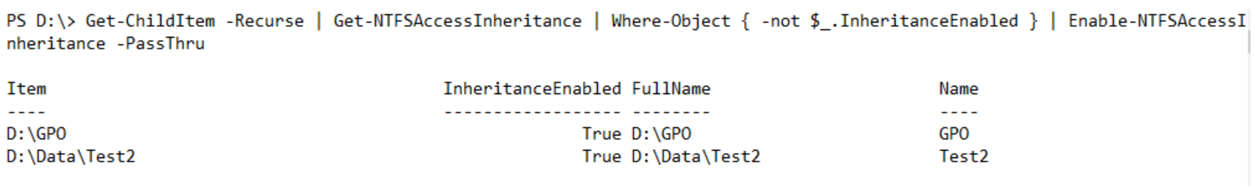
Note: *The RemoveExplicitAccessRules switch should be used with care. Using this switch by accident can cause users to not be able to access their resources.*

A common use case for Enable-NTFSAccessInheritance

If users can alter the ACL of files or folders, they might remove relevant ACEs. The result is that the administrators are no longer able to access the data. Another reason why an administrator cannot access data is that someone has disabled the inheritance and deleted all inherited ACEs.

First you need to use the recursive method provided by **Get-ChildItem** to get all files and folders and filter out the items where inheritance is enabled.The remaining items are piped to **Enable-NTFSAccessInheritance**, which informs about items where inheritance is enabled with the **PassThru** switch.

Get-ChildItem -Recurse | Get-NTFSAccessInheritance | Where-Object { -not $_.InheritanceEnabled } | Enable-NTFSAccessInheritance -PassThru

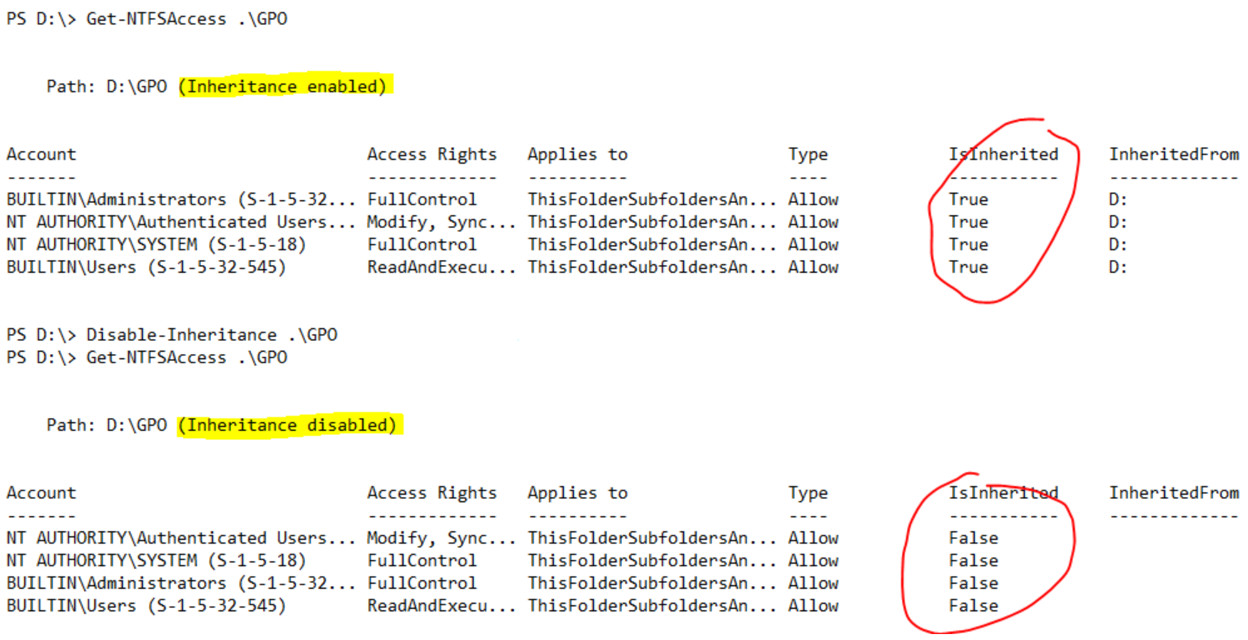


```
PS D:\> Get-ChildItem -Recurse | Get-NTFSAccessInheritance | Where-Object { -not $_.InheritanceEnabled } | Enable-NTFSAccessI
nheritance -PassThru

Item                                        InheritanceEnabled FullName                       Name
----                                        ------------------ --------                       ----
D:\GPO                                                    True D:\GPO                         GPO
D:\Data\Test2                                             True D:\Data\Test2                  Test2
```

Disabling Inheritance

In NTFSSecurity, you also need to disable inheritance. When enabling inheritance you have to determine what to do with explicit ACES (usually you keep them). When disabling inheritance, the inherited ACEs can be converted into explicit ones or they can be removed. This is controlled by the **RemoveInheritedAccessRules** switch.

In the following example, the inheritance for the folder GPO is enabled. The folder is inheriting four ACEs from the parent (drive D). After disabling inheritance, the ACEs still exist as explicit ACEs.



```
PS D:\> Get-NTFSAccess .\GPO


    Path: D:\GPO (Inheritance enabled)


Account                            Access Rights   Applies to                Type            IsInherited     InheritedFrom
-------                            -------------   ----------                ----            -----------     -------------
BUILTIN\Administrators (S-1-5-32... FullControl     ThisFolderSubfoldersAn... Allow           True            D:
NT AUTHORITY\Authenticated Users... Modify, Sync... ThisFolderSubfoldersAn... Allow           True            D:
NT AUTHORITY\SYSTEM (S-1-5-18)     FullControl     ThisFolderSubfoldersAn... Allow           True            D:
BUILTIN\Users (S-1-5-32-545)       ReadAndExecu... ThisFolderSubfoldersAn... Allow           True            D:


PS D:\> Disable-Inheritance .\GPO
PS D:\> Get-NTFSAccess .\GPO


    Path: D:\GPO (Inheritance disabled)


Account                            Access Rights   Applies to                Type            IsInherited     InheritedFrom
-------                            -------------   ----------                ----            -----------     -------------
NT AUTHORITY\Authenticated Users... Modify, Sync... ThisFolderSubfoldersAn... Allow           False
NT AUTHORITY\SYSTEM (S-1-5-18)     FullControl     ThisFolderSubfoldersAn... Allow           False
BUILTIN\Administrators (S-1-5-32... FullControl     ThisFolderSubfoldersAn... Allow           False
BUILTIN\Users (S-1-5-32-545)       ReadAndExecu... ThisFolderSubfoldersAn... Allow           False
```

***Note:** When using the **RemoveInheritedAccessRules** switch, you need to make sure that the access is guaranteed after disabling the inheritance. If an item does not have explicit ACEs assigned and you remove all inherited ACEs, nobody will be able to access the file. However, an administrator can always use the back-up privilege or take ownership to get access again.*

Using privileges

***Note:** There was a big change in Windows 8 and Windows Server 2012 regarding how privileges can be used. The features described in this section work only on Windows 8.1, Windows 8, Windows Server 2012 R2, and Windows Server 2012.*

Sometimes permissions are hosed and access is denied, even if you are the mighty administrator. In this case, you cannot even read the existing ACL. Of course, the administrator can always take the ownership of an object, but this erases the existing ACL of the object. This is bad because you cannot determine by the ACL the people who need to have access.

Windows has a very handy concept of privileges. A privilege represents the right of an account, such as a user or group account, to perform various system-related operations on the local computer, such as shutting down the system, loading device drivers, or changing the system time. Privileges differ from access rights in two ways:

- Privileges control access to system resources and system-related tasks, whereas access rights control access to securable objects.
- A system administrator assigns privileges to user and group accounts, whereas the system grants or denies access to a securable object based on the access rights granted in the ACEs in the object's DACL.

When dealing with files and folders, three privileges are worth looking at:

- Back up files and directories:

This user right determines which users can bypass file and directory, registry, and other persistent object permissions for the purposes of backing up the system. By default, Administrators and Backup Operators can make use of this privilege.

- Restore files and directories:

This security setting determines which users can bypass file, directory, registry, and other persistent objects permissions when restoring backed up files and directories, and it determines which users can set valid security principals as the owner of an object.

Specifically, this user right is similar to granting the following permissions to the user or group in question on all files and folders on the system:

- - Traverse Folder/Execute File
  - Write

By default Administrators and Backup Operators have this privilege assigned.

- Take ownership of files or other objects:

This security setting determines which users can take ownership of any securable object in the system, including Active Directory objects, files and folders, printers, registry keys, processes, and threads. By default, this privilege is assigned to Administrators.

For a list of privileges, refer to Privileges in the TechNet Library.

To create the situation described previously, you can use the following command sequence:

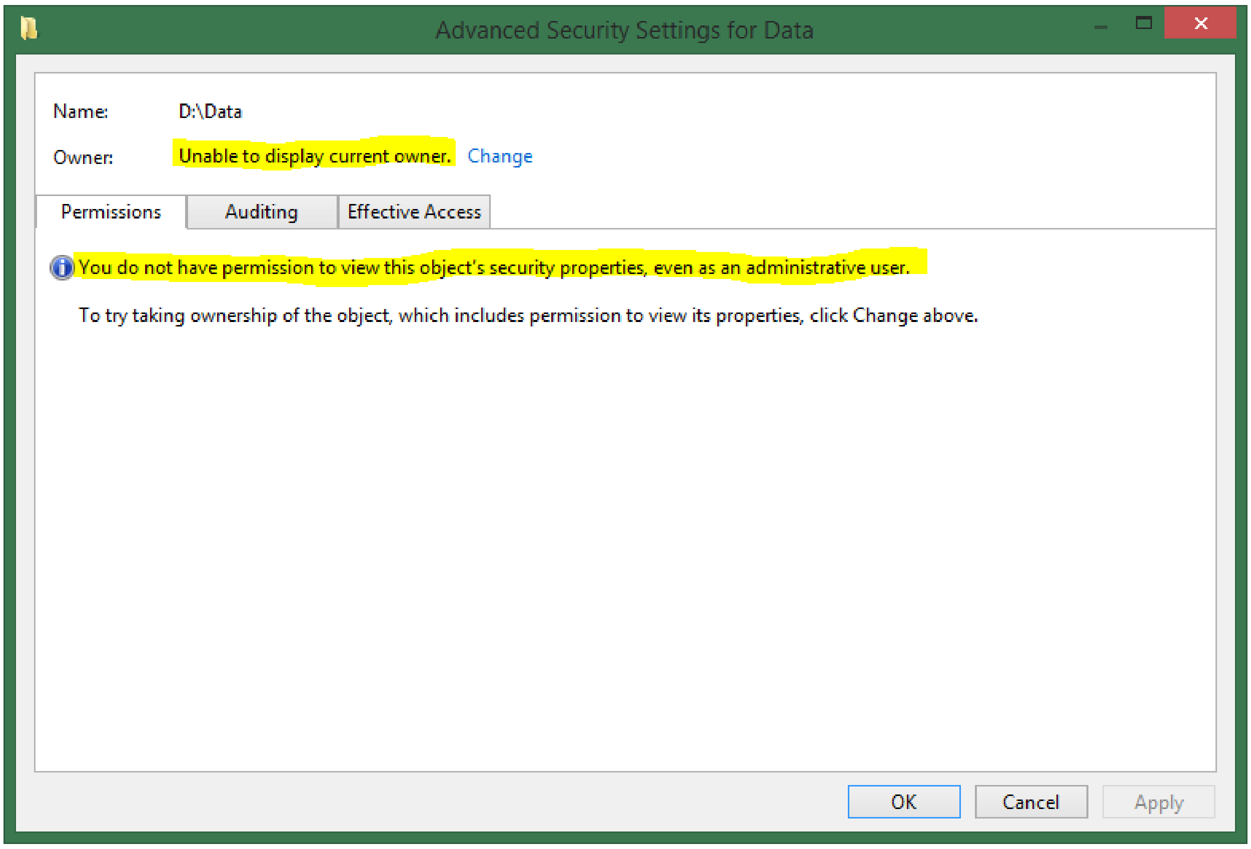
Set-NTFSOwner .\Data -Account 'NT AUTHORITY\SYSTEM'

Disable-NTFSAccessInheritance .\Data -RemoveInheritedAccessRules

The first command assigns the ownership of the Data folder to the SYSTEM account, and the second command disables the inheritance. Now only the SYSTEM account can access the item.

```
PS D:\> Get-NTFSAccess .\Data
Get-NTFSAccess : Access is denied. (Exception from HRESULT: 0x80070005 (E_ACCESSDENIED)): [\\?\D:\Data]
At line:1 char:1
+ Get-NTFSAccess .\Data
+ ~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : OpenError: (.\Data:String) [Get-NTFSAccess], UnauthorizedAccessException
    + FullyQualifiedErrorId : ReadError,NTFSSecurity.GetAccess
```

Windows Explorer is not helpful either:


Advanced Security Settings for Data

Name: D:\Data

Owner: Unable to display current owner. Change

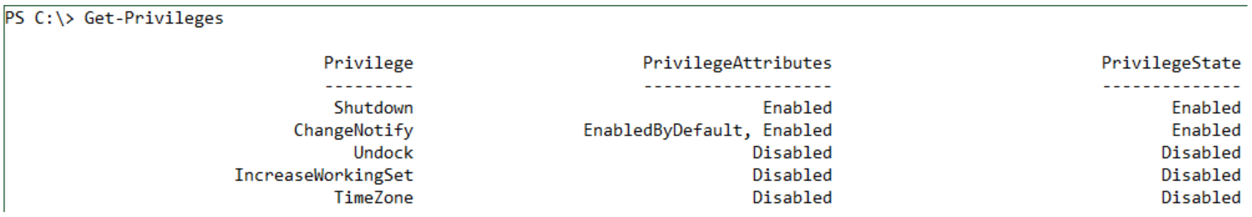
You do not have permission to view this object's security properties, even as an administrative user.

To try taking ownership of the object, which includes permission to view its properties, click Change above.

Enabling privileges

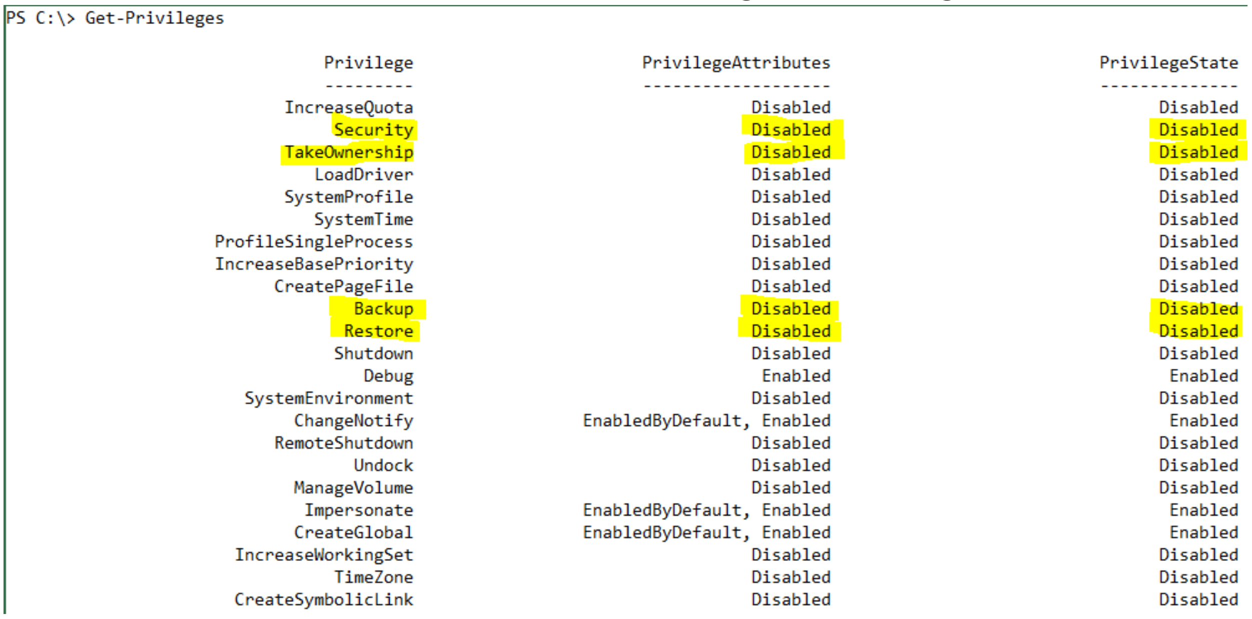
The NTFSSecurity module provides the **Enable-Privileges** cmdlet. This cmdlet enables the privileges Backup, Restore, and Security (if you have them). You can get a list of available privileges by using **Get-Privilege**.

Running **Get-Privilege** in a non-elevated Windows PowerShell console should return something like this:



```
PS C:\> Get-Privileges

                                   Privilege                       PrivilegeAttributes                            PrivilegeState
                                   ---------                       -------------------                            --------------
                                    Shutdown                                   Enabled                                   Enabled
                                ChangeNotify                     EnabledByDefault, Enabled                               Enabled
                                      Undock                                  Disabled                                  Disabled
                         IncreaseWorkingSet                                  Disabled                                  Disabled
                                    TimeZone                                  Disabled                                  Disabled
```

The privileges Backup, Restore, and Security are missing. If you run the same command in an elevated Windows PowerShell console, the list gets much longer:



```
PS C:\> Get-Privileges

                                   Privilege                       PrivilegeAttributes                            PrivilegeState
                                   ---------                       -------------------                            --------------
                               IncreaseQuota                                  Disabled                                  Disabled
                                    Security                                  Disabled                                  Disabled
                               TakeOwnership                                  Disabled                                  Disabled
                                  LoadDriver                                  Disabled                                  Disabled
                               SystemProfile                                  Disabled                                  Disabled
                                  SystemTime                                  Disabled                                  Disabled
                        ProfileSingleProcess                                  Disabled                                  Disabled
                         IncreaseBasePriority                                  Disabled                                  Disabled
                              CreatePageFile                                  Disabled                                  Disabled
                                      Backup                                  Disabled                                  Disabled
                                     Restore                                  Disabled                                  Disabled
                                    Shutdown                                  Disabled                                  Disabled
                                       Debug                                   Enabled                                   Enabled
                           SystemEnvironment                                  Disabled                                  Disabled
                                ChangeNotify                     EnabledByDefault, Enabled                               Enabled
                              RemoteShutdown                                  Disabled                                  Disabled
                                      Undock                                  Disabled                                  Disabled
                                ManageVolume                                  Disabled                                  Disabled
                                 Impersonate                     EnabledByDefault, Enabled                               Enabled
                                CreateGlobal                     EnabledByDefault, Enabled                               Enabled
                          IncreaseWorkingSet                                  Disabled                                  Disabled
                                    TimeZone                                  Disabled                                  Disabled
                          CreateSymbolicLink                                  Disabled                                  Disabled
```

All these privileges are disabled by default. Having them enabled all the time would be quite dangerous and would increase ones scope of action too much. However in some scenarios, it is required to make use of privileges, for example, when doing backup jobs, data migrations, or permission cleanups.

To enable the privileges, call Enable-Privileges. You will get a warning message that informs you about the enabled privileges.

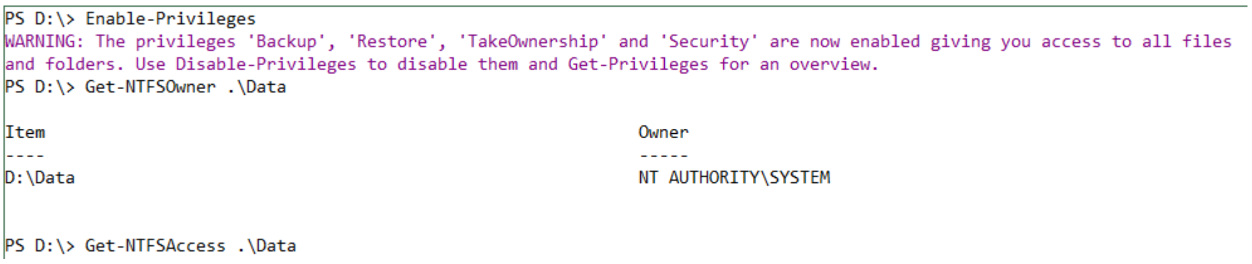
Note: *If you call the cmdlet in a non-elevated Windows PowerShell console, and hence, you do not hold the proper privileges, the following error message returns: "Enable-Privileges: Could not enable requested privileges. Cmdlets of NTFSSecurity will only work on resources you have access to."*

After calling Enable-Privileges, you can check the state of the privileges by using Get-Privileges.

Using Enabled Privileges

Enabling the privileges is pretty much all you need to do. The process that has enabled them uses them automatically.

After removing the permissions from the Data folder, there is no way to access it or display the owner or ACL. However, after enabling the privileges, Windows bypasses the ACL and grants you full access (thanks to the Backup and Restore privilege).



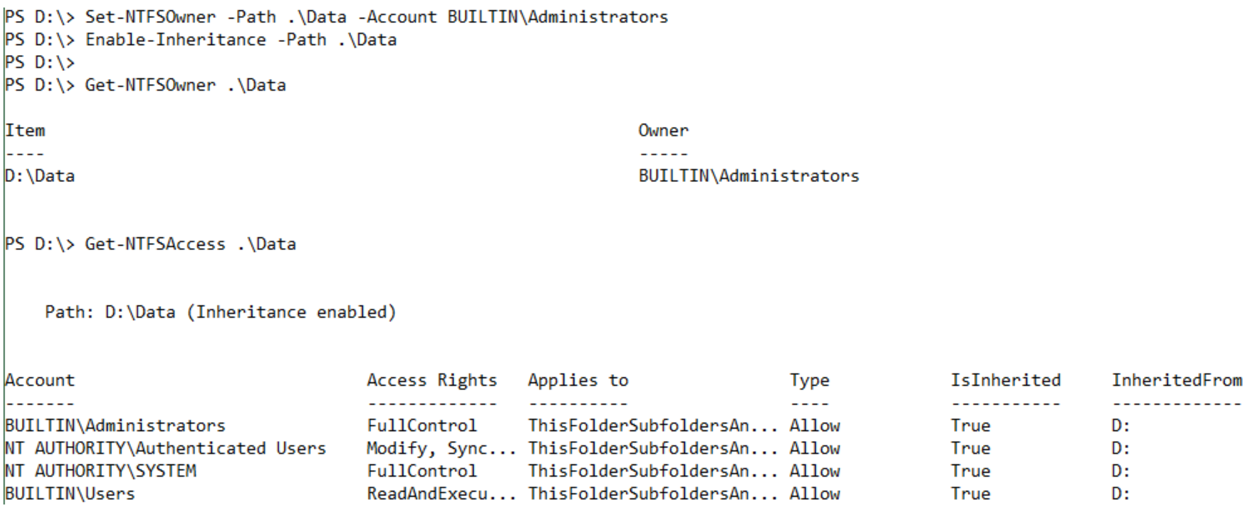
```
PS D:\> Enable-Privileges
WARNING: The privileges 'Backup', 'Restore', 'TakeOwnership' and 'Security' are now enabled giving you access to all files
and folders. Use Disable-Privileges to disable them and Get-Privileges for an overview.
PS D:\> Get-NTFSOwner .\Data

Item                                                       Owner
----                                                       -----
D:\Data                                                    NT AUTHORITY\SYSTEM


PS D:\> Get-NTFSAccess .\Data
```

Get-NTFSAccess does not return any data because there is no ACE in the ACL. Remember that all ACEs have been removed by disabling the inheritance.

Now you can take the ownership and enable inheritance again:

Set-NTFSOwner -Path .\Data -Account BUILTIN\Administrators

Enable-NTFSInheritance -Path .\Data



```
PS D:\> Set-NTFSOwner -Path .\Data -Account BUILTIN\Administrators
PS D:\> Enable-Inheritance -Path .\Data
PS D:\>
PS D:\> Get-NTFSOwner .\Data

Item                                                       Owner
----                                                       -----
D:\Data                                                    BUILTIN\Administrators


PS D:\> Get-NTFSAccess .\Data


    Path: D:\Data (Inheritance enabled)


Account                          Access Rights  Applies to                Type           IsInherited    InheritedFrom
-------                          -------------  ----------                ----           -----------    -------------
BUILTIN\Administrators           FullControl    ThisFolderSubfoldersAn... Allow          True           D:
NT AUTHORITY\Authenticated Users Modify, Sync... ThisFolderSubfoldersAn... Allow          True           D:
NT AUTHORITY\SYSTEM              FullControl    ThisFolderSubfoldersAn... Allow          True           D:
BUILTIN\Users                    ReadAndExecu... ThisFolderSubfoldersAn... Allow          True           D:
```

Now the access is how it should be again.

Disabling privileges

If you no longer need the privileges, it is strongly recommended to disable them. Use the command **Disable-Privileges** for this.

After calling **Disable-Privileges**, you can check the state of the privileges by using **Get-Privileges**.