Realtime publishers

The Essentials Series: Fundamentals of Effective File Server Security

Enumerating File and Folder Security

sponsored by

SCRIPTLOGIC®

by Greg Shields

Enumerating File and Folder Security 1
Viewing Permissions 1
Native File and Folder Enumeration Remains Painful 4

Copyright Statement

Enumerating File and Folder Security

Assigning permissions to files and folders is an important task. But you won't get far without first having a useful enumeration of your folder structures themselves. Windows Server 2008 provides numerous tools to visualize the access permissions set on your files and folders. Primary tools for this are the Windows Explorer GUI as well as command-line tools such as showacls.exe, icacls.exe, or PowerShell's Get-ACL. Unfortunately, these tools are limited in flexibility for environment sizes that go much beyond the very small. Effectively using them requires you to aggregate results from other solutions or turn to third-party solutions for a comprehensive analysis of file and folder security. With this in mind, let's take a look through the tools that are natively available, with an eye towards the features and capabilities that one might want in an external solution.

Viewing Permissions

Viewing NTFS and Share permissions from Windows Explorer requires the individual examination of each file or folder. Share permissions can be determined only by opening the properties dialog box from of the root of the share (see Figure 1).

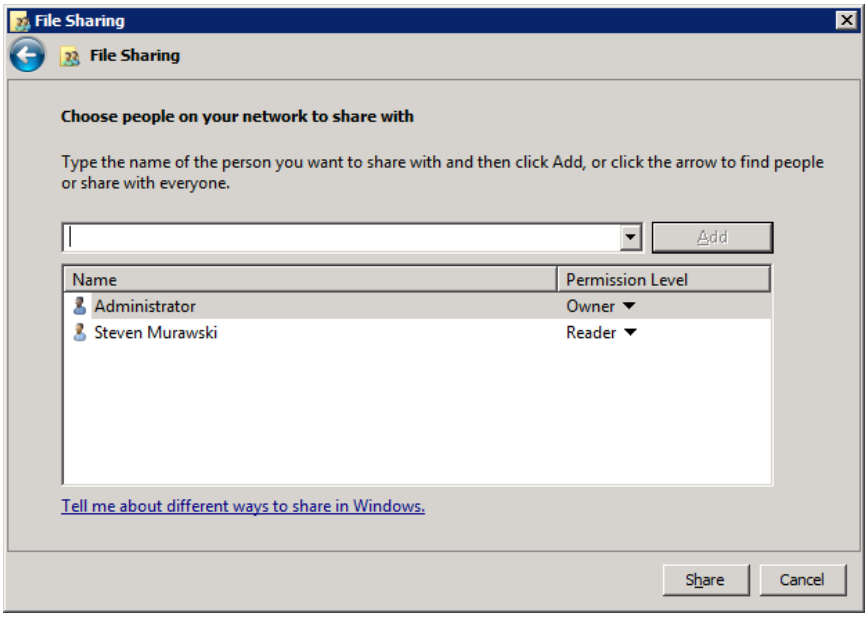

Figure 1: Viewing NTFS and Share permissions from Windows Explorer.

A similar process can be used to examine the NTFS permissions for a specific object via Windows Explorer from the properties dialog box. Administrators can dig deeper into the application of these permissions by going to the Advanced view, as shown in Figure 2.

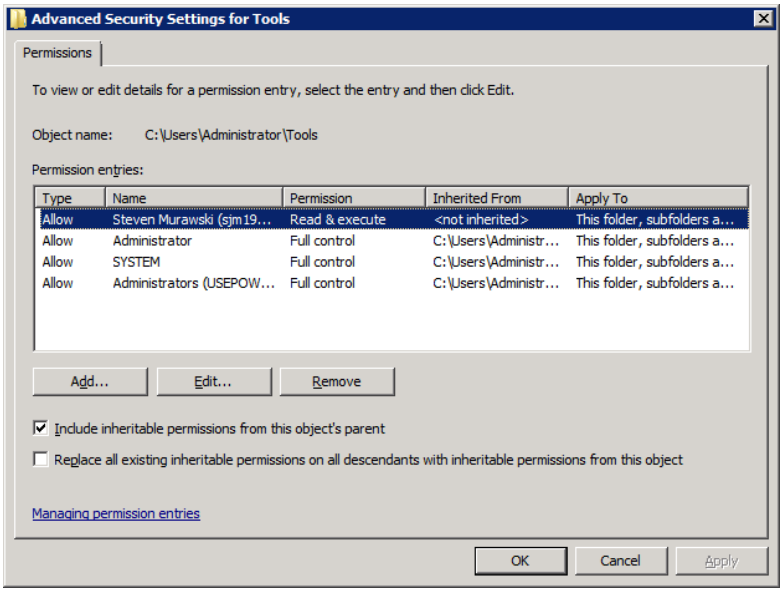

Figure 2: Digging deeper into permissions application.

These wizard screens are excellent for the single-instance application of permissions. Their use works well when you need to apply only a few permissions to a few folders. *Yet they don't scale.* The process to set each new permission can require a minimum of seven mouse clicks, with special permissions requiring an even greater number. More permissions equals more mouse clicks, which reduces your effectiveness and increases the effort required to do your job.

To combat this, Microsoft provides command-line tools for working with NTFS and Share permissions that include options for reporting. These command-line reporting options enable the creation of text output that can be redirected to a file for later viewing.

Using the net share command with no options will provide a listing of all shared folders on a system, including the share name, path, and any assigned remarks. Specifying a share name with net share will display share details, which adds information about the maximum limit of users, caching settings, and assigned permissions. As an example, to report this information about the MyFolderShare to a file called filepermissions.txt, use the following syntax:

```
subinacl.exe /share MyFolderShare /display
/outputlog="c:\securitylog\filepermissions.txt"
```

As subinacl.exe also can manage file system permission, shown below is a similar syntax which reports on NTFS permissions for files:

```
subinacl.exe /subdirectories c:\Users\Administrator\Tools /display
/outputlog="c:\securitylog\filepermissions.txt"
```

This command structure enables icacls.exe to report on a file's NTFS permissions. However, more useful is the ability to save that structure to a file for later reapplication. Consider the situation where you've created a rich set of permissions for a large file structure. Using the /save switch, as shown below, icacls.exe will create a text file that contains the folder and permissions structure in Microsoft's Security Descriptor Definition Language (SDDL) format:

```
icacls c:\Users\Administrator\Tools\* /save ACLFile /T
```

Replacing the /save switch with /restore in the previous code snippet will restore the text file's saved permissions to your file structure. The net result is the ability to quickly restore an entire structure's permissions as necessary to fix a mistaken permission or simply ensure that your permissions are set to your established standards.

Showacls.exe is another command-line tool, found in the Windows 2003 Resource Kit, which can be used for retrieving and viewing NTFS permissions. The differentiator with showacls.exe is in its ability to report on the specific permissions assigned to a user or group, similar to the Effective Permissions option found within Windows Explorer:

```
showacls /s c:\Users\Administrator\Tools\* > ACLFile.txt

showacls /s /u:MyDomain\administrator c:\Users\Administrator\Tools\* > ACLFile.txt
```

When possible, showacls.exe will use the simple permissions Read, Change, or Full Control. If the permission structure is more complex, it provides an "access mask," which attempts to sum up the access rights. More information about configuring access masks can be found in the tool's help file.

Windows PowerShell's Get-ACL cmdlet accomplishes many of the same textual visualizations seen in the previously mentioned tools but with the added power of PowerShell's rich scripting architecture. Get-ACL returns a System.Security.AccessControl.DirectorySecurity object for each file and directory it is run against, which can be later repurposed for other uses within a larger PowerShell script infrastructure. The Access property of this object contains the file or folder permissions:

```
Get-ACL c:\Users\Administrator\Tools\*
```

As a full scripting language, PowerShell provides several display options over and above the alternatives, including exporting to XML, comma separated value (CSV) files, or HTML.

Native File and Folder Enumeration Remains Painful

Each of these examples provides you with yet another view of your files and folders. But as you can obviously see, their results are almost entirely text-based. Although native tools indeed enable you to enumerate and visualize your permissions structures, you can argue that their text-based nature isn't much better than Windows Explorer alone.

Environments that need greater visibility into file server security must look to external solutions. These solutions enable the discovery of file and folder structures as well as their assigned permissions. Their aggregation of permissions information across multiple file servers into a central and consistent format enables the reporting on permissions across an entire infrastructure at once. Storing permissions centrally also enables administrators to see how and where permissions structures have evolved over time, whether by server, user, group, or combination therein. Leveraging this file server metadata with a well-formed API for accomplishing needed tasks ensures that the job is done right. Lacking a substantial scripting investment, this capability is simply not possible using native toolsets and homegrown solutions.

The needs of security are but one facet of file server management. A third-party solution becomes more valuable when the reporting is needed to meet regulatory compliance. Regulations such as the Sarbanes-Oxley Act (SOX), Health Insurance Portability and Accountability Act (HIPAA), Gramm-Leach-Bliley Act (GLBA), Federal Desktop Core Configuration (FDCC), and Payment Card Industry (PCI) require IT departments to positively show that proper access controls are in place on critical data. As you'll find out in the third article of this series, third-party solutions provide needed templates or reports that are designed to meet the auditing requirements imposed by those regulations.