Mounting Unix Shares with a Windows NFS Client

blog.netwrix.com/2022/11/18/mounting-nfs-client-windows

Network File System (NFS) is an open standard for distributing a file system across a network for multi-client access. Designed in 1984, NFS has grown to include many authentication methods at both the share (export) and file system levels, including client IP/hostname, auth_sys (Unix auth), Kerberos and NFSv4.x ACLs.

This blog post explains how to mount an NFS share on a Windows client.

Handpicked related content:

Windows Server Security Best Practices

How NSF works with Windows operating systems

While you're likely to be familiar with accessing network file shares via Server Message Block (SMB) or the Windows implementation of SMB (CIFS), NFS is still prevalent in production environments with Unix servers.

Unfortunately, NFS traditionally did not play well with environments that mix Windows with Unix: To enable Windows client access to NFS exports, each NFS export needed a Samba share equivalent (an SMB implementation for Unix).

However, this changed when Microsoft implemented NFS client and server tools. Microsoft's NFS documentation lists the following operating system support:

Operating Systems	NFS Server Versions	NFS Client Versions
Windows 7, Windows 8.1 Windows 10	N/A	NFSv2, NFSv3
Windows Server 2008, Windows Server 2008 R2	NFSv2, NFSv3	NFSv2, NFSv3
Windows Server 2012, Windows Server 2012 R2, Windows Server 2016, Windows Server 2019	NFSv2, NFSv3, NFSv4.1	NFSv2, NFSv3

How to Configure Windows as an NFS Client

Prerequisite: Enable the necessary Windows features.

Before we mount an NFS share on a Windows client, we need to enable certain features within Windows to perform NFS client operations. The PowerShell command to use depends on your client environment:

For Windows 10:

```
Enable-WindowsOptionalFeature -FeatureName ServicesForNFS-ClientOnly,
ClientForNFS-Infrastructure -Online -NoRestart
```

For Windows Server:

```
Install-WindowsFeature NFS-Client
```

Mount the NFS share (export).

Now we need to mount NFS exports from a Unix server. However, Unix and Windows use different mechanisms for identifying users and groups: In Unix-like operating systems such as Linux, users and groups are identified by user identifiers (UIDs) and group identifiers (GIDs), respectively. In Windows, users and groups are identified using security identifiers (SIDs).

Therefore, in order to authenticate to a Unix server providing NFS exports, we need to map Windows users to Unix UIDs and GIDs. With this UID/GID mapping, the Unix server will be able to determine which user created the request for the NFS export.

Here are three methods you can use to perform the identity mapping and mount the NFS export.

Method 1 (preferred). Perform identity mapping in Active Directory (AD).

If both the Unix NFS server and Windows NFS client are joined to the same Active Directory domain, then we can handle identity mapping in Active Directory. This is the preferred method for security purposes when possible.

NOTE: This method is not available if method 2 (below) is in use, since the presence of a local etcpasswd file will take precedence for identity mapping.

By default, our NFS client won't look up identity mapping in Active Directory. However, we can change that by running the following command in an elevated PowerShell session on the NFS client:

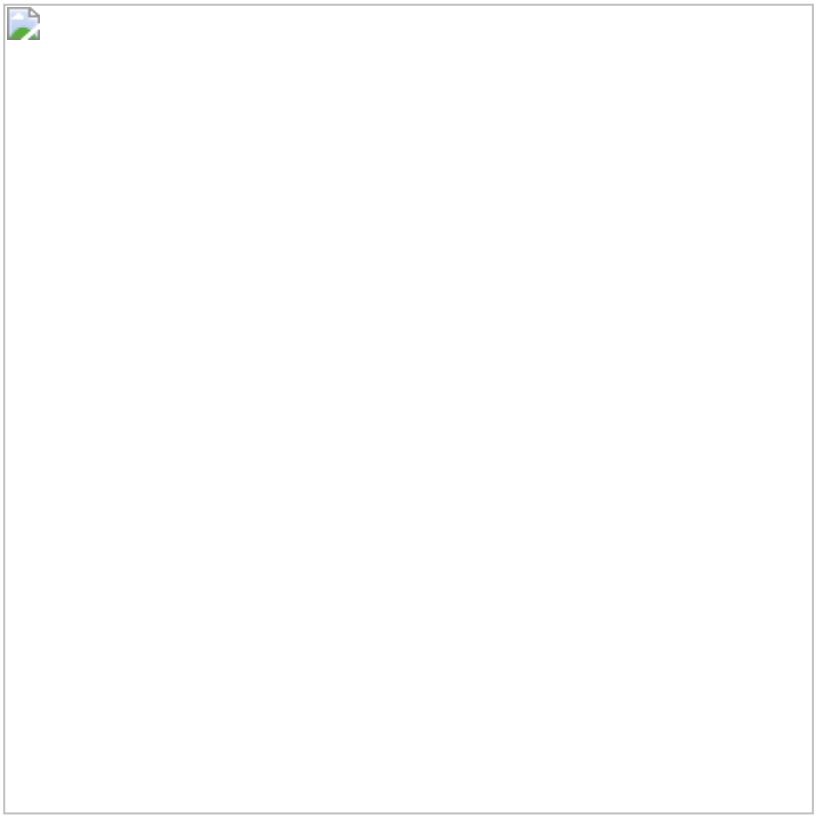
```
» Set-NfsMappingStore -EnableADLookup $True -ADDomainName "<your_domain>"
```

Now we can run the Get-NfsMappingStore command to check the current Windows user's UID/GID mapping. As you can see, ADLookupEnabled is set to True, and a domain is specified for ADDomain.

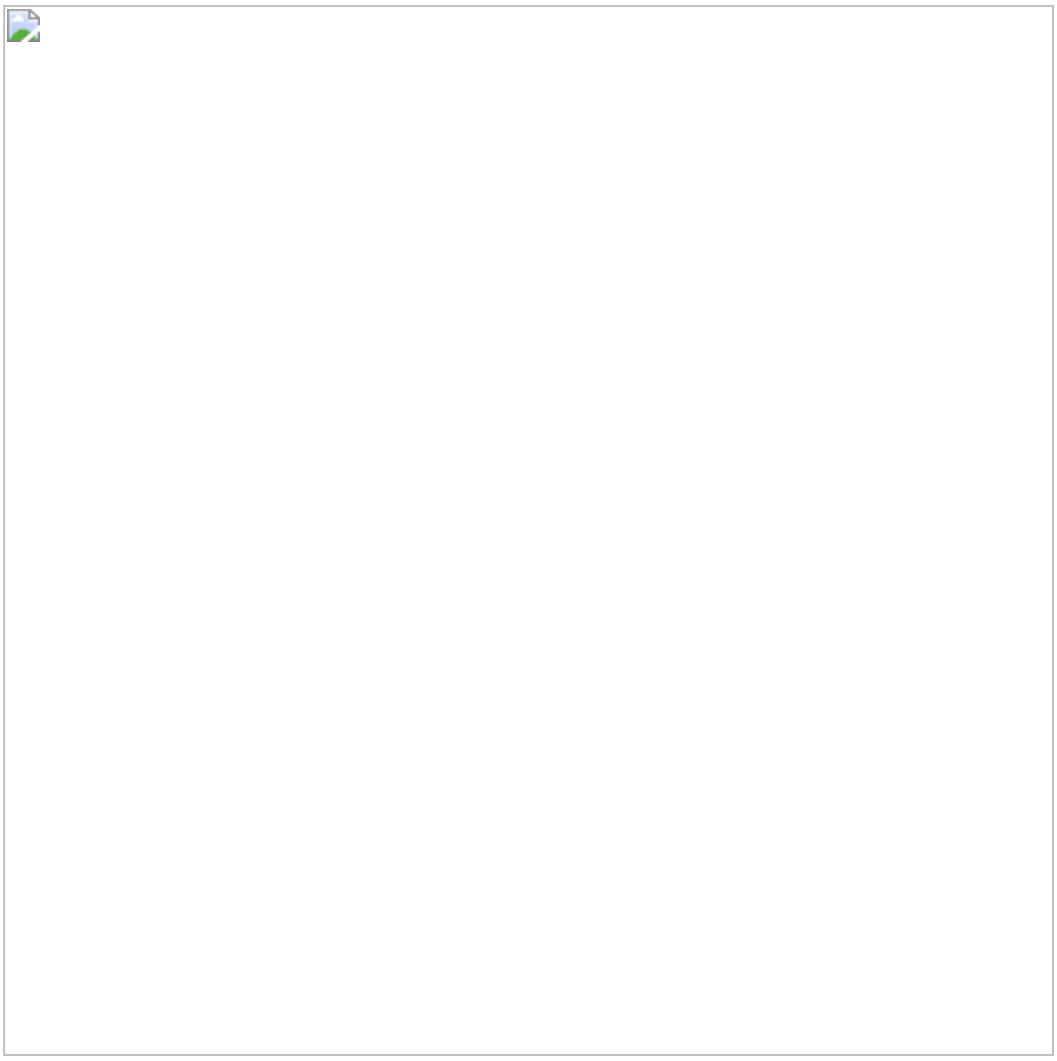
```
» Get-NfsMappingStore
```

```
UNMServer               :

UNMLookupEnabled        : False

ADDomain                : <your_domain>

ADLookupEnabled         : True

LdapServer              :

LdapNamingContext       :

LdapLookupEnabled       : False

PasswdFileLookupEnabled : False
```

Next, we need to configure our identity mapping in Active Directory Users and Computers. To view the uidNumber and gidNumber attributes for each user, make sure you have Advanced Features enabled under the View dropdown:



You'll then be able to view and edit those fields in a user or group's Properties menu, on the Attribute Editor tab:



It can be cumbersome to manually map UIDs and GIDs for many Active Directory users. To automate the process, you can use the following PowerShell command to set the appropriate attribute values for each desired user, using a CSV file with UID/GID data:

```
Set-ADUser -identity <UserPrincipalName> -add @{uidNumber="<user_unix_uid>";

gidNumber="<user_unix_gid>"}
```

.

NOTE: In the Set-ADUser command, “add” should be changed to “replace” if a user already has a value for either uidNumber or gidNumber.

Using this approach, we can now map an NFS share in Windows to an available drive letter via command prompt, and the UID/GID will be mapped per the current Active Directory user’s uidNumber and gidNumber attribute values.

```
» mount \<nfs_server_ip_address>pathtonfsexport Z:
```

The path after the NFS server’s IP is the local path to the NFS export on the NFS server, and the drive letter is any available drive letter on the Windows NFS client.

Of course, the Unix rights given to the user we've mapped to ultimately decide what kind of access we have to the export, such as read/write or read-only.

Method 2. Perform identity mapping using the Local etcpasswd file.

Since using Active Directory is the preferred method for identity mapping, we won't go into detail for the other two options. However, it's worth briefly stating that Windows can perform local identity mapping using Unix-style passwd and group files, located in %SYSTEMROOT%system32driversetc.

If the passwd file is present and has identity mapping information for the current Windows user, then the mappings specified in the passwd and group files will be used for the client's NFS mount requests rather than UID/GID mappings in Active Directory or the AnonymousUid/AnonymousGid Windows registry settings outlined below.

When running the Get-NfsMappingStore PowerShell command, you'll notice PasswdFileLookupEnabled is True whenever this workflow is in effect for the current Windows user.

This approach uses the same mount syntax as the Active Directory identity mapping approach above:

```
» mount \<nfs_server_ip_address>pathtonfsexport Z:
```

Method 3. Perform identity mapping using AnonymousUid/AnonymousGid Windows registry settings.

The final method is considered an insecure approach and is not recommended. It potentially allows any local user to mount the target NFS export(s) with read/write access, as opposed to securing write permissions to specific local Windows users via the methods above.

To map the local Windows client to the UID and GID of the Unix user that owns the desired export(s), run the following in an elevated PowerShell:

```
» New-ItemProperty HKLM:SOFTWAREMicrosoftClientForNFSCurrentVersionDefault -Name
AnonymousUID -Value <unix_export_owner_uid> -PropertyType "DWord"

» New-ItemProperty HKLM:SOFTWAREMicrosoftClientForNFSCurrentVersionDefault -Name
AnonymousGID -Value <unix_export_owner_gid> -PropertyType "DWord"
```

After adding these keys to the Windows registry, you need to reboot in order to have them take effect.

Then use the following command to mount the NFS export with read/write access (assuming the client's IP has permission to mount the export and that the UID/GID mapping is correct for each desired export):

```
» mount -o anon \<nfs_server_ip_address>pathtonfsexport Z:
```

FAQ

Does Windows 11 support NFS clients?

Yes.

How can I use NFS in Windows?

- Provide access to the same file share using both the SMB and NFS protocols by using a Windows NFS file server.
- Deploy a non-Windows operating system to provide NFS file shares accessible to non-Windows clients using the NFS protocol.
- To enable applications to be migrated from one operating system to another, store data on file shares accessible using both the SMB and NFS protocols.

What improvements are included in NFS version 4.1?

- The Remote Procedure Call (RPC)/External Data Representation (XDR) transport infrastructure offers better support and provides better scalability
- RPC port multiplexer feature
- Auto-tuned caches and thread pools
- New Kerberos privacy implementation and authentication options

For complete details, visit the Microsoft page that describes all NFS versions.

How do I add the Server for NFS role service?

In Server Manager or Windows Admin Center, use the “Add Roles” and “Features Wizard”.

Which Windows command-line administration tools does Server for NFS contain?

- **Mount** provides an NFS mount on Windows clients that maps to a local drive
- **Nfsadmin** manages configuration settings of the Server for NFS and Client for NFS components.
- **Nfsshare** sets up NFS share settings for folders that are shared via Server for NFS.
- **Nfsstat** displays or resets statistics on calls received by Server for NFS.
- **Showmount** lists the file systems that have been exported by Server for NFS.

NFS-mounted drives are unmounted using **Umount**.

Joe Dibley
Security Researcher at Netwrix and member of the Netwrix Security Research Team. Joe is an expert in Active Directory, Windows, and a wide variety of enterprise software platforms and technologies, Joe researches new security risks, complex attack techniques, and associated mitigations and detections.