

# Linux NFS Client/Server Lab

## By Michael Ambeguia

**Purpose:** Having a collaborative file sharing ability is a useful utility for companies to have. NFS is one way companies can implement this functionality into their IT environment. NFS allows companies to share documents for editing and reading across the network from a single server. The purpose of this lab is to gain hands-on experience implementing this commonly used Linux service. This lab will demonstrate how to configure and run an NFS server on a Linux server and how to set up the client devices as well. Other pertinent topics such as how to secure NFS and troubleshoot its issue are demonstrated as well.

### Topics covered:

1. NFS server configuration
2. NFS client configuration
3. Managing NFS exports and mounts
4. Troubleshooting NFS issues
5. Understanding NFS permissions and security

### Sections:

1. Introduction to NFS
2. NFS Server Setup
3. Configuring NFS Exports
4. NFS Client Setup
5. NFS Performance and Security
6. Troubleshooting NFS Issues

## Section #1 Introduction to NFS:

### 1.1a. What is NFS? How does it work?

NFS is a service that allows a system to share directories and files over the network. Users that access the remote files using NFS access files as though they were located locally on the filesystem.

This is how NFS works:

Client Request:

1. The client device requests to perform an action on a file such as opening, reading, writing, or closing a file.
2. The NFS client translates the request into an RPC (Remote Procedure Call) and sends the RPC over the network to the NFS server.

Server Processing:

1. The NFS server receives the RPC request and processes it. This involves interpreting the request and performing the appropriate file system operations.
2. The server accesses its local file system to perform the requested operation, such as reading from or writing to the file.

3. The server prepares a response, which includes the result of the operation (e.g., file data for a read request) or an acknowledgment.
4. The server sends the response back to the NFS client over the network.

#### Client Response Handling:

1. The NFS client receives the server's response and processes it.
2. The client application completes the file operation based on the response received from the server.

Let's take a look at how this would work in a hypothetical scenario:

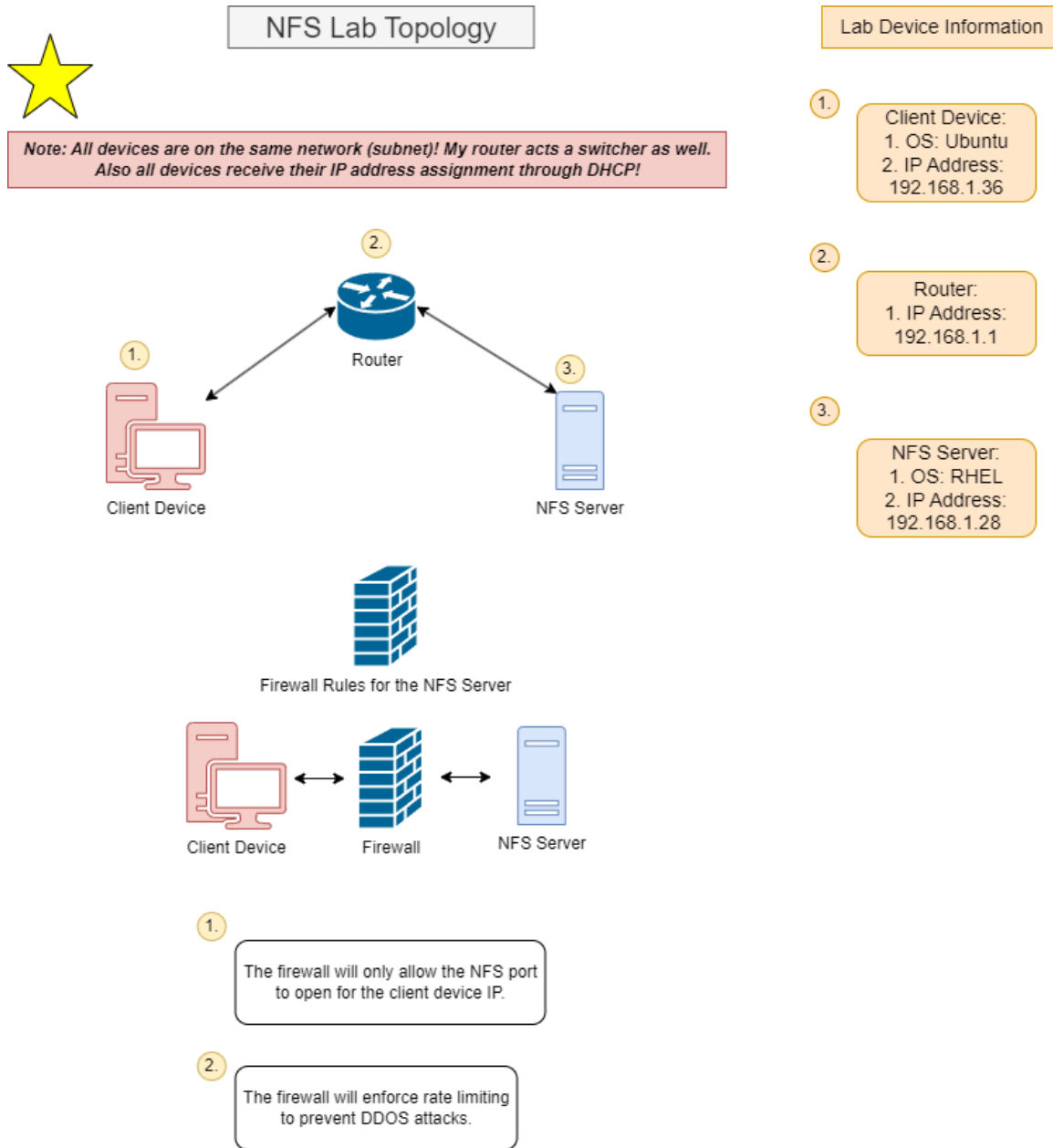
Say an Ubuntu client requests to read the file testfile.txt on an NFS server on the network it is on. The Client device will convert this read request into an RPC message and send it across the network to the NFS server. When the NFS server receives the RPC message from the Ubuntu client it will first interpret the RPC message. Once the NFS server finds out that the Ubuntu client requested to read testfile.txt, it will perform that action on its own filesystem. Once the read operation is done the NFS server creates a response to send back to the client, and this response will be the read data from testfile.txt. The client will receive the response back and will be able to view the data from testfile.txt. ***NFS works by having the server perform the requested task, and sending the result back to the client device. It might appear as though you are actually viewing the file or writing to it, but it is all an illusion since the server is the one performing the task.***

#### 1.1b. Why is NFS used?

NFS is used to provide a convenient way for users to access files on a remote server. The benefits of using NFS are the following:

1. Workstations and client devices won't have to use up disk space storing commonly used files. You can have a centralized storage location for key files used by multiple departments or groups.
2. NFS works across different systems. NFS can work across Linux, MacOS, and Windows OS making it a great service to use in a mixed computing environment.
3. NFS allows a company to control their data. Unlike cloud file sharing services like Google Cloud or DropBox the company does not have to store their data on a third party company's server. This makes NFS a useful service for sensitive industries like defense, government, and healthcare where strict data security rules must be implemented.

### 1.1d. Lab Topology:



In this lab I will use my RHEL vm as the NFS server and my Ubuntu vm as the NFS client. These devices are on my local network on the same subnet. For communication between the devices my router is used as a switch since there is no need for routing in between networks.

## Section #2 NFS Server Setup:

2.1. Make sure necessary packages are installed and start the NFS Service. Set it to run at boot time:

2.1a I will first check if NFS is installed on my RHEL vm. If it is not installed I will install it.

```
root@localhost:~  
[root@localhost ~]# dnf list --installed | grep nfs  
[root@localhost ~]#
```

I checked if any packages related to NFS are installed, and there are none installed.

2.1b. Now I will install NFS. The necessary package is called nfs-utils.

```

root@localhost:~]# dnf install nfs-utils
Updating Subscription Management repositories.
Extra Packages for Enterprise Linux 9 - x86_64                54 kB/s | 35 kB      00:00
Extra Packages for Enterprise Linux 9 - x86_64                4.0 MB/s | 22 MB     00:05
MySQL 8.4 LTS Community Server                                6.4 kB/s | 2.6 kB    00:00
MySQL Connectors Community                                    5.4 kB/s | 2.6 kB    00:00
MySQL Tools 8.4 LTS Community                                8.2 kB/s | 2.6 kB    00:00
Dependencies resolved.

=====
Package                Arch      Version              Repository              Size
=====
Installing:
nfs-utils               x86_64    1:2.5.4-25.el9       rhel-9-for-x86_64-baseos-rpms 463 k
Installing dependencies:
gssproxy                x86_64    0.8.4-6.el9          rhel-9-for-x86_64-baseos-rpms 114 k
keyutils                 x86_64    1.6.3-1.el9           rhel-9-for-x86_64-baseos-rpms 78 k
libev                    x86_64    4.33-5.el9            rhel-9-for-x86_64-baseos-rpms 56 k
libnfsidmap              x86_64    1:2.5.4-25.el9       rhel-9-for-x86_64-baseos-rpms 66 k
libverto-libev           x86_64    0.3.2-3.el9           rhel-9-for-x86_64-baseos-rpms 15 k
rpcbind                  x86_64    1.2.6-7.el9           rhel-9-for-x86_64-baseos-rpms 62 k
sssd-nfs-idmap           x86_64    2.9.4-6.el9_4.1       rhel-9-for-x86_64-baseos-rpms 44 k

Transaction Summary
=====
Install 8 Packages

Total download size: 897 k
Installed size: 2.1 M
Is this ok [y/N]:

```

Verify the packages were installed.

```
root@localhost:~  
[root@localhost ~]# dnf list --installed | grep nfs  
libnfsidmap.x86_64 1:2.5.4-25.el9 @rhel-9-for-x86_64-baseos-rpms  
nfs-utils.x86_64 1:2.5.4-25.el9 @rhel-9-for-x86_64-baseos-rpms  
sssd-nfs-idmap.x86_64 2.9.4-6.el9_4.1 @rhel-9-for-x86_64-baseos-rpms  
[root@localhost ~]#
```

2.1c Start the NFS service and ensure that it runs at boot time.

```
root@localhost:~  
[root@localhost ~]# systemctl enable --now nfs-server.service  
Created symlink /etc/systemd/system/multi-user.target.wants/nfs-server.service → /usr/lib/systemd/system/nfs-server.service.  
[root@localhost ~]#
```

I used `systemctl enable --now` to not only enable the NFS service to start running when the vm boots but to also make the service start running right away as well during the current boot.

```
root@localhost:~  
[root@localhost ~]# systemctl status nfs-server.service  
● nfs-server.service - NFS server and services  
   Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; enabled; preset: disabled)  
   Active: active (exited) since Thu 2024-08-08 13:39:20 PDT; 2min 28s ago  
     Docs: man:rpc.nfsd(8)  
           man:exportfs(8)  
  Main PID: 13570 (code=exited, status=0/SUCCESS)  
    CPU: 97ms  
  
Aug 08 13:39:19 localhost.localdomain systemd[1]: Starting NFS server and services...  
Aug 08 13:39:20 localhost.localdomain systemd[1]: Finished NFS server and services.  
[root@localhost ~]#
```

I then used `systemctl status` to check if NFS is enabled and currently running, and it is.

2.2. Check the versions of NFS supported by the server:

```
root@localhost:~  
[root@localhost ~]# cat /proc/fs/nfsd/versions  
+3 +4 +4.1 +4.2  
[root@localhost ~]# clear
```

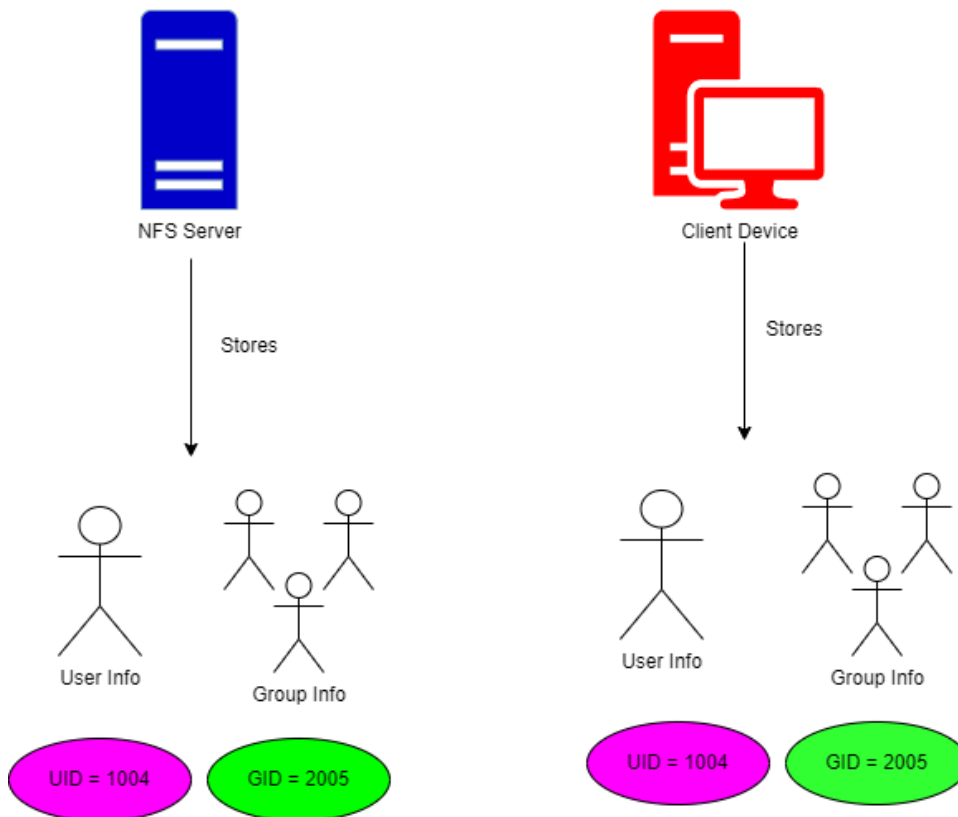
2.3. Create the necessary users and groups for NFS to function.

How do NFS user permissions work?



## NFS Permissions

Both the NFS Server and Client server will store user and group info!  
NFS requires the server and the client devices to have the same users and groups **based on uid and gid, not username or groupname!**



NFS requires the server and the client to both have users and groups with the same uids and gids. This mapping allows the server to know who can access the share on the client device and what permissions they have for the share contents.

```
root@localhost:~  
[root@localhost ~]# useradd nfsuser && groupadd nfsgroup  
[root@localhost ~]# usermod -u 2000 nfsuser && groupmod -g 2001 nfsgroup  
[root@localhost ~]# usermod -g nfsgroup nfsuser  
[root@localhost ~]# passwd nfsuser  
Changing password for user nfsuser.  
New password:  
Retype new password:  
passwd: all authentication tokens updated successfully.  
[root@localhost ~]#
```

Make sure you specify the uid and gid for the user and group! I also added the nfsuser to the nfsgroup and made nfsgroup their primary group.

Verify the user and group info:

```
root@localhost:~  
[root@localhost ~]# id nfsuser  
uid=2000(nfsuser) gid=2001(nfsgroup) groups=2001(nfsgroup)  
[root@localhost ~]#
```

2.4. Create directories and files you want to share through NFS along with necessary ownership and permissions:

***This lab will assume that the nfsuser will be the user owner of the nfs export directories and their files. Nfsgroup will be the group owner as well.***

2.4a Create the root export directory.

```
root@localhost:/home/nfsuser  
[root@localhost nfsuser]# mkdir nfs_exports & chown nfsuser:nfsgroup nfs_exports  
[1] 13925  
[1]+  Done                  mkdir nfs_exports  
[root@localhost nfsuser]# ls -l  
total 0  
drwxr-xr-x. 2 nfsuser nfsgroup 6 Aug  8 14:00 nfs_exports  
[root@localhost nfsuser]# chmod 740 nfs_exports/  
[root@localhost nfsuser]# ls -l  
total 0  
drwxr----. 2 nfsuser nfsgroup 6 Aug  8 14:00 nfs_exports  
[root@localhost nfsuser]#
```

2.4b Create a subdirectory under the /home/nfsuser/nfs\_exports/ directory.

```
root@localhost:/home/nfsuser/nfs_exports

[root@localhost nfs_exports]# mkdir spy_data
[root@localhost nfs_exports]# ls -l
total 0
drwxr-xr-x. 2 root root 6 Aug  8 14:04 spy_data
[root@localhost nfs_exports]# chown -R nfsuser:nfsgroup spy_data/
[root@localhost nfs_exports]# ls -l
total 0
drwxr-xr-x. 2 nfsuser nfsgroup 6 Aug  8 14:04 spy_data
[root@localhost nfs_exports]# chmod 740 spy_data/
[root@localhost nfs_exports]# ls -l
total 0
drwxr-----. 2 nfsuser nfsgroup 6 Aug  8 14:04 spy_data
[root@localhost nfs_exports]#
```

2.4c Create the files that will be shared through nfs.

```
root@localhost:/home/nfsuser/nfs_exports/spy_data

[root@localhost spy_data]# touch spy_data{1..10}.txt && for i in {1..10}; do echo "This document
is spy_data$i! DO NOT MENTION THE CONTENTS OF THIS FILE!" > spy_data$i.txt; done
[root@localhost spy_data]# ls -l
total 40
-rw-r--r--. 1 root root 71 Aug  8 14:14 spy_data10.txt
-rw-r--r--. 1 root root 70 Aug  8 14:14 spy_data1.txt
-rw-r--r--. 1 root root 70 Aug  8 14:14 spy_data2.txt
-rw-r--r--. 1 root root 70 Aug  8 14:14 spy_data3.txt
-rw-r--r--. 1 root root 70 Aug  8 14:14 spy_data4.txt
-rw-r--r--. 1 root root 70 Aug  8 14:14 spy_data5.txt
-rw-r--r--. 1 root root 70 Aug  8 14:14 spy_data6.txt
-rw-r--r--. 1 root root 70 Aug  8 14:14 spy_data7.txt
-rw-r--r--. 1 root root 70 Aug  8 14:14 spy_data8.txt
-rw-r--r--. 1 root root 70 Aug  8 14:14 spy_data9.txt
[root@localhost spy_data]# for file in *; do chown nfsuser:nfsgroup "$file"; done
[root@localhost spy_data]# ls -l
total 40
-rw-r--r--. 1 nfsuser nfsgroup 71 Aug  8 14:14 spy_data10.txt
-rw-r--r--. 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data1.txt
-rw-r--r--. 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data2.txt
-rw-r--r--. 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data3.txt
-rw-r--r--. 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data4.txt
-rw-r--r--. 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data5.txt
-rw-r--r--. 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data6.txt
-rw-r--r--. 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data7.txt
-rw-r--r--. 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data8.txt
-rw-r--r--. 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data9.txt
[root@localhost spy_data]#
```

I used some bash scripting to make the process a lot easier. My first script creates 10 spy\_data.txt files numbered from 1-10. It then loops through each file and appends some text to them. The second script changes the permissions for these 10 files using a loop structure as well.

Now I will change the permissions for these files:



```
[root@localhost spy_data]# for file in *; do chmod 740 "$file"; done
[root@localhost spy_data]# ls -l
total 40
-rwxr-----. 1 nfsuser nfsgroup 71 Aug  8 14:14 spy_data10.txt
-rwxr-----. 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data1.txt
-rwxr-----. 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data2.txt
-rwxr-----. 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data3.txt
-rwxr-----. 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data4.txt
-rwxr-----. 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data5.txt
-rwxr-----. 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data6.txt
-rwxr-----. 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data7.txt
-rwxr-----. 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data8.txt
-rwxr-----. 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data9.txt
[root@localhost spy_data]#
```

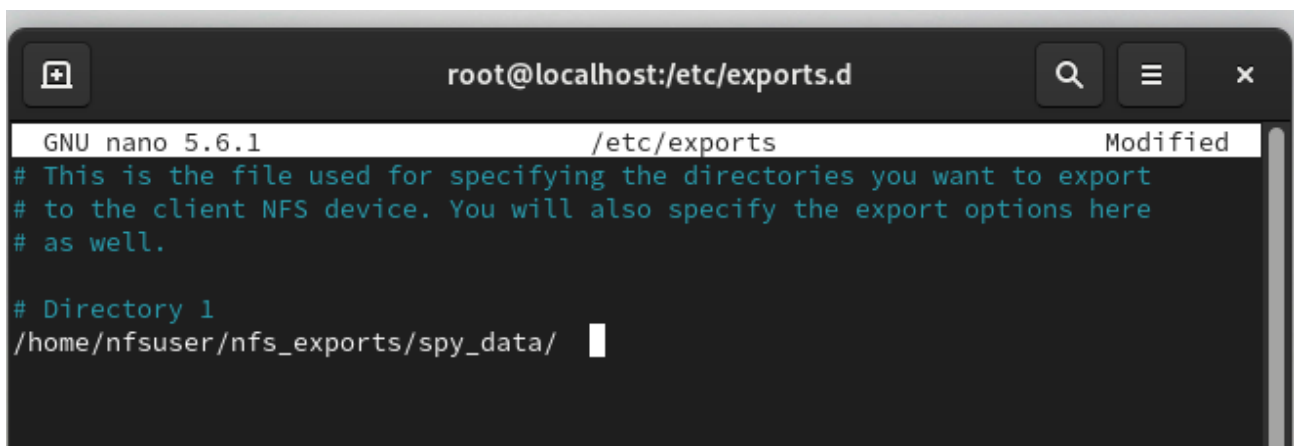
Lastly, my third bash script loops through the files in the current directory and changes their permissions from 644 to 740.

## Section #3 Configuring NFS Exports:

3.1. Edit the `/etc/exports` file to share the created files and directories:

In order for NFS to work on the server you must specify what directory paths you want to share with clients in the `/etc/exports` configuration file. In this configuration file you will also specify the clients you want each path to be shared with and the export options you want to use.

3.1a. Add the directory path you want to share:



```
root@localhost:/etc/exports.d
GNU nano 5.6.1 /etc/exports Modified
# This is the file used for specifying the directories you want to export
# to the client NFS device. You will also specify the export options here
# as well.
# Directory 1
/home/nfsuser/nfs_exports/spy_data/
```

3.1b. Add the necessary export options for the directories:  
NFS export options:

Export Option	Purpose	Example
rw	RW only grants clients read and write permissions for the files exported to the client device.	Say test1.txt was exported with rw, but it had rwx. On the client device users will only have rw. It is kind of like a netmask for limiting permissions.
ro	RO only grants clients the ability to read the contents of the files and directories exported to them.	Say test1.txt was exported with only ro. On the client device the users will only be able to read the contents of the file and its directory.
no_subtree_check	No_subtree_check will not check the permissions of the full directory tree above the file a client is trying to access. This setting will force NFS to assume that the user has permissions for the directory and subdirectories.	Say test1.txt is under the /exports/data/ directory. No_subtree_check will simply tell NFS to not check if the user on the client has permissions for the full directory tree, and to only check permissions for the files. <b>No_subtree_check will help improve nfs performance, but it comes with security risks.</b>
subtree_check	Subtree_check will actually perform the full directory tree permissions check to ensure that a user is not accessing something they shouldn't be accessing.	For instance, say test1.txt has the /data/tests/test1.txt path. With subtree check NFS will check if users have access to the /data/tests subdirectory and if they have access to the parent /data directory.

root_squash	Root squash will cause client users with root privileges (root/ users in sudo group) to have their privileges “squashed” or simply ignored when accessing NFS shares. Since privileged users could typically enter any directory they want, having root_squash will prevent abuse of this privilege.	Say on a client device user admin with sudo privileges wants to view files in the /data/secret_files/ NFS share. The NFS share is configured to technically only allow nfsuser to access it. Admin would try to access the share using sudo -i to gain root privileges, but root_squash will prevent this from allowing Admin to access the NFS share.
no_root_squash	No_root_squash will allow users on the client device to maintain their privileged permissions. Sp users that have sudo privileges can use the privileges to access the NFS share even if they shouldn't be allowed to.	For example say on a client device a user named testuser has sudo privileges. If the client device has an NFS mounted to it, the testuser can access the files in the NFS share directory even if they do not map to a uid on the server.
sync	Sync will only confirm write operations once the data has been written to the physical storage device ( ssd, hard drive). This is vital for preventing data losses. The only downside is that sync can cause NFS write operations to appear slower since the server will only acknowledge the write operation if it successfully writes data to the disk.	
async	Async will confirm the write operation once the write data is stored on the server memory (ram). Async is faster than sync, but at the cost of guaranteeing that the data is actually written on the physical storage device. If	

	something happens to the server such as a shutdown, restart, or crash, the data that was meant to be written would be lost.	
--	---	--

These are the export options I have chosen:

```

root@localhost:/etc/exports.d
GNU nano 5.6.1 /etc/exports Modified
# This is the file used for specifying the directories you want to export
# to the client NFS device. You will also specify the export options here
# as well.

# Directory 1
/home/nfsuser/nfs_exports/spy_data/ *(rw,sync,no_subtree_check,root_squash)

```

***The \* symbol represents any networked device.***

3.2. Apply the export configuration and verify it as well:

```

root@localhost:/etc/exports.d
[root@localhost exports.d]# exportfs -ra
[root@localhost exports.d]# exportfs -v
/home/nfsuser/nfs_exports/spy_data
    <world> (sync,wdelay,hide,no_subtree_check,sec=sys,rw,secure,root_squash,no_all_squash)
[root@localhost exports.d]#

```

The flags I used for the exportfs command are r,a, and v. The -r flag reloads export settings. The -a flag exports all directories listed on the /etc/exports file. Lastly, -v is used to verify what you are exporting to the client.

## Section #4 NFS Client Setup:

4.1. Check if the NFS packages are installed on the client device. If not, install them:

```
root@Laptop: ~  
root@Laptop:~# apt list --installed | grep nfs-common  
  
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.  
  
root@Laptop:~#
```

NFS is not installed on my Ubuntu client device. I will have to install it.

```
root@Laptop: ~  
root@Laptop:~# apt install nfs-common  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following packages were automatically installed and are no longer required:  
  doc-base libasn1-8-heimdal libblas3 libbcjson1 libdlt2 libev4  
  libgssapi3-heimdal libgvm21 libhcrypto4-heimdal libhdb9-heimdal  
  libheimbase1-heimdal libheimntlm0-heimdal libhx509-5-heimdal libjemalloc2  
  libjs-sphinxdoc libjs-underscore libkrb5-26-heimdal liblinear4 liblua5.1-0  
  liblzfl1 libmosquitto1 libopenvas-wmiclient22 libroken18-heimdal  
  libssh-gcrypt-4 libuuid-perl libwebsockets16 libwind0-heimdal libwpe-1.0-1  
  libwpebackend-fdo-1.0-1 libyaml-tiny-perl linux-headers-6.5.0-26-generic  
  linux-hwe-6.5-headers-6.5.0-26 linux-image-6.5.0-26-generic  
  linux-modules-6.5.0-26-generic linux-modules-extra-6.5.0-26-generic  
  lua-bitop lua-cjson lua-lpeg mosquitto nmap nmap-common notus-scanner  
  openvas-scanner openvas-smb ospd-openvas pnsnscan python-babel-localedata  
  python3-babel python3-defusedxml python3-deprecated python3-dnspython  
  python3-flask python3-gnupg python3-impacket python3-itsdangerous  
  python3-jinja2 python3-ldap3 python3-ldapdomaindump python3-ospd  
  python3-packaging python3-paho-mqtt python3-psutil python3-pyasn1  
  python3-pycryptodome python3-redis python3-requests-toolbelt  
  python3-simplejson python3-terminaltables python3-werkzeug python3-wrapt  
  redis-server redis-tools  
Use 'apt autoremove' to remove them.  
The following additional packages will be installed:  
  keyutils libevent-core-2.1-7 libnfsidmap1 rpcbind  
Suggested packages:  
  open-iscsi watchdog  
The following NEW packages will be installed:  
  keyutils libevent-core-2.1-7 libnfsidmap1 nfs-common rpcbind  
0 upgraded, 5 newly installed, 0 to remove and 8 not upgraded.  
186 not fully installed or removed.  
Need to get 475 kB of archives.  
After this operation, 1,709 kB of additional disk space will be used.  
Do you want to continue? [Y/n]
```

Verify that NFS is installed and client commands are present:

```
root@Laptop: ~  
root@Laptop:~# apt list --installed | grep nfs-common  
  
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.  
  
nfs-common/jammy-updates,now 1:2.6.1-1ubuntu1.2 amd64 [installed]  
root@Laptop:~#
```

```
root@Laptop: ~  
root@Laptop:~# which mount.nfs  
/usr/sbin/mount.nfs  
root@Laptop:~#
```

4.2. Create the necessary user and group, *ensuring that the uid and gid matches the ones on the NFS server.*

Note that the names for groups and users on the client and servers do not have to match, only the gids and uids!

```
root@Laptop: /  
root@Laptop:/# useradd nfsuser && groupadd nfsgroup  
root@Laptop:/# usermod -u 2000 nfsuser && groupmod -g 2001 nfsgroup  
root@Laptop:/# usermod -g nfsgroup nfsuser  
root@Laptop:/# passwd nfsuser  
New password:  
Retype new password:  
passwd: password updated successfully  
root@Laptop:/#
```

4.3. Mount the NFS share:

4.3a. Create a mount point to mount the share on:

Since NFS is literally named Network File System, like any file system on Linux it must be mounted on a directory so you can access its data. The mount point I will create will be directly under / (root). It is going to be called `nfs_mount` and it will be owned by `nfsuser` and the `nfsgroup` (not root even though it is at the top of the filesystem hierarchy).

```
root@Laptop: /
root@Laptop: /# mkdir nfs_mount && chown nfsuser:nfsgroup nfs_mount
root@Laptop: /# ls -l
total 2744404
lrwxrwxrwx 1 root root 7 Mar 3 17:17 bin -> usr/bin
drwxr-xr-x 4 root root 4096 Aug 9 13:59 boot
drwxrwxr-x 2 root root 4096 Mar 3 17:23 cdrom
drwxr-xr-x 19 root root 4240 Aug 9 13:53 dev
drwxr-xr-x 148 root root 12288 Aug 9 14:06 etc
drwxr-xr-x 6 root root 4096 Mar 10 16:31 home
lrwxrwxrwx 1 root root 7 Mar 3 17:17 lib -> usr/lib
lrwxrwxrwx 1 root root 9 Mar 3 17:17 lib32 -> usr/lib32
lrwxrwxrwx 1 root root 9 Mar 3 17:17 lib64 -> usr/lib64
lrwxrwxrwx 1 root root 10 Mar 3 17:17 libx32 -> usr/libx32
drwx----- 2 root root 16384 Mar 3 17:17 lost+found
drwxr-xr-x 2 root root 4096 Feb 20 11:22 media
drwxr-xr-x 2 root root 4096 Feb 20 11:22 mnt
drwxr-xr-x 2 nfsuser nfsgroup 4096 Aug 9 14:13 nfs_mount
drwxr-xr-x 2 root root 4096 Feb 20 11:22 opt
dr-xr-xr-x 275 root root 0 Aug 9 13:11 proc
drwx----- 8 root root 4096 Mar 22 17:53 root
drwxr-xr-x 46 root root 1280 Aug 9 14:13 run
lrwxrwxrwx 1 root root 8 Mar 3 17:17/sbin -> usr/sbin
drwxr-xr-x 11 root root 4096 Feb 20 11:28 snap
drwxr-xr-x 2 root root 4096 Feb 20 11:22 srv
-rw----- 1 root root 2810183680 Mar 3 17:17 swapfile
dr-xr-xr-x 13 root root 0 Aug 9 13:11 sys
drwxrwxrwt 24 root root 4096 Aug 9 13:56 tmp
drwxr-xr-x 14 root root 4096 Feb 20 11:22 usr
drwxr-xr-x 14 root root 4096 Feb 20 11:27 var
root@Laptop: /#
```

I will now change the permissions for the mount from 755 to 770 since I want nfsuser and nfsgroup to both have rwx permissions for the mount point.

```
root@Laptop: /
root@Laptop: /# chmod 770 nfs_mount/
root@Laptop: /# ls -ld nfs_mount/
drwxrwx--- 2 nfsuser nfsgroup 4096 Aug 9 14:13 nfs_mount/
root@Laptop: /#
```

#### 4.3b. Mount the NFS share:

I will now mount the NFS export onto the /nfs\_mount/ directory. Here are some of the NFS mount options. These are the mount options I have chosen:

```
root@Laptop: /
root@Laptop: /# mount -t nfs -o tcp,nosuid,noexec,vers=4,soft 192.168.1.28:
/home/nfsuser/nfs_exports/spy_data /nfs_mount
root@Laptop: /#
```

I chose the tcp, nosuid, noexec, vers=4, and soft options. The tcp option will specify that this client should communicate with the NFS server using the tcp protocol. Nosuid means that files in the share with the setuid bit set won't execute with elevated permissions. Noexec prevents binary files from executing on the device. Vers=4 is used to specify the NFS version I want to use. Lastly soft is an option that specifies if the server is unreachable the client will try a few times to reconnect before displaying an error and giving up.

#### 4.3c. Verify the mount:

```
root@Laptop: /
root@Laptop:/# df -h
Filesystem                Size      Used Avail Use% Mounted on
tmpfs                     272M        1.8M  270M   1% /run
/dev/sda3                  24G        17G   6.1G  74% /
tmpfs                     1.4G        28K   1.4G   1% /dev/shm
tmpfs                     5.0M        4.0K   5.0M   1% /run/lock
/dev/sda2                  512M        6.1M  506M   2% /boot/efi
tmpfs                     272M       120K   272M   1% /run/user/1001
192.168.1.28:/home/nfsuser/nfs_exports/spy_data 22G       5.9G   16G  28% /nfs_mount
root@Laptop:/#
```

#### 4.4. Test actions against the NFS share. Try to read/ write to it and create files in it:

##### 4.4a. Try to access the NFS share as nfsuser on the client:

```
spy2@Laptop: ~
$ cd /nfs_mount
$ ls -l
total 40
-rwxr----- 1 nfsuser nfsgroup 71 Aug  8 14:14 spy_data10.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data1.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data2.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data3.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data4.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data5.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data6.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data7.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data8.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data9.txt
$ whoami
nfsuser
$
```



4.4b. Try to read and write to files in the NFS Share:

```
spy2@Laptop: ~  
$ cd /nfs_mount  
$ ls -l  
total 40  
-rwxr----- 1 nfsuser nfsgroup 71 Aug  8 14:14 spy_data10.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data1.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data2.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data3.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data4.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data5.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data6.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data7.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data8.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data9.txt  
$ cat spy_data1.txt  
This document is spy_data1! DO NOT MENTION THE CONTENTS OF THIS FILE!  
$
```

I can read the contents of the files! Now time to test writing to them!

```
GNU nano 6.2 spy_data1.txt *  
This document is spy_data1! DO NOT MENTION THE CONTENTS OF THIS FILE!  
  
Hello from nfsuser on the Ubuntu client machine!  
  
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location  
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line
```

```
spy2@Laptop: ~  
$ ls -l  
total 40  
-rwxr----- 1 nfsuser nfsgroup 71 Aug 8 14:14 spy_data10.txt  
-rwxr----- 1 nfsuser nfsgroup 120 Aug 9 14:50 spy_data1.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug 8 14:14 spy_data2.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug 8 14:14 spy_data3.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug 8 14:14 spy_data4.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug 8 14:14 spy_data5.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug 8 14:14 spy_data6.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug 8 14:14 spy_data7.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug 8 14:14 spy_data8.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug 8 14:14 spy_data9.txt  
$ cat spy_data1.txt  
This document is spy_data1! DO NOT MENTION THE CONTENTS OF THIS FILE!  
  
Hello from nfsuser on the Ubuntu client machine!  
$
```

I was able to write to the file on the NFS server!

4.5. Configure the client to automount the NFS share on boot with Autofs:

*\*Note that the /nfs\_mount/ directory used as the mount point before in section 4.4 is not going to be used anymore! I deleted it after demonstrating manually mounting NFS shares.*

4.5a. Unmount the NFS share:

Before configuring NFS to mount automatically you need to unmount the export.

```
root@Laptop: ~  
root@Laptop:~# umount /nfs_mount  
root@Laptop:~# df -h  
Filesystem      Size  Used Avail Use% Mounted on  
tmpfs            272M  1.8M  270M   1% /run  
/dev/sda3        24G   17G   6.1G  74% /  
tmpfs            1.4G   28K   1.4G   1% /dev/shm  
tmpfs            5.0M   4.0K   5.0M   1% /run/lock  
/dev/sda2        512M   6.1M   506M   2% /boot/efi  
tmpfs            272M  128K   272M   1% /run/user/1001  
root@Laptop:~#
```

4.5b. Install autofs if it is not installed already:

Autofs is not installed, so I installed it onto the Ubuntu client:

```
root@Laptop: ~  
root@Laptop:~# apt install autofs  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following packages were automatically installed and are no longer required:  
  doc-base libasn1-8-heimdal libblas3 libcjson1 libdlt2 libev4  
  libgssapi3-heimdal libgvm21 libhcrypto4-heimdal libhdb9-heimdal  
  libheimbase1-heimdal libheimntlm0-heimdal libhx509-5-heimdal libjemalloc2  
  libjs-sphinxdoc libjs-underscore libkrb5-26-heimdal liblinear4 liblua5.1-0  
  liblzfl1 libmosquitto1 libopenvas-wmiclient2 libroken18-heimdal  
  libssh-gcrypt-4 libuuid-perl libwebsockets16 libwind0-heimdal libwpe-1.0-1  
  libwpebackend-fdo-1.0-1 libyaml-tiny-perl linux-headers-6.5.0-26-generic  
  linux-hwe-6.5-headers-6.5.0-26 linux-image-6.5.0-26-generic  
  linux-modules-6.5.0-26-generic linux-modules-extra-6.5.0-26-generic  
  lua-bitop lua-cjson lua-lpeg mosquitto nmap nmap-common notus-scanner  
  openvas-scanner openvas-smb ospd-openvas pncan python-babel-localedata  
  python3-babel python3-defusedxml python3-deprecated python3-dnspython  
  python3-flask python3-gnupg python3-impacket python3-itsdangerous  
  python3-jinja2 python3-ldap3 python3-ldapdomaindump python3-ospd  
  python3-packaging python3-paho-mqtt python3-psutil python3-pyasn1  
  python3-pycryptodome python3-redis python3-requests-toolbelt  
  python3-simplejson python3-terminaltables python3-werkzeug python3-wrapt  
  redis-server redis-tools  
Use 'apt autoremove' to remove them.  
The following NEW packages will be installed:  
  autofs  
0 upgraded, 1 newly installed, 0 to remove and 8 not upgraded.
```

4.5c. Edit the /etc/auto.master file, adding the mount point, and /etc/auto.nfs to define the nfs shares. You can also include a timeout period for limiting how long the nfs export will remain mounted on the client device without interaction or access. I am using a new mount point, /misc.

```
root@Laptop: /nfs_mount  
GNU nano 6.2 /etc/auto.master *  
# Sample auto.master file  
# This is a 'master' automounter map and it has the following format:  
# mount-point [map-type[,format]:]map [options]  
# For details of the format look at auto.master(5).  
#  
/misc /etc/auto.misc
```

4.5d. Create the spy\_data subdirectory under /misc/ directory. Then edit the /etc/auto.misc file.

Create the /misc/spy\_data subdirectory:

```
root@Laptop: /  
root@Laptop:/# mkdir -p /misc/spy_data/  
root@Laptop:/# ls -l misc/spy_data/  
total 0  
root@Laptop:/#
```

Edit the /etc/ auto.misc file:

```
GNU nano 6.2 /etc/auto.misc *  
#  
# This is an automounter map and it has the following format  
# key [ -mount-options-separated-by-comma ] location  
# Details may be found in the autofs(5) manpage  
  
cd -fstype=iso9660,ro,nosuid,nodev :/dev/cdrom  
  
# the following entries are samples to pique your imagination  
#linux -ro,soft ftp.example.org:/pub/linux  
#boot -fstype=ext2 :/dev/hda1  
#floppy -fstype=auto :/dev/fd0  
#floppy -fstype=ext2 :/dev/fd0  
#e2floppy -fstype=ext2 :/dev/fd0  
#jaz -fstype=ext2 :/dev/sdc1  
#removable -fstype=ext2 :/dev/hdd  
spy_data -fsytpe=nfs,soft,tcp 192.168.1.28:/home/nfsuser/nfs_exports/spy_data
```

4.5e. Restart the autofs service:

```
root@Laptop: ~  
root@Laptop:~# systemctl restart autofs  
root@Laptop:~#
```

4.6f. Test the mount to ensure you can access the files.

As you can see, as the nfsuser I am able to list the contents of the spy\_data NFS share. You can even see that the NFS export is mounted on my device.

```
spy2@Laptop: /nfs_mount
$ ls -l /misc/spy_data
total 40
-rwxr----- 1 nfsuser nfsgroup 71 Aug  8 14:14 spy_data10.txt
-rwxr----- 1 nfsuser nfsgroup 120 Aug  9 14:50 spy_data1.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data2.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data3.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data4.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data5.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data6.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data7.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data8.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data9.txt
$ whoami
nfsuser
$ df -h
Filesystem                                Size  Used Avail Use% Mounted on
tmpfs                                     272M  1.7M  270M   1% /run
/dev/sda3                                24G   16G   7.6G  67% /
tmpfs                                     1.4G   28K   1.4G   1% /dev/shm
tmpfs                                     5.0M   4.0K   5.0M   1% /run/lock
/dev/sda2                                512M   6.1M  506M   2% /boot/efi
tmpfs                                     272M  132K  272M   1% /run/user/2000
192.168.1.28:/home/nfsuser/nfs_exports/spy_data 22G   5.9G   16G  28% /misc/spy_data
$
```

I can even view the contents of the files!

```
spy2@Laptop: /
$ cd /misc/spy_data/
$ ls -l
total 40
-rwxr----- 1 nfsuser nfsgroup 71 Aug  8 14:14 spy_data10.txt
-rwxr----- 1 nfsuser nfsgroup 120 Aug  9 14:50 spy_data1.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data2.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data3.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data4.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data5.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data6.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data7.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data8.txt
-rwxr----- 1 nfsuser nfsgroup 70 Aug  8 14:14 spy_data9.txt
$ cat spy_data1.txt
This document is spy_data1! DO NOT MENTION THE CONTENTS OF THIS FILE!

Hello from nfsuser on the Ubuntu client machine!
$
```

## Section #5 NFS Performance and Security:

6.1. Monitor NFS performance using various tools:

### 6.1a Use the nfsstat to view client statistics:

```
spy2@Laptop: /  
$ nfsstat -c  
Client rpc stats:  
calls      retrans    authrefrsh  
150         0          150  
  
Client nfs v4:  
null        read        write       commit      open  
2           1%         0           0%          0           0%  
open_conf   open_noat   open_dgrd   close       setattr  
0           0%         0           0%          1           0%  
fsinfo      renew       setclntid   confirm     lock  
12          8%         0           0%          0           0%  
lockt       locku       access      getattr     lookup  
0           0%         0           0%          13          8%          28          18%          20          13%  
lookup_root remove      rename      link        symlink  
2           1%         0           0%          0           0%          0           0%  
create      pathconf   statfs      readlink    readdir  
0           0%         10          6%          1           0%          0           0%          2           1%  
server_caps delegreturn getacl      setacl      fs_locations  
22          14%        1           0%          0           0%          0           0%  
rel_lkowner secinfo    fsid_present exchange_id create_session  
0           0%         0           0%          0           0%          4           2%          2           1%  
destroy_session sequence   get_lease_time reclaim_comp layoutget  
1           0%         22          14%        0           0%          2           1%          0           0%  
getdevinfo  layoutcommit layoutreturn secinfo_no  test_stateid  
0           0%         0           0%          0           0%          2           1%          0           0%  
free_stateid getdevicelist bind_conn_to_ses destroy_clientid seek  
0           0%         0           0%          0           0%          1           0%          0           0%  
allocate    deallocate  layoutstats clone        0  
0           0%         0           0%          0           0%          0           0%  
  
$
```

6.1b. Use nfsstat to view server statistics:

```
Server nfs v4 operations:
op0-unused      op1-unused      op2-future      access          close
0               0               0               15              1
0               0%              0%              3%              0%
commit          create          delegpurge       delegreturn     getattr
0               0               0               1               108
0               0%              0%              0%              23%
getfh           link           lock             lockt           locku
18              3%              0               0               0
lookup          lookup_root    nverify          open            openattr
20              4%              0               1               0
open_conf       open_dgrd      putfh            putpubfh        putrootfh
0               0               114             0               4
read            readdir        readlink         remove          rename
1               0%              0               0               0
renew           restorefh      savefh           secinfo         setattr
0               0               0               0               0
setcltid        setcltidconf   verify           write           rellockowner
0               0               0               0               0
bc_ctl          bind_conn      exchange_id      create_ses      destroy_ses
0               0               4               2               1
free_stateid    getdirdeleg    getdevinfo       getdevlist      layoutcommit
0               0               0               0               0
layoutget       layoutreturn   secinfonyonam    sequence        set_ssv
0               0               2               156             0
test_stateid    want_deleg     destroy_clid     reclaim_comp     allocate
0               0               1               2               0
copy            copy_notify    deallocate        ioadvise        layouterror
0               0               0               0               0
layoutstats     offloadcancel  offloadstatus    readplus        seek
0               0               0               0               0
write_same
0               0%
[administrator@localhost ~]$
```

6.1d Use showmount to see the clients mounting to the server and to see the exports available on the server:

```
root@localhost:~
[root@localhost ~]# showmount --exports
Export list for localhost.localdomain:
/home/nfsuser/nfs_exports/spy_data 192.168.1.36
[root@localhost ~]#
```

6.2. Verify the latest version of NFS is being used:

You should use the latest version of NFS so that you are not susceptible to vulnerabilities and have access to the latest features.



To verify that you are currently using the latest version of NFS on the client you can use the `nfsstat -s` command:

```
Server nfs v4 operations:
op0-unused      op1-unused      op2-future      access          close
0              0              0              17              3%              1              0%
commit          create          delegpurge      delegreturn     getattr
0              0              0              1              0%              124             23%
getfh          link           lock            lockt           locku
18             3%            0              0              0%              0              0%
lookup         lookup_root    nverify        open            openattr
20             3%            0              0              0%              0              0%
open_conf      open_dgrd      putfh          putpubfh        putrootfh
0              0              130            24%            0              4              0%
read           readdir        readlink       remove          rename
1              0%            2              0              0%              0              0%
renew          restorefh      savefh         secinfo         setattr
0              0%            0              0              0%              0              0%
setcltid       setcltidconf   verify         write           rellockowner
0              0%            0              0              0%              0              0%
bc_ctl         bind_conn      exchange_id     create_ses      destroy_ses
0              0%            4              0%              2              0%              1              0%
free_stateid   getdirdeleg    getdevinfo      getdevlist      layoutcommit
0              0%            0              0%              0              0%              0              0%
layoutget      layoutreturn    secinfonyonam   sequence        set_ssv
0              0%            2              0%              197             37%            0              0%
test_stateid   want_deleg     destroy_clid    reclaim_comp    allocate
0              0%            1              0%              2              0%              0              0%
copy           copy_notify    deallocate      ioadvise        layouterror
0              0%            0              0%              0              0%              0              0%
layoutstats    offloadcancel  offloadstatus   readplus        seek
0              0%            0              0%              0              0%              0              0%
write_same
0              0%

[root@localhost ~]#
```

To verify the version being used on the client side, you can use the `mount` command as follows:

```
spy2@Laptop: ~
$ mount | grep nfs
192.168.1.28:/home/nfsuser/nfs_exports/spy_data on /misc/spy_data type nfs4 (rw,
relatime,vers=4.2,rsize=524288,wsiz=524288,namlen=255,soft,proto=tcp,timeo=600,
retrans=2,sec=sys,clientaddr=192.168.1.36,local_lock=none,addr=192.168.1.28)
$
```

6.3 Set up a firewall for the NFS Server. The firewall will restrict access to the NFS server from only the client device, it will enforce rate limiting to prevent DDoS attacks, and lastly it will log connection attempts to the server.

6.3a. Enable `firewalld` on the NFS server:



```
root@localhost:~  
[root@localhost ~]# systemctl enable --now firewalld.service  
Created symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service →  
/usr/lib/systemd/system/firewalld.service.  
Created symlink /etc/systemd/system/multi-user.target.wants/firewalld.service →  
/usr/lib/systemd/system/firewalld.service.
```

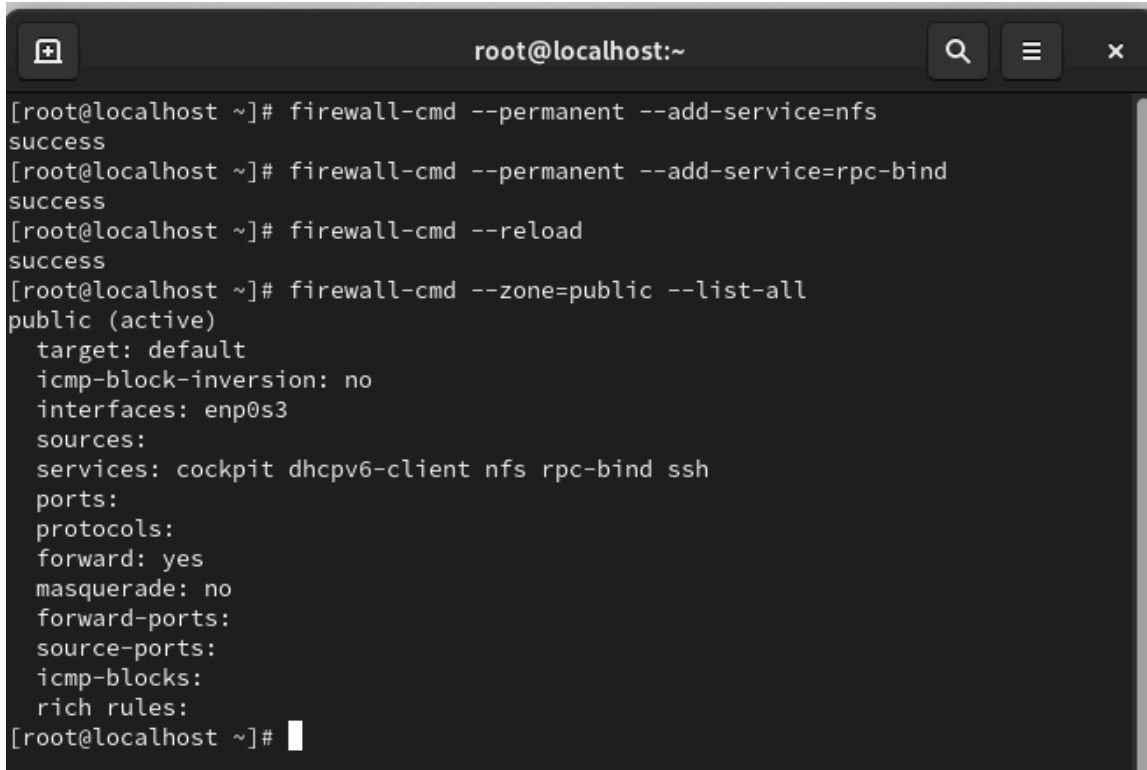
6.3b. Check what the current zone is, and what the zone is enforcing:

```
root@localhost:~  
[root@localhost ~]# firewall-cmd --get-active-zones  
public  
  interfaces: enp0s3  
[root@localhost ~]# ip addr show  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000  
    link/ether 08:00:27:af:93:38 brd ff:ff:ff:ff:ff:ff  
    inet 192.168.1.28/24 brd 192.168.1.255 scope global dynamic noprefixroute enp0s3  
        valid_lft 86112sec preferred_lft 86112sec  
    inet6 fe80::2db5:b8ac:8d1f:2f6c/64 scope link noprefixroute  
        valid_lft forever preferred_lft forever  
[root@localhost ~]#
```

```
root@localhost:~  
[root@localhost ~]# firewall-cmd --zone=public --list-all  
public (active)  
  target: default  
  icmp-block-inversion: no  
  interfaces: enp0s3  
  sources:  
  services: cockpit dhcpv6-client ssh  
  ports:  
  protocols:  
  forward: yes  
  masquerade: no  
  forward-ports:  
  source-ports:  
  icmp-blocks:  
  rich rules:  
[root@localhost ~]#
```

The RHEL server currently has the public zone set as the default zone. The only services this zone is configured to allow so far are cockpit, dhcpv6-client, and ssh.

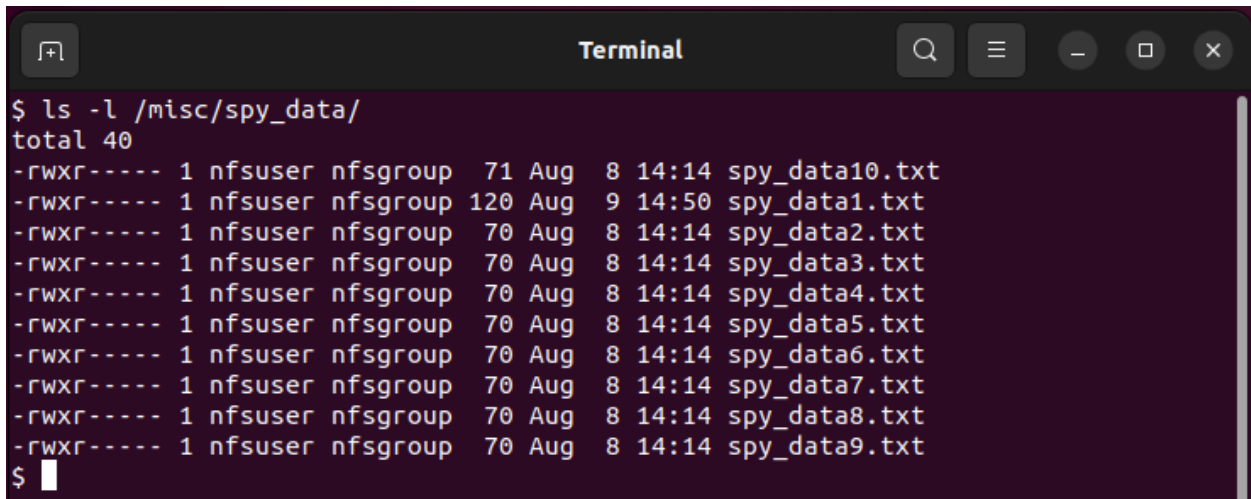
6.3c. Open the NFS and RPC Bind service ports in the public zone:



```
root@localhost:~  
[root@localhost ~]# firewall-cmd --permanent --add-service=nfs  
success  
[root@localhost ~]# firewall-cmd --permanent --add-service=rpc-bind  
success  
[root@localhost ~]# firewall-cmd --reload  
success  
[root@localhost ~]# firewall-cmd --zone=public --list-all  
public (active)  
  target: default  
  icmp-block-inversion: no  
  interfaces: enp0s3  
  sources:  
  services: cockpit dhcpv6-client nfs rpc-bind ssh  
  ports:  
  protocols:  
  forward: yes  
  masquerade: no  
  forward-ports:  
  source-ports:  
  icmp-blocks:  
  rich rules:  
[root@localhost ~]#
```

6.3d. Create a rich rule that will implement rate-limiting, only allow NFS connections to the server to occur from a designated client, and logging:

6.3e Test the firewall on the client to ensure that it works:



```
Terminal  
$ ls -l /misc/spy_data/  
total 40  
-rwxr----- 1 nfsuser nfsgroup 71 Aug 8 14:14 spy_data10.txt  
-rwxr----- 1 nfsuser nfsgroup 120 Aug 9 14:50 spy_data1.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug 8 14:14 spy_data2.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug 8 14:14 spy_data3.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug 8 14:14 spy_data4.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug 8 14:14 spy_data5.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug 8 14:14 spy_data6.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug 8 14:14 spy_data7.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug 8 14:14 spy_data8.txt  
-rwxr----- 1 nfsuser nfsgroup 70 Aug 8 14:14 spy_data9.txt  
$
```

On the client I was able to access the NFS shares after the firewall was implemented!

```
root@localhost:~  
[root@localhost ~]# journalctl | grep "nfs-access"  
Aug 15 13:57:26 localhost.localdomain kernel: nfs-accessIN=enp0s3 OUT= MAC=08:00:27:af:93:38:10:5b:ad:65:80:83:08:00 SRC=192.168.1.36 DST=192.168.1.28 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=16506 DF PROTO=TCP SPT=56843 DPT=2049 WINDOW=64240 RES=0x00 SYN URG=0  
Aug 15 13:57:27 localhost.localdomain kernel: nfs-accessIN=enp0s3 OUT= MAC=08:00:27:af:93:38:10:5b:ad:65:80:83:08:00 SRC=192.168.1.36 DST=192.168.1.28 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=2210 DF PROTO=TCP SPT=736 DPT=2049 WINDOW=64240 RES=0x00 SYN URG=0  
[root@localhost ~]#
```

On the NFS server I can even view the connection information by using journalctl and looking for the log prefix nfs-access.

#### 6.4. Specify specific network address that can mount the NFS exports:

Before I was using the \* directive for specifying the network devices that can mount the exports. The \* directives means that any network host can mount the exports. But this is not acceptable for security reasons since anyone can try to mount the exports. /

```
root@localhost:~  
GNU nano 5.6.1 /etc/exports  
# This is the file used for specifying the directories you want to export  
# to the client NFS device. You will also specify the export options here  
# as well.  
  
# Directory 1  
/home/nfsuser/nfs_exports/spy_data/ *(rw,sync,no_subtree_check,root_squash)
```

Now I will specify the exact IP address of the client device I want to allow mounting the export on:

```
root@localhost:~  
GNU nano 5.6.1 /etc/exports  
# This is the file used for specifying the directories you want to export  
# to the client NFS device. You will also specify the export options here  
# as well.  
  
# Directory 1  
/home/nfsuser/nfs_exports/spy_data/ 192.168.1.36(rw,sync,no_subtree_check,root_squash)
```

Now only the client device with the IP address 192.168.1.36 can mount the export!

#### 6.5. Use stricter mount options on the client side.

```
root@Laptop: ~  
GNU nano 6.2 /etc/auto.misc *  
#  
# This is an automounter map and it has the following format  
# key [ -mount-options-separated-by-comma ] location  
# Details may be found in the autofs(5) manpage  
  
cd -fstype=iso9660,ro,nosuid,nodev :/dev/cdrom  
  
# the following entries are samples to pique your imagination  
#linux -ro,soft ftp.example.org:/pub/linux  
#boot -fstype=ext2 :/dev/hda1  
#floppy -fstype=auto :/dev/fd0  
#floppy -fstype=ext2 :/dev/fd0  
#e2floppy -fstype=ext2 :/dev/fd0  
#jaz -fstype=ext2 :/dev/sdc1  
#removable -fstype=ext2 :/dev/hdd  
spy_data -fstype=nfs,soft,tcp,noexec,nosuid 192.168.1.28:/home/nfsuse>
```

## Section #6 Troubleshooting NFS:

### 7.1. Check NFS logs on the server:

```
root@localhost:~  
[root@localhost ~]# journalctl -u nfs-server  
Aug 08 13:39:19 localhost.localdomain systemd[1]: Starting NFS server and services...  
Aug 08 13:39:20 localhost.localdomain systemd[1]: Finished NFS server and services.  
Aug 08 15:11:21 localhost.localdomain systemd[1]: Stopping NFS server and services...  
Aug 08 15:11:21 localhost.localdomain systemd[1]: nfs-server.service: Deactivated successfully.  
Aug 08 15:11:21 localhost.localdomain systemd[1]: Stopped NFS server and services.  
-- Boot 47dee516a0c049e589810a62c4083cf3 --  
Aug 09 12:25:26 localhost.localdomain systemd[1]: Starting NFS server and services...  
Aug 09 12:25:29 localhost.localdomain systemd[1]: Finished NFS server and services.  
Aug 09 13:00:25 localhost.localdomain systemd[1]: Stopping NFS server and services...  
Aug 09 13:00:26 localhost.localdomain systemd[1]: nfs-server.service: Deactivated successfully.  
Aug 09 13:00:26 localhost.localdomain systemd[1]: Stopped NFS server and services.  
-- Boot f58f5001d42145e9a4fceb4979d4c27f --  
Aug 09 13:33:05 localhost.localdomain systemd[1]: Starting NFS server and services...  
Aug 09 13:33:08 localhost.localdomain systemd[1]: Finished NFS server and services.  
Aug 09 13:34:15 localhost.localdomain systemd[1]: Stopping NFS server and services...  
Aug 09 13:34:15 localhost.localdomain systemd[1]: nfs-server.service: Deactivated successfully.  
Aug 09 13:34:15 localhost.localdomain systemd[1]: Stopped NFS server and services.  
-- Boot b629667337d047d39ca9a60743ff1d64 --  
Aug 09 13:37:32 localhost.localdomain systemd[1]: Starting NFS server and services...  
Aug 09 13:37:33 localhost.localdomain systemd[1]: Finished NFS server and services.  
Aug 09 13:42:32 localhost.localdomain systemd[1]: Stopping NFS server and services...  
Aug 09 13:42:33 localhost.localdomain systemd[1]: nfs-server.service: Deactivated successfully.  
Aug 09 13:42:33 localhost.localdomain systemd[1]: Stopped NFS server and services.
```

```
root@localhost:~  
[root@localhost ~]# journalctl | grep -i nfs  
Jul 26 14:19:26 localhost kernel: SELinux: policy capability genfs_seclabel_symlinks=1  
Jul 26 14:21:14 localhost kernel: SELinux: policy capability genfs_seclabel_symlinks=1  
Jul 27 16:05:28 localhost kernel: SELinux: policy capability genfs_seclabel_symlinks=1  
Jul 28 13:14:13 localhost kernel: SELinux: policy capability genfs_seclabel_symlinks=1  
Jul 28 13:22:38 localhost.localdomain kernel: SELinux: policy capability genfs_seclabel_symlinks=1  
Jul 28 13:26:02 localhost.localdomain kernel: SELinux: policy capability genfs_seclabel_symlinks=1  
Jul 28 13:31:10 localhost.localdomain kernel: SELinux: policy capability genfs_seclabel_symlinks=1  
Jul 28 13:31:37 localhost.localdomain kernel: SELinux: policy capability genfs_seclabel_symlinks=1  
Jul 28 13:33:32 localhost.localdomain kernel: SELinux: policy capability genfs_seclabel_symlinks=1  
Jul 28 13:34:48 localhost.localdomain kernel: SELinux: policy capability genfs_seclabel_symlinks=1  
Jul 28 13:38:20 localhost.localdomain kernel: SELinux: policy capability genfs_seclabel_symlinks=1  
Jul 28 13:44:38 localhost.localdomain kernel: SELinux: policy capability genfs_seclabel_symlinks=1  
Jul 28 13:46:54 localhost.localdomain kernel: SELinux: policy capability genfs_seclabel_symlinks=1  
Jul 28 13:48:37 localhost kernel: SELinux: policy capability genfs_seclabel_symlinks=1  
Aug 01 13:10:22 localhost kernel: SELinux: policy capability genfs_seclabel_symlinks=1  
Aug 03 13:26:12 localhost kernel: SELinux: policy capability genfs_seclabel_symlinks=1  
Aug 03 15:05:29 localhost.localdomain sudo[4098]: administrator : TTY=pts/0 ; PWD=/home/administrator/Documents  
; USER=root ; COMMAND=/bin/systemctl status nfs  
Aug 03 15:05:36 localhost.localdomain sudo[4108]: administrator : TTY=pts/0 ; PWD=/home/administrator/Documents  
; USER=root ; COMMAND=/bin/systemctl status nfs-kernel-server  
Aug 03 15:06:58 localhost.localdomain sudo[4144]: administrator : TTY=pts/0 ; PWD=/home/administrator/Documents  
; USER=root ; COMMAND=/bin/systemctl status nfs-server  
Aug 03 15:07:12 localhost.localdomain sudo[4153]: administrator : TTY=pts/0 ; PWD=/home/administrator/Documents
```

## 7.2 Check NFS logs on the client:

```
root@Laptop: ~  
root@Laptop:~# less /var/log/syslog  
root@Laptop:~# cat /var/log/syslog | grep "nfs"  
Aug 11 12:55:27 Laptop systemd[1]: Started Session 3 of User nfsuser.  
Aug 11 12:55:27 Laptop snapd-desktop-integration.snapd-desktop-integration[1505]  
: cmd_run.go:1129: WARNING: cannot create user data directory: cannot create sna  
p home dir: mkdir /home/nfsuser: permission denied  
Aug 11 12:55:27 Laptop snapd-desktop-integration.snapd-desktop-integration[1505]  
: cannot create user data directory: /home/nfsuser/snap/snapd-desktop-integratio  
n/157: Permission denied  
Aug 11 12:55:27 Laptop kernel: [ 131.502885] audit: type=1400 audit(1723406127.  
798:66): apparmor="DENIED" operation="mkdir" class="file" profile="/usr/lib/snap  
d/snap-confine" name="/home/nfsuser/" pid=1505 comm="snap-confine" requested_mas  
k="c" denied_mask="c" fsuid=2000 ouid=2000  
Aug 11 12:55:27 Laptop pulseaudio[1504]: Failed to create secure directory (/hom  
e/nfsuser/.config/pulse): No such file or directory  
Aug 11 12:55:28 Laptop pulseaudio[1590]: Failed to create secure directory (/hom  
e/nfsuser/.config/pulse): No such file or directory  
Aug 11 12:55:28 Laptop session-migration[1593]: Failed to create directory /home  
/nfsuser/.local/share: Permission denied  
Aug 11 12:55:28 Laptop gnome-session[1641]: Failed to create directory /home/nfs  
user/.local/share: Permission denied  
Aug 11 12:55:28 Laptop pulseaudio[1662]: Failed to create secure directory (/hom  
e/nfsuser/.config/pulse): No such file or directory  
Aug 11 12:55:28 Laptop xdg-user-dirs.desktop[1671]: Can't create dir /home/nfsus
```



```
root@Laptop: ~  
root@Laptop:~# journalctl | grep -i nfs  
Aug 09 13:55:41 Laptop useradd[26524]: new user: name=statd, UID=136, GID=65534,  
home=/var/lib/nfs, shell=/usr/sbin/nologin, from=/dev/pts/2  
Aug 09 13:55:46 Laptop systemd[1]: Starting Notify NFS peers of a restart...  
Aug 09 13:55:46 Laptop systemd[1]: Condition check resulted in RPC security serv  
ice for NFS server being skipped.  
Aug 09 13:55:46 Laptop systemd[1]: Started Notify NFS peers of a restart.  
Aug 09 13:55:46 Laptop kernel: RPC: Registered tcp NFSv4.1 backchannel transport  
module.  
Aug 09 13:55:46 Laptop systemd[1]: Condition check resulted in RPC security serv  
ice for NFS client and server being skipped.  
Aug 09 13:55:46 Laptop systemd[1]: Reached target NFS client services.  
Aug 09 14:04:18 Laptop useradd[42975]: new group: name=nfsuser, GID=1004  
Aug 09 14:04:18 Laptop useradd[42975]: new user: name=nfsuser, UID=1004, GID=100  
4, home=/home/nfsuser, shell=/bin/sh, from=/dev/pts/1  
Aug 09 14:04:18 Laptop groupadd[42982]: group added to /etc/group: name=nfsgroup  
, GID=1005  
Aug 09 14:04:18 Laptop groupadd[42982]: group added to /etc/gshadow: name=nfsgro  
up  
Aug 09 14:04:18 Laptop groupadd[42982]: new group: name=nfsgroup, GID=1005  
Aug 09 14:04:53 Laptop usermod[43029]: change user 'nfsuser' UID from '1004' to  
'2000'  
Aug 09 14:04:53 Laptop groupmod[43036]: group changed in /etc/group (group nfsg  
roup/1005, new gid: 2001)
```

7.3. Check the network connection between the client and server:

Since NFS is a networking service, issues with network connections can occur. Luckily there are plenty of ways you can troubleshoot such issues.

7.3a. Use ping to test if the client and server can reach each other:

- Client to server:

```
spy2@Laptop: /  
$ ping 192.168.1.28 -c 8  
PING 192.168.1.28 (192.168.1.28) 56(84) bytes of data.  
64 bytes from 192.168.1.28: icmp_seq=1 ttl=64 time=332 ms  
64 bytes from 192.168.1.28: icmp_seq=2 ttl=64 time=88.1 ms  
64 bytes from 192.168.1.28: icmp_seq=3 ttl=64 time=70.6 ms  
64 bytes from 192.168.1.28: icmp_seq=4 ttl=64 time=191 ms  
64 bytes from 192.168.1.28: icmp_seq=5 ttl=64 time=11.7 ms  
64 bytes from 192.168.1.28: icmp_seq=6 ttl=64 time=77.8 ms  
64 bytes from 192.168.1.28: icmp_seq=7 ttl=64 time=5.47 ms  
64 bytes from 192.168.1.28: icmp_seq=8 ttl=64 time=6.41 ms  
  
--- 192.168.1.28 ping statistics ---  
8 packets transmitted, 8 received, 0% packet loss, time 7014ms  
rtt min/avg/max/mdev = 5.474/97.899/332.398/105.582 ms  
$
```

I sent 8 icmp ping packets from the client device to the server. The packets were transmitted with no loss so that means that the client can reach the server over the network.

- Server to client:

```
[administrator@localhost ~]$ ping 192.168.1.36 -c 8
PING 192.168.1.36 (192.168.1.36) 56(84) bytes of data.
64 bytes from 192.168.1.36: icmp_seq=1 ttl=64 time=5.19 ms
64 bytes from 192.168.1.36: icmp_seq=2 ttl=64 time=6.81 ms
64 bytes from 192.168.1.36: icmp_seq=3 ttl=64 time=5.74 ms
64 bytes from 192.168.1.36: icmp_seq=4 ttl=64 time=6.33 ms
64 bytes from 192.168.1.36: icmp_seq=5 ttl=64 time=6.08 ms
64 bytes from 192.168.1.36: icmp_seq=6 ttl=64 time=5.78 ms
64 bytes from 192.168.1.36: icmp_seq=7 ttl=64 time=5.12 ms
64 bytes from 192.168.1.36: icmp_seq=8 ttl=64 time=5.89 ms

--- 192.168.1.36 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7020ms
rtt min/avg/max/mdev = 5.117/5.865/6.812/0.524 ms
[administrator@localhost ~]$
```

I sent 8 icmp ping packets from the server to the client device. The packets were transmitted with no loss so that means that the server can reach the client over the network.

7.3b. Use traceroute if you suspect there is an issue reaching a remote NFS server:

***\*Note that in this lab the server and client are on the same network/subnet so there really is no routing to a remote device. However, my router acts as a switcher to allow devices on the same subnet to communicate.***

- Client to server:

I needed to install traceroute on my client device:

```
root@Laptop: ~  
root@Laptop:~# apt install traceroute  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following packages were automatically installed and are no longer required:  
  doc-base libasn1-8-heimdal libblas3 libbson1 libdlt2 libev4 libgssapi3-heimdal libgvm21  
  libhcrypto4-heimdal libhdb9-heimdal libheimbase1-heimdal libheimntlm0-heimdal  
  libhx509-5-heimdal libjemalloc2 libjs-sphinxdoc libjs-underscore libkrb5-26-heimdal  
  liblinear4 liblua5.1-0 liblzfl libmosquitto1 libopenvas-wmiclient2 libroken18-heimdal  
  libssh-gcrypt-4 libuuid-perl libwebsockets16 libwind0-heimdal libwpe-1.0-1  
  libwpebackend-fdo-1.0-1 libyaml-tiny-perl lua-bitop lua-cjson lua-lpeg mosquitto nmap  
  nmap-common notus-scanner openvas-scanner openvas-smb ospd-openvas pnsnscan  
  python-babel-localedata python3-babel python3-defusedxml python3-deprecated  
  python3-dnspython python3-flask python3-gnupg python3-impacket python3-itsdangerous  
  python3-jinja2 python3-ldap3 python3-ldapdomaindump python3-ospd python3-packaging  
  python3-paho-mqtt python3-psutil python3-pyasn1 python3-pycryptodome python3-redis  
  python3-requests-toolbelt python3-simplejson python3-terminaltables python3-werkzeug  
  python3-wrapt redis-server redis-tools  
Use 'apt autoremove' to remove them.  
The following NEW packages will be installed:  
  traceroute  
0 upgraded, 1 newly installed, 0 to remove and 8 not upgraded.  
Need to get 45.4 kB of archives.  
After this operation, 152 kB of additional disk space will be used.  
Get:1 http://us.archive.ubuntu.com/ubuntu jammy/universe amd64 traceroute amd64 1:2.1.0-2 [45.4  
  kB]  
Fetched 45.4 kB in 1s (80.1 kB/s)  
Selecting previously unselected package traceroute.  
(Reading database ... 284678 files and directories currently installed.)  
Preparing to unpack .../traceroute_1%3a2.1.0-2_amd64.deb ...  
Unpacking traceroute (1:2.1.0-2) ...  
Setting up traceroute (1:2.1.0-2) ...  
update-alternatives: using /usr/bin/traceroute.db to provide /usr/bin/traceroute (traceroute) i  
n auto mode  
update-alternatives: using /usr/bin/traceroute6.db to provide /usr/bin/traceroute6 (traceroute6  
) in auto mode  
update-alternatives: using /usr/bin/lft.db to provide /usr/bin/lft (lft) in auto mode  
update-alternatives: using /usr/bin/traceproto.db to provide /usr/bin/traceproto (traceproto) i  
n auto mode
```

Now time to see how the client device reaches the server:

```
root@Laptop: ~  
root@Laptop:~# traceroute -T 192.168.1.28  
traceroute to 192.168.1.28 (192.168.1.28), 30 hops max, 60 byte packets  
 1  192.168.1.28 (192.168.1.28)  4.964 ms  5.667 ms  5.643 ms  
root@Laptop:~#
```

My client device only needed one hop to reach the NFS server. This makes sense since the client device and server are on the same subnet on my local network. But if the server was located in another city, state, or even country the number of hops would greatly increase.



- Server to client:

```
administrator@localhost:~  
[administrator@localhost ~]$ traceroute 192.168.1.36  
traceroute to 192.168.1.36 (192.168.1.36), 30 hops max, 60 byte packets  
 1  192.168.1.36 (192.168.1.36)  11.756 ms  11.369 ms  10.882 ms  
[administrator@localhost ~]$
```

Again, only one hop was needed for the server to reach the client device.

7.3c Use netstat on the server to see if the port for NFS is open and working:

Using netstat I can see the connection between the client (192.168.1.36) and the server (192.168.1.28) taking place over tcp port 2049 (which is the default NFS port). This is good since that means that the correct port is being used by the NFS service and connections between the client and server can occur.

```
[root@localhost ~]# netstat -an | grep ':2049'  
tcp        0      0 0.0.0.0:2049          0.0.0.0:*            LISTEN  
tcp        0      0 192.168.1.28:2049    192.168.1.36:973     ESTABLISHED  
tcp6       0      0 :::2049              :::*                  LISTEN  
[root@localhost ~]#
```