16

More File-Sharing Services

CERTIFICATION OBJECTIVES

Linux is designed for networking. Besides Samba, covered in Chapter 15, there are two other major services associated with sharing files on a network: NFS and FTP. Red Hat does not provide GUI tools for these services. Even if Red Hat did as such, it's fastest to learn to configure these services from the command line. If you know these services, you can do more in less time by directly editing key configuration files.

This chapter starts with a description of the Network File System (NFS), a powerful and versatile way of sharing filesystems between servers and workstations. NFS client capabilities are included with a default installation of RHEL 6 and support connections to NFS servers.

The chapter continues with the Very Secure FTP (vsFTP) daemon, which provides both basic and secure FTP server services. With vsFTP, you can secure users, directories, subdirectories, and files with various levels of access control.

These are two more network services that you might configure and/or troubleshoot on the Red Hat exams. Take the time you need to understand the configuration files associated with each of these services, and practice making them work on a Linux computer. In some cases, two or three computers running Linux will be useful to practice the lessons of this chapter.

INSIDE THE EXAM

In the RHCE objectives, the requirements for NFS are essentially the same as for Samba. Of course, the requirements associated with the configuration of NFS are quite different from those of Samba. So when you see the following objectives:

- Provide network shares to specific clients
- Provide network shares suitable for group collaboration

You can expect to study different techniques when setting up NFS to achieve those objectives. The process for limiting NFS access to specific clients is straightforward. The process to set up group collaboration for a NFS network share is based more on the shared directories configured on the NFS server.

This chapter also addresses FTP services. It's the same FTP server configured in Chapter 1 for the RHCSA exam, as an installation server for RHEL 6. The RHCE objective is to

- Configure anonymous-only download

The default installation of the vsFTP server already supports anonymous-only downloads. But it actually also supports access from regular users to their own local accounts. That's only stopped by a single SELinux boolean. In addition, the way vsFTP and NFS serves remote clients depends upon the changes made to firewall and SELinux options. And that's related to the standard RHCE objectives that apply to all services, including user- and host-based security.

CERTIFICATION OBJECTIVE 16.01

The Network File System (NFS) Server

NFS is the standard for sharing files and printers on a directory with Linux and Unix computers. It was originally developed by Sun Microsystems in the mid-1980s. Linux has supported NFS (both as a client and a server) for years, and NFS continues to be popular in organizations with Unix- or Linux-based networks.

You can create shared NFS directories directly by editing the /etc/exports configuration file. But it can be helpful to have some information on how NFS works, from the most important of the NFS files, to what you need to do to set up NFS for basic operation. With that information in hand, you'll understand what services to start, and how NFS communicates over a network. NFS communication can be configured over fixed ports with the help of the right settings in the /etc/sysconfig/nfs file. Of course, directories shared via NFS won't work without proper changes to SELinux.

NFS security can be enhanced in a number of ways, not only with the right options in **iptables**-based firewalls, but also with the help of TCP Wrappers.

NFS Options for RHEL 6

While NFS version 4 (NFSv4) is the default for RHEL 6, Linux NFS software also supports NFS versions 2 (NFSv2) and 3 (NFSv3). The differences include the way clients and servers communicate, the supported file sizes, and support for Windows-style access control lists (ACLs)

As NFSv4 is supported by default, you no longer have to set up Remote Procedure Call (RPC) communication with the rpcbind package. Nevertheless, it is still available for communications with other NFS clients. If used, you'll need to set up a number of fixed ports for NFS services; one method is based on configuration options available in the /etc/sysconfig/nfs file. NFSv4 supports ACLs.

The primary advantage of NFSv3 is support of 64-bit file sizes, which effectively allows access to more than 2GB of data over a shared directory. NFSv4 retains those advantages.

Basic NFS Installation

The primary group associated with NFS software is the "NFS file server." In other words, if you run the following command, **yum** installs the mandatory packages from that group, nfs-utils and nfs4-acl-tools:

```
# yum groupinstall "NFS file server"
```

But those are not the only packages of interest for NFS, especially for older versions of NFS (versions 2 and 3) that requires RPC support. Three other packages of interest include

- **portreserve** Supports the portreserve service, the successor to portmap for NFS (and Network Information Service [NIS]) communication. Prevents NFS from taking ports needed by other services.
- **quota** Provides quota support for shared NFS directories; not absolutely required.
- **rpcbind** Includes RPC communication support for different NFS channels.

To make sure all needed packages are installed and operational, run the following commands:

```
# /etc/init.d/rpcbind start
# /etc/init.d/nfs start
```

Depending on previous actions, it may help to substitute restart for start in these commands. After each command, you may want to check the result with the **rpcbind -p** command, as explained later in this chapter. In the next section, you'll examine the scripts relevant to NFS services.

Basic NFS Server Configuration

NFS servers are relatively easy to configure. All you need to do is export a filesystem, either generally or to a specific host, and then mount that filesystem from a remote client. Of course, you'll also need to open up the right ports in the firewall, and modify SELinux options as appropriate. NFS is controlled by a series of scripts, associated with a number of daemons. It also comes with a broad array of control commands.

NFS Service Scripts

Once appropriate packages are installed, they may be controlled by several different service scripts in the /etc/init.d directory:

- **/etc/init.d/nfs** Control script for NFS; refers to /etc/sysconfig/nfs for basic configuration. Can control NFS services via rpc.nfsd, quotas via rpc.rquotad, the general security services daemon via rpc.svcgssd, and mounts via rpc.mountd.
- **/etc/init.d/nfslock** Control script for lock files and the statd daemon, which locks and provides status for files currently in use.
- **/etc/init.d/portreserve** Replacement for the portmap script; used to set up ports for RPC services.
- **/etc/init.d/rpcbind** RPC program number converter.
- **/etc/init.d/rpcgssd** Control script for RPC-related general security services.
- **/etc/init.d/rpcidmapd** Configuration for NFS user ID mapping to LDAP and Kerberos systems.
- **/etc/init.d/rpcsvcgssd** Control script for the server side of RPC-related general security services.

To configure an NFS server, you'll want to make sure all of these scripts are active in appropriate runlevels. As some of these scripts may not already be active, make sure to apply the following commands to start each of these scripts, and make sure they're active upon reboot:

```
# /etc/init.d/script start
# chkconfig script on
```

NFS Service Daemons

While the basic NFS control script (/etc/init.d/nfs) is fairly simple, that script includes a number of service daemons, each with its own function. These service daemons may be stored either in the /sbin or the /usr/sbin directories. All but **rpc.statd** are controlled by the NFS control script.

- **rpc.idmapd** Works if /etc/idmapd.conf is configured.
- **rpc.mountd** Processes mount requests and verifies current exports.
- **rpc.nfsd** Supports client access with needed kernel threads.
- **rpc.rquotad** Works with quota information.
- **rpc.statd** Configures the status monitor, controlled by the /etc/init.d/nfslock script.

NFS Control Commands and Files

NFS includes a wide variety of commands to set up exports, to show what's available, to see what's mounted, to review statistics, and more. Except for specialized **mount** commands, these commands are can be found in the /usr/sbin directory.

The NFS mount commands are **mount.nfs**, **mount.nfs4**, **umount.nfs**, and **umount.nfs4**. Functionally, they work like regular **mount** and **umount** commands. As suggested by the extensions, they apply to filesystems shared via NFSv4 and other NFS versions. Like other mount.* commands, they have functional equivalents. For example, the **mount.nfs4** command is functionally equivalent to the **mount -t nfs4** command.

If you're mounting a directory shared via NFSv2 or NFSv3, the **mount.nfs** and **mount -t nfs** commands are available for both systems.

The packages associated with NFS include a substantial number of commands in the /usr/sbin directory. The list of commands shown here are just the ones most commonly used to configure and test NFS.

- **exportfs** The **exportfs** command can be used to manage directories shared through and configured in the /etc/exports file.
- **nfsiostat** A statistics command for input/output rates based on an existing mount point. Uses information from the /proc/self/mountstats file.
- **nfsstat** A statistics command for client/server activity based on an existing mount point. Uses information from the /proc/self/mountstats file.

- **showmount** The command most closely associated with a display of shared NFS directories, locally and remotely.

Related commands associated with ACLs are available from the nfs4_acl_tools RPM. They work only with filesystems mounted locally with the acl option, as discussed in Chapter 6. The commands themselves are straightforward, as they set (**nfs4_setfacl**), edit (**nfs4_editfacl**), and list (**nfs4_getfacl**) current ACLs of specified files. While these commands go beyond the basic operation of NFS, they are discussed in Chapter 4.

To review, on a /home directory mounted with the ACL option and then shared via NFS, I applied the **nfs4_getfacl** command on a file from the remote client and got the following output:

```
A::OWNER@:rwatTcCy
A::GROUP@:tcy
A::EVERYONE@:tcy
```

The ACLs either Allow (A) or Deny (D) the file owner (OWNER, GROUP, or EVERYONE). In this case, the extensive levels of permissions given to the owner of the directory are essentially complete and more fine-grained than regular rwx permissions. For example, write (w) and append (a) are both enabled on a normal Linux file with write permissions.

Perhaps the simplest way to edit these ACLs is with the **nfs4_setfacl -e *filename*** command, which opens the current permissions in a text editor. For example, I opened a file mounted via NFSv4 from a remote system with the following command:

```
$ nfs4_setfacl -e /test/michael/filename.txt
```

It opened the given NFSv4 ACLs in the default text editor for the user (normally vi). When I deleted the append permissions for the owner of the file and then saved the changes, it actually deleted both append and write permissions for the file, with the following result, the next time the **nfs4_getfacl** command was applied to the file:

```
D::OWNER@:wa
A::OWNER@:rtTcCy
A::GROUP@:rwatcy
A::EVERYONE@:rtcy
```

In addition, when the **ls -l** command is applied to the file, it's clear that the file owner no longer has write permissions.

Configure NFS for Basic Operation

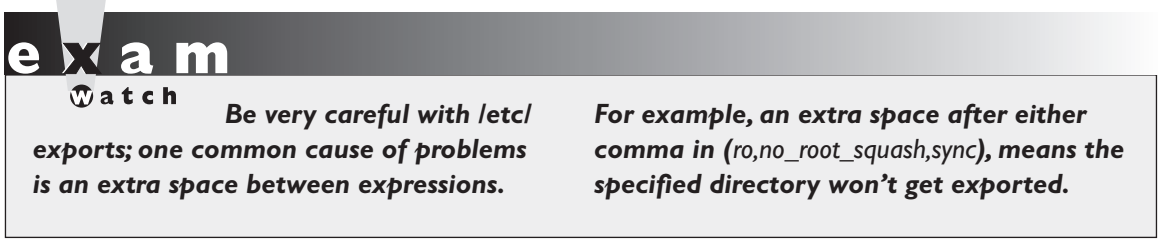
The configuration of the basic /etc/exports file is fairly simple. Once it is configured, you can export directories set up in that file with the **exportfs -a** command. Each line in this file lists the directory to be exported, the hosts to which it will be exported, and the options that apply to this export. While multiple conditions can be set, you can export a particular directory only once. Take the following examples from an /etc/exports file:

```
/pub     (ro,sync) tester1.example.com(rw,sync)
/home    *.example.com(rw,sync)
/tftp    nodisk.example.net(rw,no_root_squash,sync)
```

In this example, the /pub directory is exported to all users as read-only. It is also exported to one specific computer with read/write privileges. The /home directory is exported, with read/write privileges, to any computer on the .example.com network. Finally, the /tftp directory is exported with full read/write privileges (even for root users) to the nodisk.example.net computer.

While these options are fairly straightforward, the /etc/exports file is somewhat picky. A space at the end of a line could lead to a syntax error. A space between a hostname and the conditions in parentheses would open access to all hosts.

All of these options include the **sync** flag. This requires all changes to be written to disk before a command such as a file copy is complete. Before NFSv4, many such options included the **insecure** flag, which allows access on ports above 1024. Even though NFSv4 automatically works with port 2049 by default, the **insecure** flag can still be useful to enable access for other ports above 1024, which is discussed later.



Wildcards and Globbing

In Linux network configuration files, you can specify a group of computers with the right wildcard, which in Linux is also known as *globbing*. What can be used as a wildcard

depends on the configuration file. The NFS /etc/exports file uses "conventional" wildcards: for example, *.example.net specifies all computers within the example.net domain. In contrast, /etc/hosts.deny is less conventional; .example.net, with the leading dot, specifies all computers in that same domain.

For IPv4 networks, wildcards often require some form of the subnet mask. For example, 192.168.0.0/255.255.255.0 specifies the 192.168.0.0 network of computers with IP addresses that range from 192.168.0.1 to 192.168.0.254. Some services support the use of CIDR (Classless Inter-Domain Routing) notation. In CIDR, since 255.255.255.0 masks 24 bits, CIDR represents this with the number *24*. When configuring a network in CIDR notation, you can represent this network as 192.168.0.0/24.

More NFS Server Options

The examples of shared directories shown earlier are just three ways to share a directory. With /etc/exports, it's possible to use a number of different parameters. The parameters described in Table 16-1 and 16-2 fall into two categories: general and user access.

TABLE 16-1

NFS /etc/exports General Options

Parameter	Corresponding /etc/exports Command / Description
insecure	Supports communications above port 1024, primarily for NFS versions 2 and 3.
insecure_locks	Allows insecure file locks; suitable for older NFS clients. Does not check user permissions to a file.
no_subtree_check	Disables subtree checks. If you export a subdirectory such as /mnt/inst, this feature disables checks of higher-level directories for permissions.
sync	Syncs write operations on request. Active by default.
no_wdelay	Forces immediate data writes.
hide	Hide filesystems; if you export a directory and subdirectory such as /mnt and /mnt/inst, shares to /mnt/inst must be explicitly mounted.
mp	Export only if mounted; requires the export point to also be a mount point on the client.
fsid	Set explicit filesystem ID; specifies a numeric identifier for the exported filesystem.

Other parameters relate to how users are treated for the purpose of NFS shared directories. As shown in Table 16-2, the options are associated with the root administrative user, anonymous-only users, and other users that may be designated in the parameters.

Activate the List of Exports

It's not enough to configure the /etc/exports file, as it's simply the default set of exported directories. You need to activate them with the **exportfs -a** command. The next time RHEL 6 is booted, if the right services are activated, the **nfs** start script (/etc/init.d/nfs) automatically runs the **exportfs -r** command, which re-exports directories configured in /etc/exports.

However, if you're modifying, moving, or deleting a share, it is safest to temporarily unexport all filesystems first with the **exportfs -ua** command before reexporting the shares with the **exportfs -a** command.

Once exports are active, they're easy to check. Just run the **showmount -e** command on the server. To review the export list for a remote NFS server, just add the name of the NFS server. For example, the **showmount -e server1.example.com** command looks for the list of exported NFS directories from the server1.example.com system. If this command doesn't work, communication may be blocked with a firewall.

Special Requirements for /home Directories

Some systems store user /home directories on a central server. Such directories can be shared via NFS. Administrators can then back up the /home directories on a regular basis, perhaps supplemented by configuration in a RAID array. That works

TABLE 16-2

NFS Tool User Access Options

Parameter	Corresponding /etc/exports Command / Description
no_root_squash	Treat remote root user as local root; remote root users get root privileges on the shared directory.
all_squash	Treat all client users as anonymous users; all remote users are mapped as an anonymous user.
anonuid=*userid*	Specify local user ID for anonymous users; supports mapping of remote users to a specific user ID such as guest.
anonuid=*userid*	Specify local group ID for anonymous groups; supports mapping of remote groups to a specific group ID.

based on a central authentication database of users and passwords, such as LDAP or Kerberos. In some cases, /home directories may be shared on a small network without a shared authentication database.

In either case, you should configure the /etc/idmapd.conf file to set up how the NFS shared /home directories read the authentication database. Otherwise, such home directories may be configured with ownership by the user named nobody, which would be troublesome to regular users.

The /etc/idmapd.conf file is straightforward and well commented. In all cases, you should change the **Domain**, **Nobody-User**, and **Nobody-Group** directives to match the domain of the current network and the nfsnobody user and group, to minimize the associated privileges. For the example.com domain, that would be

```
DOMAIN = example.com
Nobody-User = nfsnobody
Nobody-Group = nfsnobody
```

While the nsswitch option shown is supposed to be the default, I've found it useful in my tests to make it explicit:

```
Method = nsswitch
```

As discussed in Chapter 8, LDAP authentication may be incorporated into the associated /etc/nsswitch.conf file. Nevertheless, additional custom options for connections to both LDAP and Kerberos services are shown in the databases.

You'll need to make the same changes to all /etc/idmapd.conf files on each NFS client. Changes are applied when you run the **/etc/init.d/rpcidmapd restart** command on both the NFS server and each client.

Fixed Ports in /etc/sysconfig/nfs

NFSv4 is easier to configure, especially with respect to firewalls. To enable communication with an NFSv4 server, the only ports you absolutely need to open are TCP port 2049 and UDP port 111. Port 2049 is the standard for NFSv4 communications. Port 111 supports RPC communications over a network. However, that does not support full functionality of the commands associated with NFS.

While NFSv4 is the default, RHEL 6 still supports NVSv2 and NFSv3. So given the publicly available information on the RHCE exam, you also need to know how to handle those versions of NFS. With associated services, NFSv2 and NFSv3 uses dynamic port numbers. Even for NFSv4, you may want to fix some of the associated

ports, to support the use of the **showmount** command. With fixed ports, you can configure a firewall with appropriate open ports to support an NFS server.

For that purpose, review the /etc/sysconfig/nfs file. It is already preconfigured with comments suggesting appropriate fixed port numbers. While you don't have to follow the suggested port numbers in the commented version of the file, you can. Generally, such port numbers do not cause trouble, as they do not conflict with any commonly used RHEL 6 services. For convenience, however, many administrators set up a series of consecutive unused ports for this purpose, such as 4000 through 4003.

The port numbers listed in Table 16-3 are listed in the order shown in the default version of the /etc/sysconfig/nfs file. The table does not include now-obsolete references to the **rpc.lockd** daemon, as that's not available for RHEL 6.

Once changes are made and saved to the /etc/sysconfig/nfs file, restart the associated service with the /etc/init.d/nfs restart command. If successful, you'll see the associated ports in the output to the rpcinfo command, which lists all communication channels associated with RPC. The following command is more precise, as it isolates actual port numbers:

```
# rpcinfo -p
```

Sample output is shown in Figure 16-1. At first glance, the lines may appear repetitive; however, every line has a purpose. Unless another RPC-related service such as the Network Information Service (NIS) is running, all of the lines shown here are required for NFS communications, when NFSv2 or NFSv3 is used. Examine the first line shown here:

```
program vers proto   port  service
100000    4   tcp    111  portmapper
```

TABLE 16-3 NFS Tool User Access Options

Port	Parameter	Description
875	RQUOTAD	Remote quota daemon
892	MOUNTD_PORT	For mount requests
662	STATD_PORT	For status requests, including locked files (port 662 is assigned to a normally unused FTP protocol)
2020	STATD_OUTGOING_PORT	Reference to outgoing communications.

FIGURE 16-1

Sample rpcinfo -p output with NFS-related ports

```
[root@MinimalRHEL6 ~]# rpcinfo -p
   program vers proto   port  service
    100000    4   tcp    111  portmapper
    100000    3   tcp    111  portmapper
    100000    2   tcp    111  portmapper
    100000    4   udp    111  portmapper
    100000    3   udp    111  portmapper
    100000    2   udp    111  portmapper
    100011    1   udp    875  rquotad
    100011    2   udp    875  rquotad
    100011    1   tcp    875  rquotad
    100011    2   tcp    875  rquotad
    100003    2   tcp   2049  nfs
    100003    3   tcp   2049  nfs
    100003    4   tcp   2049  nfs
    100227    2   tcp   2049  nfs_acl
    100227    3   tcp   2049  nfs_acl
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100003    4   udp   2049  nfs
    100227    2   udp   2049  nfs_acl
    100227    3   udp   2049  nfs_acl
    100021    1   udp  32769  nlockmgr
    100021    3   udp  32769  nlockmgr
    100021    4   udp  32769  nlockmgr
    100021    1   tcp  32803  nlockmgr
    100021    3   tcp  32803  nlockmgr
    100021    4   tcp  32803  nlockmgr
    100005    1   udp    892  mountd
    100005    1   tcp    892  mountd
    100005    2   udp    892  mountd
    100005    2   tcp    892  mountd
    100005    3   udp    892  mountd
    100005    3   tcp    892  mountd
[root@MinimalRHEL6 ~]#
```

The first line represents the arbitrary RPC program number, the NFS version, the use of TCP as a communications protocol, over port 111, with the portmapper service. Note the availability of the portmapper service to NFS versions 2, 3, and 4; communicating over the TCP and UDP protocols.

Communication through selected ports should also be allowed through any configured firewall. For example, Figure 16-2 shows the /etc/sysconfig/iptables file for a firewall that supports remote access to a local NFS server.

Of course, you can set up these firewall rules with the Firewall Configuration tool discussed in Chapter 10.

FIGURE 16-2

Firewall rules for NFS

```
# Firewall configuration written by system-config-firewall
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 2049 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 111 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 111 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
~
"/etc/sysconfig/iptables" 17L, 740C
```

Make NFS Work with SELinux

Of course, appropriate firewalls are not enough. SELinux is an integral part of the security landscape, with respect to both boolean options and files. First, there are two file types associated with NFS:

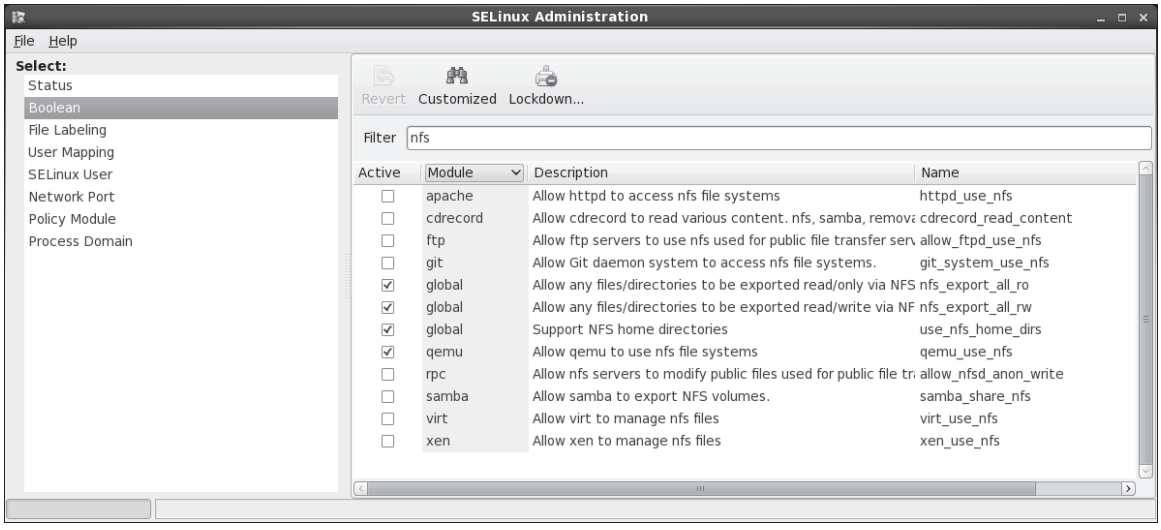
- **var_lib_nfs_t** Associated with dynamic files in the /var/lib/nfs directory. Files in this directory are updated as shares are mounted, as files from shared directories are called and locked.
- **nfsd_exec_t** Assigned to system executable files such as **rpc.mountd** and **rpc.nfsd** in the /usr/sbin directory. Closely related are the rpcd_exec_t and gssd_exec_t file types, for services associated with RPCs and communications with Kerberos servers.

☆n general, you won't have to assign a new file type to a shared NFS directory. So for most administrators, these file types are shown for reference.

So for SELinux, the boolean directives are most important. The options are shown in the Booleans section of the SELinux Administration tool, with the nfs filter, as shown in Figure 16-3. The figure reflects the default configuration; in other words, the global modules are all enabled by default.

The following directives are associated with making NFS work with SELinux in targeted mode. While most of these modules were already listed in Chapter 10,

FIGURE 16-3 NFS-related SELinux boolean options



they're worth repeating, if only to help those who fear SELinux. The first module is not shown in Figure 16-3. The remaining modules are described in the order shown in the figure.

- **allow_gssd_read_tmp** Supports the reading of temporary directories by the General Security Services daemon, **gssd**, which helps protect NFS when systems authenticate through Kerberos 5.
- **httpd_use_nfs** Supports access by the Apache Web server to shared NFS directories.
- **cd_record_read_content** Enables access to mounted NFS directories by the cdrecord command.
- **allow_ftpd_use_nfs** Allows the use of shared NFS directories by FTP servers.
- **git_system_use_nfs** Supports access of NFS shares by the git revision control system service.
- **nfs_export_all_ro** Supports read-only access to shared NFS directories.
- **nfs_export_all_rw** Supports read/write access to shared NFS directories.

- **use_nfs_home_dirs** Enables the mounting of /home on a remote NFS server.
- **qemu_use_nfs** Allows access by the quick emulator to NFS-mounted filesystems.
- **allow_nfsd_anon_write** Supports NFS servers when they modify files on public file transfer services.
- **samba_share_nfs** Allows Samba to export NFS-mounted directories.
- **virt_use_nfs** Enables access by VMs to NFS-mounted filesystems.
- **xen_use_nfs** Allows access by the Xen virtual machine monitor to NFS-mounted filesystems.

To set these directives, use the **setsebool** command. For example, to activate access by an FTP server, in a way that survives a reboot, run the following command:

```
# setsebool -P allow_ftpd_use_nfs 1
```

Quirks and Limitations of NFS

NFS does have its problems. Any administrator who controls shared NFS directories would be wise to take note of these limitations.

Statelessness

NFS is a "stateless" protocol. In other words, you don't need to log in separately to access a shared NFS directory. Instead, the NFS client normally contacts rpc.mountd on the server. The **rpc.mounted** daemon handles mount requests. It checks the request against currently exported filesystems. If the request is valid, **rpc.mounted** provides an *NFS file handle* (a "magic cookie"), which is then used for further client/server communication for this share.

The stateless protocol allows the NFS client to wait if the NFS server ever has to be rebooted. The software waits, and waits, and waits. This can cause the NFS client to hang. The client may even have to reboot or even power-cycle the system.

This can also lead to problems with insecure single-user clients. When a file is opened through a share, it may be "locked out" from other users. When an NFS server is rebooted, handling the locked file can be difficult. The security problems can be so severe that NFS communication is blocked even by the default Red Hat Enterprise Linux firewall.

In theory, the recent change to NFS, setting up sync as the default for file transfers, should help address this problem. In theory, locked-out users should not lose any data that they've written with the appropriate commands.

Absolute and Relative Symbolic Links

If you have any symbolic links on an exported directory, be careful. The client interprets a symbolically linked file from a remotely mounted directory as if it were a local link. Unless the mount point and filesystem structures are identical, the linked file can point to an unexpected location, which may lead to unpredictable consequences.

There are two ways to address this issue. You can take the time to analyze the exported directory, limiting the use of symbolic links. Alternatively, you could use the NFS server-side export option (**link_relative**) that converts absolute links to relative links; however, this can have counterintuitive results if the client mounts a subdirectory of the exported directory.

Root Squash

By default, NFS is set up to **root_squash**, which prevents root users on an NFS client from gaining root access to a share on an NFS server. Specifically, the root user on a client (with a user ID of 0) is mapped to the *nfsnobody* unprivileged account (if in doubt, check the local /etc/passwd file).

This behavior can be disabled via the **no_root_squash** server export option in /etc/exports. For exported directories so disabled, remote root users can use their root privileges on the shared NFS directory. While it can be useful, it is also a security risk, especially from crackers who use their own Linux systems to take advantage of those root privileges.

NFS Hangs

Because NFS is stateless, NFS clients may wait up to several minutes for a server. In some cases, an NFS client may wait indefinitely if a server goes down. During the wait, any process that looks for a file on the mounted NFS share will hang. Once this happens, it is generally difficult or impossible to unmount the offending filesystems. You can do several things to reduce the impact of this problem:

- Take great care to ensure the reliability of NFS servers and the network.

- Avoid mounting many different NFS servers at once. If several computers mount each other's NFS directories, this could cause problems throughout the network.
- Mount infrequently used NFS exports only when needed. NFS clients should unmount these clients after use.
- Set up NFS shares with the **sync** option, which should at least reduce the incidence of lost files.
- Avoid configuring a mission-critical computer as an NFS client.
- Keep NFS mounted directories out of the search path for users, especially that of root.
- Keep NFS mounted directories out of the root (/) directory; instead, segregate them to a less frequently used filesystem, if possible, on a separate partition.

Inverse DNS Pointers

An NFS server daemon checks mount requests. First, it looks at the current list of exports, based on /etc/exports. Then it looks up the client's IP address to find its host name. This requires a reverse DNS lookup.

This host name is then finally checked against the list of exports. If NFS can't find a host name, **rpc.mountd** will deny access to that client. For security reasons, it also adds a "request from unknown host" entry in /var/log/messages.

File Locking

Multiple NFS clients can be set up to mount the same exported directory from the same server. It's quite possible that people on different computers end up trying to use the same shared file. This is addressed by the file-locking daemon service.

While mandatory locks can now be configured with NFSv4, NFS has historically had serious problems with file locks. If you have an application that depends on file locking over NFS, test it thoroughly before putting it into production.

Performance Tips

You can do several things to keep NFS running in a stable and reliable manner. As you gain experience with NFS, you might monitor or even experiment with the following factors:

- Eight kernel NFS daemons, which is the default, is generally sufficient for good performance, even under fairly heavy loads. To increase the capacity of the service, you can add additional NFS daemons through the **RPCNFSDCOUNT** directive in the /etc/sysconfig/nfs configuration file. Just keep in mind that the extra kernel processes consume valuable kernel resources.
- NFS write performance can be extremely slow, particularly with NFS v2 clients, as the client waits for each block of data to be written to disk.
- You may try specialized hardware with nonvolatile RAM. Data that is stored on such RAM isn't lost if you have trouble with network connectivity or a power failure.
- In applications where data loss is not a big concern, you may try the **async** option. This makes NFS faster because **async** NFS mounts do not write files to disk until other operations are complete. However, a loss of power or network connectivity can result in a loss of data.
- Host name lookups are performed frequently by the NFS server; you can start the Name Switch Cache Daemon (**nscd**) to speed lookup performance.

On the Job NFS is a powerful file-sharing system. But there are risks ith NFS. If an NFS server is down, it could affect the entire network. In my personal opinion, it's also not sufficiently secure to use on the Internet. NFS is primarily used on secure networks.

NFS Security Directives

NFS includes a number of serious security problems and should never be used in hostile environments (such as on a server directly exposed to the Internet), at least not without strong precautions.

Shortcomings and Risks

NFS is an easy-to-use yet powerful file-sharing system. However, it is not without its problems. The following are a few security issues to keep in mind:

- **Authentication** NFS relies on the host to report user and group IDs. However, this can be a security risk if root users on other computers access your NFS shares. In other words, data that is accessible via NFS to *any user*

can potentially be accessed by *any other* user. This risk is addressed in part by NFSv4, if the system is configured as a Kerberos client.

- **Privacy** Before NFSv4, such network connections were not encrypted. NFSv4 connections with the support of a Kerberos server are encrypted.
- **portmap infrastructure** Both the NFS client and server depend on the RPC portmap daemon. The portmap daemon has historically had a number of serious security holes. For this reason, RHEL 6 has replaced it with the portmapper service.

Security Tips

If NFS *must* be used in or near a hostile environment, you can reduce the security risks:

- Educate yourself in detail about NFS security. If possible, set up encrypted NFSv4 communications with the help of Kerberos. Otherwise, restrict NFS to friendly, internal networks protected with a good firewall.
- Export as little data as possible, and export filesystems as read-only if possible.
- Unless absolutely necessary, don't supersede the **root_squash** option. Otherwise, crackers on allowed clients may have root-level access to exported filesystems.
- If an NFS client has a direct connection to the Internet, use separate network adapters for the Internet connection and the LAN. Then limit use of NFS to the network adapter connected to the internal network.
- Use appropriate firewall settings to deny access to the portmapper and nfsd ports, except from explicitly trusted hosts or networks. If you're using NFSv4, it's good enough to open the following ports:

```
111     TCP/UDP     portmapper     (server and client)
2049    TCP/UDP     nfsd           (server)
```

- Set fixed port numbers for the services associated with NFS. As discussed earlier, it's possible in /etc/sysconfig/nfs. If you prefer a continuous group of port numbers, one option is 4000:4003:

```
LOCKD_TCPPORT=4000
LOCKD_UDPPORT=4000
MOUNTD_PORT=4001
STATD_PORT=4002
RQUOTAD_PORT=4003
```

- Use a port scanner such as **nmap** to verify that these ports are blocked for untrusted network(s).

Options for Host-Based Security

To review, host-based security on NFS systems is based primarily on the systems allowed to access a share in the /etc/exports file. Of course, host-based security can also include limits based on **iptables** firewall rules.

Options for User-Based Security

As NFS mounts should reflect the security associated with a common user database, the standard user-based security options should apply. That includes the configuration of a common group, as discussed in Chapter 8.

exam Watch

As long as there's a common user database, such as LDAP, the permissions associated with a common group directory carry over to a mount shared via NFS.

EXERCISE 16-1

NFS

This exercise requires two systems: one set up as an NFS server, the other as an NFS client. Then, on the NFS server, take the following steps:

1. Set up a group named IT for the Information Technology group in /etc/group.
2. Create the /MIS directory. Assign ownership to the MIS group with the **chgrp** command.
3. Set the SGID bit on this directory to enforce group ownership.
4. Update the /etc/exports file to allow read and write permissions to the share for the local network. Run the following command to set it up under NFS:

```
# exportfs -a
```

5. Make sure the SELinux booleans are compatible; specifically, make sure the nfs_export_all_ro and nfs_export_all_rw booleans are both still set. You can do so either with the **getsebool** command or the SELinux Management tool.
6. Set fixed ports for the parameters described in the /etc/sysconfig/nfs file.
7. Open those ports in the firewall. Restart the firewall with the **/etc/init.d/iptables restart** command.
8. Restart the NFS service with the **/etc/init.d/nfs restart** command.

Then, on an NFS client, take the following steps:

9. Create a directory for the server share called /mnt/MIS.
10. Mount the shared NFS directory on /mnt/MIS.
11. List all exported shares from the server and save this output as /mnt/MIS/shares.list.
12. Make this service a permanent connection in the /etc/fstab file. Assume that the connection might be troublesome and add the appropriate options, such as soft mounting.
13. Run the **mount -a** command to reread /etc/fstab. Check to see if the share is properly remounted.
14. Test the NFS connection. Stop the NFS service on the server, and then try copying a file to the /mnt/MIS directory. While the attempt to copy will fail, it should not hang the client.
15. Restart the NFS service on the server.
16. Edit /etc/fstab again. This time, assume that NFS is reliable, and remove the special options added in Step 12.
17. Now shut down the server and test what happens. The mounted NFS directory on the client should hang when you try to access the service.
18. The client computer may lock. If so you can boot into single-user mode, as described in Chapter 5, to avoid the pain of a reboot. Restore the original configuration.

CERTIFICATION OBJECTIVE 16.02

Test an NFS Client

Now you can mount a shared NFS directory from a client computer. The commands and configuration files are similar to those used for any local filesystem. In the preceding section, you configured an NFS server. For now, stay on the NFS server system, as the first client test can be run directly from the server system.

NFS Mount Options

Before doing anything elaborate, you should check for the list of shared NFS directories. Then you can test the shared NFS directory from a second Linux system, presumably a RHEL 6 system (or equivalent). To that end, the **showmount** command displays available shared directories.

The two options of significance are **-d** and **-e**; when coupled with the hostname or IP address of the NFS server, the command displays the shared directories, possibly including the host limits of the share. For example, given a simple share of the /mnt and /home directories on a given NFS server, the **showmount -d server1.example.com** command provides the following result:

```
Directories on server1.example.com:
/home
/mnt
```

As suggested, the **showmount -e server1.example.com** command provides more information:

```
Export list for server1.example.com:
/mnt  192.168.100.0/24
/home 192.168.122.0/24
```

If you don't see a list of shared directories, log in to the NFS server system. Repeat the **showmount** command, substituting localhost or 127.0.0.1 for the host name or IP address. If there's still no output, review the steps described earlier in this chapter. Make sure the /etc/exports file is configured properly. Remember to export the shared directories. Use the command

```
# /etc/init.d/nfs status
```

to see if the nfsd, rpc.mountd, and rpc.rquotad services are running. Unless the system is a Kerberos client, don't be concerned by the following message in the **nfs status** command output:

```
rpc.svcgssd is stopped
```

The advertising of shared NFS directories depends on RPC. And that starts with the rpc.statd service, controlled by the /etc/init.d/rpcbind script. If all such services are in operation, one more thing to check is the output to the **rpcinfo -p** command. As shown back in Figure 16-1, it lists a number of services and ports.

For NFSv2 and NFSv3, if the ports are not fixed per /etc/sysconfig/nfs, it won't be possible to set up an iptables-based firewall to support communication through those ports. If not all of the services shown in Figure 16-1 show up on the local system, something may be missing.

Now to mount this directory locally, you'll need an empty local directory. Create a directory such as /remotemnt. You can then mount the shared directory from a system like 192.168.122.50 with the following command:

```
# mount.nfs4 192.168.122.50:/mnt /remotemnt
```

This command mounts the NFS shared /mnt directory from the computer on the noted IP address. If desired, you could substitute the **mount -t nfs4** command for **mount.nfs4**. When it works, you'll be able to access files from the remote /mnt directory as if it were a local directory.

Configure NFS in /etc/fstab

You can also configure an NFS client to mount a remote NFS directory during the boot process, as defined in /etc/fstab. For example, the following entry in a client /etc/fstab mounts the /homenfs share from the computer named nfsserv, based on NFSv4, on the local /nfs/home directory:

```
nfsserv:/homenfs   /nfs/home  nfs4  soft,timeo=100  0  0
```

exam Watch

NFS-specific options to the* mount *command that can also be used in /etc/fstab can be found in the nfs man page.

The **soft** and **timeo** options are part of a variety of specialized NFS mount options. Such options as shown here can also be used to customize how mounts are done during the boot process in the /etc/fstab file. For more

information on these and related options, see the nfs man page, available with the **man nfs** command.

Alternatively, an automounter can be used to mount NFS filesystems dynamically as required by the client computer. The automounter can also unmount these remote filesystems after a period of inactivity. For more information on the governing autofs service, see Chapter 6.

Without a timeout, NFS mounts through /etc/fstab can be troublesome. For example, if the network or the NFS server is down, the lack of a timeout can hang the client. (I discussed this issue in more detail earlier in this chapter.) A hang is a big problem if it happens during the boot process, because the boot may never complete, and you would have to restore your system by booting into some runlevel where networking is not started, such as **1** or **single**.

Diskless Clients

NFS supports diskless clients, which are computers that do not store the operating system locally. A diskless client may use a boot floppy or a boot programmable read-only memory (PROM) chip to get started. Then embedded commands can mount the appropriate root (/) directory, set up swap space, set the /usr directory as read-only, and configure other shared directories such as /home in read/write mode. If your computer uses a boot PROM, you'll also need access to DHCP and TFTP servers for network and kernel information.

Red Hat Enterprise Linux includes features that support diskless clients. While they are not listed as part of the current Red Hat exam requirements or related course outlines, I would not be surprised to see such requirements in the future. For more information on an open-source Red Hat project on the topic, see the Cobbler project at https://fedorahosted.org/cobbler/.

Soft Mounting

Consider using the **soft** option when mounting NFS filesystems. When an NFS server fails, a soft-mounted NFS filesystem will fail rather than hang. However, this risks the failure of long-running processes due to temporary network outages.

In addition, you can use the **timeo** option to set a timeout interval, in tenths of a second. For example, the following command would mount /nfs/home with a timeout of 30 seconds (**timeo** uses tenths of a second):

```
# mount -o soft,timeo=300 myserver:/home /nfs/home
```

Current NFS Status

The current status of NFS services, mostly, is documented in two directories: /var/lib/nfs and /proc/fs/nfsd. If there's a problem with NFS, look at some of the files in these directories. Take these directories one at a time. First, there are two key files in the /var/lib/nfs directory:

- **etab** Includes a full description of exported directories, including default options.
- **rmtab** Specifies the state of shared directories currently mounted.

Take a look at the contents of the /proc/fs/nfsd directory. As a virtual directory, files in the /proc directory tree have a size of zero. However, as dynamic files, they can contain valuable information. Perhaps the key option for basic operation is the file named versions. The contents of that file specifies the currently recognized versions of NFS.

The normal content of this file is just a little cryptic, which suggests that the current NFS server can communicate using NFSv2, NFSv3, and NFSv4, but not NFS version 4.1.

```
+2 +3 +4 -4.1
```

So when I accidentally activated the **#RPCNFSDARGS="-N 4"** option in the /etc/sysconfig/nfs file, the contents of the versions file changed to

```
+2 +3 -4 -4.1
```

The difference is subtle, but tells me that the local NFS server does not currently recognize NFSv4 connections. Of course, with the advantages associated with NFSv4, you should not activate the **RPCNFSDARGS="-N 4"** option.

CERTIFICATION OBJECTIVE 16.03

The Very Secure FTP Server

The File Transfer Protocol is one of the original network applications developed with the TCP/IP protocol suite. It follows the standard model for network services, as FTP requires a client and a server. The **lftp** command line client is easy to install

from a package of the same name. The FTP Server package group includes the default Red Hat FTP Server, the very secure FTP (vsFTP) daemon.

This section is focused on the configuration of the FTP server. You'll test the result with the **lftp** client, described in Chapter 2. While there are many other excellent FTP servers for Linux, vsFTP is the only one supported on RHEL 6.

Basic vsFTP Configuration

The simplest way to install vsFTP with dependencies is with the following command:

```
# yum install vsftpd
```

Of course, if you're installing vsFTP in production or during an exam, it's important to run commands such as the following:

```
# /etc/init.d/vsftpd start
# chkconfig vsftpd on
```

As noted in Chapter 11, these commands start the noted service and make sure the service starts the next time the system is rebooted. Once installed, associated configuration files can be found in the /etc/vsftpd directory. There's one additional configuration file, vsftpd, in the /etc/pam.d directory. A check with the **ldd** and **strings** commands, applied to the vsftpd daemon in the /usr/sbin directory, confirms that vsFTP can be protected with the TCP Wrappers files described in Chapter 10. However, before vsFTP can be protected with TCP Wrappers, you need to make sure to keep a key option at the end of the vsftpd.conf configuration file.

The man page for the vsftpd.conf configuration file has a full list of available directives, split into three categories:

- **boolean** Like SELinux options, may be set to yes or no.
- **numeric** Can be set to a specific numeric value.
- **string** Associated with values such as filenames.

The Main vsFTP Configuration File

The main vsFTP configuration file is vsftpd.conf, in the /etc/vsftpd directory. Normally, it's pretty secure. But if it is configured incorrectly, you could end up providing access to the top-level root directory for users who connect with regular user accounts. To start securing a vsFTP server, you can configure vsFTP to disable logins from regular

users. You can configure vsFTP through the vsftpd.conf configuration file, in the /etc/vsftpd directory.

Many standard settings in the vsftpd.conf file are different from the actual default value of a directive listed in the file. In this section, I refer to the original version of the vsftpd.conf file, as installed, as the "standard" version of the file. Before starting, there's one critical directive not listed in the file:

```
ftp_username
```

It's the default user for anonymous connections. In other words, when a remote user connects with username anonymous, that user connects on the server as a user named ftp. In the /etc/passwd file, that user is listed with the following settings:

```
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
```

In other words, users can't log in directly as the user named ftp. However, anonymous users on the vsFTP server are taken to that user's home directory, /var/ftp. So if you want to change the default directory associated with anonymous logins, change the home directory associated with the local user named ftp.

The following is an analysis of the default RHEL 6 version of this file. Some of it may be reformatted for clarity, some for desirable options. In a couple of cases, directives are analyzed in an order different from that shown in the file. First note the following comment:

```
# The default compiled in settings are fairly paranoid. This sample
# file loosens things up a bit, to make the ftp daemon more usable.
```

The vsFTP service is designed for anonymous access. The following comment emphasizes it. To disable anonymous access, you'll have to set this to **NO**.

```
# Allow anonymous FTP? (Beware - allowed by default if you comment
# this out).
anonymous_enable=YES
```

The standard vsftpd.conf file supports access by users configured in the local authentication database. The RHCE objectives suggests "anonymous-only download." To set up the server for anonymous-only access, change this option to **local_enable=NO**. Since that's the default, it's sufficient to comment out the directive as shown here:

```
# local_enable=YES
```

If there's actually a need to enable local users, you'll also have to enable the SELinux ftp_home_dir boolean. By default, such users are also able to access the top-level root directory, unless you include the following directive:

```
chroot_local_user=YES
```

While the standard version of the vsftpd.conf file supports write commands by users, the default value of the write_enable directive is no. As the relevant RHCE objective requires support only for anonymous downloads, be prepared to comment out or change this directive:

```
# write_enable = yes
```

The target directory also needs a special SELabel: public_content_rw_t. Changes are also required to activate either the allow_ftpd_anon_write or the ftp_home_dir boolean, depending on the users who are allowed to write files through the FTP server.

If writes are enabled, files are written with some permissions. New files on a local system are given permissions based on the value of umask, as discussed in Chapter 4. The umask associated with writes is set with the following directive:

```
local_umask=022
```

While support for anonymous-only downloads is listed in the RHCE objectives, there are circumstances where you might allow anonymous users to upload files. Yes, allowing anonymous users to write anything to a server can be a security risk.

As suggested by the associated comment, it also requires an appropriate directory, writable by the FTP user, and configured with the public_content_rw_t SELinux file type.

```
# Uncomment this to allow the anonymous FTP user to upload files.
# This only has an effect if the above global write enable is
# activated. Also, you will

# obviously need to create a directory
# writable by the FTP user.
#anon_upload_enable=YES
```

In a similar fashion, you could set up the server to go one step further, and allow anonymous users to actually create directories:

```
#anon_mkdir_write_enable=YES
```

Further down in the file, a couple of directives can be set to change ownership of uploaded files. The default user owner for files uploaded by anonymous users is root. And that's an additional security risk. What if a cracker were able to upload and run

a script on a server with root privileges? So if you do enable anonymous uploads, make sure to set the chown_username directive to a nonprivileged user such as nobody.

```
#chown_uploads=YES
#chown_username=whoever
```

A couple of useful features for users help welcome them to the vsFTP server on a successful login. Skipping down in the file, the following command provides a message for users who are logging in to the local vsFTP server:

```
ftpd_banner=Welcome to blah FTP service
```

This next message looks for a .message file in each directory and sends it to the client:

```
dirmessage_enable=YES
```

Unfortunately, these messages don't work for users who log in to using a client such as **lftp**.

The following option enables logging of both uploads and downloads:

```
xferlog_enable=YES
```

By itself, it enables logging to the /var/log/vsftpd.log file. But there are a couple of related directives a few lines down. First, the following directive sets the actual log file:

```
#xferlog_file=/var/log/vsftpd.log
```

And then the next xferlog directive configures a standard format for logging, including a date, a time, an IP address, the file in question, the user, and more. Unless superseded, it also directs logging to the /var/log/xferlog file.

```
xferlog_std_format=YES
```

While there is no requirement to open port 20 in a firewall for the standard **lftp** client, some other FTP clients may require the use of that port. So while the following directive enables such communication over port 20, in most cases, you need not be concerned with that port:

```
connect_from_port_20=YES
```

One standard security measure automatically logs out a user after a period of inactivity. The default is 300 seconds. The following commented directive suggests 600 seconds:

```
idle_session_timeout=600
```

Sometimes data transfers can stall for various reasons, such as an overloaded server. While the default timeout is 300 seconds, the following commented directive suggests a shorter period:

```
#data_connection_timeout=120
```

The default nonprivileged user is named nobody, which does exist in the standard user authentication database. If you change the directive as suggested here, make sure the noted user actually exists in the authentication database.

```
#nopriv_user=ftpsecure
```

Some FTP clients may require the use of this feature to abort requests such as big file transfers. It's disabled by default:

```
#async_abor_enable=YES
```

Further down in the file is one other option associated with certain FTP clients, associated with recursive file lists. If you're configuring an FTP server for such clients, consider enabling the following directive:

```
#ls_recurse_enable=YES
```

As ASCII transfers to FTP servers are a known security risk, they are disabled by default. The following commented directives make it possible to enable such transfers:

```
#ascii_upload_enable=YES
#ascii_download_enable=YES
```

The first FTP servers were developed in a world where most Internet users could be trusted. Users who logged in anonymously were asked to supply their e-mail addresses as the password. An early security measure denied access to users who supplied e-mail addresses listed in the associated banned_email_file:

```
#deny_email_enable=YES
#banned_email_file=/etc/vsftpd/banned_emails
```

The following two directives are important for vsFTP, especially on a network where both IPv4 and IPv6 networks have been enabled. First, one of these directives should be enabled to allow the control of vsFTP from the vsftpd script in the /etc/init.d directory. Second, both parameters can't be activated on the same vsFTP server. In other words, vsFTP can't communicate on both IPv4 and IPv6 networks simultaneously.

```
listen=YES
#listen_ipv6=YES
```

The final three directives in the standard vsftpd.conf file have been added by Red Hat. First, RHEL 6 includes user lists associated with Pluggable Authentication Modules (PAM) described in Chapter 10.

```
pam_service_name=vsftpd
```

If you allow something more than anonymous-only access, it's important to keep out certain users such as root, as well as any other users with privileges. By default, vsFTP is configured to disable logins from sensitive users such as root, bin, and mail. The following directive implicitly refers to the user_list file in the /etc/vsftpd directory, through the userlist_deny directive.

```
userlist_enable=YES
```

While vsFTP has been compiled with TCP Wrappers libraries, you can't use the associated /etc/hosts.allow and/or the /etc/hosts.deny files unless this final directive is kept active:

```
tcp_wrappers=YES
```

As vsFTP also uses PAM for security, it also disables the users in /etc/vsftpd/ftpusers. The lists of users in the default versions of these files are identical.

Other vsFTP Configuration Files

The contents of the user_list and the ftpusers files in the /etc/vsftpd directory are identical. They include a standard list of service users, including the root administrative user. In general, you don't want to give any such users login privileges, under any circumstances. The aforementioned **userlist_enable=YES** directive activates the **userlist_deny=YES** directive, which denies all listed in the user_list file.

The ftpusers file is used by the /etc/pam.d/vsftpd configuration file. Chapter 10, Lab 6, gave you an opportunity to explore and reconfigure that file. To review, the appropriate PAM directive in that file is

```
auth required pam_listfile.so item=user sense=deny
file=/etc/vsftpd/ftpusers onerr=succeed
```

The directive denies all users in the ftpusers file. There's one other file in the directory, vsftpd_conf_migrate.sh, which links configuration files created in previous versions of the vsFTP server.

Configure SELinux Support for vsFTP

There are a variety of SELinux settings available for vsFTP. The major settings are related to file types and boolean options. If you've run Linux without SELinux before, note that some of these options are required to make vsFTP work in ways otherwise expected.

Of course, appropriate firewalls are not enough. SELinux is an integral part of the security landscape, with respect to boolean options as well as files. First, there are four major file types associated with FTP servers:

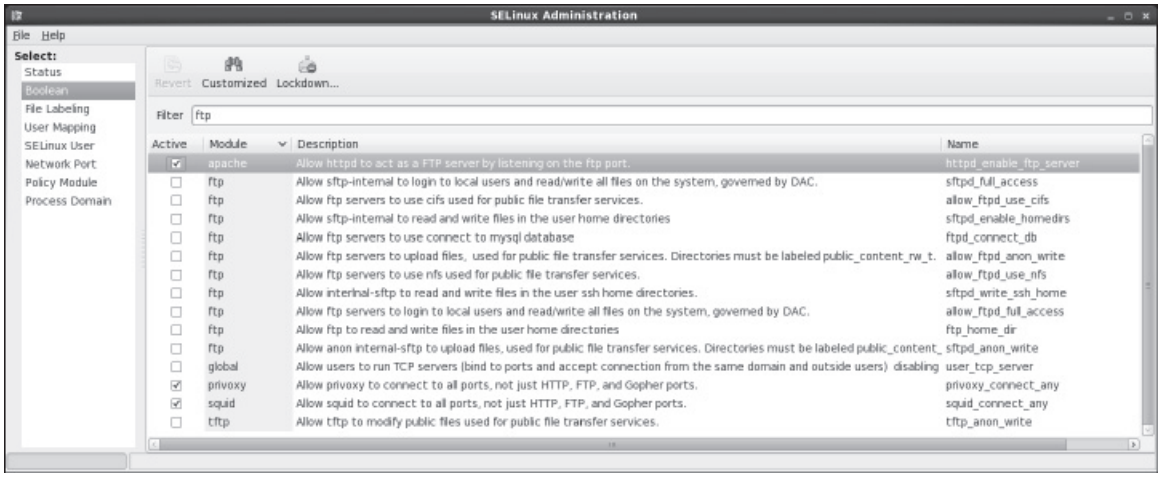
- **ftpd_exec_t** Used for the executable daemon, vsftpd, in the /usr/sbin directory.
- **xferlog_t** Assigned to the xferlog file for vsFTP logs, in the /var/log directory.
- **public_content_t** Required for files shared via an FTP server, unless associated with a user home directory.
- **public_content_rw_t** Required for directories where users can write files when logged in via an FTP server, unless associated with a user home directory.

Of course, there are also boolean options associated with the vsFTP server. All options associated with FTP are shown in the Booleans section of the SELinux Administration tool, with the ftp filter, as shown in Figure 16-4. The figure reflects the default configuration; in other words, no module directly related to FTP services is normally enabled.

Most of these directives were covered in general terms in Chapter 11. For the vsFTP server, you may use the following boolean options under certain circumstances. For example,

- To set up directories where anonymous users can write files, you'll need to activate the **allow_ftpd_anon_write** boolean and configure those directories with the **public_content_rw_t** file type.
- To allow regular users access to all files on a system with an FTP server, set the **allow_ftpd_full_access** boolean.
- To let regular users full access their home directories, protected by their usernames and passwords, set the **ftp_home_dir** boolean.

FIGURE 16-4 FTP-related SELinux boolean options



To set these directives, use the **setsebool** command. For example, to go beyond the RHCE requirement for anonymous-only download, by activating read/write access from FTP on user home directories, run the following command:

```
# setsebool -P ftp_home_dir 1
```

Ports, Firewalls, and vsFTP

The configuration of a firewall for an FTP server is a relatively simple process. In most cases, communication requires access to just one TCP port, 21. The standard firewall rule for that purpose, as may be configured in the /etc/sysconfig/iptables file, is

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 21 -j ACCEPT
```

If desired, you can also limit access to one or more IP addresses. For example, the following variation on the **iptables** rule limits access to systems on the 192.168.122.0/255.255.255.0 network:

```
-A INPUT -m state --state NEW -m tcp -p tcp -s 192.168.122.0/24 --dport 80 -j
ACCEPT
```

Some rare FTP clients may also need access to TCP port 20. To configure vsFTP to allow connections from such clients, you should apply any firewall changes made for port 21 also to port 20.

Assuming you retain the **tcp_wrappers=YES** option in the vsftpd.conf configuration file, you can further protect this service with the files associated with TCP Wrappers, /etc/hosts.allow and /etc/hosts.deny. To review and cite an example, the following entry in /etc/hosts.allow would permit access from user donna on system tester1.example.com:

```
vsftpd : donna@tester1.example.com
```

That limit would work if coupled with the following entry in /etc/hosts.deny:

```
vsftpd : ALL
```

EXERCISE 16-2

Configure a Basic vsFTP Server

In this exercise, you'll install and activate a basic vsFTP server on a RHEL system. This exercise assumes that you've configured an open port 21 through a local firewall. If you've enabled SELinux and want to support an FTP server just for downloads, you'll also need to modify SELinux policies to "Allow Ftpd To Read/Write Files In The User Home Directories," which corresponds to the boolean **ftp_home_dir** option described earlier.

1. Make sure the vsFTP server is installed. The easiest way is with the following command:

   ```
   # rpm -q vsftpd
   ```

2. If it isn't already installed, use the techniques discussed earlier to install the vsFTP RPM package.
3. Activate the vsFTP server with the **service vsftpd start** command.
4. Make sure this server is automatically activated the next time the local system boots with the following command:

   ```
   # chkconfig vsftpd on
   ```

5. Log in to the vsFTP server as a regular user, from a remote system.
6. Once logged in, run the **cd ..** command twice (remember the space between the command and the two dots). Explore the local directory. You should see a danger here, as this is the root directory for the FTP server computer.
7. Close the FTP client session.

8. If you're concerned about the security issues, deactivate the **ftp_home_dir** SELinux boolean.

Anonymous-Only Download Configuration

While you've just seen a lot can be done with the vsftpd.conf file, the configuration of an anonymous-only download configuration for the vsFTP service is relatively simple. To sum up, all you need to do is disable the following directive, which allows locally configured users to log in with their regular accounts:

```
#local_enable=yes
```

If you're asked to set up anonymous only downloads elsewhere than the /var/ftp tree, a few additional steps are required:

- Create the target directory.
- Populate the new target directory with files to be downloaded.
- Set the target directory (and files therein) with the public_content_t file type.
- Change the home directory for the ftp user to match the new target.

SCENARIO & SOLUTION	
You're having trouble configuring a firewall for NFS	Fix the ports associated with various NFS services. Open firewall ports for those services. Also open port 111 for the portmapper.
You want to prohibit read/write access to shared NFS directories	Make sure shares are configured with the **ro** parameter in /etc/exports. Disable the nfs_export_all_rw SELinux boolean.
You need to set up automatic mounts of a shared NFS directory	Configure the shared directory in /etc/fstab, with options such as **soft** and **timeo**.
You want to disable user-based access to a vsFTP server	Set the **local_enable=no** directive in the vsftpd.conf file in the /etc/vsftpd directory. Don't activate the ftp_home_dir boolean.
You need to set up a nonstandard directory for anonymous-only vsFTP downloads	Configure the new directory as the home directory for the ftp user, and assign it the SELinux public_content_t file type.

CERTIFICATION SUMMARY

NFS allows you to share filesystems between Linux and Unix computers. It is an efficient way to share files between such systems, but there are many security concerns involved with its use. Be careful when setting up an NFS share on an unprotected network.

While RHEL 6 supports NFSv4, it also supports access by NFSv2 and NFSv3 clients. It depends on the portreserve service to keep NFS from taking ports needed by other services. It depends on the rpcbind service to set up RPC support. It can even set up quota support for shared NFS directories. It's controlled by a group of scripts in the /etc/init.d directory (nfs, nfslock, portreserve, rpcbind, rpcidmapd). If it's supported by Kerberos-based user authentication, it's also controlled by the rpcgssd and rpcsvcgssd scripts in the same directory. Each of these scripts is associated with daemons in the /sbin and /usr/sbin directories. Configuration of these scripts is configured primarily in the /etc/sysconfig/nfs file. Related commands include **exportfs** and **showmount**. It even supports ACLs with the **nfs4_setfacl**, **nfs4_getfacl**, and the **nfs4_editfacl** commands.

In most cases, basic configuration of NFS is accomplished by relatively simple directives in the /etc/exports file. Once the NFS service is running, such exports are activated through the **exportfs** command. Firewalls should be configured based on the ports fixed in the /etc/sysconfig/nfs file. Appropriate active ports and services can be confirmed with the **rpcinfo -p** command.

Generally, for basic NFS operation, no SELinux changes are required. As mounted NFS directories may appear local, user security, as well as directories configured for group collaboration, can be set up as if the filesystem were local. It works if the systems use the same or matching authentication databases. NFS mounts can be automated in /etc/fstab or through the automounter. The current status of NFS is documented in various files in the /var/lib/nfs and /proc/fs/nfsd directories.

Red Hat includes one FTP server, the very secure FTP service. You can configure it in detail through the /etc/vsftpd/vsftpd.conf configuration file. Anonymous users are configured through the default ftp user's home directory, /var/ftp. Other vsftpd.conf directives can be used to set up access by regular users, various forms of anonymous access, and more. It also supports standard options for PAM and TCP Wrappers security.

Generally, vsFTP requires access only through TCP port 21. Unless it involves access to home directories, access requires labeling with the public_content_t or public_content_rw_t file types. Home directory access requires an active ftp_home_dir boolean. In general, to set up the anonymous-only download access specified in the RHCE objectives, you need to set **local_enable=no** and keep the ftp_home_dir boolean off. If it's a nonstandard directory, you'll need to set the public_content_t file type on that directory.

TWO-MINUTE DRILL

Here are some of the key points from the certification objectives in Chapter 16.

The Network File System (NFS) Server

- ❑ NFS is the standard for sharing files and printers between Linux and Unix computers. RHEL 6 supports NFS versions 2, 3, and 4.
- ❑ Key NFS processes are **rpc.mountd** for mount requests, **rpc.rquotad** for quota requests, and **nfsd** for each network share.
- ❑ Configuration options for these processes, as well as parameters to fix key ports can be found in the /etc/sysconfig/nfs file.
- ❑ NFS shares are configured in /etc/exports and activated with the **exportfs -a** command.
- ❑ Firewalls can be set based on ports fixed in /etc/sysconfig/nfs as well as port 111 for the portmapper and 2049 for NFSv4.
- ❑ In most cases, required booleans for SELinux are already active. In fact, to disallow read/write access, you may choose to disable the nfs_export_all_rw boolean.
- ❑ When NFS directories are mounted, they should appear seamless. User authentication works with a common or matching authentication database.

Test an NFS Client

- ❑ Clients can make permanent connections for NFS shares through /etc/fstab.
- ❑ Clients can review shared directories with the **showmount** command.
- ❑ The **mount.nfs4** command is designed to mount directories shared via NFSv4. The **mount.nfs** command works with NFSv2 and NFSv3.
- ❑ If an NFS server fails, it can "hang" an NFS client. The **soft** and **timeo** options to the **mount** command can help prevent such hangs, which could otherwise force users to reboot a system.
- ❑ NFS and portmap have security problems. Limit their use when possible to secure internal networks protected by an appropriate firewall.

The Very Secure FTP Server

- RHEL includes the vsFTP server. The default configuration allows anonymous and real user access.
- To configure anonymous-only download access, you'll need to disable the **local_enable** directive.
- To set up anonymous-only access on a nonstandard directory, you'll have to change the home directory of the default user named ftp and set up that directory with the public_content_t SELinux file type.
- You can customize vsFTP through the /etc/vsftpd/vsftpd.conf configuration file. It also uses authentication files in the /etc/vsftpd/ directory: ftpusers and user_list.
- The ftpusers file is cited by a PAM configuration file, /etc/pam.d/vsftpd, to disallow access from cited system users. That can be used as a model to limit or allow user access in other ways.

SELF TEST

The following questions will help you measure your understanding of the material presented in this chapter. As no multiple-choice questions appear on the Red Hat exams, no multiple-choice questions appear in this book. These questions exclusively test your understanding of the chapter. It is okay if you have another way of performing a task. Getting results, not memorizing trivia, is what counts on the Red Hat exams. There may be more than one answer to many of these questions.

The Network File System (NFS) Server

1. In the /etc/exports file, to export the /data directory as read-only to all hosts and grant read and write permission to the hostname superv in example.com, what directive would you enter in that file?

   _____

2. Once you've configured /etc/exports, what command exports these shares?

   _____

3. What port number is associated with the port mapper?

   _____

4. What port number is associated with NFSv4?

   _____

5. What is the NFS configuration option that supports access by the root administrative user?

   _____

Test an NFS Client

6. You're experiencing problems with NFS clients for various reasons, including frequent downtime on the NFS server and network outages between NFS clients and servers. What type of mounting can prevent problems on NFS clients?

   _____

7. What is the command that can display NFS shared directories from the outsider1.example.org system?

   _____

The Very Secure FTP Server

8. What default directive in /etc/vsftpd/vsftpd.conf should you disable to prevent users from logging in to their accounts through the vsFTP server?

   _____

9. What directive should you include to keep regular users from getting to the top-level root directory (/) on a vsFTP server?

   _____

10. Based on the default RHEL 6 configuration, what file includes a list of users not allowed to log in to the vsFTP server?

    _____

11. What user account determines the default directory for anonymous users who connect to the vsFTP server?

    _____

12. What additional directives do you need to add to the default vsFTP configuration file to allow security using PAM and TCP wrappers?

    _____

LAB QUESTIONS

Several of these labs involve configuration exercises. You should do these exercises on test machines only. It's assumed that you're running these exercises on virtual machines such as KVM. For this chapter, it's also assumed that you may be changing the configuration of a physical host system for such virtual machines.

Red Hat presents its exams electronically. For that reason, the labs in this and future chapters are available from the CD that accompanies the book, in the Chapter16/ subdirectory. In case you haven't yet set up RHEL 6 on a system, refer to Chapter 1 for installation instructions.

The answers for each lab follow the Self Test answers for the fill-in-the-blank questions.

SELF TEST ANSWERS

The Network File System (NFS) Server

1. The following entry in /etc/exports would export the /data directory as read-only to all hosts and grant read and write permission to the host superv in example.com:

```
/data (ro,sync)  superv.example.com(rw,sync)
```

2. Once you've revised /etc/exports, the **exportfs -a** command exports all filesystems. Yes, you can re-export filesystems with the **exportfs -r** command. But there's no indication that NFS shares have yet been exported.
3. The port number associated with the portmapper is 111.
4. The port number associated with NFSv4 is 2049.
5. The NFS configuration option that supports access by the root administrative user is no_root_squash.

Test an NFS Client

6. Soft mounting and time-outs associated with the **soft** and **timeo** options can prevent problems such as lockups with NFS clients.
7. The command that can display NFS shared directories from the named remote system is **showmount -e outsider1.example.org**.

The Very Secure FTP Server

8. The default directive in /etc/vsftpd/vsftpd.conf that you should disable to prevent users from logging into their accounts through the vsFTP server is **local_enable=YES**.
9. The directive that prevents keep regular users from getting to the top-level root directory (/) after logging into a vsFTP server is **chroot_local_user=YES**.
10. Based on the default RHEL 6 configuration, both ftpusers and user_list in the /etc/vsftpd directory include a list of users not allowed to log in to the vsFTP server. Strictly speaking, the user_list file is the right answer, as it's associated with the userlist_enable directive. But the active contents of the standard ftpusers file are identical.
11. The default directory for anonymous access is the home directory of the user named ftp.

12. The additional directives to add to the default vsFTP configuration file to allow security using PAM and TCP wrappers are **pam_service_name=vsftpd** and **tcp_wrappers=YES**.

LAB ANSWERS

Lab 1: Basic NFS Configuration

When this lab is complete, you'll see the following features on the system with the NFS server:

- Installed NFS packages, including related packages such as portreserve and rpcbind. Without these packages, communication won't be possible with NFS clients.
- An active NFSv4 service, which can be confirmed in the output to the **rpcinfo -p** command.
- A firewall that supports access through the noted ports. It should also be limited by IP address network.

In addition, you'll be able to perform the following tasks from the NFS client:

- Run the **showmount -e server1.example.com** command, where server1.example.com is the name of the NFS server system (substitute if and as needed).
- Mount the shared directory as the root user with the **mount.nfs4 server1.example.com:/shared /testing** command.
- The first time the share is mounted, you should be able to copy local files as the root user to the /testing directory.
- The second time the share is mounted, with the no_root_squash directive in effect, such copying should not work, at least from the client root user account.

Lab 2: Standard NFS Configuration

This lab is the first step toward creating a single /home directory for your network. Once you get it working on a single client/server combination, you can set it up on all clients and servers. You can then use the LDAP server described in Chapter 8 to set up a single Linux/Unix database of usernames and passwords for the network. Alternatively, matching usernames (with matching UID and GID numbers) on different local systems should also work. On the NFS server, take the following steps:

1. Set up a couple of users and identifying files such as user1 and user1.txt on the system being used as the NFS server.
2. Configure the /etc/idmapd.conf file for the current domain, appropriate users, and the translation method for authentication. As discussed in the chapter, that refers to

```
Domain = example.com
Nobody-User = nfsnobody
```

```
Nobody-Group = nfsnobody
Method = nsswitch
```

3. Make the system reread the modified configuration file, using the /etc/init.d/rpcidmapd script.
4. Share the /home directory in /etc/exports on the server1.example.com client. You can do this in this file with the following command:

```
/home nfsclient(rw,sync)
```

5. Export this directory with the following command:

```
# exportfs -a
```

6. Restart the NFS service:

```
# service nfs restart
```

7. Make sure that the exported /home directory shows in the export list. On the local server, you can do this with the following command:

```
# showmount -e server1.example.com
```

8. If problems appear during this process, check the /etc/exports file carefully. Make sure there aren't extra spaces in /etc/exports, even at the end of a code line. Make sure the NFS service is actually running with the **service nfs status** command.
9. You may also want to check your firewall and make sure the appropriate services described in this chapter are running with the **rpcinfo -p** command.
10. Remember to make sure that the NFS server starts automatically the next time that system is booted. One way to do so is with the following command:

```
# chkconfig nfs on
```

Now on the NFS client, take the following steps to connect to the shared /home directory:

1. First, make the same changes to the /etc/idmapd.conf file as was done on the server, and restart the rpcidmapd service script available from the /etc/init.d directory.
2. Make sure that you can see the shared /home directory. If there is no DNS server or the /etc/hosts file does not include the IP address of the server1.example.com system, substitute the IP address:

```
# showmount -e server1.example.com
```

3. Now **mount** the share that is offered on the local /remote directory:

```
# mount.nfs4 server1.example.com:/home /remote
```

4. Check to see that the mounting has worked. If it did, you'll see the NFS mount in the output to the **mount** command.
5. Now look through the mounted /home directory for the special files that created in step 1. If found, you've succeeded in creating and connecting to the /home directory share.
6. To make the mount permanent, add it to the /etc/fstab file on the client. Once you've added a line such as the following to that file, the Linux client automatically mounts the shared /home directory from the NFS server.

```
server1.example.com:/home    /remote nfs4 soft,timeout=100  0  0
```

Lab 3: NFS and SELinux

The reference to SELinux is deliberate and should provide an important hint. If you have to modify every directory shared and configured in the /etc/exports file on each NFS server, that can be time consuming. Perhaps the simplest way to prevent writes to shared NFS directories is to deactivate the associated SELinux boolean setting. While there are other methods, the quickest way to do so on a permanent basis is with the following command:

```
# setsebool -P nfs_export_all_rw off
```

You should then be able to test the result with the next mounting of a shared NFS directory.

Lab 4: Configure a vsFTP Server with Messages

The vsFTP server can be installed from one RPM package. So if you have not installed this server earlier, you could install it with the following command:

```
# yum install vsftpd
```

This also installs configuration files in the /etc and /etc/vsftpd directories. The main configuration file is /etc/vsftpd/vsftpd.conf. Based on the RHEL default version of this file, you can make a couple of changes to enable messages. But by default, messages are already enabled for directory access on an FTP server, courtesy of the following command:

```
dirmessage_enable=yes
```

The actual configuration of a message is a matter of creating a text file and saving it as .message in the desired directories, /var/ftp and /var/ftp/pub. You could add a simple line such as "root directory for the FTP server" or "main download directory."

Finally, to configure the Red Hat FTP server to start, run the **service vsftpd start** command. To make sure it starts the next time you boot your computer, run the **chkconfig vsftpd on** command.

Lab 5: Configure a vsFTP Server for Anonymous-Only Downloads

This lab can be run as a continuation of Lab 4. To allow only anonymous access, comment out the following line in the /etc/vsftpd/vsftpd.conf file:

```
local_enable=yes
```

Anonymous users are already prevented from uploading files in the standard vsFTP server configuration. You could enable uploads by activating the **anon_upload_enable=yes** command, and by setting the upload directory to the public_content_rw_t SELinux file type.

Lab 6: Configure a vsFTP Server on a Special Directory

This lab is more complex than it sounds. The following highlights the required changes:

- Change the home directory of the user ftp to the /ftp directory; make sure the change is shown in /etc/passwd.
- Create the /ftp directory.
- Make sure the /ftp directory is set to the public_content_r SELinux file type.
- Make sure the public_content_r SELinux file type is documented in the file_contexts.local file in the /etc/selinux/targeted/contexts/files directory. As discussed in several previous chapters, that's possible with the **semanage** command; in this case, the command would be

```
# semanage fcontext -a -s system_u -t public_content_t /ftp/
```