CHAPTER 15

Linux FTP Server Setup

In This Chapter

- FTP Overview
- Problems with FTP and Firewalls
- How to Download and Install VSFTPD
- How to Get VSFTPD Started
- Testing the Status of VSFTPD
- The `vsftpd.conf` File
- FTP Security Issues
- Troubleshooting FTP
- Tutorial
- Conclusion

The **File Transfer Protocol** (**FTP**) is one of the most common means of copying files between servers over the Internet. Most Web-based download sites use the built-in FTP capabilities of Web browsers, and, therefore, most server-oriented operating systems usually include an FTP server application as part of the software suite. Linux is no exception.

This chapter will show you how to convert your Linux box into an FTP server using the default **Very Secure FTP Daemon** (**VSFTPD)** package included in Fedora.

FTP Overview

FTP relies on a pair of TCP ports to get the job done. It operates using two connection channels:

- **FTP control channel, TCP Port 21**: All commands you send, as well as the FTP server's responses to those commands, go over the control connection, but any data sent back (such as `ls` directory lists or actual file data in either direction) will go over the data connection.
- **FTP data channel,** TCP Port 20: This port is used for all subsequent data transfers between the client and server.

In addition to these channels, there are several varieties of FTP.

Types of FTP

From a networking perspective, the two main types of FTP are active and passive. In **active FTP**, the FTP server initiates a data transfer connection back to the client. For **passive FTP,** the connection is initiated from the FTP client. These are illustrated in Figure 15.1.

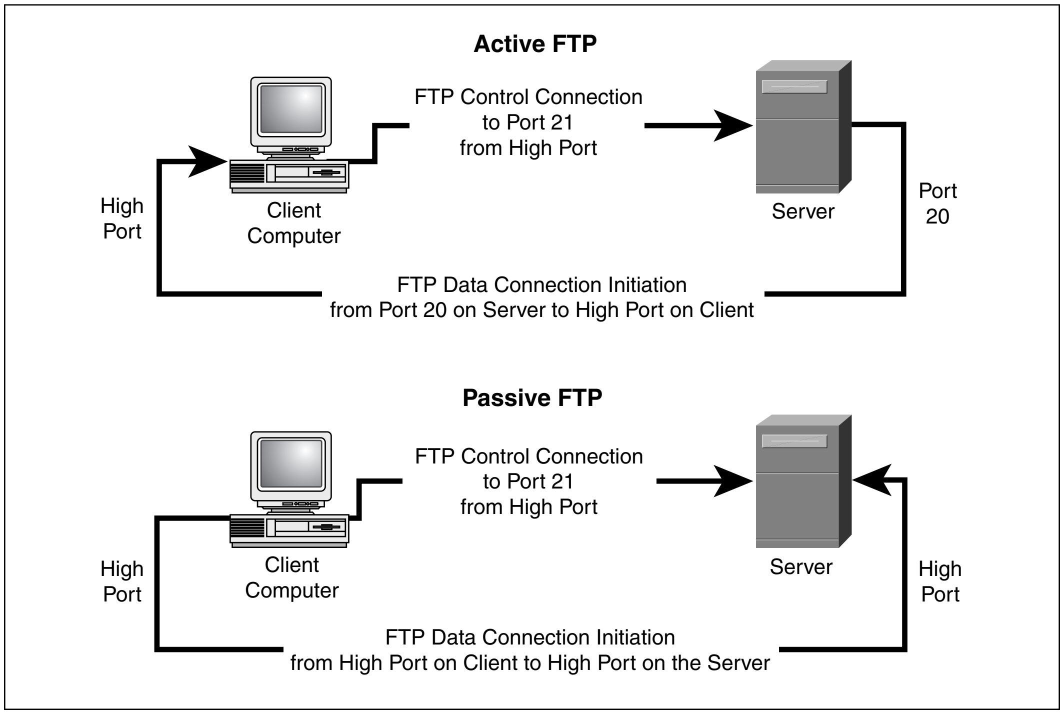

Figure 15.1 Active and passive FTP.

From a user management perspective, there are two additional types of FTP: **regular FTP**, in which files are transferred using the username and password of a regular user FTP server, and **anonymous FTP,** in which general access is provided to the FTP server using a well known universal login method.

Take a closer look at each type.

Active FTP

The sequence of events for active FTP is:

1. Your client connects to the FTP server by establishing an FTP control connection to port 21 of the server. Your commands such as `ls` and `get` are sent over this connection.
2. Whenever the client requests data over the control connection, the server initiates data transfer connections back to the client. The source port of these data transfer connections is always port 20 on the server, and the destination port is a high port (greater than 1024) on the client.
3. Thus the `ls` listing that you asked for comes back over the port 20 to high port connection, not the port 21 control connection.

FTP active mode, therefore, transfers data in a counter intuitive way to the TCP standard, as it selects port 20 as its source port (not a random high port that's greater than 1024) and connects back to the client on a random high port that has been pre-negotiated on the port 21 control connection.

Active FTP may fail in cases where the client is protected from the Internet via many to one NAT (masquerading), because the firewall will not know which of the many servers behind it should receive the return connection.

Passive FTP

Passive FTP works differently:

1. Your client connects to the FTP server by establishing an FTP control connection to port 21 of the server. Your commands such as `ls` and `get` are sent over that connection.
2. Whenever the client requests data over the control connection, the client initiates the data transfer connections to the server. The source port of these data transfer connections is always a high port on the client with a destination port of a high port on the server.

Passive FTP should be viewed as the server never making an active attempt to connect to the client for FTP data transfers. Because the client always initiates the required connections, passive FTP works better for clients protected by a firewall.

As Windows defaults to active FTP and Linux defaults to passive, you'll probably have to accommodate both forms when deciding upon a security policy for your FTP server.

Regular FTP

By default, the VSFTPD package allows regular Linux users to copy files to and from their home directories with an FTP client using their Linux usernames and passwords as their login credentials.

VSFTPD also has the option of allowing this type of access to only a group of Linux users, enabling you to restrict the addition of new files to your system to authorized personnel.

The disadvantage of regular FTP is that it isn't suitable for general download distribution of software as everyone either has to get a unique Linux user account or has to use a shared username and password. Anonymous FTP allows you to avoid this difficulty.

Anonymous FTP

Anonymous FTP is the choice of Web sites that need to exchange files with numerous unknown remote users. Common uses include downloading software updates and MP3s and uploading diagnostic information for a technical support engineers' attention. Unlike regular FTP where you login with a preconfigured Linux username and password, anonymous FTP requires only a username of anonymous and your e-mail address for the password. Once logged into a VSFTPD server, you automatically have access to only the default anonymous FTP directory (`/var/ftp` in the case of VSFTPD) and all its subdirectories.

As seen in Chapter 6, "Installing RPM Software," using anonymous FTP as a remote user is fairly straightforward. VSFTPD can be configured to support user-based and/or anonymous FTP in its configuration file, as you'll see later.

PROBLEMS WITH FTP AND FIREWALLS

FTP frequently fails when the data has to pass through a firewall, because firewalls are designed to limit data flows to predictable TCP ports and FTP uses a wide range of unpredictable TCP ports. You have a choice of methods to overcome this.

Note

Appendix II, "Codes, Scripts, and Configurations," contains examples of how to configure the **VSFTPD** Linux firewall to function with both active and passive FTP.

Client Protected by a Firewall Problem

Typically firewalls don't allow any incoming connections at all, which frequently blocks active FTP from functioning. With this type of FTP failure, the active FTP connection appears to work when the client initiates an outbound connection to the server on port 21. The connection then appears to hang, however, as soon as you use the `ls`, `dir`, or `get commands`. The reason is that the firewall is blocking the return connection from the server to the client (from port

20 on the server to a high port on the client). If a firewall allows all outbound connections to the Internet, then passive FTP clients behind a firewall will usually work correctly as the clients initiate all the FTP connections.

Solution

Table 15.1 shows the general rules you'll need to allow FTP clients through a firewall.

Table 15.1 Client Protected by Firewall: Required Rules for FTP

Method	Source Address	Source Port	Destination Address	Destination Port	Connection Type
Allow outgoing control connections to server					
Control channel	FTP client/ network	High[1]	FTP server[2]	21	New
	FTP server[2]	21	FTP client/ network	High	Established[3]
Allow the client to establish data channels to remote server					
Active FTP	FTP server[2]	20	FTP client/ network	High	New
	FTP client/ network	High	FTP server[2]	20	Established[3]
Passive FTP	FTP client/ network	High	FTP server[2]	High	New
	FTP server[2]	High	FTP client/ network	High	Established[3]

[1] Greater than 1024.

[2] In some cases, you may want to allow all Internet users to have access, not just a specific client, server, or network.

[3] Many home-based firewall routers automatically allow traffic for already established connections. This rule may not be necessary in all cases.

Server Protected by a Firewall Problem

Typically, firewalls don't let any connections come in at all. When an incorrectly configured firewall protects an FTP server, the FTP connection from the client doesn't appear to work at all for both active and passive FTP.

Solution

Table 15.2 outlines the general rules needed to allow FTP servers through a firewall.

Table 15.2 Server Protected by Firewall: Required Rules for FTP

Method	Source Address	Source Port	Destination Address	Destination Port	Connection Type
Allow incoming control connections to server					
Control channel	FTP client/ network[1]	High[2]	FTP server	21	New
	FTP server	21	FTP client/ network[1]	High	Established[3]
Allow server to establish data channel to remote client					
Active FTP	FTP server	20	FTP client/ network[1]	High	New
	FTP client/ network[1]	High	FTP server	20	Established[3]
Passive FTP	FTP client/ network[1]	High	FTP server	High	New
	FTP server	High	FTP client/ network[1]	High	Established[3]

[1] In some cases, you may want to allow all Internet users to have access, not just a specific client, server, or network.

[2] Greater than 1024.

[3] Many home-based firewall routers automatically allow traffic for already established connections. This rule may not be necessary in all cases.

How to Download and Install VSFTPD

Most Red Hat and Fedora Linux software products are available in the RPM format. Downloading and installing RPMs isn't hard. If you need a refresher, Chapter 6 covers how to do this in detail. It is best to use the latest version of VSFTPD.

When searching for the file, remember that the VSFTPD RPM's filename usually starts with the word "`vsftpd`" followed by a version number, as in `vsftpd-1.2.1-5.i386.rpm`.

How to Get VSFTPD Started

You can start, stop, or restart VSFTPD after booting using these commands:

```
[root@bigboy tmp]# service vsftpd start
[root@bigboy tmp]# service vsftpd stop
[root@bigboy tmp]# service vsftpd restart
```

To configure VSFTPD to start at boot, use the `chkconfig` command:

```
[root@bigboy tmp]# chkconfig vsftpd on
```

Note

In Red Hat Linux version 8.0 and earlier, VSFTPD operation is controlled by the `xinetd` process, which is covered in Chapter 16, "TELNET, TFTP, and XINETD." You can find a full description of how to configure these versions of Linux for VSFTPD in Appendix III, "Fedora Version Differences."

TESTING THE STATUS OF VSFTPD

You can always test whether the VSFTPD process is running by using the `netstat -a` command, which lists all the TCP and UDP ports on which the server is listening for traffic. This example shows the expected output:

```
[root@bigboy root]# netstat -a | grep ftp
tcp        0        0        *:ftp        *:*        LISTEN
[root@bigboy root]#
```

If VSFTPD wasn't running, there would be no output at all.

THE *VSFTPD.CONF* FILE

VSFTPD reads the contents of its `vsftpd.conf` configuration file only when it starts, so you'll have to restart VSFTPD each time you edit the file in order for the changes to take effect.

This file uses a number of default settings you need to know about:

☞ **VSFTPD runs as an anonymous FTP server:** Unless you want any remote user to log into to your default FTP directory using a username of `anonymous` and a password that's the same as their e-mail address, I suggest turning this off. You can set the configuration file's `anonymous_enable` directive to no to disable this feature. You'll also need to simultaneously enable local users to be able to log in by removing the comment symbol (#) before the `local_enable` instruction.

☞ **VSFTPD allows only anonymous FTP downloads to remote users, not uploads from them**: You can change this by modifying the `anon_upload_enable` directive shown later.

☞ **VSFTPD doesn't allow anonymous users to create directories on your FTP server**: You can change this by modifying the `anon_mkdir_write_enable` directive.

☞ **VSFTPD logs FTP access to the `/var/log/vsftpd.log` log file**: You can change this by modifying the `xferlog_file` directive.

☞ **VSFTPD expects files for anonymous FTP to be placed in the `/var/ftp` directory**: You can change this by modifying the `anon_root` directive. There is always the risk with anonymous FTP that users will discover a way to write files to your anonymous FTP directory. You run the risk of filling up your `/var` partition if you use the default setting. It is best to make the anonymous FTP directory reside in its own dedicated partition.

The configuration file is fairly straightforward as you can see in the snippet:

```
# Allow anonymous FTP?
anonymous_enable=YES



# Uncomment this to allow local users to log in.
local_enable=YES

# Uncomment this to enable any form of FTP write command.
# (Needed even if you want local users to be able to upload files)
write_enable=YES

# Uncomment to allow the anonymous FTP user to upload files. This only
# has an effect if global write enable is activated. Also, you will
# obviously need to create a directory writable by the FTP user.
#anon_upload_enable=YES

# Uncomment this if you want the anonymous FTP user to be able to
create
# new directories.
#anon_mkdir_write_enable=YES

# Activate logging of uploads/downloads.
xferlog_enable=YES

# You may override where the log file goes if you like.
# The default is shown# below.
#xferlog_file=/var/log/vsftpd.log

# The directory which vsftpd will try to change
# into after an anonymous login. (Default = /var/ftp)
#anon_root=/data/directory
```

To activate or deactivate a feature, remove or add the # at the beginning of the appropriate line.

Other *vsftpd.conf* Options

There are many other options you can add to this file:

- Limiting the maximum number of client connections (`max_clients`)
- Limiting the number of connections by source IP address (`max_per_ip`)
- Setting the maximum rate of data transfer per anonymous login (`anon_max_rate`)
- Setting the maximum rate of data transfer per non-anonymous login (`local_max_rate`)

Descriptions on this and more can be found in the `vsftpd.conf` man pages.

FTP SECURITY ISSUES

FTP has a number of security drawbacks, but you can overcome them in some cases. You can restrict an individual Linux user's access to non-anonymous FTP, and you can change the configuration to not display the FTP server's software version information, but unfortunately, though very convenient, FTP logins and data transfers are not encrypted.

The */etc/vsftpd.ftpusers* File

For added security, you may restrict FTP access to certain users by adding them to the list of users in the `/etc/vsftpd.ftpusers` file. The VSFTPD package creates this file with a number of entries for privileged users that normally shouldn't have FTP access. As FTP doesn't encrypt passwords, thereby increasing the risk of data or passwords being compromised, it is a good idea to let these entries remain and add new entries for additional security.

Anonymous Upload

If you want remote users to write data to your FTP server, then you should create a write-only directory within `/var/ftp/pub`. This will allow your users to upload but not access other files uploaded by other users. The commands you need are:

```
[root@bigboy tmp]# mkdir /var/ftp/pub/upload
[root@bigboy tmp]# chmod 722 /var/ftp/pub/upload
```

FTP Greeting Banner

Change the default greeting banner in the `vsftpd.conf` file to make it harder for malicious users to determine the type of system you have. The directive in this file is:

```
ftpd_banner= New Banner Here
```

Using SCP as Secure Alternative to FTP

One of the disadvantages of FTP is that it does not encrypt your username and password. This could make your user account vulnerable to an unauthorized attack from a person eavesdropping on the network connection. **Secure Copy** (**SCP**) and **Secure FTP** (**SFTP**) provide encryption and could be considered as an alternative to FTP for trusted users. `SCP` does not support anonymous services, however, a feature that FTP does support.

TROUBLESHOOTING FTP

You should always test your FTP installation by attempting to use an FTP client to log into your FTP server to transfer sample files.

The most common sources of day-to-day failures are incorrect usernames and passwords.

Initial setup failures could be caused by firewalls along the path between the client and server blocking some or all types of FTP traffic. Typical symptoms of this are either connection timeouts or the ability to use the `ls` command to view the contents of a directory without the ability to either upload or download files. Follow the firewall rule guidelines to help overcome this problem. Connection problems could also be the result of typical network issues outlined in Chapter 4, "Simple Network Troubleshooting."

TUTORIAL

FTP has many uses, one of which is allowing numerous unknown users to download files. You have to be careful, because you run the risk of accidentally allowing unknown persons to upload files to your server. This sort of unintended activity can quickly fill up your hard drive with illegal software, images, and music for the world to download, which in turn can clog your server's Internet access and drive up your bandwidth charges.

FTP Users with Read-Only Access to a Shared Directory

In this example, anonymous FTP is not desired, but a group of trusted users need to have read-only access to a directory for downloading files. Here are the steps:

1. Disable anonymous FTP. Comment out the `anonymous_enable` line in the `vsftpd.conf` file:

```
# Allow anonymous FTP?
# anonymous_enable=YES
```

2. Enable individual logins by making sure you have the `local_enable` line uncommented in the `vsftpd.conf` file:

```
# Uncomment this to allow local users to log in.
local_enable=YES
```

3. Start VSFTP.

```
[root@bigboy tmp]# service vsftpd start
```

4. Create a user group and shared directory. In this case, use `/home/ftp-users` and a user group name of ftp-users for the remote users:

```
[root@bigboy tmp]# groupadd ftp-users
[root@bigboy tmp]# mkdir /home/ftp-docs
```

5. Make the directory accessible to the ftp-users group:

```
[root@bigboy tmp]# chmod 750 /home/ftp-docs
[root@bigboy tmp]# chown root:ftp-users /home/ftp-docs
```

6. Add users, and make their default directory `/home/ftp-docs:`

```
[root@bigboy tmp]# useradd -g ftp-users -d /home/ftp-docs user1
[root@bigboy tmp]# useradd -g ftp-users -d /home/ftp-docs user2
[root@bigboy tmp]# useradd -g ftp-users -d /home/ftp-docs user3
[root@bigboy tmp]# useradd -g ftp-users -d /home/ftp-docs user4
[root@bigboy tmp]# passwd user1
[root@bigboy tmp]# passwd user2
[root@bigboy tmp]# passwd user3
[root@bigboy tmp]# passwd user4
```

7. Copy files to be downloaded by your users into the `/home/ftp-docs` directory.
8. Change the permissions of the files in the `/home/ftp-docs` directory to read-only access by the group:

```
[root@bigboy tmp]# chown root:ftp-users /home/ftp-docs/*
[root@bigboy tmp]# chmod 740 /home/ftp-docs/*
```

Users should now be able to log in via FTP to the server using their new usernames and passwords. If you absolutely don't want any FTP users to be able to write to any directory, then you should set the `write_enable` line in your `vsftpd.conf` file to no:

```
write_enable = NO
```

Remember, you must restart VSFTPD for the configuration file changes to take effect.

Sample Login Session to Test Functionality

Here is a simple test procedure you can use to make sure everything is working correctly:

1. Check for the presence of a test file on the FTP client server.

```
[root@smallfry tmp]# ll
total 1
-rw-r--r-- 1 root root 0 Jan 4 09:08 testfile
[root@smallfry tmp]#
```

2. Connect to Bigboy via FTP:

```
[root@smallfry tmp]# ftp 192.168.1.100
Connected to 192.168.1.100 (192.168.1.100)
220 ready, dude (vsFTPd 1.1.0: beat me, break me)
Name (192.168.1.100:root): user1
331 Please specify the password.
Password:
230 Login successful. Have fun.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

As expected, you can't do an upload transfer of testfile to bigboy:

```
ftp> put testfile
local: testfile remote: testfile
227 Entering Passive Mode (192,168,1,100,181,210)
553 Could not create file.
ftp>
```

But we can view and download a copy of the VSFTPD RPM on the FTP server bigboy:

```
ftp> ls
227 Entering Passive Mode (192,168,1,100,35,173)
150 Here comes the directory listing.
-rwxr----- 1 0 502 76288 Jan 04 17:06 vsftpd-1.1.0-1.i386.rpm
```

```
226 Directory send OK.
ftp> get vsftpd-1.1.0-1.i386.rpm vsftpd-1.1.0-1.i386.rpm.tmp
local: vsftpd-1.1.0-1.i386.rpm.tmp remote: vsftpd-1.1.0-
1.i386.rpm
227 Entering Passive Mode (192,168,1,100,44,156)
150 Opening BINARY mode data connection for vsftpd-1.1.0-
1.i386.rpm (76288 bytes).
226 File send OK.
76288 bytes received in 0.499 secs (1.5e+02 Kbytes/sec)
ftp> exit
221 Goodbye.
[root@smallfry tmp]#
```

As expected, an anonymous FTP fails:

```
[root@smallfry tmp]# ftp 192.168.1.100
Connected to 192.168.1.100 (192.168.1.100)
220 ready, dude (vsFTPd 1.1.0: beat me, break me)
Name (192.168.1.100:root): anonymous
331 Please specify the password.
Password:
530 Login incorrect.
Login failed.
ftp> quit
221 Goodbye.
[root@smallfry tmp]#
```

Now that testing is complete, you can make this a regular part of your FTP server's operation.

CONCLUSION

FTP is a very useful software application that can have enormous benefit to a Web site or to collaborative computing in which files need to be shared between business partners. Although insecure, it is universally accessible, because FTP clients are a part of all operating systems and Web browsers. If data encryption security is of great importance to you, then you should probably consider SCP as a possible alternative. You can find more information on it in Chapter 17, "Secure Remote Logins and File Copying."