Introduction

The File Transfer Protocol (FTP) is used as 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 in two connection channels as I'll explain:

FTP Control Channel, TCP Port 21: All commands you send and the ftp server's responses to those commands will 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.

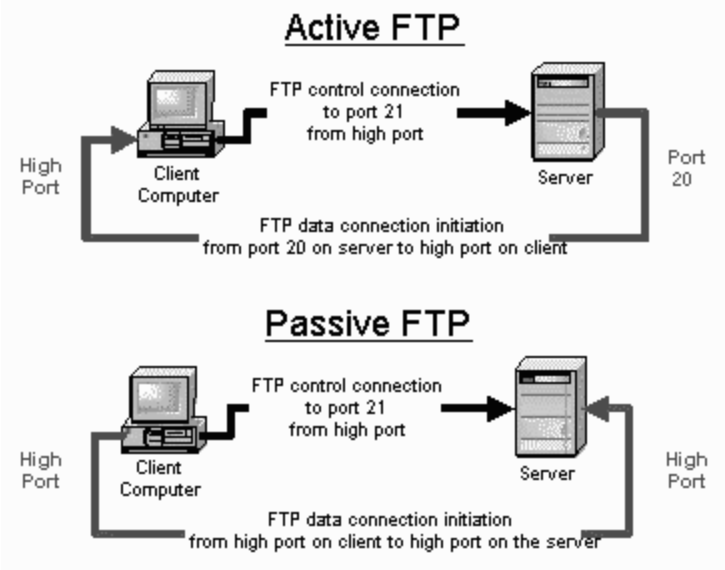
FTP Data Channel, TCP Port 20: This port is used for all subsequent data transfers between the client and server.

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 Illustrated



From a user management perspective there are also two 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 it's 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). This is 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 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 email address for the password. Once logged in to 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 Linux Software", using anonymous FTP as a remote user is fairly straight forward. VSFTPD can be configured to support user-based and or anonymous FTP in its configuration file which 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: The 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 a 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 Rules needed to allow FTP servers through a firewall.

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

How To Download And Install VSFTPD

Most Linux software products are available in a precompiled package format. Downloading and installing packages isn't hard. If you need a refresher, Chapter 6, "Installing Linux Software", 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 packages' filename usually starts with the word vsftpd followed by a version number, as in `vsftpd-1.2.1-5.i386.rpm` for Redhat/Fedora or `vsftpd_2.0.4-0ubuntu4_i386.deb` for Ubuntu.

How To Get VSFTPD Started

With Fedora, Redhat, Ubunbtu and Debian You can start, stop, or restart VSFTPD after booting by using these commands:

```
[root@bigboy tmp]# /etc/init.d/vsftpd start
[root@bigboy tmp]# /etc/init.d/vsftpd stop
[root@bigboy tmp]# /etc/init.d/vsftpd restart
```

With Redhat / Fedora you can configure VSFTPD to start at boot you can use the chkconfig command.

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

With Ubuntu / Debian the sysv-rc-conf command can be used like this:

```
root@u-bigboy:/tmp# sysv-rc-conf on
```

Note: In RedHat 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 only 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. The file may be located in either the `/etc` or the `/etc/vsftpd` directories depending on your Linux distribution.

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 email address, I would suggest turning this off. The configuration file's anonymous_enable directive can be set 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.
- If you enable anonymous FTP with VSFTPD, remember to define the root directory that visitors will visit. This is done with the anon_root directive.

```
anon_root=/data/directory
```

- VSFTPD allows only anonymous FTP downloads to remote users, not uploads from them. This can be changed 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.
- By default 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 straight forward as you can see in the snippet below where we enable anonymous FTP and individual accounts simultaneously.

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

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)
- The maximum rate of data transfer per anonymous login. (anon_max_rate)
- 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 in to 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 Only Read 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 like this:

```
# Allow anonymous FTP?
anonymous_enable=NO
```

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

```
# 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 for 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, we 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 located 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, 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

NAME

vsftpd.conf - config file for vsftpd

DESCRIPTION

vsftpd.conf may be used to control various aspects of vsftpd's behaviour. By default, vsftpd looks for this file at the location **/etc/vsftpd.conf**. However, you may override this by specifying a command line argument to vsftpd. The command line argument is the pathname of the configuration file for vsftpd. This behaviour is useful because you may wish to use an advanced inetd such as **xinetd** to launch vsftpd with different configuration files on a per virtual host basis.

FORMAT

The format of vsftpd.conf is very simple. Each line is either a comment or a directive. Comment lines start with a # and are ignored. A directive line has the format:

option=value

It is important to note that it is an error to put any space between the option, = and value.

Each setting has a compiled in default which may be modified in the configuration file.

BOOLEAN OPTIONS

Below is a list of boolean options. The value for a boolean option may be set to **YES** or **NO**.

allow_anon_ssl

Only applies if **ssl_enable** is active. If set to YES, anonymous users will be allowed to use secured SSL connections.

Default: NO

anon_mkdir_write_enable

If set to YES, anonymous users will be permitted to create new directories under certain conditions. For this to work, the option **write_enable** must be activated, and the anonymous ftp user must have write permission on the parent directory.

Default: NO

anon_other_write_enable

If set to YES, anonymous users will be permitted to perform write operations other than upload and create directory, such as deletion and renaming. This is generally not recommended but included for completeness.

Default: NO

anon_upload_enable

If set to YES, anonymous users will be permitted to upload files under certain conditions. For this to work, the option **write_enable** must be activated, and the anonymous ftp user must have write permission on desired upload locations. This setting is also required for virtual users to upload; by default, virtual users are treated with anonymous (i.e. maximally restricted) privilege.

Default: NO

anon_world_readable_only

When enabled, anonymous users will only be allowed to download files which are world readable. This is recognising that the ftp user may own files, especially in the presence of uploads.

Default: YES

anonymous_enable

Controls whether anonymous logins are permitted or not. If enabled, both the usernames **ftp** and **anonymous** are recognised as anonymous logins.

Default: YES

ascii_download_enable

When enabled, ASCII mode data transfers will be honoured on downloads.

Default: NO

ascii_upload_enable

When enabled, ASCII mode data transfers will be honoured on uploads.

Default: NO

async_abor_enable

When enabled, a special FTP command known as "async ABOR" will be enabled. Only ill advised FTP clients will use this feature. Additionally, this feature is awkward to handle, so it is disabled by default. Unfortunately, some FTP clients will hang when cancelling a transfer unless this feature is available, so you may wish to enable it.

Default: NO

background

When enabled, and vsftpd is started in "listen" mode, vsftpd will background the listener process. i.e. control will immediately be returned to the shell which launched vsftpd.

Default: NO

check_shell

Note! This option only has an effect for non-PAM builds of vsftpd. If disabled, vsftpd will not check /etc/shells for a valid user shell for local logins.

Default: YES

chmod_enable

When enabled, allows use of the SITE CHMOD command. NOTE! This only applies to local users. Anonymous users never get to use SITE CHMOD.

Default: YES

chown_uploads

If enabled, all anonymously uploaded files will have the ownership changed to the user specified in the setting **chown_username**. This is useful from an administrative, and perhaps security, standpoint.

Default: NO

chroot_list_enable

If activated, you may provide a list of local users who are placed in a chroot() jail in their home directory upon login. The meaning is slightly different if chroot_local_user is set to YES. In this case, the list becomes a list of users which are NOT to be placed in a chroot() jail. By default, the file containing this list is /etc/vsftpd.chroot_list, but you may override this with the **chroot_list_file** setting.

Default: NO

chroot_local_user

If set to YES, local users will be (by default) placed in a chroot() jail in their home directory after login. **Warning:** This option has security implications, especially if the users have upload permission, or shell access. Only enable if you know what you are doing. Note that these security implications are not vsftpd specific. They apply to all FTP daemons which offer to put local users in chroot() jails.

Default: NO

connect_from_port_20

This controls whether PORT style data connections use port 20 (ftp-data) on the server machine. For security reasons, some clients may insist that this is the case. Conversely, disabling this option enables vsftpd to run with slightly less privilege.

Default: NO (but the sample config file enables it)

debug_ssl

If true, OpenSSL connection diagnostics are dumped to the vsftpd log file. (Added in v2.0.6).

Default: NO

delete_failed_uploads

If true, any failed upload files are deleted. (Added in v2.0.7).

Default: NO

deny_email_enable

If activated, you may provide a list of anonymous password e-mail responses which cause login to be denied. By default, the file containing this list is /etc/vsftpd.banned_emails, but you may override this with the **banned_email_file** setting.

Default: NO

dirlist_enable

If set to NO, all directory list commands will give permission denied.

Default: YES

dirmessage_enable

If enabled, users of the FTP server can be shown messages when they first enter a new directory. By default, a directory is scanned for the file .message, but that may be overridden with the configuration setting **message_file**.

Default: NO (but the sample config file enables it)

download_enable

If set to NO, all download requests will give permission denied.

Default: YES

dual_log_enable

If enabled, two log files are generated in parallel, going by default to **/var/log/xferlog** and **/var/log/vsftpd.log**. The former is a wu-ftpd style transfer log, parseable by standard tools. The latter is vsftpd's own style log.

Default: NO

force_dot_files

If activated, files and directories starting with . will be shown in directory listings even if the "a" flag was not used by the client. This override excludes the "." and ".." entries.

Default: NO

force_anon_data_ssl

Only applies if **ssl_enable** is activated. If activated, all anonymous logins are forced to use a secure SSL connection in order to send and receive data on data connections.

Default: NO

force_anon_logins_ssl

Only applies if **ssl_enable** is activated. If activated, all anonymous logins are forced to use a secure SSL connection in order to send the password.

Default: NO

force_local_data_ssl

Only applies if **ssl_enable** is activated. If activated, all non-anonymous logins are forced to use a secure SSL connection in order to send and receive data on data connections.

Default: YES

force_local_logins_ssl

Only applies if **ssl_enable** is activated. If activated, all non-anonymous logins are forced to use a secure SSL connection in order to send the password.

Default: YES

guest_enable

If enabled, all non-anonymous logins are classed as "guest" logins. A guest login is remapped to the user specified in the **guest_username** setting.

Default: NO

hide_ids

If enabled, all user and group information in directory listings will be displayed as "ftp".

Default: NO

implicit_ssl

If enabled, an SSL handshake is the first thing expect on all connections (the FTPS protocol). To support explicit SSL and/or plain text too, a separate vsftpd listener process should be run.

Default: NO

listen

If enabled, vsftpd will run in standalone mode. This means that vsftpd must not be run from an inetd of some kind. Instead, the vsftpd executable is run once directly. vsftpd itself will then take care of listening for and handling incoming connections.

Default: YES

listen_ipv6

Like the listen parameter, except vsftpd will listen on an IPv6 socket instead of an IPv4 one. This parameter and the listen parameter are mutually exclusive.

Default: NO

local_enable

Controls whether local logins are permitted or not. If enabled, normal user accounts in /etc/passwd (or wherever your PAM config references) may be used to log in. This must be enable for any non-anonymous login to work, including virtual users.

Default: NO

lock_upload_files

When enabled, all uploads proceed with a write lock on the upload file. All downloads proceed with a shared read lock on the download file. WARNING! Before enabling this, be aware that malicious readers could starve a writer wanting to e.g. append a file.

Default: YES

log_ftp_protocol

When enabled, all FTP requests and responses are logged, providing the option xferlog_std_format is not enabled. Useful for debugging.

Default: NO

ls_recurse_enable

When enabled, this setting will allow the use of "ls -R". This is a minor security risk, because a ls -R at the top level of a large site may consume a lot of resources.

Default: NO

mdtm_write

When enabled, this setting will allow MDTM to set file modification times (subject to the usual access checks).

Default: YES

no_anon_password

When enabled, this prevents vsftpd from asking for an anonymous password - the anonymous user will log straight in.

Default: NO

no_log_lock

When enabled, this prevents vsftpd from taking a file lock when writing to log files. This option should generally not be enabled. It exists to workaround operating system bugs such as the Solaris / Veritas filesystem combination which has been observed to sometimes exhibit hangs trying to lock log files.

Default: NO

one_process_model

If you have a Linux 2.4 kernel, it is possible to use a different security model which only uses one process per connection. It is a less pure security model, but gains you performance. You really don't want to enable this unless you know what you are doing, and your site supports huge numbers of simultaneously connected users.

Default: NO

passwd_chroot_enable

If enabled, along with **chroot_local_user** , then a chroot() jail location may be specified on a per-user basis. Each user's jail is derived from their home directory string in /etc/passwd. The occurrence of /./ in the home directory string denotes that the jail is at that particular location in the path.

Default: NO

pasv_addr_resolve

Set to YES if you want to use a hostname (as opposed to IP address) in the **pasv_address** option.

Default: NO

pasv_enable

Set to NO if you want to disallow the PASV method of obtaining a data connection.

Default: YES

pasv_promiscuous

Set to YES if you want to disable the PASV security check that ensures the data connection originates from the same IP address as the control connection. Only enable if you know what you are doing! The only legitimate use for this is in some form of secure tunnelling scheme, or perhaps to facilitate FXP support.

Default: NO

port_enable

Set to NO if you want to disallow the PORT method of obtaining a data connection.

Default: YES

port_promiscuous

Set to YES if you want to disable the PORT security check that ensures that outgoing data connections can only connect to the client. Only enable if you know what you are doing!

Default: NO

require_cert

If set to yes, all SSL client connections are required to present a client certificate. The degree of validation applied to this certificate is controlled by **validate_cert** (Added in v2.0.6).

Default: NO

require_ssl_reuse

If set to yes, all SSL data connections are required to exhibit SSL session reuse (which proves that they know the same master secret as the control channel). Although this is a secure default, it may break many FTP clients, so you may want to disable it. For a discussion of the consequences, see http://scarybeastsecurity.blogspot.com/2009/02/vsftpd-210-released.html (Added in v2.1.0).

Default: YES

run_as_launching_user

Set to YES if you want vsftpd to run as the user which launched vsftpd. This is useful where root access is not available. MASSIVE WARNING! Do NOT enable this option unless you totally know what you are doing, as naive use of this option can create massive security problems. Specifically, vsftpd does not / cannot use chroot technology to restrict file access when this option is set (even if launched by root). A poor substitute could be to use a **deny_file** setting such as {/*,*..*}, but the reliability of this cannot compare to chroot, and should not be relied on. If using this option, many restrictions on other options apply. For example, options requiring privilege such as non-anonymous logins, upload ownership changing, connecting from port 20 and listen ports less than 1024 are not expected to work. Other options may be impacted.

Default: NO

secure_email_list_enable

Set to YES if you want only a specified list of e-mail passwords for anonymous logins to be accepted. This is useful as a low-hassle way of restricting access to low-security content without needing virtual users. When enabled, anonymous logins are prevented unless the password provided is listed in the file specified by the **email_password_file** setting. The file format is one password per line, no extra whitespace. The default filename is /etc/vsftpd.email_passwords.

Default: NO

session_support

This controls whether vsftpd attempts to maintain sessions for logins. If vsftpd is maintaining sessions, it will try and update utmp and wtmp. It will also open a pam_session if using PAM to authenticate, and only close this upon logout. You may wish to disable this if you do not need

session logging, and you wish to give vsftpd more opportunity to run with less processes and / or less privilege. NOTE - utmp and wtmp support is only provided with PAM enabled builds.

Default: NO

setproctitle_enable

If enabled, vsftpd will try and show session status information in the system process listing. In other words, the reported name of the process will change to reflect what a vsftpd session is doing (idle, downloading etc). You probably want to leave this off for security purposes.

Default: NO

ssl_enable

If enabled, and vsftpd was compiled against OpenSSL, vsftpd will support secure connections via SSL. This applies to the control connection (including login) and also data connections. You'll need a client with SSL support too. NOTE!! Beware enabling this option. Only enable it if you need it. vsftpd can make no guarantees about the security of the OpenSSL libraries. By enabling this option, you are declaring that you trust the security of your installed OpenSSL library.

Default: NO

ssl_request_cert

If enabled, vsftpd will request (but not necessarily require; see **require_cert)**a**certificate**on**incoming**SSL**connections.**Normally**this** should not cause any trouble at all, but IBM zOS seems to have issues. (New in v2.0.7).

Default: YES

ssl_sslv2

Only applies if **ssl_enable** is activated. If enabled, this option will permit SSL v2 protocol connections. TLS v1 connections are preferred.

Default: NO

ssl_sslv3

Only applies if **ssl_enable** is activated. If enabled, this option will permit SSL v3 protocol connections. TLS v1 connections are preferred.

Default: NO

ssl_tlsv1

Only applies if **ssl_enable** is activated. If enabled, this option will permit TLS v1 protocol connections. TLS v1 connections are preferred.

Default: YES

strict_ssl_read_eof

If enabled, SSL data uploads are required to terminate via SSL, not an EOF on the socket. This option is required to be sure that an attacker did not terminate an upload prematurely with a faked TCP FIN. Unfortunately, it is not enabled by default because so few clients get it right. (New in v2.0.7).

Default: NO

strict_ssl_write_shutdown

If enabled, SSL data downloads are required to terminate via SSL, not an EOF on the socket. This is off by default as I was unable to find a single FTP client that does this. It is minor. All it affects is our ability to tell whether the client confirmed full receipt of the file. Even without this option, the client is able to check the integrity of the download. (New in v2.0.7).

Default: NO

syslog_enable

If enabled, then any log output which would have gone to /var/log/vsftpd.log goes to the system log instead. Logging is done under the FTPD facility.

Default: NO

tcp_wrappers

If enabled, and vsftpd was compiled with tcp_wrappers support, incoming connections will be fed through tcp_wrappers access control. Furthermore, there is a mechanism for per-IP based configuration. If tcp_wrappers sets the VSFTPD_LOAD_CONF environment variable, then the vsftpd session will try and load the vsftpd configuration file specified in this variable.

Default: NO

text_userdb_names

By default, numeric IDs are shown in the user and group fields of directory listings. You can get textual names by enabling this parameter. It is off by default for performance reasons.

Default: NO

tilde_user_enable

If enabled, vsftpd will try and resolve pathnames such as ~chris/pics, i.e. a tilde followed by a username. Note that vsftpd will always resolve the pathnames ~ and ~/something (in this case the ~ resolves to the initial login directory). Note that ~user paths will only resolve if the file **/etc/passwd** may be found within the _current_ chroot() jail.

Default: NO

use_localtime

If enabled, vsftpd will display directory listings with the time in your local time zone. The default is to display GMT. The times returned by the MDTM FTP command are also affected by this option.

Default: NO

use_sendfile

An internal setting used for testing the relative benefit of using the sendfile() system call on your platform.

Default: YES

userlist_deny

This option is examined if **userlist_enable** is activated. If you set this setting to NO, then users will be denied login unless they are explicitly listed in the file specified by **userlist_file**. When login is denied, the denial is issued before the user is asked for a password.

Default: YES

userlist_enable

If enabled, vsftpd will load a list of usernames, from the filename given by **userlist_file**. If a user tries to log in using a name in this file, they will be denied before they are asked for a password. This may be useful in preventing cleartext passwords being transmitted. See also **userlist_deny**.

Default: NO

validate_cert

If set to yes, all SSL client certificates received must validate OK. Self-signed certs do not constitute OK validation. (New in v2.0.6).

Default: NO

virtual_use_local_privs

If enabled, virtual users will use the same privileges as local users. By default, virtual users will use the same privileges as anonymous users, which tends to be more restrictive (especially in terms of write access).

Default: NO

write_enable

This controls whether any FTP commands which change the filesystem are allowed or not. These commands are: STOR, DELE, RNFR, RNTO, MKD, RMD, APPE and SITE.

Default: NO

xferlog_enable

If enabled, a log file will be maintained detailling uploads and downloads. By default, this file will be placed at /var/log/vsftpd.log, but this location may be overridden using the configuration setting **vsftpd_log_file**.

Default: NO (but the sample config file enables it)

xferlog_std_format

If enabled, the transfer log file will be written in standard xferlog format, as used by wu-ftpd. This is useful because you can reuse existing transfer statistics generators. The default format is more readable, however. The default location for this style of log file is /var/log/xferlog, but you may change it with the setting **xferlog_file**.

Default: NO

NUMERIC OPTIONS

Below is a list of numeric options. A numeric option must be set to a non negative integer. Octal numbers are supported, for convenience of the umask options. To specify an octal number, use 0 as the first digit of the number.

accept_timeout

The timeout, in seconds, for a remote client to establish connection with a PASV style data connection.

Default: 60

anon_max_rate

The maximum data transfer rate permitted, in bytes per second, for anonymous clients.

Default: 0 (unlimited)

anon_umask

The value that the umask for file creation is set to for anonymous users. NOTE! If you want to specify octal values, remember the "0" prefix otherwise the value will be treated as a base 10 integer!

Default: 077

chown_upload_mode

The file mode to force for chown()ed anonymous uploads. (Added in v2.0.6).

Default: 0600

connect_timeout

The timeout, in seconds, for a remote client to respond to our PORT style data connection.

Default: 60

data_connection_timeout

The timeout, in seconds, which is roughly the maximum time we permit data transfers to stall for with no progress. If the timeout triggers, the remote client is kicked off.

Default: 300

delay_failed_login

The number of seconds to pause prior to reporting a failed login.

Default: 1

delay_successful_login

The number of seconds to pause prior to allowing a successful login.

Default: 0

file_open_mode

The permissions with which uploaded files are created. Umasks are applied on top of this value. You may wish to change to 0777 if you want uploaded files to be executable.

Default: 0666

ftp_data_port

The port from which PORT style connections originate (as long as the poorly named **connect_from_port_20** is enabled).

Default: 20

idle_session_timeout

The timeout, in seconds, which is the maximum time a remote client may spend between FTP commands. If the timeout triggers, the remote client is kicked off.

Default: 300

listen_port

If vsftpd is in standalone mode, this is the port it will listen on for incoming FTP connections.

Default: 21

local_max_rate

The maximum data transfer rate permitted, in bytes per second, for local authenticated users.

Default: 0 (unlimited)

local_umask

The value that the umask for file creation is set to for local users. NOTE! If you want to specify octal values, remember the "0" prefix otherwise the value will be treated as a base 10 integer!

Default: 077

max_clients

If vsftpd is in standalone mode, this is the maximum number of clients which may be connected. Any additional clients connecting will get an error message.

Default: 0 (unlimited)

max_login_fails

After this many login failures, the session is killed.

Default: 3

max_per_ip

If vsftpd is in standalone mode, this is the maximum number of clients which may be connected from the same source internet address. A client will get an error message if they go over this limit.

Default: 0 (unlimited)

pasv_max_port

The maximum port to allocate for PASV style data connections. Can be used to specify a narrow port range to assist firewalling.

Default: 0 (use any port)

pasv_min_port

The minimum port to allocate for PASV style data connections. Can be used to specify a narrow port range to assist firewalling.

Default: 0 (use any port)

trans_chunk_size

You probably don't want to change this, but try setting it to something like 8192 for a much smoother bandwidth limiter.

Default: 0 (let vsftpd pick a sensible setting)

STRING OPTIONS

Below is a list of string options.

anon_root

This option represents a directory which vsftpd will try to change into after an anonymous login. Failure is silently ignored.

Default: (none)

banned_email_file

This option is the name of a file containing a list of anonymous e-mail passwords which are not permitted. This file is consulted if the option **deny_email_enable** is enabled.

Default: /etc/vsftpd.banned_emails

banner_file

This option is the name of a file containing text to display when someone connects to the server. If set, it overrides the banner string provided by the **ftpd_banner** option.

Default: (none)

ca_certs_file

This option is the name of a file to load Certificate Authority certs from, for the purpose of validating client certs. Regrettably, the default SSL CA cert paths are not used, because of vsftpd's use of restricted filesystem spaces (chroot). (Added in v2.0.6).

Default: (none)

chown_username

This is the name of the user who is given ownership of anonymously uploaded files. This option is only relevant if another option, **chown_uploads**, is set.

Default: root

chroot_list_file

The option is the name of a file containing a list of local users which will be placed in a chroot() jail in their home directory. This option is only relevant if the option **chroot_list_enable** is enabled. If the option **chroot_local_user** is enabled, then the list file becomes a list of users to NOT place in a chroot() jail.

Default: /etc/vsftpd.chroot_list

cmds_allowed

This options specifies a comma separated list of allowed FTP commands (post login. USER, PASS and QUIT and others are always allowed pre-login). Other commands are rejected. This is a powerful method of really locking down an FTP server. Example: cmds_allowed=PASV,RETR,QUIT

Default: (none)

cmds_denied

This options specifies a comma separated list of denied FTP commands (post login. USER, PASS, QUIT and others are always allowed pre-login). If a command appears on both this and **cmds_allowed** then the denial takes precedence. (Added in v2.1.0).

Default: (none)

deny_file

This option can be used to set a pattern for filenames (and directory names etc.) which should not be accessible in any way. The affected items are not hidden, but any attempt to do anything to them (download, change into directory, affect something within directory etc.) will be denied. This option is very simple, and should not be used for serious access control - the filesystem's permissions should be used in preference. However, this option may be useful in certain virtual user setups. In particular aware that if a filename is accessible by a variety of names (perhaps due to symbolic links or hard links), then care must be taken to deny access to all the names. Access will be denied to items if their name contains the string given by hide_file, or if they match the regular expression specified by hide_file. Note that vsftpd's regular expression matching code is a simple implementation which is a subset of full regular expression functionality. Because of this, you will need to carefully and exhaustively test any application of this option. And you are recommended to use filesystem permissions for any important security policies due to their greater reliability. Supported regex syntax is any number of *, ? and unnested {,} operators. Regex matching is only supported on the last component of a path, e.g. a/b/? is supported but a/?/c is not. Example: deny_file={*.mp3,*.mov,.private}

Default: (none)

dsa_cert_file

This option specifies the location of the DSA certificate to use for SSL encrypted connections.

Default: (none - an RSA certificate suffices)

dsa_private_key_file

This option specifies the location of the DSA private key to use for SSL encrypted connections. If this option is not set, the private key is expected to be in the same file as the certificate.

Default: (none)

email_password_file

This option can be used to provide an alternate file for usage by the **secure_email_list_enable** setting.

Default: /etc/vsftpd.email_passwords

ftp_username

This is the name of the user we use for handling anonymous FTP. The home directory of this user is the root of the anonymous FTP area.

Default: ftp

ftpd_banner

This string option allows you to override the greeting banner displayed by vsftpd when a connection first comes in.

Default: (none - default vsftpd banner is displayed)

guest_username

See the boolean setting **guest_enable** for a description of what constitutes a guest login. This setting is the real username which guest users are mapped to.

Default: ftp

hide_file

This option can be used to set a pattern for filenames (and directory names etc.) which should be hidden from directory listings. Despite being hidden, the files / directories etc. are fully accessible to clients who know what names to actually use. Items will be hidden if their names contain the string given by hide_file, or if they match the regular expression specified by hide_file. Note that vsftpd's regular expression matching code is a simple implementation which is a subset of full regular expression functionality. See **deny_file** for details of exactly what regex syntax is supported. Example: hide_file={*.mp3,.hidden,hide*,h?}

Default: (none)

listen_address

If vsftpd is in standalone mode, the default listen address (of all local interfaces) may be overridden by this setting. Provide a numeric IP address.

Default: (none)

listen_address6

Like listen_address, but specifies a default listen address for the IPv6 listener (which is used if listen_ipv6 is set). Format is standard IPv6 address format.

Default: (none)

local_root

This option represents a directory which vsftpd will try to change into after a local (i.e. non-anonymous) login. Failure is silently ignored.

Default: (none)

message_file

This option is the name of the file we look for when a new directory is entered. The contents are displayed to the remote user. This option is only relevant if the option **dirmessage_enable** is enabled.

Default: .message

nopriv_user

This is the name of the user that is used by vsftpd when it wants to be totally unprivileged. Note that this should be a dedicated user, rather than nobody. The user nobody tends to be used for rather a lot of important things on most machines.

Default: nobody

pam_service_name

This string is the name of the PAM service vsftpd will use.

Default: ftp

pasv_address

Use this option to override the IP address that vsftpd will advertise in response to the PASV command. Provide a numeric IP address, unless **pasv_addr_resolve** is enabled, in which case you can provide a hostname which will be DNS resolved for you at startup.

Default: (none - the address is taken from the incoming connected socket)

rsa_cert_file

This option specifies the location of the RSA certificate to use for SSL encrypted connections.

Default: /usr/share/ssl/certs/vsftpd.pem

rsa_private_key_file

This option specifies the location of the RSA private key to use for SSL encrypted connections. If this option is not set, the private key is expected to be in the same file as the certificate.

Default: (none)

secure_chroot_dir

This option should be the name of a directory which is empty. Also, the directory should not be writable by the ftp user. This directory is used as a secure chroot() jail at times vsftpd does not require filesystem access.

Default: /usr/share/empty

ssl_ciphers

This option can be used to select which SSL ciphers vsftpd will allow for encrypted SSL connections. See the **ciphers** man page for further details. Note that restricting ciphers can be a useful security precaution as it prevents malicious remote parties forcing a cipher which they have found problems with.

Default: DES-CBC3-SHA

user_config_dir

This powerful option allows the override of any config option specified in the manual page, on a per-user basis. Usage is simple, and is best illustrated with an example. If you set **user_config_dir** to be **/etc/vsftpd_user_conf** and then log on as the user "chris", then vsftpd will apply the settings in the file **/etc/vsftpd_user_conf/chris** for the duration of the session. The format of this file is as detailed in this manual page! PLEASE NOTE that not all settings are effective on a per-user basis. For example, many settings only prior to the user's session being

started. Examples of settings which will not affect any behviour on a per-user basis include listen_address, banner_file, max_per_ip, max_clients, xferlog_file, etc.

Default: (none)

user_sub_token

This option is useful is conjunction with virtual users. It is used to automatically generate a home directory for each virtual user, based on a template. For example, if the home directory of the real user specified via **guest_username** is **/home/virtual/$USER**, and **user_sub_token** is set to **$USER**, then when virtual user fred logs in, he will end up (usually chroot()'ed) in the directory **/home/virtual/fred**. This option also takes affect if **local_root** contains **user_sub_token**.

Default: (none)

userlist_file

This option is the name of the file loaded when the **userlist_enable** option is active.

Default: /etc/vsftpd.user_list

vsftpd_log_file

This option is the name of the file to which we write the vsftpd style log file. This log is only written if the option **xferlog_enable** is set, and **xferlog_std_format** is NOT set. Alternatively, it is written if you have set the option **dual_log_enable**. One further complication - if you have set **syslog_enable**, then this file is not written and output is sent to the system log instead.

Default: /var/log/vsftpd.log

xferlog_file

This option is the name of the file to which we write the wu-ftpd style transfer log. The transfer log is only written if the option **xferlog_enable** is set, along with **xferlog_std_format**. Alternatively, it is written if you have set the option **dual_log_enable**.

Default: /var/log/xferlog