Installing vsftpd

While there are a variety of FTP server tools available for Linux, one of the most popular and mature options is vsftpd.

Begin by SSHing into your server as root and use the apt-get command to install *vsftpd*:

```
apt-get update

apt-get install vsftpd
Reading package lists... Done
Building dependency tree
Reading state information... Done
[...]
The following NEW packages will be installed:
  vsftpd
0 upgraded, 1 newly installed, 0 to remove and 18 not upgraded.
Need to get 111 kB of archives.
After this operation, 361 kB of additional disk space will be used.
Get:1 http://mirrors.digitalocean.com/ubuntu/ trusty-updates/main vsftpd amd64
3.0.2-1ubuntu2.14.04.1 [111 kB]
Fetched 111 kB in 0s (231 kB/s)
Preconfiguring packages ...
Selecting previously unselected package vsftpd.
(Reading database ... 175600 files and directories currently installed.)
Preparing to unpack .../vsftpd_3.0.2-1ubuntu2.14.04.1_amd64.deb ...
Unpacking vsftpd (3.0.2-1ubuntu2.14.04.1) ...
Processing triggers for man-db (2.6.7.1-1) ...
Processing triggers for ureadahead (0.100.0-16) ...
Setting up vsftpd (3.0.2-1ubuntu2.14.04.1) ...
vsftpd start/running, process 18690
Processing triggers for ureadahead (0.100.0-16) ...
```

Configuration

The next step is to change any configuration settings for vsftpd. Open the /etc/vsftpd.conf file in your preferred text editor:

```
nano /etc/vsftpd.conf
```

Edit the file so it resembles the following:

```
# Example config file /etc/vsftpd.conf
# ...
# Run standalone?  vsftpd can run either from an inetd or as a standalone
# daemon started from an initscript.
listen=YES
#
# Allow anonymous FTP? (Disabled by default)
anonymous_enable=NO
#
# Uncomment this to allow local users to log in.
local_enable=YES
#
# Uncomment this to enable any form of FTP write command.
#write_enable=YES
# You may restrict local users to their home directories. See the FAQ for
# the possible risks in this before using chroot_local_user or
# chroot_list_enable below.
```

```
#chroot_local_user=YES
```

The critical settings seen above are outlined below:

- `listen=YES` tells vsftpd to run as a standalone daemon (the simplest method for getting up and running). anonymous_enable=NO disallows anonymous FTP users, which is generally preferred for security reasons but can be enabled for testing purposes.
- `local_enable=YES` allows any user account defined in the /etc/passwd file access to the FTP server and is generally how most FTP users will connect.
- `write_enable=YES` is commented out by default, but removing the hash (#) allows files to be uploaded to the FTP server. chroot_local_user=YES restricts users to their home directory and is also commented out by default.

To begin your testing and make sure everything is working, start with the following settings for the above parameters:

```
listen=YES
anonymous_enable=YES
local_enable=YES
write_enable=YES
chroot_local_user=YES
```

Save the `vsftpd.conf` file then restart the vsftpd service for the changes to take effect:

```
sudo service vsftpd restart
vsftpd stop/waiting
vsftpd start/running, process 18954
```

Testing Your FTP Server

To quickly determine if your server was installed properly and is up and running, try to connect to the FTP server from your active shell, using the name anonymous and a blank password:

```
ftp localhost
Connected to localhost.
220 (vsFTPd 3.0.2)
Name (localhost:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
```

With both `anonymous_enable` and `local_enable` set to "YES" in the configuration, you should be able to successfully login to your local FTP server as seen above!

With that out of the way, simply enter quit at the ftp> prompt to cancel out:

```
quit
221 Goodbye.
```

With the test complete, you may wish to disable anonymous access once again by setting `anonymous_enable=NO` in the `/etc/vsftpd.conf` file and restarting the service:

```
nano /etc/vsftpd.conf
```

Edit the file to resemble this:

```
# Set to NO to disable anonymous access
anonymous_enable=NO
sudo service vsftpd restart
vsftpd stop/waiting
vsftpd start/running, process 18996
```

Adding an FTP User

If this is a new server it may be advisable to add a specific user for FTP access. Doing so is a fairly simple process but begin by creating a new user:

```
sudo adduser foobar
Adding user `foobar' ...
Adding new group `foobar' (1000) ...
Adding new user `foobar' (1000) with group `foobar' ...
Creating home directory `/home/foobar' ...
Copying files from `/etc/skel' ...

Enter new UNIX password:

Retype new UNIX password:
passwd: password updated successfully
Changing the user information for foobar

Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n]

Y
```

With a new user added you can now connect to your server remotely with an FTP client such as FileZilla, but you will immediately run into an error:

```
Status:     Connecting to 104.131.170.253:21...
Status:     Connection established, waiting for welcome message...
Response:     220 (vsFTPd 3.0.2)
Command:     USER foobar
Response:     331 Please specify the password.
Command:     PASS ****************
Response:     500 OOPS: vsftpd: refusing to run with writable root inside
chroot()
```

The "500 OOPS" error vsftpd returns is a security measure designed to prevent *writable* root access for FTP users by default. To resolve this issue there are two main options available.

Allowing Writable User-root Access

The simplest method is to alter the `/etc/vsftpd.conf` file once again and enable one particular setting:

```
nano /etc/vsftpd.conf
```

Edit the file so it resembles the following:

```
# Allow users to write to their root directory
allow_writeable_chroot=YES
```

With allow_writeable_chroot enabled following a service vsftpd restart, you can now successfully FTP into your server remotely as your newly created user:

```
Status:    Connecting to 104.131.170.253:21...
Status:    Connection established, waiting for welcome message...
Response:    220 (vsFTPd 3.0.2)
Command:    USER foobar
Response:    331 Please specify the password.
Command:    PASS ****************
Response:    230 Login successful.
```

Using Writeable Subdirectories

The other option to maintain slightly stronger security is not to enable allow_writeable_chroot as outlined above, but instead to create a new subdirectory in the user's root directory with write access:

```
sudo chown root:root /home/foobar

sudo mkdir /home/foobar/uploads

sudo chown foobar:foobar /home/foobar/uploads

sudo service vsftpd restart
```

Now when you connect remotely to your FTP server as the new user, that user will not have write access to the root directory, but will instead have full write access to upload files into the newly created uploads directory instead.

Securing Your FTP With SSL

While standard unencrypted FTP access as outlined so far is sufficient in many cases, when transferring sensitive information over FTP it is useful to utilize a more secure connection using SSL.

To begin you'll likely need to generate a new SSL certificate with the following command, following the prompts as appropriate to complete the process:

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout
/etc/ssl/private/vsftpd.pem -out /etc/ssl/private/vsftpd.pem
```

Now you must ensure that vsftpd is aware of the SSL certificate. Open the `/etc/vsftpd.conf` file once again:

```
sudo nano /etc/vsftpd.conf
```

Look near the bottom of the file for two rsa_ settings like this, indicating the location of the SSL certificate that was just created:

```
rsa_cert_file=/etc/ssl/private/vsftpd.pem
rsa_private_key_file=/etc/ssl/private/vsftpd.pem
```

If those lines don't exist or match the appropriate path to the SSL certificate created, update them accordingly.

Additionally, there are a number of configuration settings to handle SSL connections, particularly forcing use of the TLS protocol which is ideal:

```
ssl_enable=YES
allow_anon_ssl=NO
force_local_data_ssl=YES
force_local_logins_ssl=YES
ssl_tlsv1=YES
ssl_sslv2=NO
ssl_sslv3=NO
require_ssl_reuse=NO
ssl_ciphers=HIGH
```

Some of the settings are self-explanatory, but the key components are the overall enabling of SSL, the restriction to use only TLS, and disallowing anonymous access.

With the settings added and the file saved, once again restart the vsftpd service:

```
sudo service vsftpd restart
```

Now your FTP server is ready to accept secure connections using "FTP over TLS" encryption. Using a client such as FileZilla, you will be presented with a certificate popup asking to verify the newly created SSL certification.

Upon accepting you will now be securely connected and transfers will be encrypted via SSL:

```
Status:    Connecting to 104.131.170.253:21...
Status:    Connection established, waiting for welcome message...
Response:    220 (vsFTPd 3.0.2)
Command:    AUTH TLS
Response:    234 Proceed with negotiation.
Status:    Initializing TLS...
Status:    Verifying certificate...
Command:    USER foobar
Status:    TLS/SSL connection established.
Response:    331 Please specify the password.
Command:    PASS ****************
Response:    230 Login successful.
```

To allow users with a shell of **/usr/sbin/nologin** access to FTP, but have no shell access, edit **/etc/shells** adding the nologin shell:

```
# /etc/shells: valid login shells
/bin/csh
/bin/sh
/usr/bin/es
/usr/bin/ksh
/bin/ksh
/usr/bin/rc
/usr/bin/tcsh
```

```
/bin/tcsh
/usr/bin/esh
/bin/dash
/bin/bash
/bin/rbash
/usr/bin/screen
/usr/sbin/nologin
```

This is necessary because, by default vsftpd uses PAM for authentication, and the /etc/pam.d/vsftpd configuration file contains:

```
auth    required        pam_shells.so
```

The shells PAM module restricts access to shells listed in the /etc/shells file.