Prerequisites

Step 1 — Installing vsftpd

Step 2 — Opening the Firewall

Step 3 — Preparing the User Directory

Step 4 — Configuring FTP Access

Step 5 — Testing FTP Access

Step 6 — Securing Transactions

Step 7 — Testing TLS with FileZilla

Step 8 — Disabling Shell Access (Optional)

Conclusion

**Mark as Complete**

[×](javascript:;)

Sign up for our newsletter.

Get the latest tutorials on SysAdmin and open source topics.

Sign Up

[**Sign Up**](https://www.digitalocean.com/community/auth/digitalocean?display=sessionless+register)

[**Log In**](https://www.digitalocean.com/community/auth/digitalocean)

[**Tutorials**](https://www.digitalocean.com/community/tutorials)

[**Questions**](https://www.digitalocean.com/community/questions)

[**Projects**](https://www.digitalocean.com/community/projects)

[**Meetups**](https://www.meetup.com/pro/digitalocean/?utm_source=do_community)

[**Main Site**](https://www.digitalocean.com/)

[DigitalOcean DigitalOcean](https://www.digitalocean.com/)

**Community**

[**Tutorials**](https://www.digitalocean.com/community/tutorials)

[Questions](https://www.digitalocean.com/community/questions)

[Projects](https://www.digitalocean.com/community/projects)

[Meetups](https://www.meetup.com/pro/digitalocean/?utm_source=do_community)

[**Sign Up**](https://www.digitalocean.com/community/auth/digitalocean?display=sessionless+register)

[**Log In**](https://www.digitalocean.com/community/auth/digitalocean)

[View All Results](https://www.digitalocean.com/community/search)

[](https://www.digitalocean.com/community/users/melissaanderson)

[Melissa Anderson](https://www.digitalocean.com/community/users/melissaanderson)

Subscribe

Share

45

How To Set Up vsftpd for a User's Directory on Ubuntu 16.04

Updated February 20, 2018 378.8k

views

[Linux Basics](https://www.digitalocean.com/community/tags/linux-basics?type=tutorials) [Security](https://www.digitalocean.com/community/tags/security?type=tutorials) [Ubuntu](https://www.digitalocean.com/community/tags/ubuntu?type=tutorials) [Ubuntu 16.04](https://www.digitalocean.com/community/tags/ubuntu-16-04?type=tutorials)

**Introduction**

FTP, short for File Transfer Protocol, is a network protocol that was once widely used for moving files between a client and server. It has since been replaced by faster, more secure, and more convenient ways of delivering files. Many casual Internet users expect to download directly from their web browser with https , and command-line users are more likely to use secure protocols such as the scp or [sFTP](https://www.digitalocean.com/community/tutorials/how-to-use-sftp-to-securely-transfer-files-with-a-remote-server) .

FTP is still used to support legacy applications and workflows with very specific needs. If you have a choice of what protocol to use, consider exploring the more modern options. When you do need FTP, however, vsftpd is an excellent choice. Optimized for security, performance, and stability, vsftpd offers strong protection against many security problems found in other FTP servers and is the default for many Linux distributions.

In this tutorial, we'll show you how to configure vsftpd to allow a user to upload files to his or her home directory using FTP with login credentials secured by SSL/TLS.

Prerequisites

To follow along with this tutorial you will need:

**An Ubuntu 16.04 server with a non-root user with sudo privileges** : You can learn more about how to set up a user with these privileges in our [Initial Server Setup with Ubuntu 16.04](https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-16-04) guide.

Once you have an Ubuntu server in place, you're ready to begin.

Step 1 — Installing vsftpd

We'll start by updating our package list and installing the vsftpd daemon:

sudo apt-get update

sudo apt-get install vsftpd

When the installation is complete, we'll copy the configuration file so we can start with a blank configuration, saving the original as a backup.

sudo cp /etc/vsftpd.conf /etc/vsftpd.conf.orig

With a backup of the configuration in place, we're ready to configure the firewall.

Step 2 — Opening the Firewall

We'll check the firewall status to see if it’s enabled. If so, we’ll ensure that FTP traffic is permitted so you won’t run into firewall rules blocking you when it comes time to test.

sudo ufw status

In this case, only SSH is allowed through:

Output

Status: active To Action From -- ------ ---- OpenSSH ALLOW Anywhere OpenSSH (v6) ALLOW Anywhere (v6)

You may have other rules in place or no firewall rules at all. Since only ssh traffic is permitted in this case, we’ll need to add rules for FTP traffic.

We'll need to open ports 20 and 21 for FTP, port 990 for later when we enable TLS, and ports 40000-50000 for the range of passive ports we plan to set in the configuration file:

sudo ufw allow 20/tcp

sudo ufw allow 21/tcp

sudo ufw allow 990/tcp

sudo ufw allow 40000:50000/tcp

sudo ufw status

Now our firewall rules looks like:

Output

Status: active To Action From -- ------ ---- OpenSSH ALLOW Anywhere 990/tcp ALLOW Anywhere 20/tcp ALLOW Anywhere 21/tcp ALLOW Anywhere 40000:50000/tcp ALLOW Anywhere OpenSSH (v6) ALLOW Anywhere (v6) 20/tcp (v6) ALLOW Anywhere (v6) 21/tcp (v6) ALLOW Anywhere (v6) 990/tcp (v6) ALLOW Anywhere (v6) 40000:50000/tcp (v6) ALLOW Anywhere (v6)

With vsftpd installed and the necessary ports open, we're ready to proceed to the next step.

Step 3 — Preparing the User Directory

For this tutorial, we're going to create a user, but you may already have a user in need of FTP access. We'll take care to preserve an existing user’s access to their data in the instructions that follow. Even so, we recommend you start with a new user until you've configured and tested your setup.

First, we’ll add a test user:

sudo adduser sammy

Assign a password when prompted and feel free to press "ENTER" through the other prompts.

FTP is generally more secure when users are restricted to a specific directory. vsftpd accomplishes this with chroot jails. When chroot is enabled for local users, they are restricted to their home directory by default. However, because of the way vsftpd secures the directory, it must not be writable by the user. This is fine for a new user who should only connect via FTP, but an existing user may need to write to their home folder if they also shell access.

In this example, rather than removing write privileges from the home directory, we're will create an ftp directory to serve as the chroot and a writable files directory to hold the actual files.

Create the ftp folder, set its ownership, and be sure to remove write permissions with the following commands:

sudo mkdir /home/ sammy /ftp

sudo chown nobody:nogroup /home/ sammy /ftp

sudo chmod a-w /home/ sammy /ftp

Let's verify the permissions:

sudo ls -la /home/sammy/ftp

Output

total 8 4 dr-xr-xr-x 2 nobody nogroup 4096 Aug 24 21:29 . 4 drwxr-xr-x 3 sammy sammy 4096 Aug 24 21:29 ..

Next, we'll create the directory where files can be uploaded and assign ownership to the user:

sudo mkdir /home/ sammy /ftp/files

sudo chown sammy : sammy /home/ sammy /ftp/files

A permissions check on the files directory should return the following:

sudo ls -la /home/sammy/ftp

Output

total 12 dr-xr-xr-x 3 nobody nogroup 4096 Aug 26 14:01 . drwxr-xr-x 3 sammy sammy 4096 Aug 26 13:59 .. drwxr-xr-x 2 sammy sammy 4096 Aug 26 14:01 files

Finally, we'll add a test.txt file to use when we test later on:

echo "vsftpd test file" | sudo tee /home/ sammy /ftp/files/test.txt

Now that we've secured the ftp directory and allowed the user access to the files directory, we'll turn our attention to configuration.

Step 4 — Configuring FTP Access

We're planning to allow a single user with a local shell account to connect with FTP. The two key settings for this are already set in vsftpd.conf . Start by opening the config file to verify that the settings in your configuration match those below:

sudo nano /etc/vsftpd.conf

/etc/vsftpd.conf

. . . # Allow anonymous FTP? (Disabled by default). anonymous\_enable=NO # # Uncomment this to allow local users to log in. local\_enable=YES . . .

Next we'll need to change some values in the file. In order to allow the user to upload files, we’ll uncomment the write\_enable setting so that we have:

/etc/vsftpd.conf

. . . write\_enable= YES . . .

We’ll also uncomment the chroot to prevent the FTP-connected user from accessing any files or commands outside the directory tree.

/etc/vsftpd.conf

. . . chroot\_local\_user= YES . . .

We’ll add a user\_sub\_token in order to insert the username in our local\_root directory path so our configuration will work for this user and any future users that might be added.

/etc/vsftpd.conf

user\_sub\_token=$USER local\_root=/home/$USER/ftp

We'll limit the range of ports that can be used for passive FTP to make sure enough connections are available:

/etc/vsftpd.conf

pasv\_min\_port=40000 pasv\_max\_port=50000

**Note:** We pre-opened the ports that we set here for the passive port range. If you change the values, be sure to update your firewall settings.

We will also add a directive telling vsftpd to listen on a particular port for incoming FTP connections:

/etc/vsftpd.conf

listen\_port=45000

Since we’re only planning to allow FTP access on a case-by-case basis, we’ll set up the configuration so that access is given to a user only when they are explicitly added to a list rather than by default:

/etc/vsftpd.conf

userlist\_enable=YES userlist\_file=/etc/vsftpd.userlist userlist\_deny=NO

userlist\_deny toggles the logic. When it is set to "YES", users on the list are denied FTP access. When it is set to "NO", only users on the list are allowed access. When you're done making the change, save and exit the file.

Finally, we’ll create and add our user to the file. We'll use the -a flag to append to file:

echo " sammy " | sudo tee -a /etc/vsftpd.userlist

Double-check that it was added as you expected:

cat /etc/vsftpd.userlist

Output

sammy

Restart the daemon to load the configuration changes:

sudo systemctl restart vsftpd

Now we're ready for testing.

Step 5 — Testing FTP Access

We've configured the server to allow only the user sammy to connect via FTP. Let's make sure that's the case.

**Anonymous users should fail to connect** : We disabled anonymous access. Here we'll test that by trying to connect anonymously. If we've done it properly, anonymous users should be denied permission:

ftp -p 203.0.113.0

Output

Connected to 203.0.113.0. 220 (vsFTPd 3.0.3) Name (203.0.113.0:default): anonymous 530 Permission denied. ftp: Login failed. ftp>

Close the connection:

bye

**Users other than sammy should fail to connect** : Next, we'll try connecting as our sudo user. They, too, should be denied access, and it should happen before they're allowed to enter their password.

ftp -p 203.0.113.0

Output

Connected to 203.0.113.0. 220 (vsFTPd 3.0.3) Name (203.0.113.0:default): sudo\_user 530 Permission denied. ftp: Login failed. ftp>

Close the connection:

bye

**sammy should be able to connect, as well as read and write files** : Here, we'll make sure that our designated user *can* connect:

ftp -p 203.0.113.0

Output

Connected to 203.0.113.0. 220 (vsFTPd 3.0.3) Name (203.0.113.0:default): sammy 331 Please specify the password. Password: your\_user's\_password 230 Login successful. Remote system type is UNIX. Using binary mode to transfer files. ftp>

We'll change into the files directory, then use the get command to transfer the test file we created earlier to our local machine:

cd files

get test.txt

Output

227 Entering Passive Mode (203,0,113,0,169,12). 150 Opening BINARY mode data connection for test.txt (16 bytes). 226 Transfer complete. 16 bytes received in 0.0101 seconds (1588 bytes/s) ftp>

We'll turn right back around and try to upload the file with a new name to test write permissions:

put test.txt upload.txt

Output

227 Entering Passive Mode (203,0,113,0,164,71). 150 Ok to send data. 226 Transfer complete. 16 bytes sent in 0.000894 seconds (17897 bytes/s)

Close the connection:

bye

Now that we've tested our configuration, we'll take steps to further secure our server.

Step 6 — Securing Transactions

Since FTP does *not* encrypt any data in transit, including user credentials, we'll enable TTL/SSL to provide that encryption. The first step is to create the SSL certificates for use with vsftpd.

We'll use openssl to create a new certificate and use the -days flag to make it valid for one year. In the same command, we'll add a private 2048-bit RSA key. Then by setting both the -keyout and -out flags to the same value, the private key and the certificate will be located in the same file.

We'll do this with the following command:

sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/vsftpd.pem -out /etc/ssl/private/vsftpd.pem

You'll be prompted to provide address information for your certificate. Substitute your own information for the questions below:

Output

Generating a 2048 bit RSA private key ............................................................................+++ ...........+++ writing new private key to '/etc/ssl/private/vsftpd.pem' ----- You are about to be asked to enter information that will be incorporated into your certificate request. What you are about to enter is what is called a Distinguished Name or a DN. There are quite a few fields but you can leave some blank For some fields there will be a default value, If you enter '.', the field will be left blank. ----- Country Name (2 letter code) [AU]: US State or Province Name (full name) [Some-State]: NY Locality Name (eg, city) []: New York City Organization Name (eg, company) [Internet Widgits Pty Ltd]: DigitalOcean Organizational Unit Name (eg, section) []: Common Name (e.g. server FQDN or YOUR name) []: your\_IP\_address Email Address []:

For more detailed information about the certificate flags, see [OpenSSL Essentials: Working with SSL Certificates, Private Keys and CSRs](https://www.digitalocean.com/community/tutorials/openssl-essentials-working-with-ssl-certificates-private-keys-and-csrs)

Once you've created the certificates, open the vsftpd configuration file again:

sudo nano /etc/vsftpd.conf

Toward the bottom of the file, you should two lines that begin with rsa\_ . Comment them out so they look like:

/etc/vsftpd.conf

# rsa\_cert\_file=/etc/ssl/certs/ssl-cert-snakeoil.pem # rsa\_private\_key\_file=/etc/ssl/private/ssl-cert-snakeoil.key

Below them, add the following lines which point to the certificate and private key we just created:

/etc/vsftpd.conf

rsa\_cert\_file=/etc/ssl/private/vsftpd.pem rsa\_private\_key\_file=/etc/ssl/private/vsftpd.pem

After that, we will force the use of SSL, which will prevent clients that can't deal with TLS from connecting. This is necessary in order to ensure all traffic is encrypted but may force your FTP user to change clients. Change ssl\_enable to YES :

/etc/vsftpd.conf

ssl\_enable= YES

After that, add the following lines to explicitly deny anonymous connections over SSL and to require SSL for both data transfer and logins:

/etc/vsftpd.conf

allow\_anon\_ssl=NO force\_local\_data\_ssl=YES force\_local\_logins\_ssl=YES

After this we'll configure the server to use TLS, the preferred successor to SSL by adding the following lines:

/etc/vsftpd.conf

ssl\_tlsv1=YES ssl\_sslv2=NO ssl\_sslv3=NO

Finally, we will add two more options. First, we will not require SSL reuse because it can break many FTP clients. We will require "high" encryption cipher suites, which currently means key lengths equal to or greater than 128 bits:

/etc/vsftpd.conf

require\_ssl\_reuse=NO ssl\_ciphers=HIGH

When you're done, save and close the file.

Now, we need to restart the server for the changes to take effect:

sudo systemctl restart vsftpd

At this point, we will no longer be able to connect with an insecure command-line client. If we tried, we'd see something like:

ftp -p 203.0.113.0

Connected to 203.0.113.0.

220 (vsFTPd 3.0.3)

Name (203.0.113.0:default): sammy

530 Non-anonymous sessions must use encryption.

ftp: Login failed.

421 Service not available, remote server has closed connection

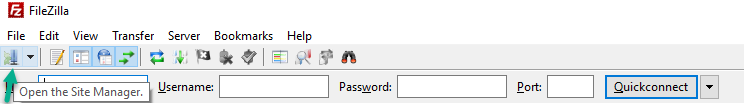
ftp>

Next, we'll verify that we can connect using a client that supports TLS.

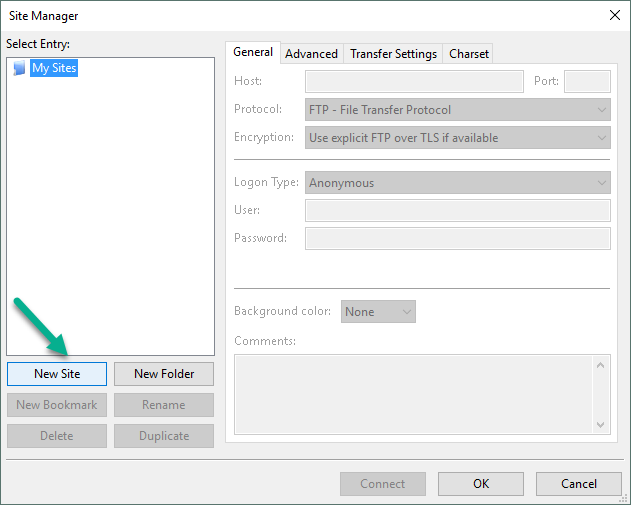
Step 7 — Testing TLS with FileZilla

Most modern FTP clients can be configured to use TLS encryption. We will demonstrate how to connect using FileZilla because of its cross platform support. Consult the documentation for other clients.

When you first open FileZilla, find the Site Manager icon just below the word File, the left-most icon on the top row. Click it:

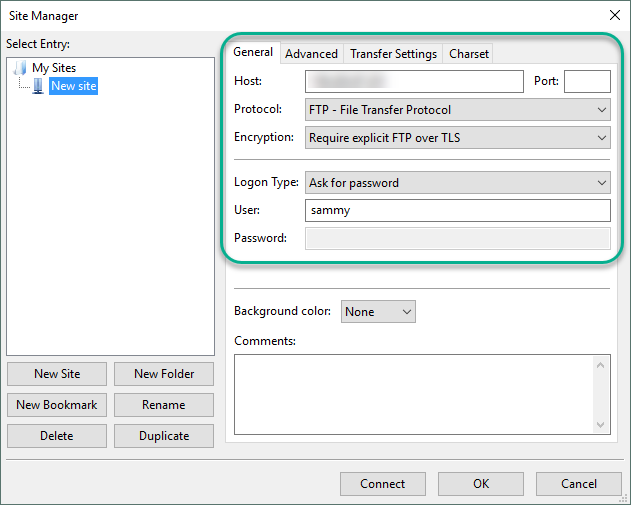


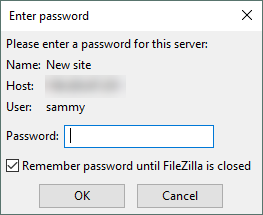
A new window will open. Click the "New Site" button in the bottom right corner:

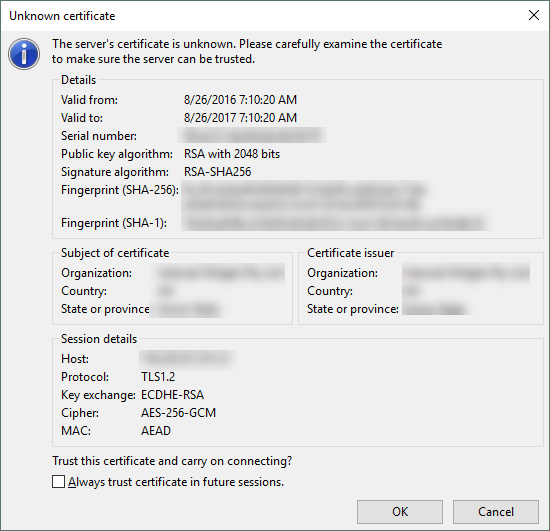
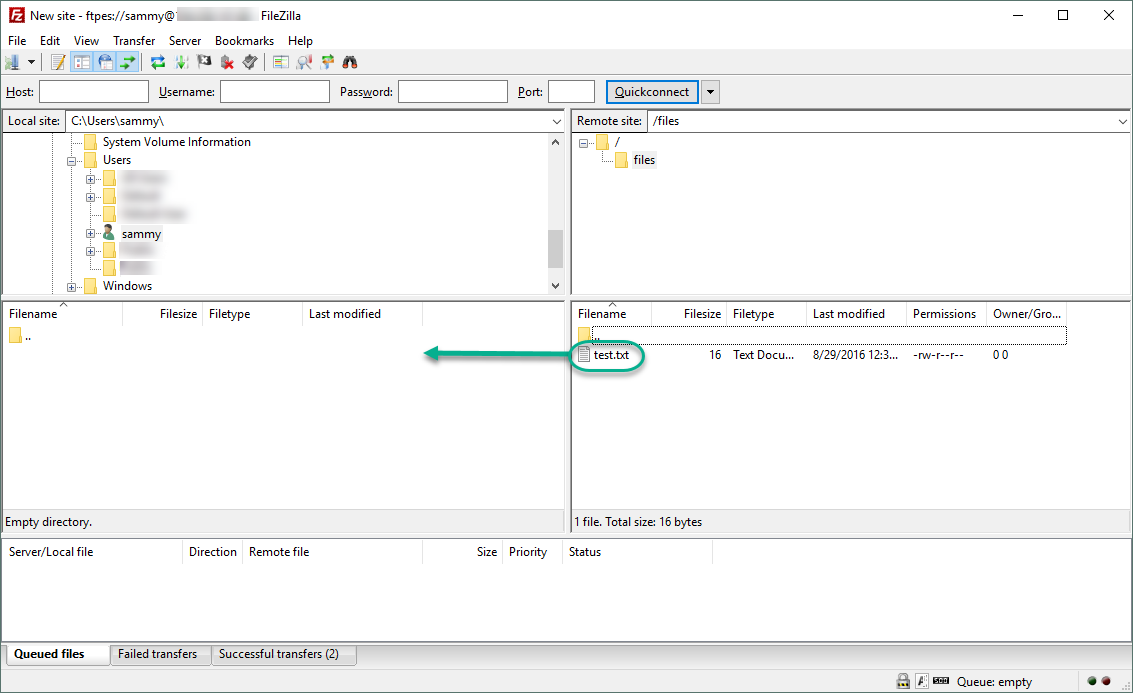
  
Under "My Sites" a new icon with the words "New site" will appear. You can name it now or return later and use the Rename button.

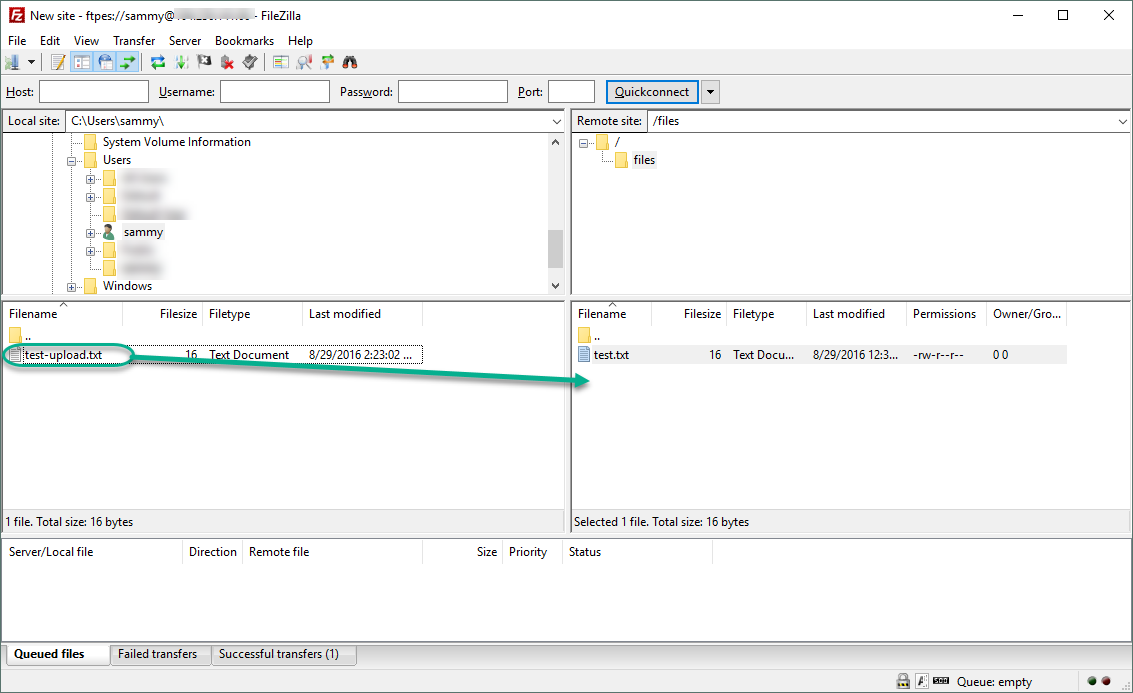
You must fill out the "Host" field with the name or IP address. Under the "Encryption" drop down menu, select "Require explicit FTP over TLS". You also want to specify that Filezilla should use port 45000 by filling out the “Port” field.

For "Logon Type", select "Ask for password". Fill in the FTP user you created in the "User" field:

  
Click "Connect" at the bottom of the interface. You will be asked for the user's password:

  
Click "OK" to connect. You should now be connected with your server with TLS/SSL encryption.

  
When you’ve accepted the certificate, double-click the files folder and drag upload.txt to the left to confirm that you’re able to download files.   
  
  
When you’ve done that, right-click on the local copy, rename it to upload-tls.txt` and drag it back to the server to confirm that you can upload files.

  
You’ve now confirmed that you can securely and successfully transfer files with SSL/TLS enabled.

Step 8 — Disabling Shell Access (Optional)

If you're unable to use TLS because of client requirements, you can gain some security by disabling the FTP user's ability to log in any other way. One relatively straightforward way to prevent it is by creating a custom shell. This will not provide any encryption, but it will limit the access of a compromised account to files accessible by FTP.

First, open a file called ftponly in the bin directory:

sudo nano /bin/ftponly

We'll add a message telling the user why they are unable to log in. Paste in the following:

#!/bin/sh echo "This account is limited to FTP access only."

Change the permissions to make the file executable:

sudo chmod a+x /bin/ftponly

Open the list of valid shells:

sudo nano /etc/shells

At the bottom, add:

/etc/shells

. . . /bin/ftponly

Update the user's shell with the following command:

sudo usermod sammy -s /bin/ftponly

Now try logging in as sammy:

ssh sammy@ 203.0.113.0

You should see something like:

Output

This account is limited to FTP access only. Connection to 203.0.113.0 closed.

This confirms that the user can no longer ssh to the server and is limited to FTP access only.

Conclusion

In this tutorial we covered setting up FTP for users with a local account. If you need to use an external authentication source, you might want to look into vsftpd's support of virtual users. This offers a rich set of options through the use of PAM, the Pluggable Authentication Modules, and is a good choice if you manage users in another system such as LDAP or Kerberos.

Spaces: Object Storage on DigitalOcean

A simple and cost-effective way to store, serve, backup, and archive a virtually infinite amount of media, content, images, and static files for your apps.