CONTENTS

Introduction

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

RELATED

Initial Server Setup with CentOS 6
View

Initial Server Setup with Ubuntu 12.04
View

// Tutorial //

How To Set Up vsftpd for a User's Directory on Ubuntu 20.04

Published on October 1, 2021

Linux Basics Security Ubuntu Ubuntu 20.04

By Melissa Anderson, Kathleen Juell and Jeanelle Horcasitas

Not using Ubuntu 20.04?
Choose a different version or distribution.

Ubuntu 20.04

Introduction

FTP, which is short for *File Transfer Protocol*, is a network protocol that was once widely used for moving files between a client and server. FTP is still used to support legacy applications and workflows with very specific needs. If you have a choice on protocol, consider modern options that are more efficient, secure, and convenient for delivering files. For example, Internet users who download directly from their web browser with `https`, and command line users who use secure protocols such as the `scp` or SFTP.

vsftpd, *very secure FTP daemon*, is an FTP server for many Unix-like systems, including Linux, and is often the default FTP server for many Linux distributions as well. vsftpd is beneficial for optimizing security, performance, and stability. It also provides strong protection against security problems found in other FTP servers. vsftpd can handle virtual IPD configurations, encryption support with SSL integration, and more.

In this tutorial, you'll configure vsftpd to allow a user to upload files to their home directory using FTP with login credentials secured by SSL/TLS. You'll also connect your server using FileZilla, an open-source FTP client, to test the TLS encryption.

Prerequisites

To follow along with this tutorial you will need:

- The first thing you need is an Ubuntu 20.04 server, a non-root user with sudo privileges, and an enabled firewall. You can learn more about how to do this in our Initial Server Setup with Ubuntu 20.04 guide.
- The second thing you need is FileZilla, an open-source FTP client, installed and configured on your local machine. This will allow you to test whether the client can connect to your server over TLS. You can find instructions for installing FileZilla on Debian and Ubuntu systems from this tutorial, along with links to instructions for installing it on other systems.

Step 1 – Installing vsftpd

Start by updating your package list:

```
$ sudo apt update
```

Copy

Next, install the `vsftpd` daemon:

```
$ sudo apt install vsftpd
```

Copy

When the installation is complete, copy the configuration file so you can start with a blank configuration, while also saving the original as a backup:

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

Copy

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

Step 2 – Opening the Firewall

First, check the firewall status to see if it's enabled. If it is, then you'll make adjustments to ensure that FTP traffic is permitted so firewall rules don't block the tests.

Check the firewall status:

```
$ sudo ufw status
```

Copy

This output reveals that the firewall is active and 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, you'll need to add rules for FTP traffic.

Start by opening ports `20`, `21`, and `990` so they're ready when you enable TLS:

```
$ sudo ufw allow 20,21,990/tcp
```

Copy

Next, open ports `40000-50000` for the range of passive ports you will be setting in the configuration file:

```
$ sudo ufw allow 40000:50000/tcp
```

Copy

Check the status of your firewall:

```
$ sudo ufw status
```

Copy

The output of your firewall rules should now appear as the following:

```
Output
Status: active

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

With `vsftpd` installed and the necessary ports open, now it's time to create a dedicated FTP user.

Step 3 – Preparing the User Directory

In this step, you will create a dedicated FTP user. However, you may already have a user in need of FTP access. This guide outlines how to preserve an existing user's access to their data, but, even so, we recommend that you start with a new dedicated FTP user until you've configured and tested your setup before reconfiguring any existing users.

Start by adding a test user:

```
$ sudo adduser sammy
```
Copy

Assign a password when prompted. Feel free to press `ENTER` to skip through the following prompts, as those details aren't important for the purposes of this step.

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. Since `vsftpd` secures the directory in a specific way, 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 have shell access.

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

Create the `ftp` folder:

```
$ sudo mkdir /home/sammy/ftp
```
Copy

Set its ownership:

```
$ sudo chown nobody:nogroup /home/sammy/ftp
```
Copy

Remove write permissions:

```
$ sudo chmod a-w /home/sammy/ftp
```
Copy

Verify the permissions:

```
$ sudo ls -la /home/sammy/ftp
```
Copy

```
Output
total 8
dr-xr-xr-x 2 nobody nogroup 4096 Sep 14 20:28 .
drwxr-xr-x 3 sammy sammy  4096 Sep 14 20:28 ..
```

Next, create the directory for file uploads:

```
$ sudo mkdir /home/sammy/ftp/files
```
Copy

Then assign ownership to the user:

```
$ sudo chown sammy:sammy /home/sammy/ftp/files
```
Copy

A permissions check on the `ftp` directory should return the following output:

```
$ sudo ls -la /home/sammy/ftp
```
Copy

```
Output
total 12
dr-xr-xr-x 3 nobody nogroup 4096 Sep 14 20:30 .
drwxr-xr-x 3 sammy sammy  4096 Sep 14 20:28 ..
drwxr-xr-x 2 sammy sammy  4096 Sep 14 20:30 files
```

Finally, add a `test.txt` file to use for testing:

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

Copy

```
Output
vsftpd test file
```

Now that you've secured the `ftp` directory and allowed the user access to the `files` directory, next you will modify our configuration.

Step 4 – Configuring FTP Access

In this step, you will 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`. Open this file using your preferred text editor. Here, we'll use `nano`:

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

Copy

Once you've opened the file, confirm that the `anonymous_enable` directive is set to `NO` and the `local_enable` directive is set to `YES`:

/etc/vsftpd.conf

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

These settings prevent anonymous logins and permit local logins, respectively. Keep in mind that enabling local logins means that any normal user listed in the `/etc/passwd` file can be used to log in.

Some FTP commands allow users to add, change, or remove files and directories on the filesystem. Enable these commands by uncommenting the `write_enable` setting. You can do this by removing the pound sign (`#`) preceding this directive:

/etc/vsftpd.conf

```
. . .
write_enable=YES
. . .
```

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
. . .
```

Next, add a `user_sub_token` directive whose value is the `$USER` environment variable. Then add a `local_root` directive and set it to the path shown, which also includes the `$USER` environment variable. This setup ensures that the configuration will allow for this user and future users to be routed to the appropriate user's home directory when logging in. Add these settings anywhere in the file:

/etc/vsftpd.conf

```
. . .
user_sub_token=$USER
local_root=/home/$USER/ftp
```

Limit the range of ports that can be used for passive FTP to ensure enough connections are available:

/etc/vsftpd.conf

```
. . .
pasv_min_port=40000
pasv_max_port=50000
```

Note: In Step 2, you opened the ports that are set here for the passive port range. If you change these values, be sure to update your firewall settings.

To allow FTP access on a case-by-case basis, set the configuration so that users have access 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 changes, save the file and exit the editor. If you used `nano` to edit the file, you can do so by pressing `CTRL + X`, `Y`, then `ENTER`.

Finally, add your user to `/etc/vsftpd.userlist`. Use the `-a` flag to append to the file:

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

Copy

Check that it was added as you expected:

```
$ cat /etc/vsftpd.userlist
```

Copy

```
Output
sammy
```

Restart the daemon to load the configuration changes:

```
$ sudo systemctl restart vsftpd
```

Copy

With the configuration in place, now you can test FTP access.

Step 5 – Testing FTP Access

We've configured the server to allow only the user **sammy** to connect via FTP. Now we will make sure that this works as expected.

Since you've disabled anonymous access, you can test it by trying to connect anonymously. If the configuration is set up properly, anonymous users should be denied permission. Open another terminal window and run the following command. Be sure to replace `203.0.113.0` with your server's public IP address:

```
$ ftp -p 203.0.113.0
```

Copy

When prompted for a username, try logging in as a nonexistent user such as **anonymous** and you will receive the following output:

```
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:

```
ftp> bye
```

Copy

Users other than **sammy** should also fail to connect. Try connecting as your sudo user. They should also be denied access, and it should happen before they're allowed to enter their password:

```
$ ftp -p 203.0.113.0
```

Copy

```
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:

```
ftp> bye
```

Copy

The user **sammy**, on the other hand, should be able to connect, read, and write files. Make sure that your designated FTP user can connect:

```
$ ftp -p 203.0.113.0
```

Copy

```
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>
```

Now change into the `files` directory:

```
ftp> cd files
```

Copy

```
Output
250 Directory successfully changed.
```

Next, run `get` to transfer the test file you created earlier to your local machine:

```
ftp> get test.txt
```

Copy

```
Output
227 Entering Passive Mode (203,0,113,0,169,12).
150 Opening BINARY mode data connection for test.txt (17 bytes).
226 Transfer complete.
17 bytes received in 0.00 secs (4.5496 kB/s)
ftp>
```

Next, upload the file with a new name to test write permissions:

```
ftp> put test.txt upload.txt
```

Copy

```
Output
227 Entering Passive Mode (203,0,113,0,164,71).
150 Ok to send data.
226 Transfer complete.
17 bytes sent in 0.00 secs (5.3227 kB/s)
```

Close the connection:

```
ftp> bye
```

Now that you’ve tested your configuration, next you’ll take steps to further secure your server.

Step 6 — Securing Transactions

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

Use `openssl` to create a new certificate and use the `-days` flag to make it valid for one year. In the same command, add a private 2048-bit RSA key. 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:

```
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/v
```

You’ll be prompted to provide address information for your certificate. Substitute your own information for the highlighted values:

```
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_server_ip
Email Address []:
```

For more detailed information about the certificate flags, read 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, there will be two lines that begin with `rsa_`. Comment them out by preceding each line with a pound sign (`#`):

/etc/vsftpd.conf

```
. . .
# rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
# rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
. . .
```

After those lines, add the following lines that point to the certificate and private key you created:

/etc/vsftpd.conf

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

Now you will force the use of SSL, which will prevent clients that can't handle TLS from connecting. This is necessary to ensure that all traffic is encrypted, but it may force your FTP user to change clients. Change `ssl_enable` to `YES`:

/etc/vsftpd.conf

```
. . .
ssl_enable=YES
. . .
```

Next, add the following lines to explicitly deny anonymous connections over SSL and 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
. . .
```

Then 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
. . .
```

Lastly, add two final options. The first will not require SSL reuse because it can break many FTP clients. The second 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
. . .
```

Here is how this section of the file should appear after all of these changes have been made:

/etc/vsftpd.conf

```
# This option specifies the location of the RSA certificate to use for SSL
# encrypted connections.
#rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
#rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
rsa_cert_file=/etc/ssl/private/vsftpd.pem
rsa_private_key_file=/etc/ssl/private/vsftpd.pem
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
```

When you're done, save and close the file. If you used `nano`, you can exit by pressing `CTRL + X`, `Y`, then `ENTER`.

Restart the server for the changes to take effect:

```
$ sudo systemctl restart vsftpd
```

Copy

At this point, you'll no longer be able to connect with an insecure command line client. If you tried, you'd get the following message:

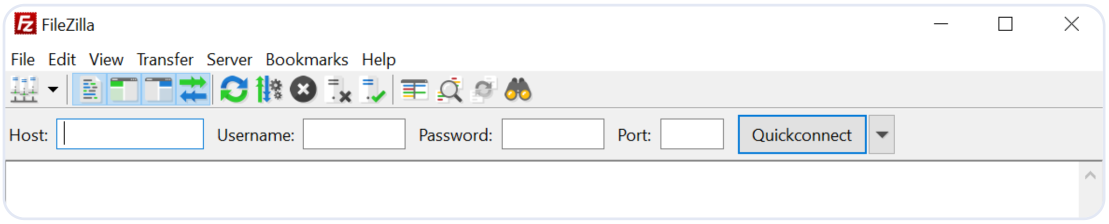
```
Output
ftp -p 203.0.113.0
Connected to 203.0.113.0.
220 (vsFTPd 3.0.3)
Name (203.0.113.0:default): sammy
530 Non-anonymous sessions must use encryption.
ftp: Login failed.
421 Service not available, remote server has closed connection
ftp>
```

Next, verify that you can connect using a client that supports TLS, such as FileZilla.

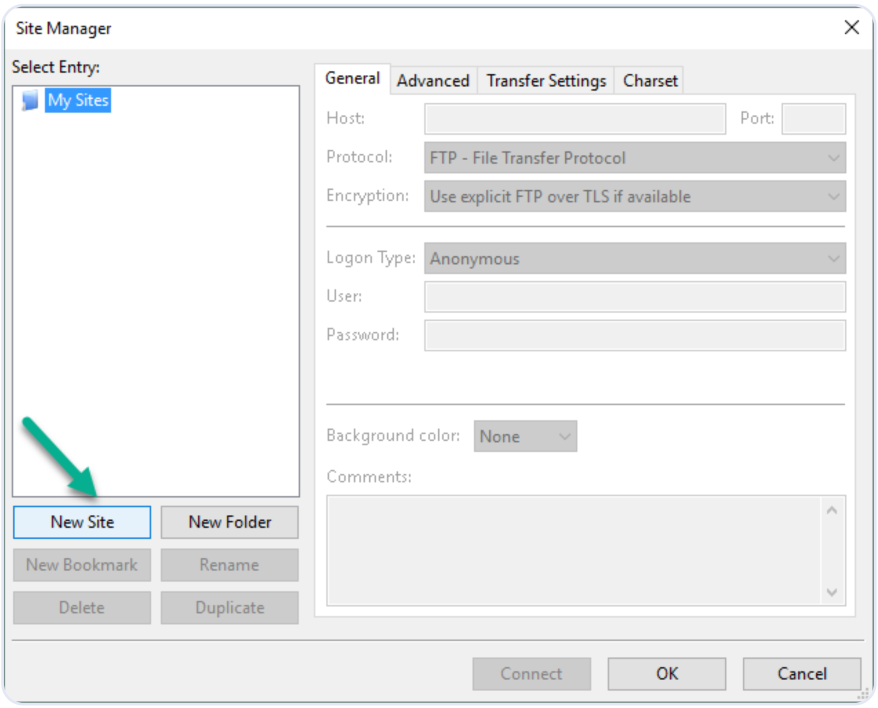
Step 7 – Testing TLS with FileZilla

Most modern FTP clients can be configured to use TLS encryption. For our purposes, we will demonstrate how to connect with FileZilla because of its cross-platform support. Consult the documentation for other clients.

When you first open FileZilla, find the **Site Manager** icon located above the word **Host**, the leftmost icon on the top row. Click this button:



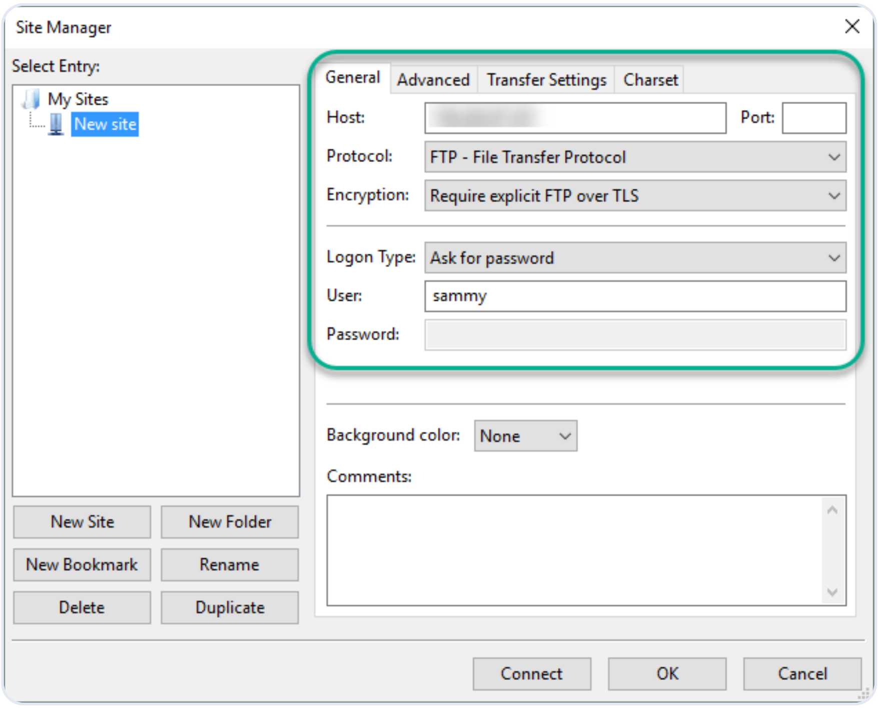
A new window will open. Click the **New Site** button in the bottom right corner:



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.

Fill out the **Host** field with the name or IP address. Under the **Encryption** drop-down menu, select **Require explicit FTP over TLS**.

For **Logon Type**, select **Ask for password**. Fill in your FTP user in the **User** field:


Site Manager

Select Entry:

My Sites
New site

General | Advanced | Transfer Settings | Charset

Host: | Port:

Protocol: FTP - File Transfer Protocol

Encryption: Require explicit FTP over TLS

Logon Type: Ask for password

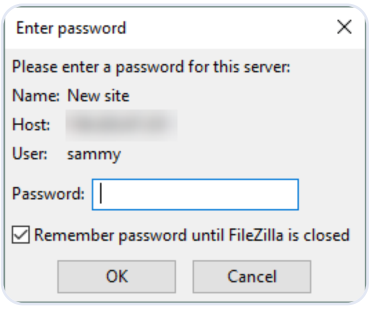
User: sammy

Password:

Background color: None

Comments:

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


Enter password

Please enter a password for this server:

Name: New site

Host:

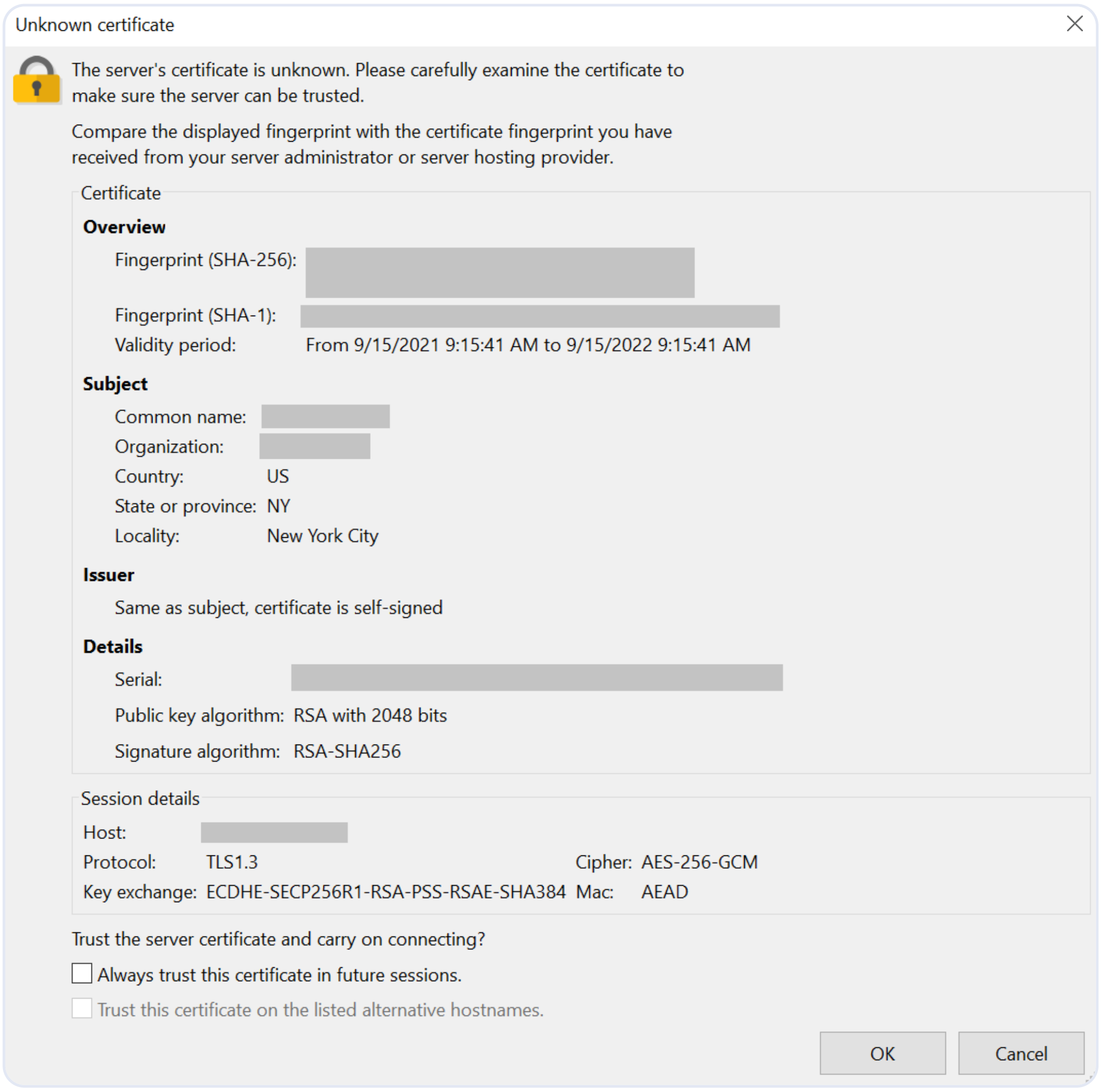
User: sammy

Password:

☑ Remember password until FileZilla is closed

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

Next, you will be presented with a server certificate that looks like the following:



Unknown certificate

The server's certificate is unknown. Please carefully examine the certificate to make sure the server can be trusted.

Compare the displayed fingerprint with the certificate fingerprint you have received from your server administrator or server hosting provider.

Certificate

Overview

Fingerprint (SHA-256):	[redacted]
Fingerprint (SHA-1):	[redacted]
Validity period:	From 9/15/2021 9:15:41 AM to 9/15/2022 9:15:41 AM

Subject

Common name:	[redacted]
Organization:	[redacted]
Country:	US
State or province:	NY
Locality:	New York City

Issuer

Same as subject, certificate is self-signed

Details

Serial:	[redacted]
Public key algorithm:	RSA with 2048 bits
Signature algorithm:	RSA-SHA256

Session details

Host:	[redacted]		
Protocol:	TLS1.3	Cipher:	AES-256-GCM
Key exchange:	ECDHE-SECP256R1-RSA-PSS-RSAE-SHA384	Mac:	AEAD

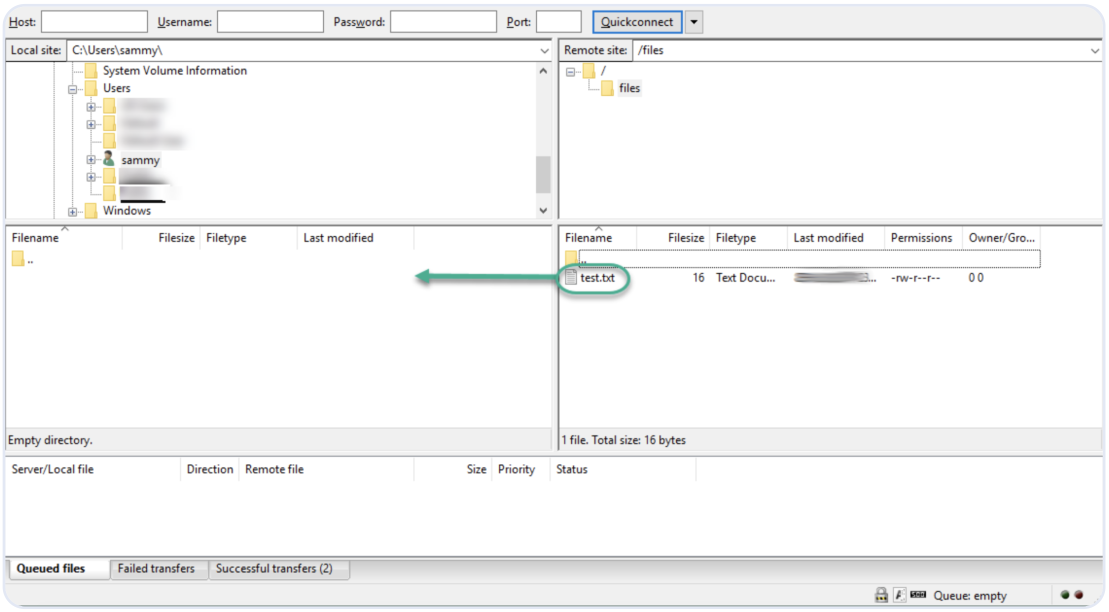
Trust the server certificate and carry on connecting?

☐ Always trust this certificate in future sessions.

☐ Trust this certificate on the listed alternative hostnames.

OK Cancel

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:



Host: Username: Password: Port: Quickconnect

Local site: C:\Users\sammy\

Remote site: /files

System Volume Information
Users
sammy
Windows

/
files

Filename	Filesize	Filetype	Last modified
..			

Empty directory.

Filename	Filesize	Filetype	Last modified	Permissions	Owner/Gro...
..					
test.txt	16	Text Docu...	[redacted]	-rw-r--r--	0 0

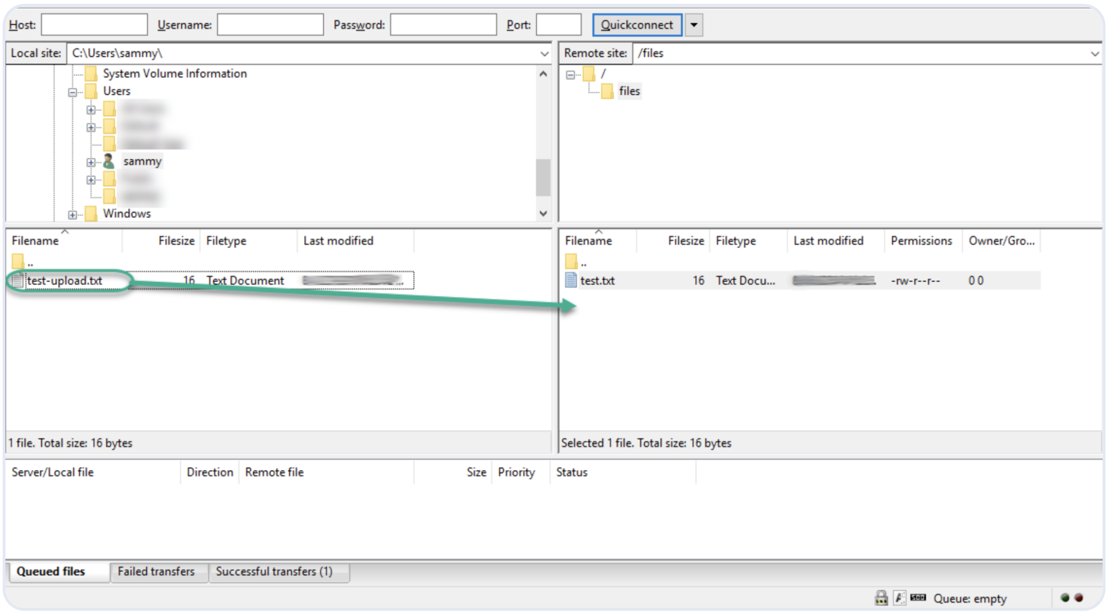
1 file. Total size: 16 bytes

Server/Local file	Direction	Remote file	Size	Priority	Status

Queued files | Failed transfers | Successful transfers (2)

Queue: empty

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 way to prevent it is by creating a custom shell. Although this will not provide any encryption, it may be worth doing so as to 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
```

Copy

Add a message telling the user why they are unable to log in:

/bin/ftponly

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

Save the file and exit your editor. If you used `nano`, you can exit by pressing `CTRL + X`, `Y`, then `ENTER`.

Then, change the permissions to make the file executable:

```
$ sudo chmod a+x /bin/ftponly
```

Copy

Open the list of valid shells:

```
$ sudo nano /etc/shells
```

Copy

At the bottom add:

/etc/shells

```
. . .
/bin/ftponly
```

Update the user's shell with the following command:

```
$ sudo usermod sammy -s /bin/ftponly
```

Copy

Now, try logging into your server as **sammy**:

```
$ ssh sammy@your_server_ip
```

Copy

You will receive the following message :

```
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. Please note, if you received an error message when logging into your server, this could mean that your server does not accept password authentication. Using password-based authentication can leave your server vulnerable to attacks, and this is why you may want to consider disabling password authentication. If you've already configured SSH-key-based authentication, you can learn more about how to disable password authentication on your server on Step 4 of this tutorial.

Conclusion

In this tutorial, we explained how to set up FTP for users with a local account. If you need to use an external authentication source, you might want to explore `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. You can also read about vsftpd features, latest releases, and updates to learn more.

About the authors

Melissa Anderson Author
Developer and author at DigitalOcean.

Kathleen Juell Author
Developer and author at DigitalOcean.

Jeanelle Horcasitas Author
Technical Writer

Educator and writer committed to empowering our community by providing access to the knowledge and tools for making creative ideas into a reality