**Set up two-factor authentication for SSH on Fedora**

**Posted by** [**Curt Warfield**](https://fedoramagazine.org/author/rcurtiswarfield/) **on** [**February 20, 2019**](https://fedoramagazine.org/two-factor-authentication-ssh-fedora/)[**0 Comments**](https://fedoramagazine.org/two-factor-authentication-ssh-fedora/#respond)



**More Like This**

[[](https://fedoramagazine.org/find-systems-easily-lan-mdns/)](https://fedoramagazine.org/find-systems-easily-lan-mdns/)

**[Find your systems easily on a LAN with mDNS](https://fedoramagazine.org/find-systems-easily-lan-mdns/)**

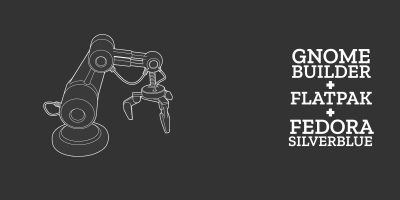
[[](https://fedoramagazine.org/test-new-features-fedora-29-atomic-cloud/)](https://fedoramagazine.org/test-new-features-fedora-29-atomic-cloud/)

**[Test the new features in Fedora 29 Atomic and Cloud](https://fedoramagazine.org/test-new-features-fedora-29-atomic-cloud/)**

[[](https://fedoramagazine.org/whats-new-fedora-29-workstation/)](https://fedoramagazine.org/whats-new-fedora-29-workstation/)

**[What’s New in Fedora 29 Workstation](https://fedoramagazine.org/whats-new-fedora-29-workstation/)**

**Recent Posts**

[[](https://fedoramagazine.org/building-flatpak-apps-in-gnome-builder-on-fedora-silverblue/)](https://fedoramagazine.org/building-flatpak-apps-in-gnome-builder-on-fedora-silverblue/)

**[Building Flatpak apps in Gnome Builder on Fedora Silverblue](https://fedoramagazine.org/building-flatpak-apps-in-gnome-builder-on-fedora-silverblue/)**

[[](https://fedoramagazine.org/how-to-watch-for-releases-of-upstream-projects/)](https://fedoramagazine.org/how-to-watch-for-releases-of-upstream-projects/)

**[How to watch for releases of upstream projects](https://fedoramagazine.org/how-to-watch-for-releases-of-upstream-projects/)**

[[](https://fedoramagazine.org/python-3-8-alpha-in-fedora/)](https://fedoramagazine.org/python-3-8-alpha-in-fedora/)

**[Python 3.8 alpha in Fedora](https://fedoramagazine.org/python-3-8-alpha-in-fedora/)**

Every day there seems to be a security breach reported in the news where our data is at risk. Despite the fact that SSH is a secure way to connect remotely to a system, you can still make it even more secure. This article will show you how.

That’s where two-factor authentication (2FA) comes in. Even if you disable passwords and only allow SSH connections using public and private keys, an unauthorized user could still gain access to your system if they steal your keys.

With two-factor authentication, you can’t connect to a server with just your SSH keys. You also need to provide the randomly generated number displayed by an authenticator application on a mobile phone.

The Time-based One-time Password algorithm (TOTP) is the method shown in this article. [Google Authenticator](https://en.wikipedia.org/wiki/Google_Authenticator) is used as the server application. Google Authenticator is available by default in Fedora.

For your mobile phone, you can use any two-way authentication application that is compatible with TOTP. There are numerous free applications for Android or IOS that work with TOTP and Google Authenticator. This article uses [FreeOTP](https://freeotp.github.io/) as an example.

**Install and set up Google Authenticator**

First, install the Google Authenticator package on your server.

$ sudo dnf install -y google-authenticator

Run the application.

$ google-authenticator

The application presents you with a series of questions. The snippets below show you how to answer for a reasonably secure setup.

Do you want authentication tokens to be time-based (y/n) **y**  
Do you want me to update your "/home/user/.google\_authenticator" file (y/n)? **y**

The app provides you with a secret key, verification code, and recovery codes. Keep these in a secure, safe location. The recovery codes are the **only** way to access your server if you lose your mobile phone.

The application asks further questions. The example below shows you how to answer to set up a reasonably secure configuration.

Do you want to disallow multiple uses of the same authentication token? This restricts you to one login about every 30s, but it increases your chances to notice or even prevent man-in-the-middle attacks (y/n) **y**  
By default, tokens are good for 30 seconds. In order to compensate for possible time-skew between the client and the server, we allow an extra token before and after the current time. If you experience problems with poor time synchronization, you can increase the window from its default size of +-1min (window size of 3) to about +-4min (window size of 17 acceptable tokens).   
Do you want to do so? (y/n) **n**  
If the computer that you are logging into isn't hardened against brute-force login attempts, you can enable rate-limiting for the authentication module. By default, this limits attackers to no more than 3 login attempts every 30s.  
Do you want to enable rate-limiting (y/n) **y**

**Set up mobile phone authentication**

Install the authenticator application (FreeOTP) on your mobile phone. You can find it in Google Play if you have an Android phone, or in the iTunes store for an Apple iPhone.

A QR code is displayed on the screen. Open up the FreeOTP app on your mobile phone. To add a new account, select the QR code shaped tool at the top on the app, and then scan the QR code. After the setup is complete, you’ll have to provide the random number generated by the authenticator application every time you connect to your server remotely.

Now you have to set up SSH to take advantage of the new two-way authentication.

**Configure SSH**

Before completing this step, **make sure you’ve already established a working SSH connection** using public SSH keys, since we’ll be disabling password connections. If there is a problem or mistake, having a connection will allow you to fix the problem.

On your server, use [*sudo*](https://fedoramagazine.org/howto-use-sudo/) to edit the */etc/pam.d/sshd* file.

$ sudo vi /etc/pam.d/ssh

Comment out the *auth substack password-auth* line:

#auth       substack     password-auth

Add the following line to the bottom of the file.

auth sufficient pam\_google\_authenticator.so

Save and close the file. Next, edit the /etc/ssh/sshd\_config file.

$ sudo vi /etc/ssh/sshd\_config

Look for the *ChallengeResponseAuthentication* line and change it to *yes*.

ChallengeResponseAuthentication yes

Look for the *PasswordAuthentication* line and change it to *no*.

PasswordAuthentication no

Add the following line to the bottom of the file.

AuthenticationMethods publickey,password publickey,keyboard-interactive

Save and close the file, and then restart SSH.

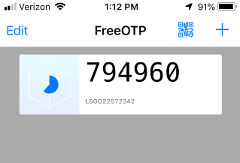
$ sudo systemctl restart sshd

**Testing your two-factor authentication**

When you attempt to connect to your server you’re now prompted for a verification code.

[user@client ~]$ ssh user@example.com  
Verification code:

The verification code is randomly generated by your authenticator application on your mobile phone. Since this number changes every few seconds, you need to enter it before it changes.



If you do not enter the verification code, you won’t be able to access the system, and you’ll get a permission denied error:

[user@client ~]$ ssh user@example.com  
Verification code:  
Verification code:  
Verification code:  
Permission denied (keyboard-interactive).  
[user@client ~]$

**Conclusion**

By adding this simple two-way authentication, you’ve now made it much more difficult for an unauthorized user to gain access to your server.