‹ Articles (/docs/)

How to Use SSH Public Key Authentication

Overview

Public key authentication is a way of logging into an SSH/SFTP (getting-started-with-ssh-and-sftp.html) account using a cryptographic key rather than a password.

If you use very strong SSH/SFTP passwords, your accounts are already safe from brute force attacks. However, using public key authentication provides many benefits when working with multiple developers. For example, with SSH keys you can

- allow multiple developers to log in as the same system user without having to share a single password between them;
- revoke a single developer's access without revoking access by other developers; and
- make it easier for a single developer to log in to many accounts without needing to manage many different passwords.

How Public Key Authentication Works

Keys come in pairs of a public key and a private key. Each key pair is unique, and the two keys work together.

These two keys have a very special and beautiful mathematical property: if you have the private key, you can prove you have it without showing what it is. It's like proving you know a password without having to show someone the password.

Public key authentication works like this:

1. Generate a key pair.
2. Give someone (or a server) the public key.
3. Later, anytime you want to authenticate, the person (or the server) asks you to prove you have the private key that corresponds to the public key.
4. You prove you have the private key.

You don't have to do the math or implement the key exchange yourself. The SSH server and client programs take care of this for you.

Generate an SSH Key Pair

You should generate your key pair on your laptop, not on your server. All Mac and Linux systems include a command called `ssh-keygen` that will generate a new key pair.

If you're using Windows, you can generate the keys on your server. Just remember to copy your keys to your laptop and delete your private key from the server after you've generated it.

To generate an SSH key pair, run the command **`ssh-keygen`**.

```
ssh-keygen
```

It will look like this when you run it:

```
laptop1:~ yourname$ ssh-keygen

Generating public/private rsa key pair.
```

You'll be prompted to choose the location to store the keys. The default location is good unless you already have a key. Press **Enter** to choose the default location.

```
Enter file in which to save the key (/Users/yourname/.ssh/id_rsa):
```

Next, you'll be asked to choose a password. Using a password means a password will be required to use the private key. It's a good idea to use a password on your private key.

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

After you choose a password, your public and private keys will be generated. There will be two different files. The one named `id_rsa` is your private key. The one named `id_rsa.pub` is your public key.

```
Your identification has been saved in /Users/yourname/.ssh/id_rsa.
Your public key has been saved in /Users/yourname/.ssh/id_rsa.pub.
```

You'll also be shown a fingerprint and "visual fingerprint" of your key. You do not need to save these.

```
The key fingerprint is:
d7:21:c7:d6:b8:3a:29:29:11:ae:6f:79:bc:67:63:53 yourname@laptop1
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|           . o   |
|      .   . * .  |
|     . .   = o   |
|      o S . o    |
|     . . o oE    |
|    . .oo +.     |
|     .o.o.*.     |
|     ....= o     |
+-----------------+
```

Configure an SSH/SFTP User for Your Key

Method 1: Using `ssh-copy-id`

Now that you have an SSH key pair, you're ready to configure your app's system user so you can SSH or SFTP in using your private key.

To copy your public key to your server, run the following command. Be sure to replace "x.x.x.x" with your server's IP address and `SYSUSER` with the name of the the system user your app belongs to.

```
ssh-copy-id SYSUSER@x.x.x.x
```

Method 2: Manual Configuration

If you don't have the `ssh-copy-id` command (for example, if you are using Windows), you can instead SSH in to your server and manually create the `.ssh/authorized_keys` file so it contains your public key.

First, run the following commands to make create the file with the correct permissions.

```
(umask 077 && test -d ~/.ssh || mkdir ~/.ssh)
(umask 077 && touch ~/.ssh/authorized_keys)
```

Next, edit the file `.ssh/authorized_keys` using your preferred editor. Copy and paste your `id_rsa.pub` file into the file.

Log In Using Your Private Key

You can now SSH or SFTP into your server using your private key. From the command line, you can use:

```
ssh SYSUSER@x.x.x.x
```

If you didn't create your key in the default location, you'll need to specify the location:

```
ssh -i ~/.ssh/custom_key_name SYSUSER@x.x.x.x
```

If you're using a Windows SSH client, such as PuTTy, look in the configuration settings to specify the path to your private key.

Granting Access to Multiple Keys

The `.ssh/authorized_keys` file you created above uses a very simple format: it can contain many keys as long as you put one key on each line in the file.

If you have multiple keys (for example, one on each of your laptops) or multiple developers you need to grant access to, just follow the same instructions above using `ssh-copy-id` or manually editing the file to paste in additional keys, one on each line.

When you're done, the `.ssh/authorized_keys` file will look something like this (don't copy this, use your own public keys):

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDSkT3A1j89RT/540ghIMHXIVwNlAEM3WtmqVG7YN/wYwtsJ8iCszg4/lXQsfLFxYmEVe8
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCzlL9Wo8ywEFXSvMJ8FYmxP6HHHMDTyYAWwM3AOtsc96DcYVQIJ5VsydZf5/4NWuq55Mc
```

Additional Information

Retrieve Your Public Key from Your Private Key

The following command will retrieve the public key from a private key:

```
ssh-keygen -y
```

This can be useful, for example, if your server provider generated your SSH key for you and you were only able to download the private key portion of the key pair.

Note that you cannot retrieve the private key if you only have the public key.

Correcting Permissions on the `.ssh` Directory

The instructions in this article will create your server's `.ssh` directory and `.ssh/authorized_keys` file with the correct permissions. However, if you've created them yourself and need to fix permissions, you can run the following commands on your server while SSH'd in as your app's system user.

```
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys
```

Still Have Questions?

Don't hesitate to contact us (/contact) if you can't find the answers to your questions.

ServerPilot (/)

(https://twitter.com/ServerPilot) (https://www.facebook.com/ServerPilot)

OVERVIEW

Pricing (/pricing)

WordPress Hosting (/wordpress-hosting)

Magento Hosting (/magento-hosting)

Laravel Hosting (/laravel-hosting)

PHP Hosting (/php-hosting)

Terms (/terms) / Privacy (/privacy)

GDPR (/gdpr/)

Features (/features)

About (/about)

Blog (/blog/)

Security (/security)

API (/docs/how-to-use-the-serverpilot-api)

AutoSSL (/docs/how-to-use-autossl)

RESOURCES

Documentation (/docs/)

Sign Up (https://manage.serverpilot.io/signup)

Getting Started (/docs/how-to-connect-a-server-to-serverpilot)

Support (/support)

Referral Program (/referral-program)

Logos and Badges (/logos-and-badges)

ARTICLES

How ServerPilot Works (/docs/how-serverpilot-works)

Install WordPress on DigitalOcean (/docs/how-to-install-wordpress-on-digitalocean)

DigitalOcean Control Panel (/docs/digitalocean-control-panel)

cPanel Alternative (/docs/cpanel-alternative)

Plesk Alternative (/docs/plesk-alternative)

HostGator Alternative (/docs/hostgator-alternative)

Bitnami Alternative (/docs/bitnami-alternative)