SSH authentication

Key-Based SSH Authentication

SSH keys, similarly to cryptocurrency keys, consist of public and private keys. You should store the private key on a machine you trust. The corresponding public key is what you will add to your server to secure it.

Be sure to securely store a secure backup of your private SSH key. From your local machine that you plan to SSH from, generate an SSH key. This is likely going to be your laptop or desktop computer. Use the following command if you are using OSX or Linux:

```

Copy ssh-keygen-tecdsa

```

Decide on a name for your key and proceed through the prompts.

```

Copy Youridentificationhasbeensavedin/Users/myname/.ssh/id_ecdsa. Yourpublickeyhasbeensavedin/Users/myname/.ssh/id_ecdsa.pub Thekeyfingerprintis: ae:89:72:0b:85:da:5a:f4:7c:1f:c2:43:fd:c6:44:38myname@mymac.local Thekey's randomart image is: +--[ ECDSA 256]---+ | | | . | | E . | | . . o | | o . . S . | | + + o . + | |. + o = o + | | o...o * o | |. oo.o . | +-----------------+

```

Copy the contents of your public key.

Your file name will differ from the command below based on how you named your key. ```

Copy cat/Users/myname/.ssh/id_rsa.pub

```

Give the ssh folder the correct permissions.

```

Copy mkdir-p~/.ssh&&chmod700~/.ssh

```

Chmod 700 (chmod a+rwx,g-rwx,o-rwx) sets permissions so the user or owner can read, write and execute, and the Group or Others cannot read, write or execute. Copy the contents of your newly generated public key.

```

Copy cat/Users/myname/.ssh/id_ecdsa.pub

```

CopySSH Public Key To Your Server

Now log into the server that you want to protect with your new SSH key and create a copy of the pubkey.

Create a file and paste in the public key information you copied from the previous step. Be sure to save the file.

```

Copy nanokey.pub

```

Now add the pubkey to the authorized keys list.

```

Copy catkey.pub>>~/.ssh/authorized_keys&&chmod600~/.ssh/authorized_keys

```
```

Once you've confirmed that you can login via the new key, you can proceed to lock down the server to only allow access via the key.

Edit sshd_config to disable password-based authentication.

```

Copy sudonano/etc/ssh/sshd_config

```

Change PasswordAuthentication yes" to "PasswordAuthentication no" and then save.

```

Copy

# Example of overriding settings on a per-user basis

# Match User anoncvs

# X11Forwarding no

# AllowTcpForwarding no

# PermitTTY no

# ForceCommand cvs server

PasswordAuthentication no

```

Restart ssh process for settings to take effect.

```

Copy servicesshrestart

```

Securing SSH with FIDO U2F (YubiKey)

For additional security node operators may choose to secure their SSH connections with FIDO U2F hardware security devices such as YubiKey, SoloKey, or a Nitrokey. A security key ensures that SSH connections will not be possible using the private and public SSH key-pair without the security key present and activated. Even if the private key is compromised, adversaries will not be able to use it to create SSH connections without its associated password and security key.

This tutorial will go over how to set up your SSH connection with FIDO U2F using a YubiKey, but the general process should work with other FIDO U2F security devices.

NOTE: More information on how to get started using a YubiKey can be foundHERE . You should have a general understanding of how to use a YubiKey before attempting this ssh guide.

SSH Requirements

For SSH secured with FIDO U2F to work both the host and server must be running SSH version 8.2 or higher. Check what version of SSH is on your local machine, and your server by running:

```

Copy ssh-V

```

It does not matter if there are mismatched versions between the host machine and server; as long as they are both using version 8.2 or higher you will be able to secure your ssh connection using FIDO U2F.

Creating SSH Key-Pairs With FIDO U2F Authentication

SSH key-pairs with FIDO U2F authentication use 'sk' in addition to the typical commands you would expect to generate SSH key-pairs with and support both ecdsa-sk and ed25519-sk.

YubiKeys require firmware version 5.2.3 or higher to support FIDO U2F using ed25519-sk to generate SSH key-pairs. To check the firmware version of a YubiKey, connect the YubiKey to your host machine and execute the following command:

```

Copy lsusb-v2>/dev/null|grep-A2Yubico|grep"bcdDevice"|awk'{print 2}'

```

To allow your host machine to communicate with a FIDO device through USB to verify attestation and assert signatures the libsk-fido2 library must be installed.

```

Copy sudoaptinstalllibfido2-dev

# macos users will need to run the command found below

brewinstalllibfido2

```

Generate an ed25519-sk key-pair with the following command with your YubiKey connected to your host machine (NOTE: you will be prompted to touch your YubiKey to authorize SSH key-pair generation):

```

Copy ssh-keygen-ted25519-sk-C"(hostname)-(date+'%d-%m-%Y')-yubikey1"

```

You can now use your new ed25519-sk key-pair to secure SSH connections with your servers. Part of the key-pair is from the YubiKey, and is used to secure the SHH connection as part of a challenge-response from the devices.

Last updated11 months ago On this page *Key-Based SSH Authentication * Copy SSH Public Key To Your Server * Securing SSH with FIDO U2F (YubiKey)* SSH Requirements * Creating SSH Key-Pairs With FIDO U2F Authentication

Was this helpful? Edit on GitHub Export as PDF