A nice `ssh` setup

Vedant Modi

Contents

0 Introduction

This document serves as a setup guide for a **secure**, **fast**, and **convenient** configuration of `ssh`. In place of authentication via a password, which is a little antiquated, we will use an `ssh` key. With a key, authentication and access to your server from your local machine will be password-less, but still secure.

I hope this is more convenient for access to servers you can connect to, like an institution's server, or cloning a GitHub repository on to your local machine.

Note: This guide assumes basic knowledge and access, of Unix, or Unix-like systems like macOS.

1 Connecting via ssh

First, establish a connection to your server via `ssh`. This can be done with `ssh user@hostname`, for example, to get into my Tufts EECS server, I use my username `vmodi01`, and the hostname `homework.cs.tufts.edu` to login. The command, there, would be

```
ssh vmodi01@homework.cs.tufts.edu
```

Sometimes, we say to use the `-X` or `-Y` flag, but this is usually only needed if you need to display a **GUI**, so these flags are not required if one is dealing with only terminal output (most cases). If using a program like `kcachegrind` or `display`, then you probably need it.

After entering the `ssh` command described above, you will be prompted for credentials (username and password). Enter them, and ensure that you can login.

2 Creating an ssh key

Next, we will create an `ssh` key. To start, open your system terminal, and ensure that you are *not* connected to the `ssh` from step 1. Generate an `ssh` key by doing

```
ssh-keygen -t ed25519 -C "user@hostname"
```

The `-t` flag specifies the `ed25519` algorithm to generate the key, and the `-C` flag specifies a nickname that you can see in the generated file. The command will ask for a location to save it to. By default, it should save to `~/.ssh/id_ed25519`.

Feel free to just press `Enter` through the prompts for the password, it's up to you if you want to add one, but it's just an extra step when you would be logged in to your computer anyway.

If you are confused during this step, here's what some output *could* look like

```
Generating public/private ed25519 key pair.
Enter file in which to save the key (/Users/vedantmodi/.ssh/id_ed25519):
Created directory '/Users/vedantmodi/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/vedantmodi/.ssh/id_ed25519
Your public key has been saved in /Users/vedantmodi/.ssh/id_ed25519.pub
```

```
The key fingerprint is:
[SHA256:wP4b8fyo6WexQnJDxKE+dJMFFA3PnAmCeV1jd6LxUuU vedantmodi@MACBOOK-PRO]
/* Don't worry I am not using this key anywhere */
The key's randomart image is:
+--[ED25519 256]--+
|     o.+BB* +.o  |
|    o..o+O.X +   |
|     .=.+ O . E  |
|     + o.. .     |
|      +.S        |
|      .o++.      |
|       +o.oo     |
|         .++o    |
|        .==. .   |
+----[SHA256]-----+
```

Once you have generated these keys, go to their location to check them out. Do this by typing `cd DIRECTORY_OF_KEY` (in my case, `cd /Users/vedantmodi/.ssh/`). Here, once you `ls`, you should see at least two files

```
id_ed25519      id_ed25519.pub
```

Under no circumstance should you `id_ed25519` share with anyone. `id_ed25519` is a **private key** corresponding to the public key, `id_ed25519.pub`. Feel free to share the **public key** . If you're curious about this process, feel free to check out this article on asymmetric-key cryptography.

If you share, or suspect someone has access to `id_ed25519`, `rm id_ed25519 id_ed25519.pub` from your system, and delete the key from the host. (You will add the key to the host in the next step, and the removal step is similar)

3 Installing the `ssh-key` on a server

Now, we have the `ssh` key-pair for secure connection to a server. But we have to tell the server our public key, so we can use our private key for authentication when we want to access the server (basically the whole point). To do this, copy the *public key* using `ssh-copy-id` to the server.

```
ssh-copy-id -i ~/.ssh/id_ed25519.pub user@hostname
```

If your host is GitHub, you may have to go into your user settings and paste the contents of your public key.

Like all good practitioners, we should test that this worked. To do this, try

```
ssh user@hostname
```

You will know if this worked if you aren't asked for a password!

4 Adding a shortname for the host

This next step will make your life a little easier to `ssh` into your server. Instead of typing in `ssh user@hostname` (a mouthful!), you can type something like `ssh tufts`, which is naturally easier to type. Add the following to the file `~/.ssh/config`.

```
Host NICKNAME /* This can be anything you want: tufts, banana, venator, etc */
  HostName HOSTNAME /* Something like homework.cs.tufts.edu */
  User USER /* Something vmodi01 */
  ForwardAgent yes /* Now your other local keys will work on the server */
```

Now, test again!

You can test by doing `ssh NICKNAME`, which should work if you followed the guide exactly.

5 Making SFTP more secure

For the last step, we will leverage our previous work to patch a **security flaw** that maybe present in your SFTP setup. Before I set the key-pair up, I had a `json` file with password to a server I was syncing to in *plaintext*, an inherent security flaw. I advise you to search for ways to provide your private key to the SFTP setup, in my VSCode SFTP setup, it was as easy as adding the property `"privateKeyPath":"~/.ssh/id_ed25519"`. We can see that `~/.ssh/id_ed25519` is just our *private key* from before.

6 Wrap up

If you're reading this, you have a newly founded cracked setup — congratulations! Please feel free to reach out to me, Vedant Modi, if you have any questions or have found any errors in my guide, and I hope you enjoyed setting this up!