This is a collaborative doc in Puppy Linux Users Group.

Edit

How To: Setup Open SSH Server

JOHN CREIGHTON · SATURDAY, APRIL 3, 2021 ·

This doc is a refference doc for myself on how to setup an openssh server. I’ve tested this on debia/ubuntu based puppies such as fossapup64 and upuphh+d. When I’m happy with this doc, I will see if any of this info should be moved to the wiki. I’m posting it here because it should be easier to find since there are likely very few how-to docs created on this facebook group.

Step #1 Install an ssh server

I typically install openssh because it has more features than dropbear, however others have suggested dropbear (see post). On debian/ubuntu based puppies the package that you need to install is called “openssh-server”, you can install this either via the ppm or alternatively via “pkg”. For instance to install via pkg use the following command:

```
pkg --get openssh-server
```

Step #2 Create or modify /etc/ssh/sshd_config

Often a given linux package will have a configuration file which may or may not be necessary. When it isn’t necessary the defaults are sufficient. When the defaults aren’t sufficient such a file must be created and/or modified. If after installing openssh there is no configuration file included you can use the following file as a starting point:
https://pastebin.com/WNWLCRJM
*or alternatively search for online for examples.

For information on this file see:
https://linux.die.net/man/5/sshd_config

Step #3.A Generate you Host Keys

First generate a ssh hostkey:

```
 ssh-keygen -t ecdsa -b 521
```

https://www.ssh.com/ssh/keygen/ (related post)

when asked for the password, do not type a password and just press enter instead. Host keys cannot have passwords. When asked where to save the key, save it with the following path:

```
/etc/ssh/ssh_host_ecdsa_key
```

Step #3.B Generate Client Keys (if you haven't already done so)

Secure Shell (ssh) client keys are generated similarly to server keys except that you type in a passphrase when asked, and also these types of keys are saved at a different path. Typically `ssh-keygen` will suggest a good location to store the ssh client keys such as:

```
/root/.ssh/id_ecdsa
```

remember the client keys need to be generated on the client computer and the host keys should be generated on the ssh server computer. For good measure, one could generate both types of keys on each computer.

Step #4 Specified Allowed Hosts via authorized keys on the Server Computer

An ssh server needs to specify which hosts are allowed to connect to the server. This can be key-based or IP address based. Typically both will be used and the key based permissions also server as an authentication method. When key based authentication isn't used then password based authentication should be specified in /etc/ssh/sshd_config.

When specifying allowed hosts by key this information is typically stored in one of two places, either:
```
~/.ssh/authorized_keys
```
or alternatively:
```
/etc/ssh/keys/root/authorized_keys
```

The first location is the default location. The second location is a typical setup if one wants to centralize all the authorized_key files in a single directory. In the latter case one should have the following line in their sshd_config file:
```
AuthorizedKeysFile /etc/ssh/keys/%u/authorized_keys
```

The above sshd_config example has this line in it and consequently centralizes the authorized_keys files. For each type of client key added to the authorized_key file, One needs a corresponding host key generated and these other host keys need to also be specified in the sshd_config file. So for instance if one wants the server to support both ecdsa and ed25519 keys then they need the following two lines in the sshd_config file:

```
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
```

the ed25519 key can be generated as follows:

```
 ssh-keygen -t ed25519
```

Step #5, verify that the clients aren't blocked by the tcp wrapper

Based on the files /etc/hosts.allow and /etc/hosts.deny the tcp wrapper will decide which hosts are allowed to connect to the ssh server via IP address. For instance say you wanted to allow all hosts on the local subnet 192.168.*.*. One way to do this is to add the following line to /etc/hosts.allow

```
 sshd : 192.168.*.*
```

Further reading:
https://www.crybit.com/allow-ssh-connection-on-server/

Troubleshooting and Initial Testing

Prior to enabling your ssh server at boot you should first test that it is working properly. First locate the path to the sshd utility with the following command:

```
which sshd
```

Then launch the ssh server by specifying the full path with the following options (see post):

```
 /usr/sbin/sshd -deD
```

use the ifconfig command to find the ip adress of the server, and then connect to the server with some assortment of options (see post):

```
ssh -vvv -Y  root@IP_ADDRESS  #where  IP_ADDRESS is the ip adress of the server
```

The -vvv option gives maximum debugging information. For less debugging information use fewer or no v’s. The -Y means to do XForwarding in trusted mode. This is faster because the xserver networks directly over the tcp connection rather than tunneling it through ssh. However, it is less secure. For greater security use the -X option instead of -Y. In either case the mode of operation has to be allowed in the sshd_config file. In the above sshd_config example untrusted mode is allowed and this is because generally people trust their own local network.