How to use SSH keys to connect without a password.

OpenSSH is a very secure way to connect remotely to a Slackware machine. But the easiest way to use SSH is to use its key facility.

The concept of public/private keys can be hard to explain, we will try to go through it in as simple a manner as possible.

Allow me to say this again, for all of you, crypto nerds out there: yes, I know this is a very simplified version of SSH. This is created for all the SSH newbies... mmmmkay?

Essentially, SSH keys are based on public key cryptography. This means you create two keys: one is called the public key, and it is used to encrypt data that only you can decrypt. You can give your public key to anyone, since its only function is to encrypt data - there is not much more you can do with it. The other key is called the PRIVATE key, and it is this key that is used to decrypt data encrypted with the public key.

So far so good... Now, how is this used with SSH?

Whenever you contact a Slackware machine (or any machine running OpenSSH, really) through the SSH protocol, your SSH client (the program, installed on the computer in front of you, that you use to connect) will talk to the SSH server installed on the distant machine. They will determine together the capabilities that they both can use, and the protocol version they should use to communicate securely.

Then, they will try to determine how you (the user) will login on the remote machine. If keys are not used, SSH will usually (but not always) default to asking you a password. On the other hand, if keys are used, the machines are going to use them in the following order:

1. The SSH server will encrypt a short message (technically a hash value) with your public key and send it to your computer.
2. Your SSH client will decrypt this message with the private key (whose only copy should be on your computer), and send it back to the SSH server.
3. The SSH server will then be satisfied that you “are you” so to speak, since you are theoretically the only person able to decrypt the message sent, and will grant you access immediately.

If this all seems a bit complicated, just remember this simple thing: you have a public key and a private key. The public key should be on the computer you want access to, or the “remote” computer. The private key should be on your computer.

Let's go through this process step by step!

Create the public/private key pair

To create a public and a private key, use the OpenSSH `ssh-keygen` utility. This will automatically

generate a key pair, using the default values. Here is a small example:

```
noryungi@mypc:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/noryungi/.ssh/id_rsa): TEST.rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in TEST.rsa.
Your public key has been saved in TEST.rsa.pub.
The key fingerprint is:
1a:99:51:a6:12:69:53:aa:d8:f6:c2:56:66:6e:68:5a noryungi@udon
The key's randomart image is:
+--[ RSA 2048]----+
|    .o. o        |
|    +o +         |
|   .o.o          |
| o . . +         |
|. + + + S        |
| o B   o         |
|  E + .          |
| = o             |
|.                |
+-----------------+
noryungi@mypc:~$ ls TEST*
TEST.rsa  TEST.rsa.pub
```

OK, what is going on here? First, `ssh-keygen` is going to generate a key pair. So far, so good, please make sure you read the ssh-keygen man page to understand all the options, and there are many of them.

Next, `ssh-keygen` is going to respond that it is creating the RSA key pair (public and private key). RSA is the name of the encryption algorithm used. There are three such encryption possible: DSA, RSA and ECDSA. Which one is the best is left as an exercice for the reader... 

It is then going to ask you where to save the key. Here, `TEST.rsa` is entered, since there are other keys on the system. Giving a proper name to the key is important, since it makes it much easier to remember which key connects to what.

For instance, if you had an account on a machine named: `stang.slackware.com`, a good name for the key pair would be `stang_slackware_com.rsa` or some such.

Next, `ssh-keygen` asks you for a passphrase. It is always a good idea to enter a passphrase! This allows you to protect your private key, even if it falls into the wrong hands. If you are absolutely, 100% sure that your private key is **not** going to fall into the wrong hands (how optimistic you are!), just press `Enter` here.

The rest are just informative messages, and you will note that the key pair has been saved as follows:

1. The private key is named `TEST.rsa`.
2. The public key - the one you want to copy on the remote machine - is named `TEST.rsa.pub`

Congratulations! You are halfway there!

Configure your public key on the remote computer

All right, now, how to use the public/private key? That's simple enough: copy the public key (named TEST.rsa.pub as we have seen) onto the remote computer. The best way to do this is to use scp the OpenSSH secure copy program. For instance:

```
noryungi@mypc$ scp TEST.rsa.pub
nr@test.example.com:/home/nr/.ssh/authorized_keys
```

In the example above, I copy the public key TEST.rsa.pub on the remote machine named test.example.com, as user nr. The file is renamed authorized_keys which is the name of the file that contains all the public keys authorized to connect to the server.

A word of caution here: do not execute the scp command above if you already have an authorized_keys file on the remote computer! This will replace the content of the file with your public key!! If you already have an authorized_keys file, execute a cat TEST.rsa.pub » authorized_keys on the remote machine to add your public key at the end of authorized keys.

Since all the SSH keys you use should be placed in the .ssh directory, that's where it goes on the remote machine.

So, are we done? Not really, there is just one small thing to do, but it is truly important, since it is the source of a lot of problems...

Check the public key permissions on the remote machine

Since the private and public keys are very sensitive, they should be protected against prying eyes. To do this, on both the remote and the local machine, enter the following command:

```
nr@test.example.com$ chmod -R -v g-rwx,o-rwx ./.ssh/
mode of `./.ssh/' changed from 0755 (rwxr-xr-x) to 0700 (rwx------)
mode of `./.ssh/authorized_keys' changed from 0644 (rw-r--r--) to 0600 (rw--
------)
```

This command allows you to make sure nobody (except you and the SSH server) can read the public key.

Please note: if the permissions on the authorized_keys file OR the .ssh directory are not correct OpenSSH won't use the keys! If you have any problem with a public/private key, check the permissions and/or run the above command to make sure they are correct!

Connect using your newly created SSH key

Let's try to connect, back on the local machine, to the remote server named `test.example.com`:

```
noryungi@mypc$ ssh -i TEST.rsa nr@test.example.com
```

Please note that I have entered the option `-i` right after the `ssh` command: this option selects the private key to be used to connect, as user `nr`, to the remote server named `test.example.com`.

If you have chosen to protect the private key with a passphrase, `ssh` will ask you this passphrase before connecting. If not... Well, if the permissions are correct (see above), you should see the equivalent of the following:

```
nr@test.example.com$
```

That's it! You are connected to a remote machine, without ever entering a password, and with a much better security than with a password - which can be guessed, while a key is way too long to be ever guessed.

What could go wrong at this point?

Well, not much, really, except the possibility that your system administrator does not want you to connect with a public/private key pair...

In which case, let's face it, he is not a very good system administrator. This can be checked, however, with the following command on the remote machine:

```
nr@test.example.com$ grep -i pubkeyauth /etc/ssh/sshd_config
#PubkeyAuthentication yes
```

Please note the `#PubkeyAuthentication yes` line: this is the default value for public key authentication, and, as you can see, it is set to `yes`. You are good to go and use your key! On the other hand, if you see something like this:

```
nr@test.example.com$ grep -i pubkeyauth /etc/ssh/sshd_config
PubkeyAuthentication no
```

Then you are not allowed to connect to the remote machine (`test.example.com` above) with a public key. Time to contact your system administrator or your security administrator and request politely to be able to use them.

(Yes, there are other, more sneaky, ways to connect without entering a password, but none of them are as secure as a public/private key pair - maybe in another documentation on this wiki?) 🙂

You have reached the end of this short documentation - go forth, and use OpenSSH keys!

See Also

- The OpenSSH manual pages (online)

Sources

- Originally written by Noryungi

howtos, security, ssh, sshkeys, author noryungi