Passwordless SSH

I was sure this topic had been covered on Unix Tutorial pages before, but apparently it wasn’t – so without further ado, let me introduce you to one of the most fundamental ways of improving your daily sysadmin or developer life in Linux and Unix environments: **passwordless SSH**.

What Passwordless SSH Really Is

Passwords are used to protect all sorts of things in our digital lives. So anything that is passwordless sounds like a compromise and a degraded security. Want to make it clear: that is not what’s happening with **passwordless SSH**! There is no degraded security, only added flexibility to your daily SSH
use.

Passwords are indeed the default authentication method for SSH access. You access remote server, it learns your username and asks for your password. If password is wrong, you get connection denied.

**Passwordless SSH** switches authentication mode: instead of using password, your client uses SSH key to authenticate against remote server. Just like in case with passwords, the remote SSH server must already have your SSH key in order to authenticate you and to accept your login. But depending on how you have your SSH keys management configured, you may not get asked for any password as part of logging in.

**Passwordless SSH** means you won’t get asked for your user password, but remote SSH server will instead query your client for your SSH key. If your SSH key is protected with a passphrase, you’ll still get asked for it. But if you are using SSH agent – then nothing may be asked.

Overall, **passwordless SSH** is a great improvement because it relies on SSH agent for managing your local SSH keys. You can load your SSH key into SSH agent once, and then SSH into hundreds of systems that accept that key, all without having to type passwords.

**What passwordless SSH Isn’t**

**Passwordless SSH** is not:
Configure Passwordless SSH

Okay, here is a brief example of configuring passwordless SSH. Remember: simplicity and flexibility are in the way this is used, not in the way this is deployed. So it takes a bit of effort to configure, but then becomes a breeze when it comes to adding new servers access.

For this example I’ll use the trusted Raspberry Pi system called becky.

Step 1: Generate your SSH keypair

This step is done on your local system: laptop or desktop. Better go into /home/USERNAME/.ssh directory (note the dot in front of ssh! it’s one of the dotfiles and dot directories mostly used), where these SSH keys (called identities) are usually stored. So for me, I’m in the /home/greys/.ssh directory.
DO NOT skip the passphrase – this is an important bit! Set it to some memorable phrase but not one of your existing passwords. You’ll need this pass phrase to access and use your SSH key.

```
greys@xps:~ $ ssh-keygen -t rsa -b 4096 -f unixtutorial
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in unixtutorial.
Your public key has been saved in unixtutorial.pub.
The key fingerprint is:
SHA256:tzMm/yh6rfiyEk7bfx04UDi3dxcZISljk/YWkbn2XRE greys@xps
The key's randomart image is:
+---[RSA 4096]----+
|    . .o=E= |
|   o oB =.+ |
|  +o.= o o |
|  . . .=. o |
| S..oo...o |
|  o .o.. ..|
| o + ..=o . |
| + o.o+o+. |
|  .+B++o.. |
+----[SHA256]-----+
```

Excellent. Here is our SSH identity, also called SSH keypair:

- unixtutorial is a private key (protected by the passphrase)
- unixtutorial.pub is the public key

```
greys@xps:~ $ ls -al unixtutorial*
-rw------- 1 greys 3414 Sep 11 08:49 unixtutorial
-rw-r--r-- 1 greys  735 Sep 11 08:49 unixtutorial.pub
```
Step 2: Deploy your SSH keypair onto remote server

Now it’s time to share the public key (unixtutorial.pub) with the remote system I want to access in the future. That system is server called becky, but you need to specify the IP or hostname of your own remote server. As you remember, you need to have a username/password access to that server already – if you don’t have them yet then you can’t proceed.

We use the **ssh-copy-id command** for this purpose: you specify the public key file and the hostname of remote server:

```
greys@xps:~ $ ssh-copy-id -i unixtutorial.pub becky
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "unixtutorial.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
greys@192.168.1.XX's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'becky'" and check to make sure that only the key(s) you wanted were added.

**BY THE WAY:** I have a number of small VPS servers online for the purpose of teaching Linux basics. If you want, I can create you an account there and you can test SSH access procedure there – just [contact me](mailto:).
Step 3: Connect to remote server using your new SSH keypair

Okay, now let’s try and connect to remote server using our new SSH identity. Just use the ssh command and specify the SSH private key (notice how it doesn’t say .pub at the end of unixtutorial filename):

```bash
greys@xps:~ $ ssh -i unixtutorial becky
Enter passphrase for key 'unixtutorial':
Linux becky 4.14.94-v7+ #1193 SMP Tue Jan 22 15:34:30 GMT 2019 armv7l
```

The programs included with the Debian GNU/Linux system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/*/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.

Last login: Wed Sep 11 09:00:27 2019 from 192.168.1.XX

greys@becky:~ $ echo "Woohoo!"
Woohoo!

And just like that – we’re on the remote server! So yes, we had to specify passphrase in order to use unixtutorial SSH key, but after that remote server becky didn’t ask us for a password to my account on it – it trusted my SSH key instead. So it’s been a password-less SSH access.
Step 4: Use ssh-agent for your SSH keypair

**ssh-agent** is a special tool that comes packaged with SSH. It asks for your passphrase to each private key you want to use, but then keeps the key in memory and uses it for remote access as needed. You type your passphrase once for ssh-agent, and then enjoy truly passwordless SSH to remote servers: no questions, passwords or passphrases asked.

Let’s add our key to the **ssh-agent**:

```
greys@xps:~ $ ssh-add unixtutorial
Enter passphrase for unixtutorial:
Identity added: unixtutorial (greys@xps)
```

Perfect. Now try accessing the remote server again, exactly the way we’ve done it in Step 3. Only this time we won’t get asked for a passphrase, because **ssh-agent** keeps it in memory:

```
greys@xps:~ $ ssh -i unixtutorial becky
Linux becky 4.14.94-v7+ #1193 SMP Tue Jan 22 15:34:30 GMT 2019 armv7l
```

The programs included with the Debian GNU/Linux system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.
Last login: Wed Sep 11 09:00:37 2019 from 192.168.1.XX
greys@becky:~ $ echo "Hurray!"
Hurray!

And we’re done! Congrats on your newly setup **passwordless SSH**!

Did you like this article? Leave a comment to let me know! If you have questions – feel free to ask and I’ll update the article.

**See Also**

- [SSH Reference](#)
- [SSH Port Forwarding](#)
- [SSH port](#)
- [Check SSH key fingerprints](#)
- [How To: Find ssh key by fingerprint](#)
- [Using multiple ports with SSH](#)
- [Access encrypted homedir with SSH](#)