SSH Copy ID for Copying SSH Keys to Servers

`ssh-copy-id` installs an SSH key on a server as an authorized key. Its purpose is to provide access without requiring a password for each login. This facilitates automated, passwordless logins and single sign-on using the SSH protocol.

The `ssh-copy-id` tool is part of OpenSSH.

Contents

Setting up public key authentication

Generate an SSH Key

Copy the key to a server

Test the new key

Troubleshooting

How ssh-copy-id works

Some best practices for SSH keys

Use a passphrase when possible

Add a command restriction when possible

Managing SSH keys

Command-line options

Ssh-copy-id on Mac

Installation using Homebrew

Installation from MacPorts

Installation using Curl

Setting up public key authentication

Key based authentication in SSH is called public key authentication. The purpose of `ssh-copy-id` is to make setting up public key authentication easier. The process is as follows.

Generate an SSH Key

With OpenSSH, an SSH key is created using ssh-keygen. In the simplest form, just run `ssh-keygen` and answer the questions. The following example illustates this.

```
# ssh-keygen Generating public/private rsa key pair. Enter file in which to save the key
(/home/ylo/.ssh/id_rsa): mykey Enter passphrase (empty for no passphrase): Enter same
passphrase again: Your identification has been saved in mykey. Your public key has been
saved in mykey.pub. The key fingerprint is:
SHA256:GKW7yzA1J1qkr1Cr9MhUwAbHbF2NrIPEgZXeOUOz3Us ylo@klar The key's randomart image is: +--
-[RSA 2048]----+ |.*++ o.o.        | |.+B + oo.        | | +++ *+.         | | .o.Oo.+E
| |    ++B.S.       | |   o * =.        | |  + = o          | | + = = .         | |  + o o
| +----[SHA256]-----+ #
```

Creating a key pair (public key and private key) only takes a minute. The key files are usually stored in the `~/.ssh` directory.

Copy the key to a server

Once an SSH key has been created, the `ssh-copy-id` command can be used to install it as an **authorized key** on the server. Once the key has been authorized for SSH, it grants access to the server without a password.

Use a command like the following to copy SSH key:

```
ssh-copy-id -i ~/.ssh/mykey user@host
```

This logs into the server host, and copies keys to the server, and configures them to grant access by adding them to the **authorized_keys** file. The copying may ask for a password or other authentication for the server.

Only the public key is copied to the server. The private key should never be copied to another machine.

Test the new key

Once the key has been copied, it is best to test it:

```
ssh -i ~/.ssh/mykey user@host
```

The login should now complete without asking for a password. Note, however, that the command might ask for the passphrase you specified for the key.

Troubleshooting

There are a number of reasons why the test might fail:

- The server might not be configured to accept public key authentication. Make sure **/etc/ssh/sshd_config** on the server contains `PubkeyAuthentication yes`. Remember to restart the **sshd** process on the server.
- If trying to login as **root**, the server might not be configured to allow root logins. Make sure `/etc/sshd_config` includes `PermitRootLogin yes`, `PermitRootLogin prohibit-password`, or `without-password`. If it is set to `forced-commands-only`, the key must be manually configured to use a forced command (see `command=` option in `~/.ssh/authorized_keys`.
- Make sure the client allows public key authentication. Check that **/etc/ssh/config** includes `PubkeyAuthentication yes`.

- Try adding `-v` option to the `ssh` command used for the test. Read the output to see what it says about whether the key is tried and what authentication methods the server is willing to accept.
- OpenSSH only allows a maximum of five keys to be tried authomatically. If you have more keys, you must specify which key to use using the `-i` option to `ssh`.

How ssh-copy-id works

`ssh-copy-id` uses the **SSH protocol** to connect to the target host and upload the SSH user key. The command edits the `authorized_keys` file on the server. It creates the `.ssh` directory if it doesn't exist. It creates the authorized keys file if it doesn't exist. Effectively, ssh key copied to server.

It also checks if the key already exists on the server. Unless the `-f` option is given, each key is only added to the authorized keys file once.

It further ensures that the key files have appropriate permissions. Generally, the user's home directory or any file or directory containing keys files should not be writable by anyone else. Otherwise someone else could add new authorized keys for the user and gain access. Private key files should not be readable by anyone else.

Some best practices for SSH keys

SSH keys are very useful, but can lead to problems if they are not properly managed. They are access credentials just like user names and passwords. If they are not properly removed when people leave or systems are decommissioned, no-one may any longer know who really has access to which systems and data. Many large organizations have ended up having millions of SSH keys.

Use a passphrase when possible

It is recommended that keys used for single sign-on have a passphrase to prevent use of the key if it is stolen or inadvertatly leaked. The **ssh-agent** and **ssh-add** programs can be used to avoid having to enter the passphrase every time the key is used.

Generally all keys used for interactive access should have a passphrase. Keys without a passphrase are useful for fully automated processes. They allow shell scripts, programs, and management tools to log into servers unattended. This is often used for backups and data transfers between information systems.

Add a command restriction when possible

The `copy-id` tool does not automatically add command restrictions to keys. Using command restrictions is highly recommended when the key is used for automating operations, such as running a report for fetching some files. A command restriction is basically a `command="<permitted command>"` option added to the beginning of the line in the server's **authorized_keys** file.

Managing SSH keys

Anyone having more than a few dozen servers is strongly recommended to **manage SSH keys**. Not managing the keys exposes the organization to substantial risks, including loss of confidentiality, insertion of fraudulent transactions, and outright destruction of systems.

The `copy-id` tool can be dangerous. It can easily accidentally install multiple keys or unintended keys as authorized. The logic for choosing which key to install is convoluted. Extra authorized keys grant permanent access. They can later be used to spread attacks host-to-host, and the more keys there are, the higher the risk. It also violates all **regulatory compliance requirements**.

The **Universal SSH Key Manager** is a widely used product for managing SSH keys.

Command-line options

The sample below presents *ssh-copy-id* command line syntax:

```
ssh-copy-id [-f] [-n] [-i identity file] [-p port] [-o ssh_option] [user@]hostname
```

The options have the following meaning:

-f Don't check if the key is already configured as an authorized key on the server. Just add it. This can result in multiple copies of the key in `authorized_keys` files.

-i Specifies the identity file that is to be copied (default is `~/.ssh/id_rsa`). If this option is not provided, this adds all keys listed by `ssh-add -L`. Note: it can be multiple keys and **adding extra authorized keys can easily happen accidentally**! If `ssh-add -L` returns no keys, then the most recently modified key matching `~/.ssh/id*.pub`, excluding those matching `~/.ssh/*-cert.pub`, will be used.

-n Just print the key(s) that would be installed, without actually installing them.

-o ssh_option Pass `-o ssh_option` to the SSH client when making the connection. This can be used for overriding configuration settings for the client. See **ssh command line options** and the possible configuration options in **ssh_config**.

-p port Connect to the specifed SSH port on the server, instead of the default port 22.

-h or **-?** Print usage summary.

Ssh-copy-id on Mac

While MacOS includes SSH, it does not include `ssh-copy-id` out of the port. However, according to some sources MacOS 10.12.4 includes it, and presumably newever versions include it as well.

You can test whether your Mac has it by opening a terminal window (Finder / Go / Utilities / Terminal) and typing `ssh-copy-id`.

If your system does not have it, there are many ways to install ssh-copy-id Mac version.

Installation using Homebrew

To install it using Homebrew, use the following command. You need to have the `brew` command installed.

```
brew install ssh-copy-id
```

Installation from MacPorts

The following command will install it using MacPorts. You need to have the `port` command installed.

```
sudo port install openssh +ssh-copy-id
```

Installation using Curl

The following command can be used to install a Mac version directly. Note that as a general rule we do not recommend piping any commands from the network to the shell, like this does. Only use this method if you fully trust the source. The advantage of this method is that it does not need any special software - `curl` comes preinstalled.

```
curl -L https://raw.githubusercontent.com/beautifulcode/ssh-copy-id-for-OSX/master/install.sh
```