Git Conversations With The SSH Agent Conversation

Christian Külker

2023-05-05

Contents

When using the `ssh` command, `ssh` uses a key stored in `~/.ssh` and you rarely need to worry about the details. Recently, however, in the context of moving identities and permissions away from passwords, `ssh` **public** keys are increasingly being used to control access. This is also the case with `git`. While on most systems this is handled transparently, sometimes the wrong key is used and access is denied when it should not be. This document covers `ssh` key management, **agents** and `git`.

For the sake of education, let us assume that we have generated a private and public key that will only be used to access a commercial site under `~/.ssh` called `id_ed25519_github`. So we have the following files:

- `~/.ssh/id_ed25519_github` (private key - we never touch this!)
- `~/.ssh/id_ed25519_github.pub` (public key)

As we are in a dangerous world to help with this kind of things we need an **agent**. Luckily `SSH` provides such an **agent** called `ssh-agent`.

Usually this **agent** is already started on your favorite operating system: Debian. It is very common to start the **agent** together with the `X` session aka desktop. However, for some reason unknown to me, it is usually not very talkative (or dormant if `X` is not started), at least when using the system remotely.

To understand if our **agent** is alive and willing to communicate with a mortal, you can see if the **agent** is listed in the process list:

```
ps ax|grep ssh-agent|grep -v grep
  33178 ?        Ss     0:02 /usr/bin/ssh-agent x-session-manager
```

After understanding that we have an **agent** alive, we could ask him for our keys, oddly we do not execute `ssh-agent ask` or something, but `ssh-add`, even though we are not adding anything. Who would have thought?

```
ssh-add -l
```

This will list the fingerprints of all identities currently represented by the **agent** without using a magnifying glass.

However, in some cases the **agent** is just too secretive and will not talk to us:

```
ssh-add -l
Could not open a connection to your authentication agent.
```

To change his mind, we invoke him and execute his answer with an `eval` (or similar) command. It works like Eliza, you reflect the communication back, and as you can see, communication with an **agent** is two-dimensional. The `eval` command basically sets the variables: `SSH_AUTH_SOCK` and `SSH_AGENT_PID` so that the `ssh` commands know how to talk to the **agent**. It depends on the ticks, if you do not use a back-tick you will get:

```
 eval 'ssh-agent'
SSH_AUTH_SOCK=/tmp/ssh-fMGM9OrtVYR4/agent.990266; export SSH_AUTH_SOCK;
SSH_AGENT_PID=990267; export SSH_AGENT_PID;
echo Agent pid 990267;
```

This will not update the `ssh-agent` and the connection will be denied. If you are using back-ticks, the situation is different.

```
eval `ssh-agent`
Agent pid 990667
```

If you do not like backticks, there are other approaches. There seems to be no difference between using the `-s` or not, even if this option is used in other examples online. The `-s` options generate Bourne shell commands on `stdout`. This is the default when `SHELL` doesn't look like a csh-style shell. So it is better to let the **agent** guess the shell and only if you are smarter than the **agent** tell him how to speak (csh or Bourne).

```
eval $(ssh-agent)`
Agent pid 5079
```

```
ssh-add -l
The agent has no identities.
```

Now we can add an identity.

```
ssh-add ~/.ssh/id_ed25519_github
Identity added: /home/$USER/.ssh/id_ed25519_github ($USER@$HOST.$TLD)
ssh-add -l
256 SHA256:Cekfa54+saeceiR4ooVegXGYqu+Veixae7rooroDefs $USER@$HOST.$TLD
 ↪  (ED25519
```

Once it is clear that the **agent** is using the correct key, you could test access to https://gitub.com, for example.

```
ssh -vT git@github.com
```

Of course, you could tell `ssh` directly what key to use, and bypass the **agent** entirely if you **don't** want `ssh` to talk to him. Add this to your `~/.ssh/config`.

```
Host github.com
 HostName github.com
 IdentityFile ~/.ssh/id_ed25519_github
```

1 History

Version	Date	Notes
0.1.1	2023-05-05	Improve writing
0.1.0	2022-06-22	Initial release

2 Disclaimer of Warranty

3 Limitation of Liability

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE INFORMATION, DOCUMENTS OR PROGRAMS AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE INFORMATION, DOCUMENTS OR PROGRAMS (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE INFORMATION, DOCUMENTS OR PROGRAMS TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.