

Git Conversations With The SSH Agent Conversation

Christian Külker

2022-06-22

Contents

1 History	3
2 Disclaimer of Warranty	3
3 Limitation of Liability	3

When using the `ssh` command, `ssh` uses one key stored in `~/.ssh` and one seldom need to deal with the details. However recently in the context of switching identities and authorization away from passwords more often than not `ssh` **public** keys are 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 even though it should not. This document deals with the `ssh` key management, **agents** and `git`.

For the sake of education, let's assume we generated a private and public key to be used solely for the access to a commercial site under `~/.ssh` called `id_ed25519_github`. So the following files exist:

- `~/.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**. Nicely `SSH` provide such an **agent** called `ssh-agent`.

Usually this **agent** is already started on your favorite OS: Debian. It is very common to start the **agent** together with the `X` session aka desktop. However for some reasons unknown to me he usually is 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, one can see if the **agent** is enlisted in the process list:

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

After we understood that we have an **agent** alive, we might ask him about 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 fingerprints of all identities currently represented by the **agent** without using a magnifying glass.

However in some case 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 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 knows 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-fMGM90rtVYR4/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 refused. If you are using back-ticks it looks different.

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

If you do not like back-ticks, other approaches also work. There seems to be no difference if you use 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 if `SHELL` does not look like it's a csh style of 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 -s)
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)
```

After it is clear that the **agent** is using the correct key, one could test the access to <https://github.com> for example.

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

You could of course tell `ssh` directly what key to use, and bypass the **agent** entirely, if you do **not** like `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
0.1.0	2022-06-22	Initial release

2 Disclaimer of Warranty

THERE IS NO WARRANTY FOR THIS INFORMATION, DOCUMENTS AND PROGRAMS, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE INFORMATION, DOCUMENT OR THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE INFORMATION, DOCUMENTS AND PROGRAMS IS WITH YOU. SHOULD THE INFORMATION, DOCUMENTS OR PROGRAMS PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

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.