ssh passwordless login

- Overview
- Setup procedure
- Usage
  - Bash aliases for convenience
- Problem analysis
  - auth.log messages

Overview

ssh passwordless login is done using a public/private key pair. The private key is is kept on the system you want to login from; the public key is copied to the system and user you want to login to.

Setup procedure

⚠ root's authorized_keys file is distributed from Blue Light's git. In case any extra keys are required, for example to run backups, they are put in authorized_keys2 so they are not overwritten the next time authorized_keys is updated from git.

1. Generate a public/private key pair following the procedure on ssh key generation
2. Copy the public key to the system and user you want to login to, adding it to the user's ~/.ssh/authorized_keys file. If the file does not exist, create it and set the permissions to read and write for the user only (`rw-------`).
3. For passwordless login as root to work, /etc/ssh/sshd_config on the system you want to login to must have "PermitRootLogin without password".

Usage

`ssh <username>@<host>`

If username is not specified, it defaults to the current user name

Examples:

```
ssh root@backupserver
ssh backupserver
ssh bl@192.168.28.2
```

Gotcha: the first time such a command is used for a particular host, ssh prompts interactively for confirmation to proceed. When confirmation is given, it adds the host to ~/.ssh/known_hosts. Thereafter there is no such interactive prompt. If the passwordless login is to be used from a script, it needs to be done manually the first time or the script will hang.

In case you prefer a password protected key or you want to "hop" from system to system, SSH Agent Forwarding is helpful.

Bash aliases for convenience

Bash aliases of ssh commands for most of the Blue Light supported computers are in the Blue Light git as "bash library/aliases.BL.sh" and functions.BL.sh. They are intended for sourcing from .bashrc or similar. Once they have been sourced the ag function can be used to display aliases containing an arbitrary string (ag is short for alias | grep). The aliases have been designed with that in mind so they include nickname and user name strings.

An example showing user names:

```
$ ag work
1) uma='ssh root@ac002.workcom.av'
2) valli='ssh root@ac001.workcom.av'
#?
```

Where necessary the aliases hop via an intermediate host (Blue Light policy is to configure Internet-exposed servers to accept root logon only from specific addresses):

```
$ ag hot
1) rad='ssh -A -t root@blav.bluelightav.org ssh root@hotspot.bluelightav.org'
#?
```

An example showing a hosting service provider name:

```
$ ag online
1) online.net='ssh -A root@sd-44498.dedibox.fr'
2) online.net.backup='ssh -p 2222 root@sd-44498.dedibox.fr'
#?
```

Problem analysis

The ~/.ssh directories must have 700 permissions, for example:

```
root@ac001.blue:~# ls -ld .ssh
drwx------ 2 root root 4096 Feb 5 10:24 .ssh
```

If the identity_file is not specified on the ssh command (common), has it been added to the ssh agent? This command lists what has been added

```
ssh-add -l
```

If an identity_file has been added, is its corresponding public key in the server's ~/.ssh/authorized_keys file for the user you are trying to log in as?

If there is a firewall on either ssh client or server, does it allow port 22 as required?

If the above check list has not identified the problem, generate more information by:

- Tailing the ssh server's /var/log/auth.log file while re-trying the failing ssh command.
- Use ssh's -v option. It can be used up to three times for greater verbosity.

auth.log messages

- **sshd[*]: User * not allowed because account is locke**d: Commonly caused by /etc/shadow having "!!" in the password field, in which case it can be worked around by setting a password. Further causes are documented in the sshd man page in the AUTHENTICATION section.
- .

Still stuck? Update this page when you find a solution 🙂