
Exercise Introduction

In this exercise, you will learn how to

- Use authselect to configure the authentication sources
- Create local groups and users, including SSH access
- Set passwords for local users

Contents

Task 1: Install needed packages (3 min)

Install the following packages on the head node:

- authselect
- mkpasswd
- sssd

and the following in the Warewulf image for the compute nodes (no need to reboot yet):

- authselect
- sssd

Task 2: Using authselect (7 min)

On the head node, check which profiles are available for authselect with

```
authselect list
```

The two most commonly used ones are `minimal` and `sssd`. Read the information on each one and what features they have available with

```
authselect show PROFILE
```

You can see what the `/etc/nsswitch.conf` is for each profile by running

```
authselect test PROFILE --nsswitch [FEATURE1 ...]
```

Quickly compare the head node's existing `/etc/nsswitch.conf` and the one of each profile. It helps to filter each one with `grep -E '^[^#F]' | sort` and put the results in two separate files and compare them with `diff`.

On both the head node and in the Warewulf image of the nodes, select the `minimal` profile with

```
authselect select PROFILE
```

You may have to add the `--force` option. Then check the current configuration with `authselect check` and `authselect current`

You do not have to reboot the compute nodes yet.

Task 3: Create a group for your users (3 min)

It is common on clusters to have one or more top-level POSIX groups that all the normal usernames are in since that can help management and file permissions in various places. First come up with the minimum GIDs and UIDs you will use. It is considered best practice to separate GID and UID ranges based on role and source, with the lowest range for system users/groups (e.g. 0 – 1000), a middle range for local non-system users/groups, and the whole upper range for those from remote sources. This lets authentication using different sources with SSSD put limits to the ranges so that a remote LDAP isn't used for local users/groups (low and middle UID/GIDs) and local files are never used for the centrally managed ones (high UID/GIDs). After you have picked your mininmum UID and GID for your ranges, edit `/etc/login.defs` on the head node and set the values. Then create the POSIX group on the head node for your users with `groupadd NAME` and check its GID with `getent group NAME`

Task 4: Create a normal user (7 min)

Pick your username and the shell you prefer (the most common choice is `/bin/bash`). Then create your user on the head node using `useradd [OPTIONS] NAME` such that its primary group is the group you made in the last task and its HOME directory is created under where you decided to put HOME directories on the head node's NFS share (e.g. `/nfs/home`). Check that user exists with `id NAME` and `getent passwd NAME` and check that its HOME directory was created. Check that its password is disabled in `/etc/shadow` (2nd field should be `!!`).

On your local machine, either pick an SSH key you already have or generate a new SSH key that you will use for this user. In the username's HOME directory on the head node, create their `$HOME/.ssh` directory owned by the user and their primary group and with the permissions set so only that username can access it (`700`). Then for that username, create their `$HOME/.ssh/authorized_keys` file and paste the public key on one line (**DO NOT PASTE THE PRIVATE KEY**). Then change the ownership of the file to be the username and its primary group and set its permissions to that only that username can access it (`600`).

If you try to login with SSH with that username now, it will fail with permission denied. This is because OpenStack when building the head node, configured the SSH server to only accept SSH logins from the `cloud` user. Run `id cloud` to get the primary group of the `cloud` user. Edit `/etc/ssh/sshd_config` and find the

line `AllowUsers cloud` and delete that line. Then add the line `AllowGroups CLOUDGROUP YOURGROUP` where `CLOUDGROUP` is the name of the primary group of the `cloud` username (you don't want to lock yourself out) and `YOURGROUP` is the name of the group you created earlier. Run `sshd -t` to check that the SSH server configuration is valid. Without exiting your current SSH session, restart the SSH server with `systemctl restart sshd`

Check that you can still login to the node with the `cloud` username using a different SSH session (do not close your current one since you will need it to fix the config if you locked yourself out). Check that you can login with your new username.

Now rebuild the node overlays so that the `syncuser` overlay pulls the new group and username into the node's runtime overlay.

Task 5: Set a root password for your compute nodes (15 min)

In real clusters, sometimes a node loses its network connections or has some other problem that prevents login via SSH but the admins still need to login to it to look at the logs and diagnose the problem. Usually this is done by hooking a keyboard and monitor to the node in person or over the BMC. But for this to work, there has to be a local user the admins can login as. Usually, this is done with the `root` username. This requires a strong `root` password to be set on the machines that the cluster's users or any attackers have no hope of guessing or brute forcing.

Create a strong root password for your compute nodes and write it down some place safe. Now, you will create the salted hash of the password so you can put it in the node images.

On the head node, run `man 5 crypt` and look for the available hashing methods section to see the hashing methods the head node supports. Note, if your head node and compute nodes are running different linux distributions or version of them, you would need to check both and only consider hashing methods both support. There are two groups of hashing methods. Ones using traditional hash algorithms like `sha256crypt` and `sha512crypt` that are memory efficient and CPU time cost parameter is linear (number of rounds). Ones using dedicated password hash algorithms like `yescrypt` and `scrypt` that are meant to take a while on the CPU and cost a lot of RAM, typically having a logarithmic CPU time cost parameter. The former can be made to take a long time, but is easy to parallelize with GPUs or custom silicon. The latter is very costly to parallelize due to the needed RAM, which makes it expensive to brute force.

Try making hashed passwords for a method in each group for some trivial password (e.g. "ab") and steadily increase the CPU cost parameter by

```
mkpasswd --method=METHOD --rounds=COMPLEXITY
```

Adjust the `COMLEXITY` till it takes a few seconds or maxes out for the hash method. Once you have picked out the hash method and complexity, then either on your own machine if you have `mkpasswd` or on the head node, hash the root password that you chose. Then inside the Warewulf image for your compute nodes, set the `root` password either by copy-pasting the hash into the respective line in `/etc/shadow` (2nd field) or with

```
usermod -p 'PASSWORD' USERNAME
```

Reboot the compute nodes. After one has rebooted, login to it from the head node using the username you created earlier. Then, become root with

```
su
```

and entering the `root` password you set. You should now be `root`.