Linux - Users & Ownership

Overview

User commands in Ubuntu linux.

Creating a User

First add a new user using:

sudo useradd <username>

Then give the user a password using and following the prompts:

sudo passwd <username>

The password command adds the user password to the /etc/passwd file. The file contains a list of users, each with seven fields. For example, for the user 'noreply' the entry would be:

noreply:x:1010:1010::/home/noreply:/bin/sh

The fields are:

* Username
* Password - stored in /etc/shadow in encrypted format
* User ID - UID > 999 (1-99 are predefined accounts, 100-999 are groups)
* Group ID - GID related to user groups based in /etc/group
* User info - extra information for user such as full name
* Home directory
* Shell

There are various flags which can be used to adjust the fields with the useradd command:

* -d <custom-home-directory>
* -u <custom-UID>
* -g <custom-GID>
* -G <comma-seperated-groups>

Check User Details

Use the 'id' command to check user details and joined groups:

id <username>

Create a New Group

Groups can be used to control almost everything about a user, with one of the most important aspects being a specific users access to files through permissions. Create a group using the groupadd command:

sudo groupadd <groupname>

View Groups

All groups along with their members and ID are stored in /etc/group, view groups using:

sudo vim /etc/group

Add a User to a Group

Once a user is made, if there are further groups to add to the user, the usermod append command can be used:

sudo usermod -a -G <group> <username>

Change a Users Primary Group

A user always has a primary group, which is the group which will be assinged to the files and folders the user creates. Change the primary group using the -g flag:

sudo usermod -g <group> <username>

Preventing User Shell Login

For security it is important to remove shell login for some users incase accounts have their credentails leaked.

First change the user shell to nologin in the usersettings:

sudo usermod -s /usr/sbin/nologin <username>

Then ensure the ssh daemon only allows login for the specific users required:

sudo vim /etc/ssh/sshd\_config

AllowUsers <allowed-user1> <allowed-user-2>

PasswordAuthentication no

# Sudo

The super users (sudoers) group allows users to perform commands as another user, without being logged, commonly used to allow basic users to execute commands reserved for the root user. To perform a command as root a sudoer prepends sudo to the command they are running. Having the sudoers group is particularly useful, since it means there is no need to allow direct root login, as any admins can simply have sudoer access.

Add user to sudoers

Since sudo is just another group, simply add the user to the group:

sudo usermod -a -G sudo <username>

Allow specific commands

While many commands on linux require root privildge, sometime a user might only need to to a couple tasks. The /etc/sudoers file can be used to allow specific sudo only tasks without a password for users.

visudo is exclusively used to edit /etc/sudoers, so to edit the file first login to root:

sudo -i

visudo

Then define which user and commands should be passwordless, for example:

fred ALL = NOPASSWD:/sbin/reboot

/etc/sudoers syntax

The syntax of the sudoers file is:

<user> <places>=(<as-user>) [NOPASSWD:] <command>

* user - the users who the item applies to
* places - the places where the sudo command can be lanched
* as user - the users which the the first users can act as
* NOPASSWD - optional flag to define that no password is required to run the command
* command - the commands which can be ran using the sudo command

The first line in the sudoers file is normally:

root ALL=(ALL) ALL

which says, the root users can execute all commands as all users from all places.

# umask

umask is the user file-creation mode mask and is used to determine the file permission for newly created files. umask is setup in the shell profile for all users, edit using:

vim /etc/profile or vim /etc/bashrc

umask 022

Common umasks

umask 002:

* default normal user
* directories 775
* files 664

umask 022:

* default for root user
* directories 755
* files 644

umask 007:

* exclude users who are outside group
* 770 permissions

umask 077:

* private system, no other users can read or write
* 700 permissions

Calculate umask:

* 0 : read, write and execute
* 1 : read and write
* 2 : read and execute
* 3 : read only
* 4 : write and execute
* 5 : write only
* 6 : execute only
* 7 : no permissions

Syntax:

<owner><group><others>

File base permissions are 666, directory base permissions are 777. Therefore, subtract the umask from the base permissions to get the resulting permission.

Access Control Lists

Since traditional permissions only allow tight control via group and others, it is not possible to allow two groups to access a file without allowing all to access the file. For example, the accounting group may own a file, but sales might need access. But adding sale users to the account group may give them access to files which are not meant for their viewing.

Access Control Lists (ACL)s are used to set advanced permission controls for files and directories.

Install acl on linux using apt:

sudo apt install acl

Check the ACL on a directory or file using the acl command:

getfacl <path-to-file-or-directory>

Modify the ACL using setfacl and the following options:

* -m - modify
* -x - remove
* -d - default

followed by user or group flag, its name, and its permissions. For example:

setfacl -m <user-or-group>:<name>:<permissions> <path>

setfacl -m g:accounting:rwx /accounting

Note base permissions still apply, so if removing permissions for a group etc, ensure that the base permissions for the file or directory do not still allow access.