Responsible use of sudo

Previously, you explored authorization, authentication, and Linux commands with *sudo*, *useradd*, and *userdel*. The *sudo* command is important for security analysts because it allows users to have elevated permissions without risking the system by running commands as the root user. You'll continue exploring authorization, authentication, and Linux commands in this reading and learn two more commands that can be used with *sudo*: *usermod* and *chown*.

Responsible use of sudo

To manage authorization and authentication, you need to be a **root user,** or a user with elevated privileges to modify the system. The root user can also be called the "super user." You become a root user by logging in as the root user. However, running commands as the root user is not recommended in Linux because it can create security risks if malicious actors compromise that account. It's also easy to make irreversible mistakes, and the system can't track who ran a command. For these reasons, rather than logging in as the root user, it's recommended you use *sudo* in Linux when you need elevated privileges.

The *sudo* command temporarily grants elevated permissions to specific users. The name of this command comes from "super user do." Users must be given access in a configuration file to use *sudo*. This file is called the "sudoers file." Although using *sudo* is preferable to logging in as the root user, it's important to be aware that users with the elevated permissions to use *sudo* might be more at risk in the event of an attack.

You can compare this to a hotel with a master key. The master key can be used to access any room in the hotel. There are some workers at the hotel who need this key to perform their work. For example, to clean all the rooms, the janitor would scan their ID badge and then use this master key. However, if someone outside the hotel's network gained access to the janitor's ID badge and master key, they could access any room in the hotel. In this example, the janitor with the master key represents a user using *sudo* for elevated privileges. Because of the dangers of *sudo*, only users who really need to use it should have these permissions.

Additionally, even if you need access to *sudo*, you should be careful about using it with only the commands you need and nothing more. Running commands with *sudo* allows users to bypass the typical security controls that are in place to prevent elevated access to an attacker.

Note: Be aware of *sudo* if copying commands from an online source. It's important you don't use *sudo* accidentally.

Authentication and authorization with sudo

You can use *sudo* with many authentication and authorization management tasks. As a reminder, **authentication** is the process of verifying who someone is, and **authorization** is the concept of granting access to specific resources in a system. Some of the key commands used for these tasks include the following:

useradd

The *useradd* command adds a user to the system. To add a user with the username of *fgarcia* with *sudo*, enter *sudo useradd fgarcia*. There are additional options you can use with *useradd*:

- *-g*: Sets the user's default group, also called their primary group
- *-G*: Adds the user to additional groups, also called supplemental or secondary groups

To use the *-g* option, the primary group must be specified after *-g*. For example, entering *sudo useradd -g security fgarcia* adds *fgarcia* as a new user and assigns their primary group to be *security*.

To use the *-G* option, the supplemental group must be passed into the command after *-G*. You can add more than one supplemental group at a time with the *-G* option. Entering *sudo useradd -G finance,admin fgarcia* adds *fgarcia* as a new user and adds them to the existing *finance* and *admin* groups.

usermod

The *usermod* command modifies existing user accounts. The same *-g* and *-G* options from the *useradd* command can be used with *usermod* if a user already exists.

To change the primary group of an existing user, you need the *-g* option. For example, entering *sudo usermod -g executive fgarcia* would change *fgarcia*'s primary group to the *executive* group.

To add a supplemental group for an existing user, you need the *-G* option. You also need a *-a* option, which appends the user to an existing group and is only used with the *-G* option. For example, entering *sudo usermod -a -G marketing fgarcia* would add the existing *fgarcia* user to the supplemental *marketing* group.

Note: When changing the supplemental group of an existing user, if you don't include the *-a* option, *-G* will replace any existing supplemental groups with the groups specified after *usermod*.

Using *-a* with *-G* ensures that the new groups are added but existing groups are not replaced.

There are other options you can use with *usermod* to specify how you want to modify the user, including:

- *-d*: Changes the user’s home directory.
- *-l*: Changes the user’s login name.
- *-L*: Locks the account so the user can’t log in.

The option always goes after the *usermod* command. For example, to change *fgarcia*’s home directory to */home/garcia_f*, enter *sudo usermod -d /home/garcia_f fgarcia*. The option *-d* directly follows the command *usermod* before the other two needed arguments.

userdel

The *userdel* command deletes a user from the system. For example, entering *sudo userdel fgarcia* deletes *fgarcia* as a user. Be careful before you delete a user using this command.

The *userdel* command doesn’t delete the files in the user’s home directory unless you use the *-r* option. Entering *sudo userdel -r fgarcia* would delete *fgarcia* as a user and delete all files in their home directory. Before deleting any user files, you should ensure you have backups in case you need them later.

Note: Instead of deleting the user, you could consider deactivating their account with *usermod -L*. This prevents the user from logging in while still giving you access to their account and associated permissions. For example, if a user left an organization, this option would allow you to identify which files they have ownership over, so you could move this ownership to other users.

chown

The *chown* command changes ownership of a file or directory. You can use *chown* to change user or group ownership. To change the user owner of the *access.txt* file to *fgarcia*, enter *sudo chown fgarcia access.txt*. To change the group owner of *access.txt* to *security*, enter *sudo chown :security access.txt*. You must enter a colon (*:*) before *security* to designate it as a group name.

Similar to *useradd*, *usermod*, and *userdel*, there are additional options that can be used with *chown*
.

Key takeaways

Authentication is the process of a user verifying their identity, and authorization is the process of determining what they have access to. You can use the *sudo* command to temporarily run commands with elevated privileges to complete authentication and authorization management tasks. Specifically, *useradd*, *userdel, usermod*, and *chown* can be used to manage users and file ownership.

Revision #1
Created 7 July 2023 11:45:20 by naruzkurai
Updated 7 July 2023 11:45:28 by naruzkurai