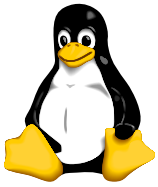

LPIC 102

Administration, the `root` User and Configuring `sudo`

1 Aim

The aim of this activity is to understand the purpose of the `root` account, and the security risks of logging in as `root`, and how to avoid them. You will also learn how to create user accounts using `useradd`. The student will appreciate that system administration with `sudo` is a useful help to security.

2 Background

You normally log into the computer system as your own user account. There are only two places where you can create or edit files on the computer system:

- in your *home directory*;
- in the `/tmp` directory.

Everywhere else is read only, or offers you no access.

However, to perform system administration, you sometimes need to change files outside of your home directory, and outside of the `/tmp` directory. How can you perform system administration? You need to temporarily become the `root` user. The `root` user account is sometimes called the *superuser*, since this account allows access to everything. This power is too great for most activities; with a simple typing mistake, you can delete the entire hard disk. Viruses and worms running under the `root` account can damage anywhere at all, and have access to everything on the computer. Buggy software running as the `root` user can do unlimited damage. Running all software as the `root` user is an accident waiting to happen. It is like using a chainsaw to cut a loaf of bread.

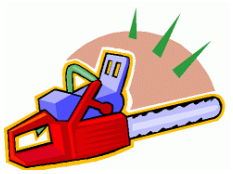

Figure 1: Doing everything as `root` is like cutting bread with a chainsaw.

There are two tools with which you should be familiar: the program `su`, and more importantly, `sudo`. Here we see how to use them.

3 Procedure

3.1 Make a User Account for Yourself

We use the program `useradd` to create a user account. Learn about this using

```
$ useradd --help
Usage: useradd [-u uid [-o]] [-g group] [-G group,...]
```

```
            [-d home] [-s shell] [-c comment] [-m [-k template]]
            [-f inactive] [-e expire]
            [-p passwd] [-M] [-n] [-r] [-l] name
    useradd -D [-g group] [-b base] [-s shell]
            [-f inactive] [-e expire]
```

The man page gives the full documentation:

```
$ man useradd
```

We assume here that you do not already have a non-root user account

1. Log in as root (the top secret password is `square`)
2. Choose a username for yourself. It should be easy to type (suggest make it all lower case), and contain no spaces. I would use just letters myself. Let's say your username is "`jim`" and your name is Jim Gettys.
3. Choose a password for yourself. It should be a "good" password.
   - **(a)** Easy for you to remember
   - **(b)** Hard for anyone else to guess

   How can you achieve these goals?

   - **(a)** Make up a sentence that has some personal meaning to you, but which no one else knows. Take one letter from each word. Add some digits and punctuation.
   - **(b)** Or just use the program `mkpasswd`, write the result on a small piece of paper and put it in your wallet or purse.
   - **(c)** Don't just change 'e' to '3' and 'l' to '1', since various password cracking programs use such simple transformations on their dictionary checking of passwords
   - **(d)** Bad passwords are one of the greatest vulnerabiilities in a computer system. Develop good habits yourself and encourage your users to do likewise.
4. Now create a user account for yourself:

   ```
   # useradd -c "Jim Gettys" jim
   ```

   Now you have an account that is in both the `/etc/passwd` and the `/etc/shadow` files, and a group has been created or you have been added to a group in `/etc/group`. But your account has no password yet, so:
5. Give yourself the password you arrived at by careful deliberation above. Let's say your password is `>Mk2sxEq7`:

   ```
   # passwd jim
   Changing password for user jim.
   New password:
   Retype new password:
   passwd: all authentication tokens updated successfully.
   ```
6. Have a look at the result of your handiwork:

```
# grep jim /etc/passwd /etc/shadow /etc/group
/etc/passwd:jim:x:500:501:Jim Gettys:/home/jim:/bin/bash
/etc/shadow:jim:$1$QJJFgTHm$cmEB/gLK1uWbxZeqtB0wT0:13013:0:99999:7:::
/etc/group:jim:x:501 :
```

3.2 Using the su Program to Become root

1. Log into your Linux system with your own user account, ***not*** as root.
2. In a terminal window, type:

   ```
   $ su -
   ```

   Do not type the $; that indicates the prompt, and that you are typing this as a normal user. When you log in as root, the prompt becomes a hash: #.

   The minus sign '-' tells su to run the login scripts of the user you are changing to (here, it is the root user).
3. When prompted, type in your root password (note, this is different from sudo).
4. Notice that your prompt has changed to a hash: #.
5. To exit (log out) from the root account, type:

   ```
   # exit
   ```

3.3 Advantages and Disadvantages of Using the su Program

There is quite a lot of convenience in opening one window, suing to root, and leaving that window open. However, when you go to get that cup of tea, someone else may come and take advantage of this. The person who uses su must also know the root password. The more people who know a secret, the less of a secret it is. It is more secure to use the program sudo, which we discuss now.

3.4 The sudo Program

The sudo program allows a senior system administrator to keep the root password to themselves, and to delegate responsibility for various tasks to other junior system administators.

There are some advantages to doing things this way, including the fact that every command executed using sudo is recorded in the system logs. This can help administrators coordinate their efforts; they can see what the other has done. Also, a cracker who breaks into the system and who gets access to the root account will be unlikely to use sudo, and the break-in will be clearly visible in the system logs (unless the person is smart enough to cover their tracks). Probably the most important thing is that the administrator does not need to know or remember the root password, thus making it easier to keep secret, and allowing it to be changed more often.

3.5 Configuring `sudo`

1. Become `root` using the `su` program, as described above in section 3.2 on the preceding page.

2. Type:

   ```
   # visudo
   ```

3. Refer to the chapter about `vi` in the Linux Workshop notes for guidance on using the `vi` text editor. You will find the '`o`' and '`i`' commands useful.

4. Or, if you prefer `emacs` to `vi` like I do, then you could type:

   ```
   # EDITOR=emacs visudo
   ```

5. Edit the file so that it looks like this, taking care to type it accurately, but put your username instead of mine:

   ```
   # sudoers file.
   #
   # This file MUST be edited with the 'visudo' command as root.
   #
   # See the sudoers man page for the details on how to write a sudoers file.
   #

   Defaults timestamp_timeout = 10

   # Host alias specification

   # User alias specification

   # Cmnd alias specification
   # User privilege specification
   root    ALL=(ALL) ALL
   nicku   ALL=(ALL) ALL
   ```

   Note that *two* lines were added, nothing else is changed. But please use *your own* user name here, not `nicku`!

   See `man sudoers` and search for `Defaults`, and also `timeout` for information about the first line. It changes the time that may elapse since you last used `sudo` before you need to enter your password again. The default is normally five minutes.

6. Save it and exit by typing (in `vi`):

   Esc `:wq`

   and `sudo` is now configured. Here we have assigned full priveleges to the user, but `sudo` can be set up to assign restricted administration rights to junior administrators. You can learn more by reading the `man` (manual) pages for `sudoers`, `sudo` and `visudo`.

7. Log out from being `root` by typing:

   ```
   # exit
   ```

   Your prompt should end with a dollar; if not, then you have run `su -` more than once. Continue typing `exit` until your prompt ends with a dollar: `$`.

3.6 Using Sudo

3.6.1 Some Background

To use `sudo`,

1. you type `sudo` in front of the command you want to execute as the `root` user.
2. The first time you do this, you will see a warning like this:

   ```
   We trust you have received the usual lecture from the local System
   Administrator. It usually boils down to these two things:

       #1) Respect the privacy of others.
       #2) Think before you type.

   Password:
   ```

3. At that prompt, you will type ***your own password***, *not* that of the `root` user.
4. If you use `sudo` within the next ten minutes (normally five) within the same terminal window, you will not need to type your password.
5. If you walk away from your computer at all, you can cause the ten minute period to expire immediately by typing:

   ```
   $ sudo -k
   ```

   If `sudo` is used immediately after this, a passsword will be required.

6. Before running a graphical program with `sudo` for the first time in one login session, on older systems, you may need to type (as your own self):

   ```
   $ xhost +localhost
   ```

   to allow any user who is logged into your machine, to display graphics on the X server (the network graphics system on which you run your graphical user interface). This includes the user `root`, who may otherwise be barred from display while you are running X from your account.

3.6.2 Exercise using `sudo`

1. Try to display the special *log* file `/var/log/secure`:

   ```
   $ cat /var/log/secure
   cat: /var/log/secure: Permission denied
   ```

   Hmm, it says permission is denied. Only system administrators can see this log file. Let's use `sudo`:

   ```
   $ sudo cat /var/log/secure
   ```

   Make sure you enter *your own* password, and not that of `root`.

3.7 Why Use `sudo`?

There are four main reasons for using `sudo`:

- Only one person needs to know the root password. A secret shared between ten system administrators is no longer a secret.
- You only run as root the commands that you need to run as root. This increases security significantly. The less you do at the highest privelege level, the better.
- A senior system administrator may *delegate* only some duties to others (i.e., backup, printing administration, ...), since `sudo` allows the senior administrator to allow priveleged access to only the required commands.
- Every command executed using `sudo` is recorded in a *log file*. On Red Hat 9, this is `/var/log/security`. For each command executed using `sudo`, the following details are recorded:
  - The time the command was executed
  - The user who executed the command
  - What user the command was effectively executed as (usually root, but you can change that with the `-u` ⟨*user*⟩ option to `sudo` — see `man sudo`).
  - What the current directory was that the command was executed in
  - What terminal was used (i.e., was the user logged in locally, or over the network?)
  - The exact command, with its exact location in the file system.

  This log allows the system administrator to go back and find out what was done when by who. If the system stops working properly, the logs can provide information about what was done at the time, and if a mistake was made, it can be identified and rectified.

3.8 Warning!

Please get into the habit of using `sudo` for system administration tasks. You will be a better system administrator for this, and your system will be less easy to crack.

Note that at your workplace, you should ensure that the number of people who can use `sudo` on the servers you are responsible for is limited as much as possible.