Managing Users and Groups

Subrahmanya Bhat B
Dep MCA,Srinivas University
Mangaluru.

users

- Users can be either people or accounts which exist for specific applications to use

groups

- Groups are logical expressions of organization, tying users together for a common purpose.

- Users within a group share the same permissions to read, write, or execute files owned by that group

- Each user is associated with a unique numerical identification number called a user ID (UID), and each group is associated with a group ID (GID)

- A user who creates a file is also the owner and group owner of that file.
- The file owner can be changed only by root
- Access permissions can be changed by both the root user and file owner.

User Private Groups

- A user private group is created whenever a new user is added to the system.

- It has the same name as the user and that user is the only member of the user private group

Shadow Passords

- To enhance the security of system authentication files Shadow Passwords concept has been introduced

Advantages in shadow passwords

- Shadow passwords improve system security by moving encrypted password hashes from the world-readable ***/etc/passwd*** file to ***/etc/shadow***, which is readable only by the root user

Advantages in shadow passwords

- Shadow passwords store information about password aging.

- Shadow passwords allow the **/etc/login.defs** file to enforce security policies

Adding a New User

- To add a new user to the system, type the following at a shell prompt as root:

useradd [options] username

Utilities for managing users & groups

- useradd, usermod, userdel
- groupadd, groupmod, groupdel
- gpasswd
- pwck, grpck
- pwconv, pwunconv
- grpconv, grpunconv

useradd, usermod, userdel

- Standard utilities for adding, modifying, and deleting user accounts.

groupadd, groupmod, groupdel

- Standard utilities for adding, modifying, and deleting groups.

gpasswd

- Standard utility for administering the **/etc/group** configuration file

pwck, grpck

- Utilities that can be used for verification of the password, group, and associated shadow files

pwconv, pwunconv

- Utilities that can be used for the conversion of passwords to shadow passwords, or back from shadow passwords to standard passwords.

grpconv, grpunconv

- Similar to the previous, these utilities can be used for conversion of shadowed information for group accounts

- By default, the ***useradd*** command creates a locked user account.
- To unlock the account, run the following command as root and assign a password:

passwd username

Ex

useradd juan

A new line for juan is created in **/etc/passwd**

juan:x:1001:1001::/home/juan:/bin/bash

A new line for juan is created in **/etc/shadow**

Juan:!!:14798:0:99999:7:::

A new line for a group named juan is created in **/etc/group:**

juan:x:1001:

Ex

A new line for a group named juan is created in **/etc/gshadow**

Juan:!::

A directory for user juan is created in the **/home** directory as **juan**

ls -ld /home/juan

drwx------. 4 juan juan 4096 Mar3 18:23 /home/juan

- At this point, a locked account called **juan** exists on the system.

- To activate it, the administrator must next assign a password using ***passwd***

Adding a New Group

- To add a new group to the system, type the following at a shell prompt as root:

groupadd [options] group_name

Creating Group Directories

- The **setgid** bit makes managing group projects that share a common directory simple
- Any files a user creates within the directory are owned by the group that owns the directory.

- For example, a group of people need to work on files in the **/opt/myproject/** directory.

- As root create directory,

mkdir **/opt/myproject**

- Add the **myproject** group in system,

groupadd **myproject**

- Associate the contents of the **/opt/myproject/** directory with the **myproject** group:

***chown* root:myproject /opt/myproject**

- Allow users in the group to create files within the directory and set the setgid bit:

chmod 2775 /opt/myproject

- To verify that the permissions have been set correctly, run the following command:

ls -ld /opt/myproject

- O/p will be

drwxrwsr-x. 3 root myproject 4096 Mar 3 18:31 /opt/myproject

- At this point, all members of the **myproject** group can create and edit files in the

/opt/myproject/ directory without the administrator having to change file permissions

every time users creates new files.

- Add users to the **myproject** group:

***usermod -aG* myproject username**

Gaining Privileges

- When a user executes the ***su*** command, they are prompted for the root password and, after

authentication, are given a **root** shell prompt.

- Since this program is powerful, administrators within an organization may want to limit who has access to the command.

- One of the simplest ways to do this is to add users to the special administrative group called wheel.
- To do this, type the following command as **root**

usermod -a -G wheel **username**

- After you add the desired users to the wheel group, it is advisable to only allow these specific users to use the ***su*** command
- To do this, edit the Pluggable Authentication Module (PAM) configuration file for ***su,*** which is in **/etc/pam.d/su.**

- Open this file in a text editor and uncomment the following line by removing the # character

#auth required pam_wheel.so use_uid

- This change means that only members of the administrative group **wheel** can switch to another user using the **su** command.

The ***sudo*** Command

- The ***sudo*** command offers another approach to giving users administrative access
- When trusted users precede an administrative command with ***sudo***, they are prompted for their own password

- When they have been authenticated and assuming that the command is permitted, the

administrative command is executed as if they were the **root** user

- The basic format of the **sudo** command is as follows:

sudo command

- ***Ex. sudo mount***

- Only users listed in the **/etc/sudoers** configuration file are allowed to use the ***sudo*** command
- The command is executed in the **user's** shell, not a **root** shell

- Each successful authentication using the ***sudo*** command is logged to the file

/var/log/messages

- The command issued along with the issuer's user name is logged to the file

/var/log/secure.

- If additional logging is required, use the **pam_tty_audit** module to enable TTY auditing for specified users by adding the following line to file **/etc/pam.d/system-auth**

session required pam_tty_audit.so disable=pattern enable=pattern

- Here pattern represents a comma-separated listing of users

- Ex

session required pam_tty_audit.so disable=* enable=root

- Another advantage of the **sudo** command is that an administrator can allow different users access to specific commands based on their needs.

- Administrators wanting to edit the **sudo** configuration file, ***/etc/sudoers***, should use the ***visudo*** command

session required pam_tty_audit.so disable=pattern enable=pattern

- The **root** user is part of the wheel group by default.

- To give someone full administrative privileges, type visudo and add a line similar to the following in the user privilege specification section:

juan ALL=(ALL) ALL

- This example states that the user, juan, can use sudo from any host and execute any command.

- The example below illustrates the granularity possible when configuring sudo :

%users localhost=/usr/sbin/shutdown -h now

- example states that any member of the users can issue the command /sbin/shutdown -h now as long as it is issued from the console

- There are several potential risks to keep in mind when using the sudo command.

- You can avoid them by editing the **/etc/sudoers** configuration file using visudo

- Leaving the **/etc/sudoers** file in its default state gives every user in the **wheel** group unlimited root access

- By default, sudo stores the sudoer's password for a five minute timeout period.

- Any subsequent uses of the command during this period will not prompt the user for a password

- This could be exploited by an attacker if the user leaves his workstation unattended and unlocked while still being logged in.

- This behavior can be changed by adding the following line to the **/etc/sudoers** file:

Defaults timestamp_timeout=value

- value is the desired timeout length in minutes.

- Setting the value to 0 causes sudo to require a password every time.

- If a sudoer's account is compromised, an attacker can use sudo to open a new shell with administrative privileges:

```
sudo /bin/bash
```

- Opening a new shell as root in this fashion gives the attacker administrative access for unlimited amount of time, bypassing the timeout period specified in the **/etc/sudoers** file
- never requiring the attacker to input a password for sudo again until the newly opened session is closed.