Previous Chapter - Table of Contents - Next Chapter

Chapter 30 User Account Management - Notes

30.3 Learning Objectives:

- Explain the purpose of individual user accounts and list their main attributes.
- Create new user accounts and modify existing account properties, as w ell as remove of lock accounts.
- Understand how user passw ords are set, encrypted, and stored, and how to require changes in passw ords over time for security purposes.
- Explain how restricted shells and restricted accounts w ork.
- Understand the role of the root account and w hen to use it.
- Use Secure Shell (**ssh**) and remove logins and commands.

30.4 User Accounts

Linux systems provide **multi-user** environment w hich permits people and processes to have separate simultaneous w ork environments.

Purposes of having individual user accounts include:

- Providing each user w ith their ow n individualized private space
- Creating particular user accounts for specific dedicated purposes
- Distinguishing privileges among users

One special user account is for the **root** user, w ho is able to do ***anything*** on the system. To avoid making costly mistakes, and for security reasons, the root account should only be used w hen absolutely necessary.

Normal user accounts are for people w ho w ill w ork on the system. Some user accounts (like the **daemon** account) exist for the purpose of allow ing processes to run as a user other than root.

In next chapter, w ill continue w ith discussion of **group** management, w here subsets of the users on system can share files, privileges, etc., according to common interests.

30.5 Attributes of a User Account

Each user on the system has a corresponding line in the `etc/passwd` file that describes their basic account attributes. (Will talk about passw ords, as w ell as this file, later). For example:

```
....
beav□1000:1000:Theodore Cleaver:/home/beav:/bin/bash
warden□1001:1001:Ward Cleaver:/home/warden:/bin/bash
dobie□1002:1002:Dobie Gillis:/home/dobie:/bin/bash
....
```

The seven elements here are:

- User name: unique name assigned to each user
- User passw ord: passw ord assigned to each user
- User identification number (**UID**): unique number assigned to user account. UID is used by system for variety of purposes,

including determination of user privileges and activity tracking
- Group identification number (**GID**): indicates primary, principal, or default **group** of user
- Comment or GECOS information: defined method to use comment field for contact information (full name, email, office, contact number) (No need to w orry about meaning of **GECOS**, very old term)
- Home directory: for most users, unique directory that offers w orking are for user. Normally, this directory ow ned by user, and except for **root** w ill be found on the system somew here under `/home`
- Login shell: normally, shell program such as `/bin/bash` or `/bin/csh`. Sometimes, how ever, alternative program referenced here for special cases. In general, this field w ill accept any executable

30.6 Creating User Accounts with useradd

The command:

```
$ sudo useradd dexter
```

w ill create an account for user `dexter`, using default algorithms for assigning user and group id, home directory, and shell choice.

Specifically, the **useradd** command above causes follow ing steps to execute:

- Next available UID greater than `UID_MIN` (specified in `/etc/login.defs`) by default assigned as `dexter`\'s `UID`
- A group called `dexter` w ith a `GID=UID` also created and assigned as `dexter`\'s primary group
- A home directory `/home/dexter` created and ow ned by `dexter`
- `dexter`\'s login shell w ill be `/bin/bash`
- Contents of `/etc/skel` copied to `/home/dexter`. By default, `/etc/skel` includes startup files for **bash** and for the X Window system
- Entry of either `!!` placed in passw ord field of `/etc/shadow` file for `dexter`\'s entry, thus requiring administrator to assign a passw ord for the account to be usable

Defaults can be easily overruled by using options to **useradd**:

```
$ sudo useradd -s /bin/csh -m -k /etc/skel -c "Bullwinkle J Moose" bmoose
```

w here explicit non-default values have been given for some of the user attributes.

30.7 Modifying and Deleting User Accounts

Root user can remove user accounts using **userdel**:

```
$ sudo userdel morgan
```

All references to user `morgan` w ill be erased from `/etc/passwd`, `/etc/shadow`, and `/etc/group`.

While this removes account, does not delete the home directory (usually `/home/morgan`) in case the account may be re-established later. If `-r` option given to **userdel**, home directory w ill also be obliterated. How ever, all other files on system ow ned by removed user w ill remain.

usermod can be used to change characteristics of a user account, such as group memberships, home directory, login name, passw ord, default shell, user id, etc. For example, the command

```
$ sudo usermod -L dexter
```

locks the account for `dexter`, so he cannot login.

Usage pretty straightforw ard. Note: **usermod** w ill take care of any modifications to files in `/etc` directory as necessary.

usermod

30.8 Locked Accounts

Linus ships w ith some system accounts that are **locked** (such as `bin`, `daemon`, `sys`), w hich means they can run programs, but can never login tot he system and have no valid passow ord associated w ith them. For example, `etc/passwd` has entries like:

```
bin□1:1:bin:/bin:/sbin/nologin
daemon□2:2:daemon:/sbin:/sbin/nologin
```

The `nologin` shell returns the follow ing if a locked user tries to login to the system:

```
This account is currently not available.
```

or w hatever message may be stored in `/et/nologin.txt`.

Such locked accounts are created for special purposes, either by system services or applications; if you scan `/etc/passwd` for users w ith the `nologin` shell, you can see w ho they are on your system.

Also possible to lock account of particular user:

```
$ sudo usermod -L dexter
```

w hich means the accounts stays on the system but logging in is impossible. Unlocking can be done w ith `-U` option.

Customary practice: lock user\'s accounts w henever they leave the organization or is on an extended leave of absence.

Another w ay to lock an account: use **chage** to change expiration date of account to date in the past:

```
$ sudo chage -E 2014-09-11 morgan
```

Actual date irrelevant as long as it is in the past. Will discuss **chage** shortly.

Another approach is to edit `etc/shadow` file and replace user\'s hashed passw ord w ith `!!` or some other invalid string.

30.9 User IDs and /etc/passwd

Have already seen how `etc/passwd` contaisn one record (one line) for each user on system:

```
beav□1000:1000:Theodore Cleaver:/home/beav:/bin/bash
rsquirrel□1001:1001:Rocket J Squirrel:/home/rsquirrel:/bin/bash
```

and have already discussed the fields in here. Each record consists of number of fields separated by colons:

- `username` - the user\'s unique name
- `password` - either the hashed passw ord (if `/etc/shadow` is not used) or a placeholder ("x" w hen `/etc/shadow` is used)
- `UID` - user identification number
- `GID` - primary group identification number for the user
- `comment` - comment area, usually the user\'s real name
- `home` - directory pathname for the user\'s home directory
- `shell` - absolutely qualified name of the shell to invoke at login

If `/etc/shadow` is not used, passw ord field contains hashed passw ord. If used, contains a placeholder ("**x**").

The convention most Linux distributions have used is that any account w ith a user ID less than `1000` is considered special and belongs to the system; normal user accounts start at `1000`. Actual value defined as `UID-MIN` and is defined in `/etc/login.defs`.

If User ID if not specified w hen using **useradd**, system w ill incrementally assign UIDS starting at `UID_MIN`

Additionally, each user gets a **Primary Group ID** w hick, by default, is the same number as the UID. These are sometimes called **User Private Groups (UPG)**.

Bad practice to edit `/etc/passwd`, `/etc/group`, `/etc/shadow` directly. Either user appropriate utilities such as **usermod**, or use **vipw** special editor to do so as it is careful about file locking, data corruption, etc.

30.10 Why Use /etc/shadow?

Use of `/etc/shadow` enables passw ord aging on a per user basis. At same time, also allow s for maintaining greater security of hashed passw ords.

Default permissions of `/etc/passwd` are **644** (`-rw-r--r--`); anyone can read the file. Unfortunately necessary because system programs and user applications need to read information contained in file. These system programs do not run as user root; in any event, only root may change the file.

Of particular concern: hashed passw ords themselves. If appear in `/etc/passwd`, anyone may make copy of hashed passw ords and then make use of utilities such as **Crack** and **John the Ripper** to guess original cleartext passw ords given hashed passw ord. Huge security risk!

`etc/shadow` has permission settings of `400` (`-r--------`), means that only root can access this file. Makes it more difficult for someone to collect hashed passw ords.

Unless compelling good reason not to, should use `/etc/shadow` file.

30.11 /etc/shadow

`/etc/shadow` contains one record (one line) for each user:

```
daemon:*:16141:0:99999:7:::
.....
beav:$6$iCZyCnBJH9rmq7P.$RYNm10Jg3wrhAtUnahBZ/mTMg.RzQE6iBXyqaXHvxxbKTYqj.d9wpoQFuRp7fPEE3hMK3W2gcIYhiXa9MIA9w1:16316:0:99
```

Colon-separated fields:

- `username` : unique user name

- `password` : hashed (**sha512**) value of the passw ord
- `lastchange` : days since Jan 1, 1970 that passw ord w as last changed
- `mindays` : minimum days before passw ord can be changed
- `maxdays` : maximum days after w hich passw ord must be changed
- `warn` : days before passw ord expires that user is w arned
- `grace` : days after passw ord expires that account is disabled
- `expire` : date that account is/w ill be disabled
- `reserved` : reserved field

Username in each record must match *exactly* that found in `etc/passwd` , and also must appear in identical order.

All dates stored as number of days since Jan. 1 1970 (**epoch** date).

Passw ord hash: string "**6**" follow ed by eight character salt value, w hich is then follow ed by **$** and an **88** character (**sha512**) passw ord hash.

30.12 Password Management

Passow rds can be changed w ith **passwd**; a normal user can change only their ow n passw ord, w hile root can change any user passw ord. When you type your passw ord, not show n; echoing back to screen suppressed.

By default, passw ord choice examined by `pam_cracklib.so` , w hich furthers making good passw ord choices.

Normal user changing passw ord:

```
$ passwd
Changing password for clyde
(current) UNIX password: <clyde\'s password>
New UNIX password: <clyde\'s-new-password>
Retype new UNIX password: <clyde\'s-new-password>
passwd: all authentication tokens updated successfully
```

Note: w hen root changes a user\'s passw ord, root not prompted for current passw ord:

```
$ sudo passwd kevin
New UNIX password: <kevin\'s-new-password>
Retype new UNIX password: <kevin\'s-new-password>
passwd: all authentication tokens updated successfully
```

Note: normal users w ill not be allow ed to set bad passw ords, such as ones that are too short, or based on dictionary w ords. How ever, root allow ed to do so.

30.13 chage: Password Aging

Generally considered important to change passw ords periodically. Limits amount of time cracked passw ord can be useful to intruder and also can be used to lock unused accounts. Dow nside: user can find policy annoying, w ind up w riting dow n ever-changing passw ords, thus making them easier to steal.

Utility to manage this: **chage**:

```
chage [-m mindays] [-M maxdays] [-d lastday] [-I inactive] [-E expiredate] [-W warndays] user
```

Examples:

```
$ sudo chage -l dexter
$ sudo chage -m 14 -M 30 kevlin
$ sudo chage -E 2012-4-1 morgan
$ sudo chage -d 0 clyde
```

Only root user can use **chage**. One exception: any user can run **chage -l** to see their aging, as show n below.

To force user to change passw ord at next login:

```
$ sudo chage -d 0 Username
```

chage

30.14 Restricted shell

Under Linux, one can use **restricted shell**, invoked as:

```
$ bash -r
```

(Some distributions may define an **rbash** command to same effect.)

Restricted shell: functions in more tightly controlled environment than standard shell, but otherw ise functions normally. In particular:

- Prevents user from using **cd** to change directories
- Prevents user from redefining follow ing environment variables: `SHELL`, `ENV`, `PATH`
- Does not permit user to specify absolute path or executable command names starting from `/`
- Prevents user from redirecting input and/or output

Note: there are other restrictions; best w ay to see them all is to do **man bash** and search for `RESTRICTED SHELL`.

Because restricted shell executes `$HOME/.bash_profile` w ithout restriction, user must have neither w rite nor execute permission on `/home` directory.

Restricted accounts can also be enabled by creating symlink to `/bin/bash`, named `/bin/rbash`, and using in `/etc/passwd`, as w ill discuss next.

30.15 Restricted Accounts

There are times w hen granting access to user necessary, but should be limited in scope. Setting up restricted user account can be useful in this context. A restricted account:

- Uses the restricted shell
- Limits available system programs and user applications
- Limits system resources
- Limits access times
- Limits access locations

From command line, or from script, restricted shell may be invoked w ith `/bin/bash -r`. How ever, flags may not be specified in `/etc/passwd` file. Simple w ay to get around this restriction w ould be to do one of the follow ing:

```
$ cd /bin ; sudo ln -s bash rbash
$ cd /bin ; sudo ln bash rbash
$ cd /bin ; sudo cp bash rbash
```

and then, use `/bin/bash` as shell in `/etc/passwd`.

When setting up such account, should avoid inadvertently adding system directories to `PATH` environment variable; this w ould grant restricted user ability to execute other system programs, such as unrestricted shell.

Restricted accounts also sometimes referred to as **limited accounts**.

30.16 The root Account

Root account should only be used for administrative purposes w hen absolutely necessary, never used as regular account. Mistakes very costly, both for integrity and stability, and for system security.

Default: root logins through netw ork generally prohibited for security reasons. Can permit **Secure Shell** logins using **ssh**, configured w ith `/etc/ssh/sshd_config`, and **PAM** (**P**luggable **A**uthentication **M**odules), discussed later, through `pam_securetty.so` module and associated `/etc/securetty` file. Root login permitted only from devices listed in `/etc/securetty`.

Generally recommended that all root access be through **su** or **sudo** (causing audit trail of all root access through **sudo**). Note: some distributions (such as Ubuntu), by default actually prohibit logging in directly to the root account.

PAM can also be used to restrict w hich users allow ed to **su** to root. Might also be w orth it to configure **auditd** to log all commands executed as root.

30.17 SSH

One often needs to login through netw ork into remote system, either w ith same user name or another. Or, one needs to transfer files to/from remote machine. In either case, one w ants to do this securely, free from interception.

SSH (**S**ecure **SH**ell) exists for this purpose. Uses encryption based on strong algorithms. Assuming proper **ssh** packages installed on system, one needs no further setup to begin using **ssh**.

To sign onto remote system:

```
$ whoami
student
$ ssh farflung.com
student@farflung.com\'s password: (type here)
$
```

w here assuming there is a student account on farflung.com. To log in as different user:

```
$ ssh root@farflung.com
root@farflung.com\'s password: (type here)
```

or

```
$ ssh -l root farflung.com
root@farflung.com\'s password: (type here)
```

To copy files from one system to another:

```
$ scp file.txt farflung.com:/tmp
$ scp file.tex student@farflung.com/home/student
$ scp -r some_dir farflung.com:/tmp/some_dir
```

(Omitted request for passw ord to save space; if configured properly w ith encryption keys as discussed next, w ill not need to supply passw ord.)

To run command on multiple machines simultaneously:

```
$ for machines in node1 node2 node3
do
      (ssh $ machines some_command &)
done
```

30.18 SSH Configuration Files

Can configure **SSH** to further expedite use, in particular to permit logging in w ithout passw ord. User-specific configuration files created under every user's home directory in hidden `.ssh` directory:

```
$ ls -l ~/.ssh
total 20
-rw-r--r-- 1 hilda hilda 1172 Sep 27  2014 authorized_keys
-rw------- 1 hilda hilda  207 Aug  9  2011 config
-rw------- 1 hilda hilda 1675 Dec  8  2010 id_rsa
-rw-r--r-- 1 hilda hilda  393 Dec  8  2010 id_rsa.pub
-rw-r--r-- 1 hilda hilda 1980 Apr 28 07:36 known_hosts
```

w hich contains:

- **`id_rsa`** : the user's private encryption key
- **`id_rsa.pub`** : the user's public encryption key
- **`authorized_keys`** : a list of public keys that are permitted to login
- **`known_hosts`** : a list of hosts from w hich logins have been allow ed in the past
- **`config`** : a configuration file for specifying various options

First, user has to generate private and public encryption keys w ith **`ssh-keygen`**:

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hilda/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/hilda/.ssh/id_rsa
Your public key has been saved in /home/hilda/.ssh/id_rsa.pub
The key fingerprint is:
76:da:d3:51:1e:c8:2d:3b:34:28:46:b2:2b:db:d1:c4 hilda@c7
The key\'s randomart image is:
+--[ RSA 2048]----+
|      . .        |
|       =   o o   |
|      . E . * +  |
|       = . . * . |
|    . o S . + .  |
|     + o + . o   |
|    . . . o .    |
|          .     |
|                |
+-----------------+
```

This w ill also generate the public key, `/.ssh/id_rsa.pub`.

Private key must never ever be shared with anyone!

Public key can be given to any machine w ith w hich you w ant to permit passw ord-less access. Should also be added to your `authorized_keys` files, together w ith all public keys from other users w ho have accounts on you machine and you w ant tot permit passw ord-less access to their accounts.

`known_hosts` file gradually built up as **ssh** accesses occur. If system detects changes in users w ho are trying to log in through **ssh**, w ill w arn you of them and afford opportunity to deny access. Note: `authorized_keys` file contains information about users and machines:

```
$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQ
...0000aSd...hilda@sbc
```

w hile `known_hosts` only contains information about computer nodes:

```
$ cat known_hosts
192.30.252.129 ssh_rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAq2A7hRGmdnm9tUDb09IDSw
....BK6tb...==
```

Can examine **man ssh_config** page to see w hat kinds of options can go into **ssh** configuration files.

30.19 Remote Graphical Login

Login into remote machine w ith full graphical desktop. Often, may use VNC (Virtual Netw ork Computing) to connect to system. Common implementation: **`tigervnc`**.

To test, first make sure that **vnc** packages installed:

```
$ sudo yum install tigervnc tigervnc-server
$ sudo zypper install tigervnc tigervnc-server
$ sudo apt install tigervnc tigervnc-server
```

using right package management system command.

Start the server as a normal user w ith:

```
$ vncserver
```

Can test w ith:

```
$ vncviewer localhost:2
```

(May have to play w ith numbers other than 2, such as 1, 3, 4…, depending on w hat you are running at the moment, and how your machine is configured.)

To view from remote machine, just slightly different:

```
$ vncviewer -via student@some_machine localhost:2
```

If you get a rather strange message about having to authenticate because of 'color profile', and no passw ords w ork, have to kill **colord** daemon on server machine:

```
$ sudo systemctl stop colord
```

This is a bug (not a feature), w ill only appear in some distributions and some systems for unclear reasons.

##

Back to top

Previous Chapter - Table of Contents - Next Chapter