-

UD 08.
LINUX: RESOURCE PROTECTION

Computer Systems
CFGS DAW

Borja Salom
b.salomsantamaria@edu.gva.es
2022/2023

Versión:221211.2018

Licencia

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

- Importante

- Atención

- Interesante

ÍNDICE DE CONTENIDO

UD08. LINUX: RESOURCE PROTECTION

1. Users

The concept of user in Linux allows separate execution environments for different purposes. Two people can work simultaneously on the same system, each having a different user, and a different home directory.

It is also very common for many internal system services to have their own user to restrict access to that service as a security mechanism. In this way, if a service sees its security compromised by an attack, the access that the user of that service has will serve as containment of the attack, and they will not be able to access files belonging to another user (person or service).

> **Daemon** is the term used in Linux to refer to a service process that runs in the background non-interactively. In general, they end with the letter **d**, like **httpd** or **ftpd**.

User configuration in Linux is essentially handled in the following two files.

1.3 File /etc/passwd

This file contains information about user accounts and their characteristics. This is the syntax used in each line:

- `name:password:UID:GID:GECOS:directory:shell`

You can see the meaning of each field with man passwd:

- **name:** user name
- **password:** The user's password in plain text, or an asterisk `*` or `x` if encrypted.
- **UID:** Unique user identification number. Users can change many parameters, including their name, but the UID should never be changed. Root's UID is 0. Service and daemon accounts have the lowest numbers, while end user accounts start at the value defined in UID_MIN in the `/etc/login.defs` file.
- **GID:** Unique group identifier number. Several users can have the same group, although when creating a user a group with the same name is created by default unless otherwise indicated. Group data appears in `/etc/group`.
- **GECOS:** Comment field that includes extra information about the user (real name, address...) Informally it is called finger information.
- **Directory:** User's home directory. End users are typically located under

`/home`.

- **Shell:** The shell that the user uses by default (in many cases it is /bin/bash). If the user has /`sbin/nologin` or /`usr/bin/false`, it means that he does not have permission to log into the system, which is common in daemons as a security measure.

An example:

```
root:x:0:0:root:/root:/bin/bash
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
avahi:x:115:120:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
usuario:x:1000:1000:usuario,,,:/home/usuario:/bin/bash
```

1.4 File /etc/shadow

It contains information about users' passwords in `/etc/passwd`, which it stores in encrypted form.

- **Name:** username
- **password**: Password encrypted. The hash function used to encrypt the password is indicated at the beginning.
- **Lastmod**: Time elapsed since the last key change.
- **Min**: Minimum number of days until the password can be changed again.
- **Max**: Maximum number of days until the system forces the user to change the password.
- **Warning**: Number of days prior to the Max in which the user is notified of his mandatory password change.
- **Inactivity**: Number of days between password expiration and account lockout.
- **Expiration**: Date the account is disabled. If left blank, the account never expires.
- **Reserved**: Field reserved for future uses.

About the password field:

- `*` is used when the account has never had a password.
- `!` it means that the account has been disabled to login by password.

When an account is locked (lock: `usermod -l user`), the user's password is not deleted, but an exclamation point is added `!` at the beginning of the password hash to indicate that it has been locked. Unlocking the user (unlock: `usermod -u user`) removes the exclamation mark, leaving the hash as it was before.

1.5 User management commands

The Shadow Tool Suite is a collection of commands that allow you to manage users without having to manipulate the `/etc/passwd` and `/etc/shadow` files directly, which is not recommended given the possibility of leaving inconsistencies in such delicate files. Some of the commands:

- `useradd` – creates a user.
- `userdel` – deletes a user.
- `usermod` – Make modifications to data in /etc/passwd. You have an option for each of the fields, except the GECOS field. Includes options to (un)lock a user (--lock and --unlock).
- `chfn` – modifies the finger information (GECOS).
- `chsh` – modifies the shell.
- `id` – prints information about the user and their groups.
  - `id -u` : print the UID
  - `id -un:` print the username
- `chage`: change age displays and modifies all password dates in /etc/shadow.

There are also the `adduser` and `deluser` commands that are more user-friendly and save some work, so they are used more frequently in practice.

> Use the man manual or the -h or --help options to see all the options these commands offer, as they can vary depending on distributions.

To set and change a password:

- `usermod -p PASSWORD USER` saves the specified password without encrypting it (it would have to be passed a hash of the password). You should not use this option.
- `passwd USER` is the command used, which allows you to enter a password securely.

With the finger command we can obtain information from the GECOS of any user:

```
$ finger theuser
Login: theuser                          Name: Juan Pérez
Directory: /home/theuser                Shell: /bin/bash
Office: 101, +34 123 456                Home Phone: +34 983 12 34 56
On since Wed Feb 04 10:26 (EST) on pts/1   3 seconds idle
     (messages off)
No mail.
No Plan.
```

2. Groups

Groups allow you to grant permissions to a set of users simultaneously.

On Linux a user has the following groups:

- **Primary group:** is the one that appears as its GID in /etc/passwd. There can only be one primary group.
- **Secondary or supplementary groups:** they are those managed in the /etc/groups file, where a user can be added to more groups.

Also, during the user session you can temporarily change the group to which the user belongs:

- **Real group:** This is your primary group that is in /etc/passwd. The group a user belongs to when they log in.
- **Effective group:** using the newgrp command you can change the primary group to which the user belongs, and the configuration is effective until you close the session or change the effective group again.

2.3 File /etc/group

This file contains information about the system groups. Its structure is similar to that of the `passwd` and `shadow` files, with the following fields:

- **group:** group name
- **password**: Password that allows a user to change groups. If it is empty, it does not require a password, and an x means that it is managed by the /etc/gshadow file.
- **GID**: Unique (numeric) identifier for the group.
- **Members**: Comma-separated list of usernames that belong to that group.

Similarly to the `/etc/shadow` file, there is the `/etc/gshadow` file, which stores the group passwords encrypted with a hash and also works with the asterisk `*` and exclamation `!` Symbols.

2.4 Group management commands

The Shadow suite also includes the commands:

- `groupadd` – add a new group
- `groupdel` – delete a group
- `groupmod` – Modifies information in /etc/groups
- `gpasswd` – Modifies the group password, reflected in /etc/gshadow

2.5 Modify user groups

Once we have a group created, we can add it as primary or secondary to a user using the usermod command, with the following options:

```
$ usermod --help
Usage: usermod [options] LOGIN

Options:
  ...
  -g, --gid GROUP               force use GROUP as new primary group
  -G, --groups GROUPS           new list of supplementary GROUPS
  -a, --append                  append the user to the supplemental GRO
UPS
                                mentioned by the -G option without remo
ving
                                him/her from other groups
```

Therefore:

- `usermod -g GROUP USER`: modify primary group
- `usermod -G GROUP USER`: replace secondary group
- `usermod -a -G GROUP USER`: Add a secondary group to the user

3. Permissions on files and directories

On Linux, each file and directory has permissions to:

- **User**: the file system stores an owner user (UID), together with the permissions associated with it.
- **Group**: an owning GID is also saved, with permissions
- **Other**: Also has permissions for users who do not have that UID or GID.

The `ls -l` command lists the files and directories including information about their permissions, as follows:

```
$ ls -l
drwxr-xr-x 6 usuario grupo  4096 Jan  5 17:37 directory
-rw-r--r-- 1 usuario grupo  2048 Jul  6 12:56 file
```

The first field indicates in the first letter if it is a regular file (`-`), directory (`d`) or link (`l`). After that, the permissions appear, organized as write, read and execute for the user, group and others.

The x execute permission on directories means that you can access inside the directory and list its contents.

3.3 Permission Modifications

The permissions of a file or directory can be changed with the command: `chmod`

Relative mode. You can treat one of the fields in isolation without touching the rest of the permissions:

- It is indicated first to whom the permission is going to be changed (you can put several):
  - `u`: user
  - `g`: group
  - `o`: others
  - `a`: all, equivalent to ugo (can also be left blank)
- Operation type:
  - `+`: add permissions
  - `-`: remove permissions
- Permissions:
  - `r`: reading
  - `w`: exwrite
  - `x`: execution

Examples:

- `chmod u+x file`
- `chmod go-x file`
- `chmod +x file`

Absolute mode. Full permit information can be replaced using a three-digit base-8 (octal) number. The permissions match those of the number in binary. The good thing about working in octal is that each character can be worked on independently.

Example of permission 754:

- 7 in binary is 111 ⇒ user permissions: rwx
- 5 in binary is 101 ⇒ group permissions: r-x
- 4 in binary is 100 ⇒ permissions from others: r--

```
$ chmod 754 file
$ ls -l file
-rwxr-xr-- 1 usuario grupo  2048 Jan  6 13:03 file
```

3.4 Mask

The operating system skin defines the permissions that are assigned by default to files and directories at the time they are created.

The parameterless `umask` command prints the value it currently has, and can be handled in symbolic mode and octal mode:

```
$ umask
0022
$ umask -S
u=rwx,g=rx,o=rx
```

If a parameter is passed to the command, the new mask is set.

Symbolic mode: Allows you to set permissions using the letters `u` (user), `g` (group), `o` (other) and `a` (all). It can be done in relative mode using the symbols + and -, or in absolute mode with `=`, and combinations of both.

```
$ # Modo relativo
$ umask g-w
$ umask a+x
$ # Modo relatuvo (u,o) y modo absoluto (g)
$ umask u-w,g=r,o+r
```

Octal mode: Allows you to set permissions numerically, always in absolute mode.

Octal mode is used based on maximum permissions, which for directories is `777` and for files `666` (no execution). Mask bits set to 1 will disable that permission to the maximum permissions. Ex: `umask = 023`

```
Umask      023: 000010011
Directorio 777: 111111111 -> 111101100 = rwxr-xr--
Archivo    666: 110110110 -> 110100100 = rw-r--r--
```

Many systems have the mask `022`, which is set with: `unmak 022`

3.5 Change of owner and group

These two attributes can be modified with the commands:

- `chown file new_owner`: Modifies the owner (associated UID).
- `chgrp file new_group`: modifies the group (associated GID).

With both, you can use the `-R` option on directories, so that it performs the operation recursively, that is, it applies it to everything it contains directly or in subdirectories, sub-subdirectories, etc.