CSNB113: System Administration - 4th Topic: User administration // Authentication and Profiles

Objectives

Managing users

- Creating new user accounts
- User groups: function, setting up
- Privileged accounts
- Deleting users accounts

Authentication and Environment

- Setting up user environment
- Profiles

Creating new user accounts

The system needs to be aware of the new users:

- Real name
- User name *
- Password *
- Home directory
- Resources allocated to the user *

*: compulsory

Example from lab exercise 2:

```
useradd -m it098765       [-m creates home directory]
passwd it098765      [sets password for user it098765]
```

As long as the password is not set, the user cannot log on

Groups: Function, setting up

Often, on systems with many users, some groups of users have similar functions in an organisation: student, staff, administration, management, cleaners, etc.

Instead of setting permissions individually, it is possible to allocate users belonging to a **group**, to a group of users on a system. This makes it easier to administrate these groups of users. On *nix, the groups on a system can be found in **/etc/group**

By default, for a new user a new group will be created.

If you want to add a new user to an existing group, you would use the option -g {group_name} to the command:

```
useradd -m -g admin it098765
```

adds the new user to the (user-)group admin.

Privileged Accounts

It is only logical, that there will be different levels of user accounts. **At least two** levels are needed:

- *System administrators*
- *Unprivileged users*

We noticed this in lab 2, when the user account created, did not have the permission to shutdown ('halt') the system.

The privileged account is the system administrator, called ***root*** on *nix machines.

Since the root user has **all privileges**, it is **very** dangerous to use the root account. One single command can erase / destroy the whole installation, operating system, applications, all data.

Therefore it is recommended, to not use the root account by default, but take the *role of root* for specific commands.

sudo

There are a number of terms for sudo, officially it is 'do as [another] user':

```
$ whoami
it12345
$ sudo -u it07777 whoami   [-u {username} identifies the user]
it07777
```

(This of course works only when a user it07777 exists on the system.)

This shows how one user can execute a command (program) as another user; or with the *privilege* of another user.

Without the -u, the program sudo executes the command as root.

Or:

Without the -u, sudo assumes you want to run the command as root:

```
$ sudo halt
```

is the same as

```
$ sudo -u root halt
```

Home Directories

For easy house-keeping, we usually put all user data into a specific directory:

/home

With two users, it07777 and it098765, there will be two subdirectories in /home:

/home/it07777/

/home/it098765/

When a user logs on, (s)he will find him-/herself in their home directory.

All personal files, data, and settings are there. It is the only place, where a user has the **permission to write** data (except the system administrator changed the *permissions*).

It is easy **to back up** all user data by **copying** /home/ to the backup medium.

Also, this is a reason, why often /home is put on an **extra partition**:

If the installation breaks, the system administrator can easily re-install the system, make sure that the partition /home/ is **not formatted**, and then all user settings and data are there, identical to what they were before the breakage.

This also allows to store all user data on another medium, another machine, or even somewhere else, over the network. This is called NFS (Network File System).

Environment

When a user is logged on, (s)he finds him-/herself in their home directory, abbreviated with ~.

But also, (s)he has access (or no access) to some resources, e.g. data, files, printer. When (s)he types a command (like 'date'), this commands needs to be interpreted and acted upon by a command interpreter, by a shell. There are a number of different shells available, like 'sh', 'ash', 'ksh', 'bash', 'csh', etc.

They all behave slightly different. You can change the shell by typing the name of the shell.

```
$ echo $SHELL
```

will show the one that you are using currently.

Environment setup

Environment is the agglomeration of all specific items for a specific user. Different users can have different environments.

When creating a new user, the files in /etc/skel [for skeleton] are copied to the home directory of the new user. Files here are all dot-files, that means they are invisible to the normal 'ls' [list] command. Such files are called hidden files.

When a user is created, the default parameters used can be seen/changed with 'useradd -D'.

Examples

```
$ useradd -D
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/sh
SKEL=/etc/skel
CREATE_MAIL_SPOOL=no
$ ls -l /etc/skel/
total 0
$ ls -la /etc/skel/
-rw-r--r--   1 root root   220 2010-04-19 10:15 .bash_logout
-rw-r--r--   1 root root  3353 2010-08-11 04:47 .bashrc
-rw-r--r--   1 root root   675 2010-04-19 10:15 .profile
$
```

Removing (Deleting) Users

Removing a user is somewhat easier: We need to remove the user from the *userbase*, and then eventually

- delete the **home directory**
- delete the **group** containing the user – only if it is empty without that user!

By default, the home-directory is not deleted:

```
$ userdel it07777
```

will leave the data of it07777 on the disk, under /home/it07777

If one wanted to **remove the home directory** as well, it would be:

```
$ userdel -r it07777
```

Permissions

(Before we continue, we **must** know about **permissions**!)

Each and every file in *nix, as well as each and every directory has a set of permissions.

The 3 basic permissions are

r(ead)

w(rite)

(e)**x**(ecute)

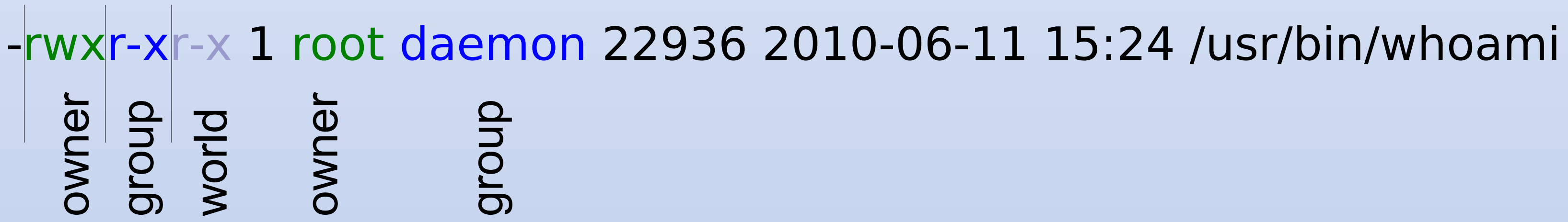
And these 3 permissions are given out three times: to the

owner ('user')

group

world ('everyone', 'all')

Permissions - Example



```
-rwxr-xr-x 1 root daemon 22936 2010-06-11 15:24 /usr/bin/whoami
```

owner is allowed to r,w,x

group is allowed to r,x

world is allowed to r,x

It is quite obvious, that only the owner is allowed to **w**rite, that is delete, change, edit the program.

It is also quite obvious that everyone is allowed to e**x**ecute, that is **run** the program. (You yourself have already used it, and you ran it!)

The other details are **size**, **timestamp** of creation or modification, and name.

Authentication

When a user logs on, (s)he must be authenticated:

It must be checked, whether

- **username** exists in the userbase
- **password** supplied is correct

That means, both must be stored. Since there is no secret in the username, everyone can view the username, and some details of the user. Since the password must be very secret, only root can be allowed to see the password.

There are two files that contain the user data:

/etc/passwd contains the 'public' data

/etc/shadow contains mainly the passwords (as *hashes*, so that they are unreadable)

References

- Textbook, p.239 - 253
- Textbook, p.70 – 71 "The Superuser: root"
- Textbook, p.328, Figure 14-1
-