Users & Privileges

In the Linux universe, every user belongs to one or more groups. These groups determine the user's access rights and privileges across the system. Think of it like a secret society, where members share certain privileges and access levels based on their group affiliations.

The most powerful group of all? The "root" group, also known as the superuser or admin group. Members of this elite group wield immense power, with the ability to perform any action on the system, from installing software to modifying system files.

Let's start by adding a new user. We can do this using the adduser command.

```
sudo adduser itachi

pass - akatsuki
```

Now let's change our shell to that of our newly created user.

```
su itachi
```

- We can also delete the created user using deluser command.

```
sudo deluser itachi
```

Like we have created a new user, we can also create a new group. Let's create a group called Akatsuki.

```
sudo groupadd Akatsuki
```

Now let's add our user itachi to our newly create group.

```
sudo usrmod -a -G Akatsuki itachi
```

Next if we want to delete the group, we can certainly do this using the groupdel comand.

```
sudo groupdel Akatsuki
```

In Linux, there are two files which are really important. One is the passwd file and the other one is the shadow file. Let's see the passwd file first.

```
cat /etc/passwd
```

Now this passwd file seems like it stores the password for the linux users. I mean, it used to do back in the day. But now, passwords are stored in the shadow file.

The great thing about the passwd file is that, any user can view its contents and it holds tremendous amount of information for hackers like us.

Starting off we have -

- Username
- x - at this place, the user password was used to store back in the day but now it only has a placeholder
- User ID (UID)
- Group ID (GID)
- Full name or comment field
- User's Home directory
- Default shell

Next, is the shadow file. Now, if we try to view its contents. It throws me an error like before. So, if you remember this from the sudo overview section, we used the sudo command to elevate our privileges as root and then we were able to view the file's content.

So, here we can see the same entries like the passwd file however the main thing to notice here is the x placeholder that we saw in the passwd file. It has been now replaced with an encrypted string which is eventually the user's password.

Managing file ownership

Moving on, let's now how we can change the owner and group permissions.

To modify the user permissions, we will use the chown command. the chown command let us change the owner of a particular file.

```
sudo chown itachi test.sh
```

Next, if we want to modify the group permissions of a file. We can do that with the chgrp command.

```
sudo chgrp Akatsuki test.sh
```

Now if we want to change both owner and group at the same time, we can use a one-liner for that.

```
sudo chown itachi:Akatsuki test.sh
```

SUID & SGID

SUID and SGID are special permissions that can be set on executable files and directories, respectively. These permissions grant temporary elevated privileges to the user executing the file or accessing the directory, allowing them to perform actions that would normally be restricted by their regular user permissions.

Think of it like a superhero's secret identity – when a regular user executes an SUID file or accesses an SGID directory, they temporarily gain superpowers, transcending their usual limitations.

Let's understand SUID first.

When the SUID (Set User ID) permission is set on an executable file, the file runs with the permissions of its owner, regardless of who executed it. This means that if an ordinary user runs an SUID file owned by root, the file will execute with root privileges, granting the user temporary elevated access.

```
-rwsr-xr-x 1 root root 166056 Apr  4  2023 /usr/bin/sudo
```

So, the `sudo` command has the SUID bit set, allowing regular users to run it with root privileges, enabling them to execute commands with elevated permissions.

We can also set up the SUID bit to our favourite scripts and files.

```
sudo chmod u+s test.sh

# Using numerical representation. SUID uses "4" and is prepended before
the file permission.

sudo chmod 4755 test.sh
```

Now, let's talk about SGID.

Similarly, the SGID (Set Group ID) permission works its magic on directories. When the SGID bit is set on a directory, any new files or subdirectories created within that directory inherit the group ownership of the parent directory, rather than the user's primary group.

```
ls -l /run
```

So, the `/run/postgres` directory has the SGID bit set, ensuring that any new files or directories created within it will belong to the "postgres" group, regardless of the user's primary group membership.

We can set up our SGID bit like we did in SUID. The main difference is that for the numerical representation, SGID uses 2 as it number and it is also prepended before the normal permissions.

```
sudo chmod g+s myfile

# Using numerical representation. SGID uses "2" and is prepended before
the file permission.

sudo chmod 2555 myfile
```
