SIC-IOT702 / task 03 : capstone Task / phase 3

Instractor : eng/ Mohamed Ahmed

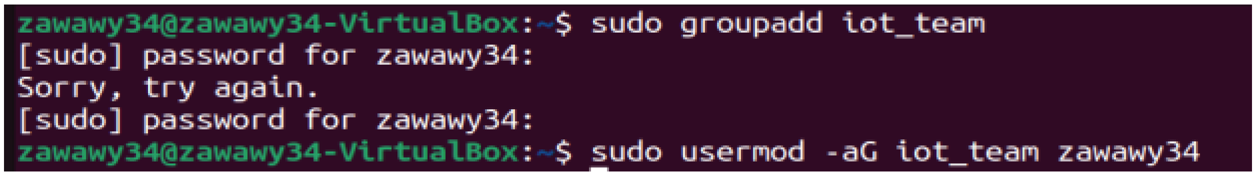
Student name: Ahmed Mohamed Elsayed

Day 3 – Phase 3: User, Group, and Permissions Management

Boss's Request: Secure the project and restrict access to authorized users only.

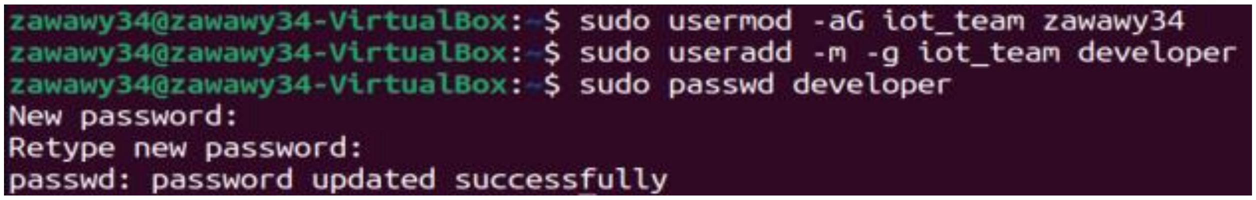
Tasks:

• Create a new group iot_team and add your user to it.



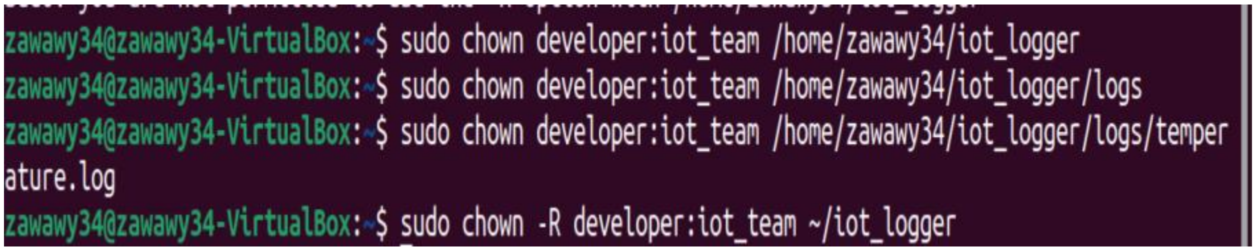
```
zawawy34@zawawy34-VirtualBox:~$ sudo groupadd iot_team
[sudo] password for zawawy34:
Sorry, try again.
[sudo] password for zawawy34:
zawawy34@zawawy34-VirtualBox:~$ sudo usermod -aG iot_team zawawy34
```

• Create a new developer user, add it to the group.



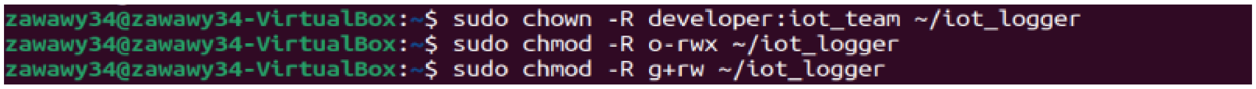
```
zawawy34@zawawy34-VirtualBox:~$ sudo usermod -aG iot_team zawawy34
zawawy34@zawawy34-VirtualBox:~$ sudo useradd -m -g iot_team developer
zawawy34@zawawy34-VirtualBox:~$ sudo passwd developer
New password:
Retype new password:
passwd: password updated successfully
```

• Change ownership of iot_logger to the developer + group.



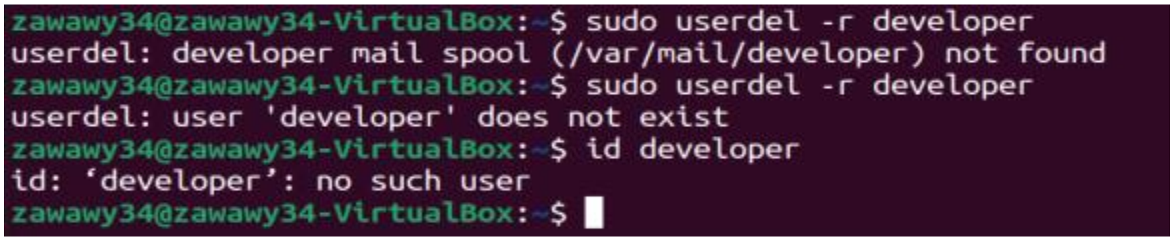
```
zawawy34@zawawy34-VirtualBox:~$ sudo chown developer:iot_team /home/zawawy34/iot_logger
zawawy34@zawawy34-VirtualBox:~$ sudo chown developer:iot_team /home/zawawy34/iot_logger/logs
zawawy34@zawawy34-VirtualBox:~$ sudo chown developer:iot_team /home/zawawy34/iot_logger/logs/temper
ature.log
zawawy34@zawawy34-VirtualBox:~$ sudo chown -R developer:iot_team ~/iot_logger
```

• Set permissions: group can read/write logs, others blocked.



```
zawawy34@zawawy34-VirtualBox:~$ sudo chown -R developer:iot_team ~/iot_logger
zawawy34@zawawy34-VirtualBox:~$ sudo chmod -R o-rwx ~/iot_logger
zawawy34@zawawy34-VirtualBox:~$ sudo chmod -R g+rw ~/iot_logger
```

• Test access as new user, then remove test user.



```
zawawy34@zawawy34-VirtualBox:~$ sudo userdel -r developer
userdel: developer mail spool (/var/mail/developer) not found
zawawy34@zawawy34-VirtualBox:~$ sudo userdel -r developer
userdel: user 'developer' does not exist
zawawy34@zawawy34-VirtualBox:~$ id developer
id: ‘developer’: no such user
zawawy34@zawawy34-VirtualBox:~$ 
```

Open-Ended Questions:

File Permissions (r, w, x) for Files vs. Directories:

- Linux file permissions control what actions users can perform on files and directories. They are defined for three user classes: the file owner (u), the group (g), and others (o).
- **For a File:**
  - r (Read): Allows the user to view the contents of the file.
  - w (Write): Allows the user to modify or delete the file's contents.
  - x (Execute): Allows the user to run the file as a program or script.
- **For a Directory:**
  - r (Read): Allows the user to list the contents of the directory (e.g., using ls).
  - w (Write): Allows the user to create, rename, or delete files within the directory.
  - x (Execute): Allows the user to enter (cd) the directory and access files inside it. Without execute permission, you cannot access the directory's contents even with read permission.
- **Example using ls -l:**
  - -rw-r--r--: This shows a regular file (-). The owner has read and write (rw-) permissions, while the group and others have only read (r--) permissions.

Octal Notation and umask:

- Octal notation is a numerical way to represent file permissions. Each permission (read, write, execute) is assigned a value: r=4, w=2, x=1. The sum of these values gives a single octal digit for each user class (owner, group, others).
- **Example Calculation:**
  - rwx = 4+2+1=7

    - rw- = 4+2+0=6
    - r-- = 4+0+0=4
    - Therefore, rwxr-xr-x is 755 in octal notation.
- The umask command controls the default permissions given to newly created files and directories. It is a three-digit octal number that represents the permissions to be **subtracted** from the default permissions. For files, the default is 666 (rw-rw-rw-), and for directories, it's 777 (rwxrwxrwx).
    - If umask is 022, the default file permissions become 666 - 022 = 644 (rw-r--r--), and directory permissions become 777 - 022 = 755 (rwxr-xr-x).

Root vs. Normal User:

- The **root user** is the system administrator account with complete and unrestricted access to the entire operating system, including all files, directories, and commands. Root can bypass all file permissions and perform any action, such as installing system-wide software, changing critical configuration files, and managing user accounts.
- A **normal user** has limited privileges and is only allowed to perform actions within their home directory and for which they have specific permissions. They cannot make system-wide changes without elevated privileges (e.g., using sudo).
- Root is considered dangerous because its power can be used to cause catastrophic damage to the system, either accidentally or maliciously. A single wrong command run as root could delete essential system files or corrupt the file system, rendering the system unusable. This is why it is a best practice to use a normal user account for daily tasks and only use sudo for administrative actions when absolutely necessary.