

Managing Access Controls in Linux

Scenario

In this activity, you will manage access controls on a Linux server. First, you will configure local users and groups on the system. Next, you will create directories and files to represent resources that need to be controlled. Next, you will configure the ownership of these directories and files. Finally, you'll configure standard Linux permissions to manage for the users and groups to access the directories and files.

Objectives

This activity is designed to test your understanding of and ability to apply content examples in the following CompTIA Security+ objectives:

- 3.8 Given a scenario, implement authentication and authorization solutions.
- 4.1 Given a scenario, use the appropriate tool to assess organizational security.

Lab

- Kali VM
- pfSense VM

Task 1

Create users and groups

You need to create two test users and two groups on the Linux virtual machine. You'll configure passwords for the two users, too.

1. Sign in to the **Kali** VM as **kali** using **Pa\$\$w0rd** as the password.
2. From the toolbar at the top of the **Desktop**, open the **Terminal**.

TIP: You should get used to managing Linux servers from the command line.

3. Run the following command to create a new user named **user01**:

```
sudo useradd user01
```

4. Run the following command to set a password for **user01**:

```
sudo passwd user01
```

5. When prompted, enter and confirm the password **Pa\$\$w0rd**

```
(kali㉿kali)-[~]
$ sudo useradd user01

(kali㉿kali)-[~]
$ sudo passwd user01
New password:
Retype new password:
passwd: password updated successfully
```

Figure 1.1 – Creating a user with `useradd` and entering a password.

TIP: Creating a user in Linux is two steps: 1) create the user by using the **useradd** command, and 2) set the password by using the **passwd** command.

6. Repeat the above steps to create **user02**, and then set **Pa\$\$w0rd** as the password.

7. Confirm that **user01** and **user02** exist typing the following command:

```
tail /etc/passwd
```

```
(kali㉿kali)-[~]
$ tail /etc/passwd
inetsim:x:128:136::/var/lib/inetsim:/usr/sbin/nologin
lightdm:x:129:137:Light Display Manager:/var/lib/lightdm:/bin/false
colord:x:130:138:colord colour management daemon,,:/var/lib/colord:/usr/sbin/nologin
geoclue:x:131:139::/var/lib/geoclue:/usr/sbin/nologin
king-phisher:x:132:140::/var/lib/king-phisher:/usr/sbin/nologin
kali:x:1000:1000:kali,,:/home/kali:/usr/bin/zsh
redis:x:133:142::/var/lib/redis:/usr/sbin/nologin
gvm:x:134:143::/var/lib/openvas:/usr/sbin/nologin
user01:x:1001:1001::/home/user01:/bin/sh
user02:x:1002:1002::/home/user02:/bin/sh
```

Figure 1.2 – Confirm user accounts with the `tail` command.

8. Run the following command to create a new group named **admins**:

```
sudo groupadd admins
```

9. Create a second group named **devs**.

```
sudo groupadd devs
```

10. Confirm that the **admins** and **devs** group exist with the following command:

```
tail /etc/group
```

```
(kali㉿kali)-[~]  
$ tail /etc/group  
geoclue:x:139:  
kpadmins:x:140:  
kali:x:1000:  
kaboxer:x:141:kali  
redis:x:142:_gvm  
_gvm:x:143:  
user01:x:1001:  
user02:x:1002:  
admins:x:1003:  
devs:x:1004:
```

Figure 1.3 – Confirm admins and devs group exist with the tail command.

11. Add **user01** to the **admins** group:

```
sudo usermod -aG admins user01
```

NOTE: Adding a user to a group is a modification of the user object, hence the use of **usermod**.

12 Run these commands to verify the group memberships:

```
id user01
```

```
tail /etc/group
```

```
(kali㉿kali)-[~]  
$ id user01  
uid=1001(user01) gid=1001(user01) groups=1001(user01) 1003(admins)  
  
(kali㉿kali)-[~]  
$ tail /etc/group  
geoclue:x:139:  
kpadmins:x:140:  
kali:x:1000:  
kaboxer:x:141:kali  
redis:x:142:_gvm  
_gvm:x:143:  
user01:x:1001:  
user02:x:1002:  
admins:x:1003:user01  
devs:x:1004:
```

Figure 1.4 – Verifying group membership.

TIP: The **tail** command in Linux displays the last 10 lines of a file. In this case, it displays the most recently-created groups.

13. Add **user02** to the **devs** group.

14 Confirm that **user02** is a member of the **devs** group.

Task 2

Create directories and files

Use the **mkdir** and **touch** commands to create directories and files to simulate resources that need to have permissions applied to them.

1. Run the following commands to create **three** directories:

```
sudo mkdir /projects
```

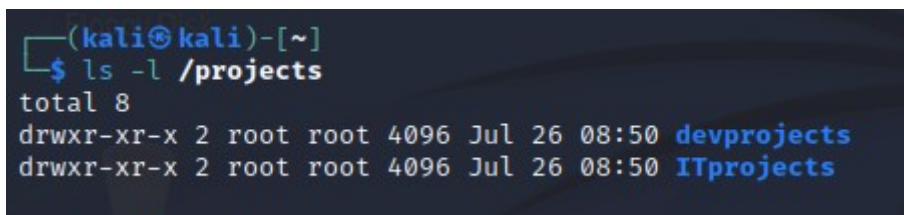
```
sudo mkdir /projects/ITprojects
```

```
sudo mkdir /projects/devprojects
```

TIP: You could create all three directories (/projects, /projects/ITprojects, /projects/devprojects) by issuing one command:

```
sudo mkdir /projects /projects/ITprojects /projects/devprojects
```

2. Confirm that the **/projects/ITprojects** and that the **/projects/devprojects** directories exists.



```
(kali㉿kali)-[~]  
$ ls -l /projects  
total 8  
drwxr-xr-x 2 root root 4096 Jul 26 08:50 devprojects  
drwxr-xr-x 2 root root 4096 Jul 26 08:50 ITprojects
```

Figure 2.1 – Confirm the directories with ls -l command.

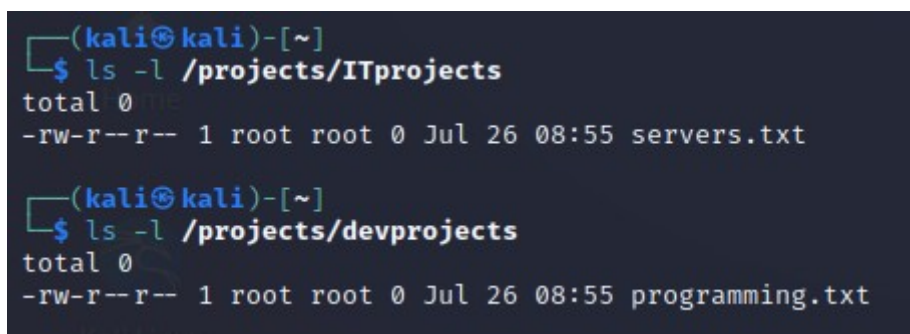
3. Run the following commands to create two files:

```
sudo touch /projects/ITprojects/servers.txt
```

```
sudo touch /projects/devprojects/programming.txt
```

TIP: The **touch** command updates the timestamp on an existing file. If the specified file does not exist, **touch** creates it.

4. Confirm that the files exist.



```
(kali㉿kali)-[~]  
$ ls -l /projects/ITprojects  
total 0  
-rw-r--r-- 1 root root 0 Jul 26 08:55 servers.txt  
  
(kali㉿kali)-[~]  
$ ls -l /projects/devprojects  
total 0  
-rw-r--r-- 1 root root 0 Jul 26 08:55 programming.txt
```

Figure 2.2 – Verifying files with ls -l command.

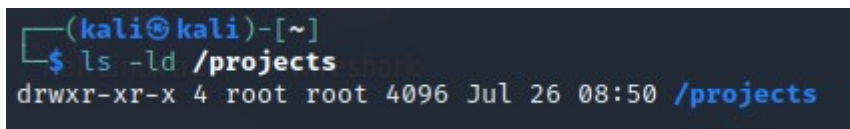
Task 3

Configure ownership

By default, the creator of a resource owns the resource. Because you used the kali user account with sudo to create the directories and files in the previous task, the root user controls access. You need to permit users and groups to own the directories and files so that they can configure permissions.

1. Run the following commands to display the current default permissions on the contents of the /projects directory:

```
ls -ld /projects
```



```
(kali㉿kali)-[~]  
$ ls -ld /projects  
drwxr-xr-x 4 root root 4096 Jul 26 08:50 /projects
```

Figure 3.1 – Display the current default permissions.

NOTE: Whoever creates the object, owns the object. Hence, root owns these objects.

2. Run the following commands to change ownership from root to the specified users and groups:

```
sudo chown -R user01:admins /projects/ITprojects
```

```
sudo chown -R user02:devs /projects/devprojects
```

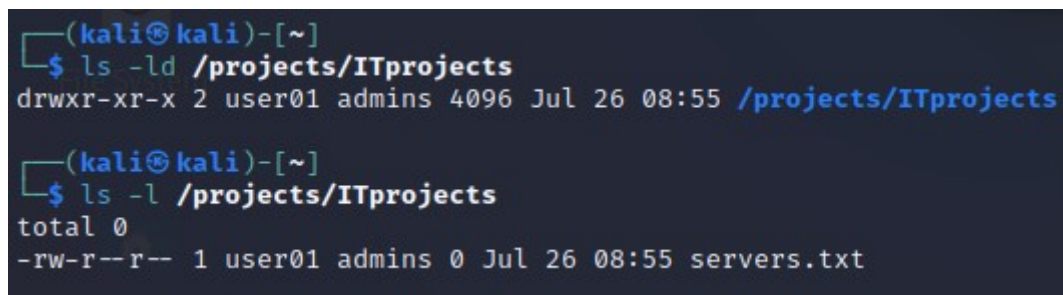
NOTE: The **-R** option makes the **chown** recursive. This applies the ownership and groups settings to the directory and everything in it.

3. Run the following command to display the user and group associations for the /projects/ITprojects directory itself.

```
ls -ld /projects/ITprojects
```

4. Run the following command to display the user and group associations for the contents of the /projects/ITprojects directory:

```
ls -l /projects/ITprojects
```



```
(kali㉿kali)-[~]  
$ ls -ld /projects/ITprojects  
drwxr-xr-x 2 user01 admins 4096 Jul 26 08:55 /projects/ITprojects  
  
(kali㉿kali)-[~]  
$ ls -l /projects/ITprojects  
total 0  
-rw-r--r-- 1 user01 admins 0 Jul 26 08:55 servers.txt
```

Figure 3.2 – Verifying ownership of the resources.

Task 4

Configure permissions

You will use the `chmod` command in symbolic mode to configure permissions for the resource owner, the associated group, and all others.

1. Run the following command to display the current permissions for the **/projects/ITprojects** directory:

```
ls -ld /projects/ITprojects
```

2. Configure the permissions according to the table below:

Resources	User/Group	Access Level
/projects/ITprojects	user01/admins	rwxrwxr-x
/projects/devprojects	user02/devs	rwxrwxr-x

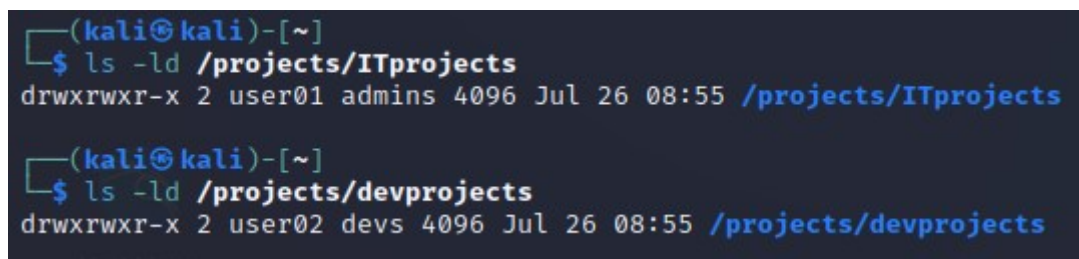
Table 4.1 – Projects table User & Group permissions.

Set the permissions by typing the following commands:

```
sudo chmod u=rwx,g=rwx,o=rx /projects/ITprojects
```

```
sudo chmod u=rwx,g=rwx,o=rx /projects/devprojects
```

3. Confirm that permissions are set correctly for the **/projects/ITprojects** and for the **/projects/devprojects** directory.



```
(kali㉿kali)-[~]  
$ ls -ld /projects/ITprojects  
drwxrwxr-x 2 user01 admins 4096 Jul 26 08:55 /projects/ITprojects  
  
(kali㉿kali)-[~]  
$ ls -ld /projects/devprojects  
drwxrwxr-x 2 user02 devs 4096 Jul 26 08:55 /projects/devprojects
```

Figure 4.1 – Verifying permissions of the directories.