Secure Files/Directories using ACLs (Access Control Lists) in Linux

Biswaraj Sahoo
•Jun 12, 2023•
7 min read

TABLE OF CONTENTS

As a **System Admin**, our first priority will be to protect and secure data from unauthorized access. We all are aware of the permissions that we set using some helpful Linux commands like **chmod**, **chown**, **chgrp**… etc. However, these default permission sets have some limitation and sometimes may not work as per our needs. For example, we cannot set up different permission sets for different users on same directory or file. Thus, **Access Control Lists** (**ACLs**) were implemented.

Let’s say, you have three users, ‘**tecmint1**‘, ‘**tecmint2**‘ and ‘**tecmint3**‘. Each having common group say ‘**acl’**. User ‘**tecmint1**‘ want that only ‘**tecmint2**‘ user can **read** and **access** files owned by ‘**tecmint1**‘ and no one else should have any access on that.

ACLs (**Access Control Lists**) allows us doing the same trick. These ACLs allow us to grant permissions for a **user**, **group** and any group of any users which are not in the group list of a user.

Note: As per Redhat Product Documentation, it provides ACL support for ext3 file system and NFS exported file systems.

1. Check Kernel for ACL Support

Run the following command to check ACL Support for file system and **POSIX_ACL=Y** option (if there is **N** instead of **Y**, then it means Kernel doesn't support ACL and need to be recompiled).

COPY

COPY

```
[root@linux ~]# grep -i acl /boot/config*



CONFIG_EXT4_FS_POSIX_ACL=y

CONFIG_REISERFS_FS_POSIX_ACL=y

CONFIG_JFS_POSIX_ACL=y

CONFIG_XFS_POSIX_ACL=y

CONFIG_BTRFS_FS_POSIX_ACL=y
```

```
CONFIG_FS_POSIX_ACL=y

CONFIG_GENERIC_ACL=y

CONFIG_TMPFS_POSIX_ACL=y

CONFIG_NFS_V3_ACL=y

CONFIG_NFSD_V2_ACL=y

CONFIG_NFSD_V3_ACL=y

CONFIG_NFS_ACL_SUPPORT=m

CONFIG_CIFS_ACL=y

CONFIG_9P_FS_POSIX_ACL=y
```

2. Check Required Packages

Before starting playing with ACLs make sure that you have required packages installed. Below are the required packages that needs to be installed using **yum** or **apt-get**.

COPY

COPY

```
[root@linux ~]# yum install nfs4-acl-tools acl libacl       [on RedHat based systems]
```

COPY

COPY

```
[tecmint@linux ~]$ sudo apt-get install nfs4-acl-tools acl    [on Debian based systems]
```

3. Check Mounted File System for ACLs Support

Now, check the mounted file system that whether it is mounted with ACL option or not. We can use **‘mount**‘ command for checking the the same as shown below.

COPY

COPY

```
[root@linux ~]# mount  | grep -i root



/dev/mapper/fedora-root on / type ext4 (rw,relatime,data=ordered)
```

But in our case its not showing acl by default. So, next we have option to remount the mounted partition again using acl option. But, before moving ahead, we have another option to make sure that partition is mounted with acl option or not, because for recent system it may be integrated with default mount option

COPY

COPY

```
[root@linux ~]# tune2fs -l /dev/mapper/fedora-root | grep acl



Default mount options:    user_xattr acl
```

In the above output, you can see that default mount option already have support for acl. Another option is to remount the partition as shown below.

COPY

COPY

```
[root@linux ~]# mount -o remount,acl /
```

Next, add the below entry to '/etc/fstab' file to make it permanent.

COPY

COPY

```
/dev/mapper/fedora-root /    ext4    defaults,acl 1 1
```

Again, remount the partition.

COPY

COPY

```
[root@linux ~]# mount -o remount  /
```

4. For NFS Server

On NFS server, if file system which is exported by NSF server supports ACL and ACLs can be read by NFS Clients, then ACLs are utilized by client System.

For disabling ACLs on NFS share, you have to add option "**no_acl**" in '**/etc/exportfs**' file on NFS Server. To disable it on NSF client side again use "**no_acl**" option during mount time.

How to Implement ACL Support in Linux Systems

There are two types of **ACLs**:

1. **Access ACLs**: Access ACLs are used for granting permissions on any file or directory.
2. **Default ACLs**: Default ACLs are used for granting/setting access control list on a specific directory only.

Difference between Access ACL and Default ACL:

1. Default ACL can be used on directory level only.
2. Any sub directory or file created within that directory will inherit the ACLs from its parent directory. On the other hand a file inherits the default ACLs as its access ACLs.
3. We make use of "**–d**" for setting default ACLs and Default ACLs are optionals.

Before Setting Default ACLs

To determine the default ACLs for a specific file or directory, use the '**getfacl**' command. In the example below, the **getfacl** is used to get the default ACLs for a folder '**Music**'.

COPY

COPY

```
[root@linux ~]# getfacl Music/



# file: Music/

# owner: root

# group: root

user::rwx

group::r-x

other::r-x

default:user::rwx
```

```
default:group::r-x

default:other::rw-
```

After Setting Default ACLs

To set the default ACLs for a specific file or directory, use the ‘**setfacl**’ command. In the example below, the **setfacl** command will set a new ACLs (**read** and **execute**) on a folder ‘**Music’**.

COPY

COPY

```
[root@linux ~]# setfacl -m d:o:rx Music/

[root@linux ~]# getfacl Music/

# file: Music/

# owner: root

# group: root

user::rwx

group::r-x

other::r-x

default:user::rwx

default:group::r-x

default:other::r-x
```

How to Set New ACLs

Use the ‘**setfacl’** command for setting or modifying on any file or directory. For example, to give **read** and **write** permissions to user ‘**tecmint1**‘.

COPY

COPY

```
# setfacl -m u:tecmint1:rw /tecmint1/example
```

How to View ACLs

Use the ‘**getfacl**‘ command for viewing ACL on any file or directory. For example, to view ACL on ‘**/tecmint1/example**‘ use below command.

COPY

COPY

```
# getfacl /tecmint1/example



# file: tecmint1/example/

# owner: tecmint1

# group: tecmint1

user::rwx

user:tecmint1:rwx

user:tecmint2:r--

group::rwx
```

```
mask::rwx

other::---
```

How to Remove ACLs

For removing ACL from any file/directory, we use **x** and **b** options as shown below.

COPY

COPY

```
# setfacl -x ACL file/directory      # remove only specified ACL from
file/directory.

# setfacl -b  file/directory          #removing all ACL from file/direcoty
```

Let’s implement ACL’s on following scenario’s.

Two Users (**tecmint1** and **tecmint2**), both having common secondary group named ‘**acl**‘. We will create one directory owned by ‘**tecmint1**‘ and will provide the **read** and **execute** permission on that directory to user ‘**tecmint2**‘.

Step 1: Create two users and remove password from both

COPY

COPY

```
[root@linux ~]# for user in tecmint1 tecmint2

> do
> useradd $user

> passwd -d $user

> done

Removing password for user tecmint1.

passwd: Success

Removing password for user tecmint2.

passwd: Success
```

Step 2: Create a Group and Users to Secondary Group.

COPY

COPY

```
[root@linux ~]# groupadd acl

[root@linux ~]# usermod -G acl tecmint1

[root@linux ~]# usermod -G acl tecmint2
```

Step 3: Create a Directory **/tecmint** and change ownership to **tecmint1**.

COPY

COPY

```
[root@linux ~]# mkdir /tecmint1

[root@linux ~]# chown tecmint1 /tecmint1/
```

COPY

COPY

```
[root@linux ~]# ls -ld /tecmint1/

drwxr-xr-x 2 tecmint1 root 4096 Apr 17 14:46 /tecmint1/
```

COPY

COPY

```
[root@linux ~]# getfacl /tecmint1

getfacl: Removing leading '/' from absolute path names
# file: tecmint1
# owner: tecmint1
# group: root
user::rwx
group::r-x
other::r-x
```

Step 4: Login with **tecmint1** and create a Directory in **/tecmint** folder.

COPY

COPY

```
[tecmint@linux ~]$ su - tecmint1

Last login: Thu Apr 17 14:49:16 IST 2014 on pts/4
```

COPY

COPY

```
[tecmint1@linux ~]$ cd /tecmint1/
[tecmint1@linux tecmint1]$ mkdir example
```

COPY

COPY

```
[tecmint1@linux tecmint1]$ ll

total 4
drwxrwxr-x 2 tecmint1 tecmint1 4096 Apr 17 14:50 example
```

COPY

COPY

```
[tecmint1@linux tecmint1]$ whoami
tecmint1
```

Step 5: Now set ACL using ‘**setfacl**‘, so that ‘**tecmint1**‘ will have all **rwx** permissions, ‘**tecmint2**‘ will have only **read** permission on ‘**example**‘ folder and other will have no permissions.

COPY

COPY

```
$ setfacl -m u:tecmint1:rwx example/

$ setfacl -m u:tecmint2:r-- example/

$ setfacl -m  other:--- example/

$ getfacl example/



# file: example

# owner: tecmint1

# group: tecmint1

user::rwx

user:tecmint1:rwx

user:tecmint2:r--

group::r-x

mask::rwx

other::---
```

Step 6: Now login with other user i.e. '**tecmint2**' on another terminal and change directory to '**/tecmint1**'. Now try to view the contents using '**ls**' command and then try to change directory and see the difference as below.

COPY

COPY

```
[tecmint@linux ~]$ su - tecmint2



Last login: Thu Apr 17 15:03:31 IST 2014 on pts/5
```

COPY

COPY

```
[tecmint2@linux ~]$ cd /tecmint1/

[tecmint2@linux tecmint1]$ ls -lR example/

example/:

total 0
```

COPY

COPY

```
[tecmint2@linux tecmint1]$ cd example/



-bash: cd: example/: Permission denied
```

COPY

COPY

```
[tecmint2@linux tecmint1]$ getfacl example/



# file: example

# owner: tecmint1

# group: tecmint1

user::rwx

user:tecmint1:rwx

user:tecmint2:r--

group::rwx

mask::rwx

other::---
```

Step 7: Now give ‘**execute**‘ permission to ‘**tecmint2**‘ on ‘**example**‘ folder and then use **‘cd**‘ command to see the effect. Now ‘**tecmint2**‘ have the permissions to view and change directory, but don’t have permissions for writing anything.

COPY

COPY

```
[tecmint1@linux tecmint1]$ setfacl -m u:tecmint2:r-x example/

[tecmint1@linux tecmint1]$ getfacl example/



# file: example

# owner: tecmint1

# group: tecmint1

user::rwx

user:tecmint1:rwx

user:tecmint2:r-x

group::rwx

mask::rwx

other::---
```

COPY

COPY

```
[tecmint@linux ~]$ su - tecmint2



Last login: Thu Apr 17 15:09:49 IST 2014 on pts/5
```

COPY

COPY

```
[tecmint2@linux ~]$ cd /tecmint1/

[tecmint2@linux tecmint1]$ cd example/

[tecmint2@linux example]$ getfacl .
```

COPY

COPY

```
[tecmint2@linux example]$ mkdir test



mkdir: cannot create directory ‘test’: Permission denied
```

COPY

COPY

```
[tecmint2@linux example]$ touch test



touch: cannot touch ‘test’: Permission denied
```

Note: After implementing ACL, you will see a extra ‘**+**’ sign for ‘ls –l’ output as below.

COPY

COPY

```
[root@linux tecmint1]# ll

total 4

drwxrwx---+ 2 tecmint1 tecmint1 4096 Apr 17 17:01 example
```

@Secure Files/Directories using ACLs (Access Control Lists) in Linux

WRITTEN BY

Biswaraj Sahoo

--AWS Community Builder | DevOps Engineer | Docker | Linux | Jenkins | AWS | Git | Terraform | Docker | kubernetes

Empowering communities via open source and education