Controlling access to files with ACL

Linux #redhat #files #users

Access control list is another layer that runs on top of your permissions, which allows you to assign permission per user.

What is ACL?

- Access control list (ACL) provides an additional, more flexible permission mechanism for file systems. It is designed to assist with UNIX file permissions. ACL allows you to give permissions for any user or group to any disc resource.

Use of ACL:

- Think of a scenario in which a particular user is not a member of group created by you but still you want to give some read or write access, how can you do it without making user a member of group, here comes in picture Access Control Lists, ACL helps us to do this trick
- Basically, ACLs are used to make a flexible permission mechanism in Linux.
- From Linux man pages, ACLs are used to define more fine-grained discretionary access rights for files and directories.
- Commands to assign and remove ACL permissions are:
    - `setfacl` and `getfacl`
    - read as "set file ACL" and "get file ACL"
        - `getfacl` will give you the information about the existing permissions of a file
        - `setfacl` will allow you to set permissions the way you want it to set

List of commands for setting up ACL

1. To add permission for a user
    - `setfacl -m u:[user]:[rwx] [/path/to/file]`
        - The `-m` options stands for "modify" and specifies to modify permissions
        - The `u` before the user name stands for "user" and specifies that the following is a user.
            - Note we have to separate `u` from user name and permissions with colons.
2. To add permissions for a group
    - If you give a permission to the group level (the middle three bits), then every group in your Linux system will have that permission, but what if you only want one group to rea it?
    - `setfacl -m g:[group]:[rw] [/path/to/file]`
        - Use the `g` keyword to specify for a group
3. To allow all files or directories to inherit ACL entries from the directory it is within (cascading)
    - `setfacl -Rm "entry" [/path/to/file]`
    - The `-R` option is for recursive
4. To remove a specific entry
    - `setfacl -x u:[user] [/path/to/file]` (For a specific user)
    - The `-x` options specifies to remove an entry.
5. To remove all entries (remove all ACL permissions to everyone)
    - `setfacl -b [/path/to/file]` (For all users)
    - The `-b` option removes all extended ACL entries.

- Note:

- As you assign the ACL permission to a file/directory, it adds + sign at the end of the permission. So when you run `ls -l` the first column which shows you the permissions, it adds the plus sign at the end of it, which shows that it has a ACL assigned to it.
- Setting with the `w` permission with ACL does NOT allow to remove a file. (you can modify but you cannot delete the file)

Checking file ACL permissions

Example using `getfacl` command:

```
[root@localhost tmp]# getfacl texas
```

- Note the file that we want to check its ACL permissions is called "texas"
- Note we are currently root users
- Note the current location is the `/tmp` directory

Output:

```
# file: tx
# owner: root
# group: root
user::rw-
group::r--
other::r--
```

- Any other user not from the root group would fall in the "other" category.
- If you assign the permission `w` (write) to this file then everyone else on this machine will be able to write to that file. To overcome this issue and just assign permissions for this file to a single user instead of everyone else we will use `setfacl`.

Add permission to a user or group

Example using `setfacl` command:

```
[root@localhost tmp]# setfacl -m u:mmarin:rw /tmp/texas
```

- Assigning the `rw` (read and write) permissions to the "mmarin" user.

Output from `getfacl texas`:

```
# file: texas
# owner: root
# group: root
user::rw-
user:mmarin:rw-
group::r--
mask::rw-
```

```
other::r--
```

- To verify we added an ACL user we can run the `getfacl` command followed by the file name.
- Note that now the user mmarin has the right to read and write.

Example using `setfacl` command:

```
[root@localhost ~]# setfacl -m g:mmarin:rw /tmp/texas
```

- Adding `rw` permissions to the "mmarin" group on the `/tmp/texas` file.

Output from `getfacl texas`:

```
# file: texas
# owner: root
# group: root
user::rw-
user:mmarin:rw-
group::r--
group:mmarin:rw-
mask::rw-
other::r--
```

- If any other user is part of the "mmarin" group they could actually go in and modify that file too.

Trying to delete a file when we have `w` permission
Example using `rm` command:

```
[root@localhost tmp]# rm texas
```

- Trying to delete the `/tmp/texas` file

Output:

```
rm: cannot remove 'texas': Operation not permitted
```

- Note that even when we have permission to write on this file we still are not able to delete it.

Cascade down to every file and directory

Example using `setfacl` command:

```
[root@localhost ~]# setfacl -Rm u:mmarin:rw /tmp/texas
```

- Using the `-Rm` option to recursively modify (cascade down) to add the entry to assign `rw` permissions to the mmarin user.

Remove ACL permissions

Example using `setfacl` command:

```
[root@localhost ~]# setfacl -x u:mmarin /tmp/texas
```

- Using the `-x` option to remove all permissions from the mmarin user.
- Note that we cannot do this for each permission individually on that user (Deleting the entry means deleting every permission (`rwx`) from a user or group)

Output from `getfacl texas`:

```
# file: texas
# owner: root
# group: root
user::rw-
group::r--
group:mmarin:rw-
mask::rw-
other::r--
```

- Note there is no other user entry as previously.
- Only the mmarin group is remaining.

What if you want to delete every permission from every user or every group
Example using `setfacl` command:

```
[root@localhost ~]# setfacl -b /tmp/texas
```

Output from `getfacl texas`:

```
# file: texas
# owner: root
# group: root
user::rw-
group::r--
other::r--
```

- You will see it's back to its original state when we created the file.