Advanced Linux File Permissions

1. Sticky bit
2. SUID - [Set User ID]
3. SGID - [Set Group ID]

What is sticky bit?

0011

- Setting the sticky bit on a file is pretty much useless, and it doesn’t do anything. On some of the older *nix flavors, a sticky bit enabled executable file will be loaded to the swap memory after 1st execution, which speeds up all subsequent execution. This is not true anymore.

- If you set the sticky bit to a directory, other users cannot delete or rename the files (or subdirectories) within that directory. When the sticky bit is set on a directory, only the owner and the root user can delete / rename the files or directories within that directory.

Set the sticky bit on Directory

- The example below enables the sticky bit on a directory.
- Use chmod command to set the sticky bit. If you are using the octal numbers in chmod, give 1 before you specify other numbered privileges, as shown below.
  - The example below, gives rwx permission to user, group and others (and also adds the sticky bit to the directory).

```
$ chmod 1777 dir
```

  - Or, you can assign only sticky bit to an existing directory (without touching any other user, group and other privileges) using chmod command as shown below.

```
$ chmod +t dir
```

- Once the sticky bit is assigned to a directory, you'll see (t) as the last character in the permission. In this example, it is drwxrwxrwt.

$ ls -ld /home/bala/dir

drwxrwxrwt 2 bala bala 4096 2011-01-28 14:09 /home/bala/dir

$ ls -l dir

total 8

0011

-rwxrwxrwx 1 bala bala 20 2011-01-28 14:12 bala.txt

-rwxrwxrwx 1 guest guest 41 2011-01-28 14:13 guest.txt

In the above example, as dir has rwx permission to everybody, all other users are allowed to do create their files or directories under this directory. However, even when the sub-directories or files under dir is having rwx permission to everybody, only the owner of those can delete or rename those files and directory. Other users cannot delete or rename it because of sticky bit.

In the above example, bala.txt has rwx to users, groups, and others. But, when guest user is trying to delete the file bala.txt, he'll see the "Operation not permission" message as shown below.

```
$ su guest
Password:
$ cd /home/bala/dir1
$ rm bala.txt
rm: cannot remove `bala.txt': Operation not permitted
```

Please note that /tmp has sticky bit enabled by default. Now you know why /tmp directory is supposed to have sticky bit enabled.

```
$ ls -ld /tmp
drwxrwxrwt 3 root root 4096 Jan 31 08:29 /tmp
```

To remove the sticky bit from a directory, do the following.

```
$ chmod -t dir
```

- Consider you have a directory " test ".

chmod 777 test

0011

  - This gives permissions for all the users to read, write and execute.

chmod +t test

  - Sticky bit set

- Example: ls -al

drwxrwxrwt	2 a1	a1 4096	Jun 13 2008	.
-rw-rw-r--	1 a1	a1 0	Jun 11 17:30	1.txt
-rw-rw-r--	1 b2	b2 0	Jun 11 22:52	2.txt

- From the above example a1 is the owner of the test directory.

  a1 can delete or rename the files 1.txt and 2.txt.

  b2 can delete or rename the file 2.txt only.

SUID - [Set User ID]

SUID bit is set for files (mainly for scripts).

The SUID permission makes a script to run as the user who is the owner of the script, rather than the user who started it.

Example:

If a1 is the owner of the script and b2 tries to run the same script, the script runs with the ownership of a1.

If the root user wants to give permissions for some scripts to run by different users, he can set the SUID bit for that particular script.

So if any user on the system starts that script, it will run under the root ownership.

$chmod u+s file1 OR chmod 4764 file1

$ls –l file1

-rwsrw-r-- 1 a1 a1 0 Jun 11 17:30 file1

SGID - [Set Group ID]

- If a file is SGID, it will run with the privileges of the files group owner, instead of the privileges of the person running the program.
  This permission set also can make a similar impact. Here the script runs under the groups ownership.
- You can also set SGID for directories.
  Consider you have given 2777 permission for a directory. Any files created by any users under this directory will come as follows.

Example:

-rw-rw-r-- 1 b2 a1 0 Jun 11 17:30 1.txt

In the above example you can see that the owner of the file 1.txt is b2 and the group owner is a1.
So both b2 and a1 will have access to the file 1.txt.

Now lets make this more interesting and complicated.
Create a directory "test". Chmod it to 2777. Add sticky bit to it.

Example:

mkdir test
chmod 2777 test
chmod +t test

- ls -al test

  drwxrwsrwt 2 a1 a1 4096 Jun 13 2008 test
- From the above permission set you can understand that SGID and sticky bit is set for the folder "test". Now any user can create files under the test directory.
- Example:

  drwxrwsrwt 2 a1 a1 4096 Jun 13 2008 .

  -rw-rw-r-- 1 b2 a1 0 Jun 11 17:30 1.txt

  -rw-rw-r-- 1 c3 a1 0 Jun 11 17:30 2.txt

  -rw-rw-r-- 1 d4 a1 0 Jun 11 17:30 3.txt

- So the a1 user has access to all the files under the test directory. He can edit, rename or remove the file.
  b2 user has access to 1.txt only, c3 has access to 2.txt only...
- If sticky bit was not set for the test directory, any user can delete any files from the test directory, since the test directory has 777 permissions.
  But now it not possible.
- Example:
  If d4 tries to remove 1.txt

```
rm -f 1.txt
rm: cannot remove `1.txt': Operation not permitted
```