Advanced File Permission

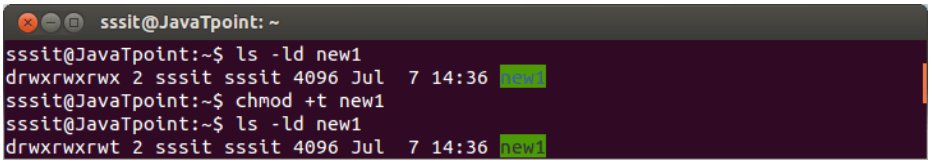
Sticky Bit on Directory

Files can be protected in a directory from getting removed by other users who do not own it by preventing it with **sticky bit**. It is displayed at the same location as the **x** permission for **others.**

It is represented by a **t** (x is also there) or a **T** (no x is there).

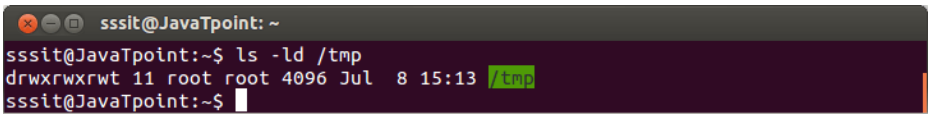
Example:

```
chmod +t new1
```



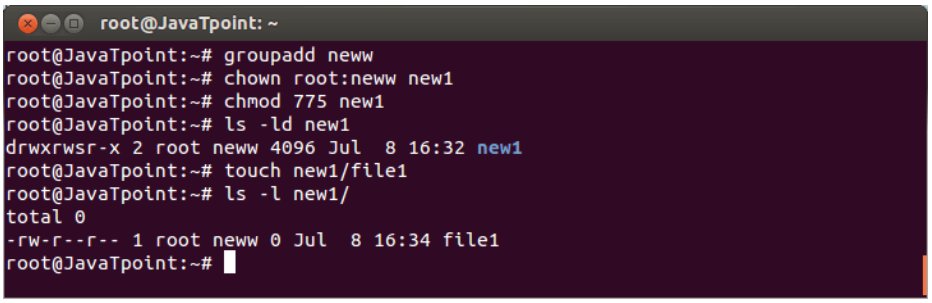
Look at the above snapshot, permission for file **new1** is changed to **t** at place of **x** permission for **others.**

Generally, sticky bit is found on **/tmp** directory.



setgid Bit on Directory

To make sure all the files in the directories are owned by the group owner of directory, setgid can be used. It is displayed on the same location as **x** permission for **group**. It is represented by a **s** (x is also there) or a **S** (no x is there).



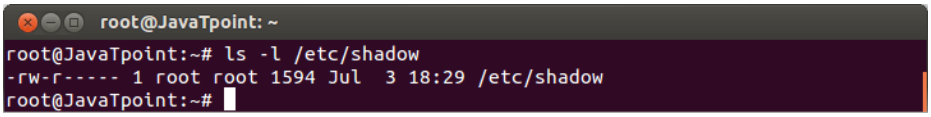
Look at the above snapshot, group owner is changed into **neww** for the directory **new1. Group** permission is changed into **s** at the place of **x** permission for **group**. You can see that **file1** which is inside directory **'new1'** has the group name as **'neww'**.

⇧

setgid and setuid on Regular Files

With the help of these two permissions, an executable file is accessed with the permissions of the **file owner** instead of the **executing owner**. It means that if a program has root user and setuid permission is set on it, then a user will run that program as root. This can be dangerous as well as good for the security.

For example, passwords which are stored in **/etc/shadow** are readable by root only as shown below.



When a user run **passwd** command, it executes with the root credentials.

← prev next →

Please Share

Learn Latest Tutorials

Tensorflow | Data Mining | Xamarin | Ansible | Matplotlib
Wireshark | Git | Jupyter | R | Bash

SetUID, SetGID, and Sticky Bits in Linux File Permissions

As explained in the article **Permissions in Linux**, Linux uses a combination of bits to store the permissions of a file. We can change the permissions using the `chmod` command, which essentially changes the 'r', 'w' and 'x' characters associated with the file.

Further, the ownership of files also depends on the `uid` (user ID) and the `gid` (group ID) of the creator, as discussed in this article. Similarly, when we launch a process, it runs with the `uid` and `gid` of the user who launched it.

1. The `setuid` bit

This bit is present for files which have executable permissions. The `setuid` bit simply indicates that when running the executable, it will set its permissions to that of the user who created it (owner), instead of setting it to the user who launched it. Similarly, there is a `setgid` bit which does the same for the `gid`.

To locate the `setuid`, look for an 's' instead of an 'x' in the executable bit of the file permissions.

An example of an executable with `setuid` permission is `passwd`, as can be seen in the following output.

```
ls -l /etc/passwd
```

This returns the following output:

```
-rwsr-xr-x root root 2447 Aug 29  2018 /etc/passwd
```

As we can observe, the 'x' is replaced by an 's' in the user section of the file permissions.

To set the `setuid` bit, use the following command.

```
chmod u+s
```

To remove the `setuid` bit, use the following command.

```
chmod u-s
```

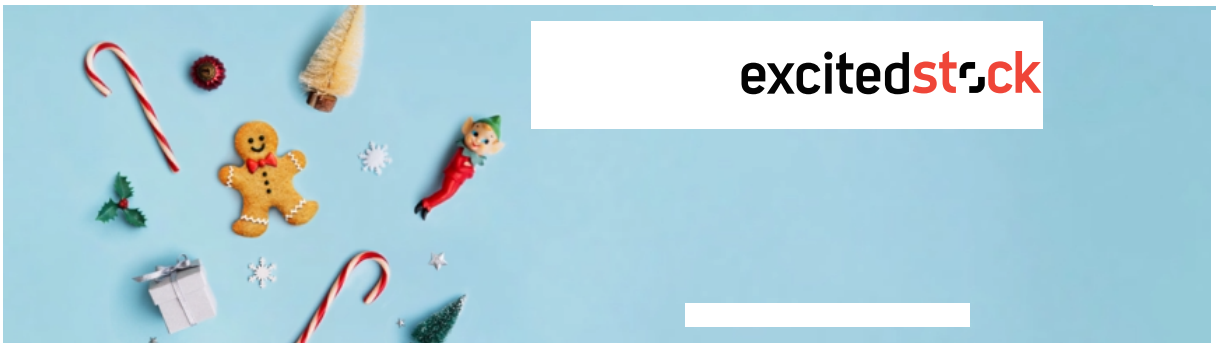
2. The `setgid` bit

The `setgid` affects both files as well as directories. When used on a file, it executes with the privileges of the group of the user who owns it instead of executing with those of the group of the user who executed it.
When the bit is set for a directory, the set of files in that directory will have the same group as the group of the parent directory, and not that of the user who created those files. This is used for file sharing since they can be now modified by all the users who are part of the group of the parent directory.

To locate the `setgid` bit, look for an 's' in the group section of the file permissions, as shown in the example below.

```
-rwxrwsr-x root root 1427 Aug 2 2019 sample_file
```

To set the `setgid` bit, use the following command.




```
chmod g+s
```

To remove the `setgid` bit, use the following command.

```
chmod g-s
```

Security Risks

The `setuid` bit is indeed quite useful in various applications, however, the executable programs supporting this feature should be carefully designed so as to not compromise on any security risks that follow, such as buffer overruns and path injection. If a vulnerable program runs with root privileges, the attacker could gain root access to the system through it. To dodge such possibilities, some operating systems ignore the `setuid` bit for executable shell scripts.

3. The sticky bit

The sticky bit was initially introduced to 'stick' an executable program's text segment in the swap space even after the program has completed execution, to speed up the subsequent runs of the same program. However, these days the sticky bit means something entirely different.

When a directory has the sticky bit set, its files can be deleted or renamed only by the file owner, directory owner and the root user. The command below shows how the sticky bit can be set.

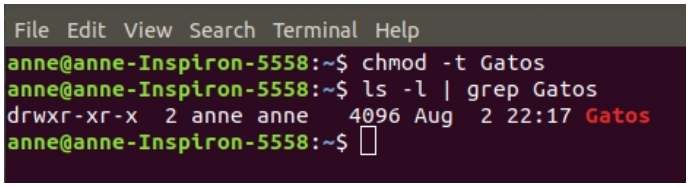
```
chmod +t
```

Simply look for a 't' character in the file permissions to locate the sticky bit. The snippet below shows how we can set the sticky bit for some directory "Gatos", and how it prevents the new user from deleting a file in the directory.

```
File  Edit  View  Search  Terminal  Help
anne@anne-Inspiron-5558:~$ ls -l | grep Gatos
drwxr-xr-x  2 anne anne   4096 Aug  2 22:17 Gatos
anne@anne-Inspiron-5558:~$ chmod +t Gatos
anne@anne-Inspiron-5558:~$ ls -l | grep Gatos
drwxr-xr-t  2 anne anne   4096 Aug  2 22:17 Gatos
anne@anne-Inspiron-5558:~$ su newuser
Password:
newuser@anne-Inspiron-5558:/home/anne$ cd Gatos
newuser@anne-Inspiron-5558:/home/anne/Gatos$ rm cat_1.jpg
rm: remove write-protected regular file 'cat_1.jpg'? Y
rm: cannot remove 'cat_1.jpg': Permission denied
newuser@anne-Inspiron-5558:/home/anne/Gatos$ exit
```

To remove the sticky bit, simply use the following command.

```
chmod -t
```



Since deleting a file is controlled by the write permission of the file, practical uses of the sticky bit involve world-writable directories such as '/tmp' so that the delete permissions are reserved only for the owners of the file.

Recommended Posts:

Permissions in Linux
File globbing in Linux
Linux File Hierarchy Structure
proc file system in Linux
file command in Linux with examples
How to copy a file's content from Linux terminal?
Linux Virtualization : Linux Containers (lxc)
Path Name in File Directory
Difference between File and Folder
Unix File System
Network File System (NFS)
File Allocation Methods
Run Levels in Linux
Linux Commands
Linux | Nmon

Anannya Uberoi 1
Check out this Author's contributed articles.

If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please Improve this article if you find anything incorrect by clicking on the "Improve Article" button below.

Article Tags : Linux-Unix | Operating Systems | linux-command