SPECIAL FILE PERMISSIONS

Special permissions make up a fourth access level in addition to **user**, **group**, and **other**.

Special permissions allow for additional privileges over the standard permission sets (as the name suggests).

Types of Special permissions

1. Setuid (SUID) - 4
2. Setgid (SGID) - 2
3. Stickybit – 1

suid	sgid	stickybit
rwx	r-x	r-x
s	s	t
rws	r-s	r-t
rws	r-S	r-T

chmod u+s file/directory

chmod g+s file/directory

chmod o+t file/directory

chmod 4755 file

chmod 2755

chmod 1744 file

chmod 0755

user + s (pecial)

- Commonly noted as **SUID**, the special permission for the user access level has a single function: A file with **SUID** always executes as the user who owns the file, regardless of the user passing the command
- Whenever SETUID permission has set on executable files, anyone executing that command (file) will inherit the permissions of the owner of the file. Its numeric value is 4.
- Now, to see this in a practical light, let's look at the **/usr/bin/passwd** command. This command, by default, has the SUID permission set:

[root@server ~]$ ls -l /usr/bin/passwd

-rwsr-xr-x. 1 root root 33544 Dec 13 2019 /usr/bin/passwd

Note the **s** where **x** would usually indicate execute permissions for the user. if a file does not have execute permissions then suid is represented by S.

To set the suid to a file

a. chmod 4755 <filename>
b. chmod u+s <filename>
c. chmod u-s <filename>

To check the files with suid set to s

a. which pwd -- > /usr/bin/passwd
b. [root@server ~]$ ls -l /usr/bin/passwd
c. [root@server ~]$ ls -l /bin/su
    a. **-rwsr-xr-x. 1 root root 57840 May 26 2022 /bin/su**
d. [root@server ~]$ ls -l /bin/mount
    a. **-rwsr-xr-x. 1 root root 49624 May 26 2022 /bin/mount**
e. [root@server ~]$ ls -l /usr/bin/crontab
    a. **-rwsr-xr-x. 1 root root 57800 May 31 2022 /usr/bin/crontab**
f. [root@server ~]$ ls -l /usr/bin/chage
    a. **-rwsr-xr-x. 1 root root 74384 Apr 25 2022 /usr/bin/chage**
g. [root@server ~]$ ls -l /usr/bin/sudo
    a. **---s--x--x. 1 root root 185456 Aug 26 2021 /usr/bin/sudo**
h. [root@server ~]$ ls -l /bin/umount
    a. **-rwsr-xr-x. 1 root root 37256 May 26 2022 /bin/umount**

    [root@machine-1 ~]# which useradd

    /usr/sbin/useradd

    [root@machine-1 ~]# which userdel

    /usr/sbin/userdel

    [root@machine-1 ~]# which fdisk

    /usr/sbin/fdisk

    [root@machine-1 ~]# which whoami

/usr/bin/whoami

For all the excutable files in /bin or /sbin the suid is set to s.

group + s (pecial)

Commonly noted as SGID, this special permission has a couple of functions:

- If set on a file, it allows the file to be executed as the group that owns the file (similar to SUID)
- If set on a directory, any files created in the directory will have their group ownership set to that of the directory owner
- The SetGID permission displays as an “s” in the group executable field. Its numeric value is 2

To set the GUID

a. chmod 2755 <file/dirname>
b. chmod g+s <file/dirname>
c. chmod g-s <file/dirname>

To check the files with GUID set to s

a. which locate
b. which wall

[root@linux ~]# which locate

/usr/bin/locate

[root@linux ~]# ls -l /usr/bin/locate

-rwx--s--x. 1 root slocate 41032 Aug 10 2021 /usr/bin/locate

other + t (sticky)

- The last special permission has been dubbed the "sticky bit.
- This permission does not affect individual files. However, at the directory level, it restricts file deletion.
- Only the **owner** (and **root**) of a file can remove the file within that directory.
- This special permission prevents to delete other user's file from public directories.
- Its numeric value is 1.
- A common example of this is the /tmp directory
  [root@linux ~]# ls -ld /tmp

  drwxrwxrwt. 32 root root 4096 Oct 20 13:28 /tmp

To set the sticky bit -t

a. chmod 1755 <file/dirname>
b. chmod o+t <file/dirname>
c. chmod o-t <file/dirname>