Set user id bit:

There are two users,user1 and user2.

* A text file,say,”foo.txt” is first created which contains some arbitrary string(eg.,”Hello world!”)
* Next, we write a C program to read the file. The owner of the program at this instance is user1.
* #include<stdio.h>  
  #include<stdlib.h>  
  #include<sys/types.h>  
  #include<sys/stat.h>  
  #include<fcntl.h>  
  #include<unistd.h>  
  int main()  
  {  
    int fd;

printf("Real User ID:%d ",getuid());

printf(“\nEffective user id:%d”,geteuid());  
  char  \*x=(char \*)malloc(25\*sizeof(char));  
  fd=open("foo.txt",O\_RDONLY);  
  read(fd,x,10);  
  printf("%s\n",x);  
 }

User1 is the owner of the text file as well.

This program(suid.c) is compiled using gcc –o suid suid.c to create an executable file called suid.

* The ownership of the program and the text file can be changed using the “chown” function. We can give ownership to either root or another user, which is related to the current owner by the others relation.

chown root foo.txt or chown user2 foo.txt

We want to access the file using user2, so the second command is given.

Now we first move ownership from user1 to root using the sudo bash command.

Using the command “chown user2 suid” user2 becomes the owner of suid.

* Now we can set permissions for the textfile using the chmod command. For example, “chmod 700 foo.txt” grants read, write and execute permissions (rwx) to owner and none of the permissions to group or others.
* “ls –ln ” shows the different types of permissions given to the text file.
* Now we use the chmod 4755 test, where 4 or 100 is the set user id bit, which is set. The set user id bit allows other users to access the file by converting their effective user ids to the owner real user id.
* On using ls –ln command,the file permissions for suid are observed as: rwsr\_xr\_x, where s denotes the ser user id bit.
* Now, we move back to user1. Though user1 does not have direct permission to access foo.txt, user1 can read the file by running the suid program. This happens because, as the executable C file has set user id bit, and user1 has permissions to execute suid, the effective user id of user1 becomes equal to the real user id of user2.(1001 in case of user2, and 0 if root is the second user).However, if the set user id bit was not set, the effective user id of user1 becomes equal to the real user id of user1 itself, and as user1 does not have direct permissions to neither the C program file nor the text file, the message “Permission Denied” comes up on the screen when suid is executed by user1.

SET GROUP ID BIT:

* Let us consider two users, user1 belongs to Group 1 and user2 belongs to Group2.
* We create a text file using user1, and the permissions are set using the command “chmod 770 foo.txt”, which implies that both user 1 and Group1 have all rwx permissions for the text file.
* Another C program is written to read the text file using user1, which is quite similar to that in case of set user id bit, only the getuid() and geteuid() functions are replaced by getguid() and getegid() functions. An executable file is created by compiling this program, say sgid. User1 is the owner of this file.
* After this, the permissions of sgid are set using “chmod 2775 sgid”, where 2(010 in binary) is the set group id bit.
* Ls –ln gives the access permissions for foo.txt as: rwx rwx \_\_\_,thus group users have full access o the file.
* The user is switched to user2 . As the sgid file has execute permissions for others, the program is executed. Also, as the set group id bit is set, the effective group id becomes equal to the owner group id, i.e. the group id of user1.

Thus ,though an user from a different group does not have direct permissions to read the text file, it can be accessed by a C program using the set group id bit .