

CSIS0230A Principle of Operating Systems(Class A)
Notes for Tutorial 11
Unix Security

1. Linux authentication: UID and GID

When users log in, their usernames are mapped to integers marking their “UID” (for “user id”) and the “GID”s (for “group id”) that they are a member of. The mappings are found in `/etc/passwd` and `/etc/group` files respectively. UID 0 is a special privileged user (role) traditionally called “root”, who can overrule most security checks and is used to administer the system. Processes are the only “subjects” in terms of security (that is, only processes are active objects). Processes can access various data objects, in particular filesystem objects (FSOs), System V Interprocess Communication (IPC) objects, and network ports.

2. Related Process Attributes

Every process has a set of security-relevant attributes, including:

- RUID, RGID - real UID and GID of the user on whose behalf the process is running. (i.e. the UID and GID of the user who launch this process)
- EUID, EGID - effective UID and GID used for privilege checks (except for the filesystem)
- FSUID, FSGID - effective UID and GID used for privilege checks for the filesystem
- SUID, SGID - Saved UID and GID; to save a UID or GID so that your program can switch to it later.

These attributes are stored in the PCB of each process, i.e., their *task_struct*. A program can get the currently used EUID and RUID using the *geteuid()* and *getuid()* functions. For example:

```
#include <unistd.h>
#include <sys/types.h>
#include <stdlib.h>

int main() {
    printf("RUID is %d. EUID is %d.\n", getuid(), geteuid());
    return 0;
}
```

Using the *setresuid*, *setfsuid*, *setregid* and *setfsgid* functions, the user id’s can be arbitrarily swapped. But for most programs, RUID and RGID store the actual identification of the user running the program, while EUID, EGID, FSUID and FSGID store the identification currently in effect. The “real”, “effective” and “file system” identification are usually the same, but sometimes they can be different. In later sections we explain how this can happen.

3. Filesystem Object Attributes

Unix filesystems store the following attributes in inodes of files. They are checked when a process open a file for reading or writing, or creating, renaming or unlinking a directory entry.

- Owner UID and GID - identifies the “owner” of the filesystem object. Only the owner or `root` can change the access control attributes.
- *read*, *write*, *execute* bits for each of user (owner), group, and other. For ordinary files, *read*,

write, and execute have their typical meanings. For directories, the “read” permission is needed to list a directory. The “execute” permission is sometimes called “search” permission, and is needed to actually change to the directory or to use its entries. The “write” permission permits adding, unlinking (i.e., removing) and renaming files in the directory. The permission values of symbolic links are never used; it’s only the values of their containing directories and the linked-to file that matter.

- *setuid, setgid* bits. They can be used for executable files to switch privileges (see below). The *setgid* bit is also be used for directories: when it is set, new files created in the directory inherit the group id the parent directory instead of the creator’s group id. Linux (indeed, System V) also use this bit for non-executable files, which requests for mandatory file locking (see *Documentation/mandatory.txt* of the kernel directory for information).
- *Sticky* bit. This bit is historically used to hint the filesystem to keep executable files in the memory, but is nowadays not used for that purpose. Instead, sticky bit is used only in directory. When it is set, non-owners of the directory cannot rename or unlink files that are not owned by himself. This is generally used in public directories like `/tmp`.

To change the permission of the file object, you can use the `chmod` command. For more information, please refer to man page.

4. Switching Privileges

It is possible to run a program with the privileges (EUID and FSUID) different from its Real UID (the user who launched it). To do this, the *setuid* bit of the executable file is turned on. The *setuid* bit is represented with an `s` when displaying the rights with the `ls` command :

```
>> ls -l /bin/su
-rwsr-xr-x 1 root root 14124 Aug 18 1999 /bin/su
>>
```

When the kernel runs a program with the *setuid* bit on, it uses the program owner’s identity as the EUID and FSUID for the process. On the other hand, the RUID doesn’t change, and is still the user who launched the program. For instance, every user can have access to the `/bin/su` command, but it runs under its owner’s identity (`root`) with every privilege on the system. Needless to say one must be very careful when writing a program with this attribute.

Similar to *setuid*, the *setgid* bit asks the kernel to use the file owner’s GID, instead of the user’s GID, as the EGID and FSGID of the process. This is usually done to grant access to a file or device only via specific programs, by granting access of a file or device to a group accessible only through a *setgid* program.

5. An Example

In this section we run an example to demonstrate the *setuid*. First, login as `root`. Compile the program list in Section 2 by the following command:

```
gcc -o testuid testuid.c
```

Then, use the `ls -l` command to see the attributes of the compile program:

```
-rwxr-xr-x 1 root root 13980 Nov 19 16:33 testuid*
```

Then, copy the file a directory which can be access by all other users, e.g. /usr/local/bin, by the following command:

```
cp testuid /usr/local/bin
```

After that, run this program by root, you will found the following output:

```
RUID is 0. EUID is 0.
```

Try to run the same program using normal user, you will get something similar to the following:

```
RUID is 501. EUID is 501.
```

Then, use root to set the program's setgid bit by the following command:

```
chmod u+s /usr/local/bin/testuid
```

Then, run the same program using normal user, you will get the following output:

```
RUID is 501. EUID is 0.
```

You HAVE ROOT ACCESS!!!!

6. References

<http://www.linuxfocus.org/English/January2001/article182.shtml>,
<http://www.theorygroup.com/Theory/FAQ/Secure-Programs-HOWTO-3.html>