Linux Setuid Programs and Their User IDs

Last updated: November 26, 2022

Written by: baeldung (https://www.baeldung.com/linux/author/baeldung)

Administration (https://www.baeldung.com/linux/category/administration)

Processes (https://www.baeldung.com/linux/category/processes)

ps (https://www.baeldung.com/linux/tag/ps)

1. Overview

Modern operating systems such as Linux provide measures to control access to their resources. One such way is the possibility to change the user of the process. **Hence, the change of privilege level follows.**

In this tutorial, **we'll learn about the setuid special file permission and three kinds of user IDs to support it.**

2. The Setuid Programs

By default, the process' user is the user who runs it. **Thus, the process is privileged as much as the user is.** However, Linux offers the setuid bit functionality, **which allows changing the process ownership to the file owner.** It is especially useful when the file belongs to *root*.

Let's apply this feature to an executable file *test_id*:

```
$ ls -all test_id    (/linux/)          (https://www.baeldung.com/linux/)
-rwxrwxr-x. 1 joe joe 23984 Apr 11 12:01 test_id
```

First, let's change the owner of the file to *root* with *chown* (/linux/chown-chmod-permissions):

```
$ sudo chown root test_id
$ ls -all test_id
-rwxrwxr-x. 1 root joe 23984 Apr 11 12:01 test_id
```

Then we should use *chmod (/linux/chown-chmod-permissions) u+s* to set the *s* bit:

```
$ sudo chmod u+s test_id
$ ls -all test_id
-rwsrwxr-x. 1 root joe 23984 Apr 11 12:01 test_id
```

With *ls (https://man7.org/linux/man-pages/man1/ls.1.html) -all*, let's check that the *s* bit appears in the place of the regular *x* executable bit of the owner. Perhaps the best-known setuid programs are *sudo* (/linux/sudo-command) and *passwd* (https://man7.org/linux/man-pages/man1/passwd.1.html).

3. Three Kinds of Linux User IDs

The operating system manages the process' ownership with three kinds of user ID:

- RUID – the real user ID identifies the user who runs the program
- EUID – the effective ID tells the kernel about the privileges of the process
- SUID – the saved user ID used when the process changes its UID

Let's explain that when the executable with the 'x' bit starts, the EUID is set to the real one. In contrast, **an executable with the special 's' bit has an initial EUID equal to the file's owner ID.**

SUID comes into play when the privileged process drops its privileges to do some ordinary user's work. **Subsequently, the process may ask the kernel to restore its initial level of privilege defined by SUID.**

4. How to Check User IDs of Process

We're going to show the real and effective user of the process with the *ps* (/linux/ps-command) command. Let's assume that we run our setuid *test_id* program:

```
$ ps -C test_id -o pid,tty,ruser,user,cmd
    PID TT       RUSER    USER     CMD
   3250 pts/1    joe      root     ./test_id
```

Let's notice that the real user, listed as *RUSER*, is different from the effective one shown in the *USER* column.

Now let's query the *proc* (https://man7.org/linux/man-pages/man5/proc.5.html) pseudo filesystem with *grep* (/linux/grep-sed-awk-differences) to obtain information about the process' UIDs.

```
$ grep "^Uid" /proc/3250/status
Uid:    1000    0       0       0
```

In detail, the meaning of IDs is, from left to right: real, effective, saved, and filesystem UID.

5. Bash Scripts Security Limitations

Modern Linux distribution doesn't allow setting the setuid bit on Bash script. **We should know that the scripts are prone to attack in many ways.**

Because the system environmental variables may be implicitly accessed by the script, **modifications to *PATH* or *IFS* (/linux/ifs-shell-variable) may lead to running malicious code.**

Moreover, **an attacker may exploit the time gap between calling the Bash interpreter and executing the script to replace its content.**

(/linux/) (https://www.baeldung.com/linux/)

6. Conclusion

In this article, we talked about the setuid programs with the ability to do privileged work for ordinary users. Then, we learned how to set the *s* bit for the executable file. Subsequently, we looked through UIDs used by the kernel to handle setuid programs.

Finally, we listed security objections to turning Bash scripts into such programs.

Comments are closed on this article!

CATEGORIES

ADMINISTRATION (/LINUX/CATEGORY/ADMINISTRATION)
FILES (/LINUX/CATEGORY/FILES)
FILESYSTEMS (/LINUX/CATEGORY/FILESYSTEMS)
INSTALLATION (/LINUX/CATEGORY/INSTALLATION)
NETWORKING (/LINUX/CATEGORY/NETWORKING)
PROCESSES (/LINUX/CATEGORY/PROCESSES)
SCRIPTING (/LINUX/CATEGORY/SCRIPTING)
SEARCH (/LINUX/CATEGORY/SEARCH)
SECURITY (/LINUX/CATEGORY/SECURITY)
WEB (/LINUX/CATEGORY/WEB)

SERIES

LINUX ADMINISTRATION (/LINUX/LINUX-ADMINISTRATION-SERIES)
LINUX FILES (/LINUX/LINUX-FILES-SERIES)

LINUX PROCESSES (/LINUX/LINUX-PROCESSES-GUIDE)

(/linux/) (https://www.baeldung.com/linux/)

ABOUT

ABOUT BAELDUNG (/ABOUT)
THE FULL ARCHIVE (HTTPS://WWW.BAELDUNG.COM/LINUX/FULL_ARCHIVE)
EDITORS (HTTPS://WWW.BAELDUNG.COM/EDITORS)

TERMS OF SERVICE (HTTPS://WWW.BAELDUNG.COM/TERMS-OF-SERVICE)
PRIVACY POLICY (HTTPS://WWW.BAELDUNG.COM/PRIVACY-POLICY)
COMPANY INFO (HTTPS://WWW.BAELDUNG.COM/BAELDUNG-COMPANY-INFO)
CONTACT (/CONTACT)