SUID Binaries in GNU/Linux: the Feature or the Bug?

Andrey M. Kanner
Department of Information Security
Moscow Institute of Physics and Technology (MIPT)
Dolgoprudny, Russia
kanner@mail.ru

Tatiana M. Kanner
Department of Information Security
Moscow Institute of Physics and Technology (MIPT)
Dolgoprudny, Russia
sheikot@mail.ru

***Abstract*—The paper considers current security issues of GNU/Linux related to the possibility to locally escalate user privileges and caused by errors in the operation of SUID programs. It provides some examples of recently identified vulnerabilities in the operation of various GNU/Linux components. Despite the fact that such vulnerabilities have already been eliminated by the main manufacturers of operating systems, SUID programs are the primary cause of their occurrence, that are still widely used in modern software solutions. There are numerous reasons for using such programs, the main of which is their historical use as part of the Unix operating system, which means that they are inherited in GNU/Linux. The authors offer possible options to solve security problems associated with the use of SUID programs that have already repeatedly arose in the past and will undoubtedly arise in the future. In the conclusion, a SUID-related dilemma is emphasized – one can either continue to use them posing a high risk of future information security problems, or restrict their use, violating the existing standards for OS compatibility.**

***Index Terms*—GNU/Linux, SUID, setuid, local privilege escalation.**

I. Introduction

The core feature contributing to the wide use of GNU/Linux, both for personal and corporate use, is the ability to customize the OS in accordance with the end user's existing needs. This is mainly due to the fact that the operating system source code is open and accessible for changes. That is why GNU/Linux grow, change and improve at a fast pace. The list of GNU/Linux distributions includes a fairly large number of various versions. Nowadays, such OSs are increasingly used in various organizations to solve a wide range of problems not only on local computers, but also in terminal and virtual systems. Moreover, along with the openness, organizations value other advantages such as reliability and stability of the operating system. Such advantages often lead to a false perception of GNU/Linux as a completely safe operating system [1], [2].

However, GNU/Linux, like other operating systems, need to ensure security of the data processed therein. There is a number of protected OSs having security mechanisms integrated at the source code level, for example, Astra Linux Special Edition and Rosa Cobalt[1]. Moreover, there are additional software, as well as software and hardware data security tools protecting from unauthorized access (DST protecting from UAA) for GNU/Linux [3], [4]. However, vulnerabilities are frequently revealed even for secured GNU/Linux and for OS having additional DST protecting from UAA, demonstrating the need to review or finalize the principles and mechanisms of protecting such operating systems. The vulnerabilities recorded and published in the information security threat data pool include the vulnerability related to SUID programs widely used in GNU/Linux [5], [6].

The privilege escalation mechanism implemented by such programs has been used in GNU/Linux for quite a long period [7], [8], and there have been numerous attempts to exclude or replace it [9]–[12]. However, such attempts have not lead to the desired result – SUID programs are still used in GNU/Linux.

II. Materials and Methods

SUID Programs are executable files in Unix-like OSs[2], including GNU/Linux, which contain setuid access rights attribute in addition to the execution attribute. This attribute changes the user identifier during the program execution and allows users to launch executable files with their owner's rights. Such programs are used when it is required for unprivileged users to perform some actions, which may only be performed with administrative access rights (Root Superuser). For example, if a user having user rights is required to change the password, he/she needs to launch a binary that has the following attributes:

/usr/bin/passwd – '-rwsr-xr-x',

This command contains an access vector with Suid bit – *"s"*. In the presence of such a vector in the SUID program, the user process will have privileges of the file owner, that is, the admin (Root Superuser), and not only user privileges.

As everyone knows, hash folds from user passwords in GNU/Linux are stored in */etc/shadow*, which is only available for changing by the Root Superuser, therefore, the SUID program is used to change the user password.

[1]https://astralinux.ru/os/ and https://rosa.ru/rosa-cobalt/

[2]setuid bit was invented by Dennis Ritchie and included in su - McIlroy, M.D., A Research Unix reader: annotated excerpts from the Programmer's Manual. 1971.

Other administrative actions for GNU/Linux include, for example, setting up and managing network connections, accessing devices, as well as rebooting the OS. All such actions may seem quite ordinary for end users, but for server solutions, these are extremely crucial operations that must not be available to unprivileged users.

As a rule, when developing SUID programs, certain programming techniques are used to counteract privilege escalation [13], [14]. For example, the process for /usr/bin/passwd must escalate privileges in such a way that only the password of such a user may be changed, who initiated the launch of this utility. However, there is always a human factor leading to the emergence of vulnerabilities in SUID programs.

Such vulnerabilities include GNU/Linux-related vulnerabilities, associated with both the disadvantages of access control in attribute processing, and with buffer overflow in dynamic memory.

The most significant of the SUID-related security threats identified in the recent years and published in the information security threats data pool [15] are the following ones: CVE-2023-0386, CVE-2023-4911 and CVE-2021-4034.

CVE-2023-0386 is the vulnerability of the stat() function of Overlayfs subsystem of the Linux kernel, which is associated with the disadvantages of access control in setuid and setgid attribute processing. Use of such a vulnerability can allow the offender to escalate their privileges. The essence of the vulnerability is as follows: if uid/gid of the mounted OverayFS file system and the user system do not correspond, when copying files through the *"cp -a"* command, files with non-existenting uid/gid are created [16].

CVE-2023-4911 (having a unique name Looney Tunables) is the vulnerability of the dynamic ld.so glibc library loader, associated with the buffer overflow in dynamic memory. Use of the vulnerability can allow the offender to execute an arbitrary code with escalated privileges, by launching binaries with SUID attributes and creating a GLIBC_TUNABLES environment variable [17].

CVE-2021-4034 is the vulnerability of Polkit library caused by the buffer overflow in the stack. Use of the vulnerability can allow the non-authenticated user, that is, the offender, to acquire the rights of the administrator (superuser privileges). At the same time, the Polkit demon can be either launched or not launched. To use the vulnerability, it is only important to have the vulnerable component available in the OS [18].

At the time of their official disclosure, all of the above-listed vulnerabilities affected many versions of GNU/Linux released over the previous few years. CVE-2021-4034 affected all the GNU/Linux versions, that had been using the vulnerable component from the moment of their first release in May 2009 (commit *c8c3d83("Add a pkexec(1) command"))*, that is, for 12 years. Such vulnerabilities are often not given so much attention due to the fact that they only allow a local privilege escalation, but not remote one. However, for a number of companies, for example, those providing hosting with GNU/Linux, local privilege escalation is no less urgent than remote one.

To counteract such vulnerabilities, it is required to control the operations of the Linux kernel related to the changes in process privileges. In Accord-X, the data security tool protecting from unauthorized access for GNU/Linux developed by the author, neither setuid operations at the level of the Linux kernel lead to a change in the user privileges if the process did not participate in the identification/authentication procedure [19]. This is implemented by access control rules of Accord-X. Therefore, even in case of an attempt to use the previously mentioned vulnerabilities in the GNU/Linux environment with the pre-installed and correctly configured Accord-X, user privileges will not be changed due to the security policy of the software and hardware complex. This is confirmed in Accord-X starting from *v1.2 (2015)*. For a long period of time, all versions of this software and hardware complex have been blocking SUID-related vulnerabilities before their occurrence.

III. Results and Discussion

Thus, there are at least two ways to solve endless problems with SUID programs. One can completely restrict their use or, at least, reduce the list of SUID programs used to reasonable limits, as proposed in [10]. Or one can resist SUID-related vulnerabilities, limiting *setuid()* operations [19]. All other ways to solve SUID-related vulnerabilities may not be called comprehensive, and as a result, we continue to face new vulnerabilities.

In the future, the authors plan to introduce the possibility of strengthening protective properties of the developed complex Accord-X in order to ensure the impossibility of unauthorized privilege change within the OS, and not only within the access policy of Accord-X. However, such an increase in protective properties can lead to the impossibility of using standard Linux applications executing setuid – su/sudo and other SUID programs, graphic subsystem – X11/wayland, gdm/lxdm and others. In other words, in order to exclude security problems related to the long-used GNU/Linux component, Posix[3]/SUS/LSB[4] specifications for this OS will have to be violated, which means that such an OS can no longer be called GNU/Linux.

References

[1] B. Spengler. (2020) 10 Years of Linux Security. 10_years_of_linux_security.pdf. [Online]. Available: https://grsecurity.net/

[2] D. Vyukov. (2020) The state of the Linux kernel security. The_state_of_the_Linux_kernel_security.pdf. [Online]. Available: https://github.com/ossf/wg-securing-critical-projects/blob/main/presentations/

[3] A. M. Kanner, T. M. Kanner, "Modeling and verification of the access control subsystem of Accord-X data security tool," *Information Security Questions*, vol. 3, pp. 6–10, 2020, (in Russian).

[3]IEEE Std 1003.1, 2018 Edition. The Open Group Technical Standard Base Specifications, Issue 7. URL: http://pubs.opengroup.org/onlinepubs/9699919799/

[4]Linux Standard Base (LSB). The Linux Foundation. URL: http://refspecs.linuxfoundation.org/lsb.shtml

[4] A. M. Kanner, “Model and algorithms for secure access control in GNU/Linux,” Ph.D. dissertation, Ural Federal Univ., Ekaterinburg, Nov. 2023. [Online]. Available: https://dissovet2.urfu.ru/mod/data/view.php?d=12&mode=single&page=383# , (in Russian).
[5] M. Payer, T. Hartmann, and T. R. Gross, “Safe Loading – A Foundation for Secure Execution of Untrusted Programs,” in *Proc. IEEE Symposium on Security and Privacy*, 2012, pp. 18-32. DOI: 10.1109/SP.2012.11
[6] B. Jain, C. C. Tsai, J. John, and D. E. Porter, “Practical techniques to obviate setuid-to-root binaries,” in *Proc. of the Ninth European Conference on Computer Systems*, 2014, pp. 1-14. DOI: 10.1145/2592798.2592811
[7] A. M. Robachevsky, S. A. Nemnugin, and O. L. Stesik, *“UNIX operating system”*, Saint Petersburg: BHV-Petersburg, 2010.
[8] M. S. Dittmer and M. V. Tripunitara, “The UNIX Process Identity Crisis: A Standards-Driven Approach to Setuid,” in *Proc. of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS ’14)*, Association for Computing Machinery, 2014, pp. 1391-1402. DOI: 10.1145/2660267.2660333
[9] N. Provos, “Improving Host Security with System Call Policies,” in *Proc. of USENIX Security Symposium*, 2003, pp. 257-272.
[10] Gentoo Linux, suidctl feature, 2023. [Online]. Available: https://devmanual.gentoo.org/eclass-reference/make.conf/index.html
[11] H. Chen, D. Wagner, and D. Dean, “Setuid demystified,” in *Proc. of 11th USENIX Security Symposium (USENIX Security 02)*, 2002.
[12] R. A. Napier, “Secure automation: achieving least privilege with SSH, Sudo and Setuid,” in *Proc. of 18th Large Installation System Administration Conference*, 2004, pp. 203-212.
[13] E. S. Raymond, *“The art of Unix programming. Security Wrappers and Bernstein Chaining”*, Addison-Wesley Professional, 2003.
[14] M. Bishop, *“How To Write a Setuid Program”*, 2001.
[15] Information security threat data pool. [Online]. Available: https://bdu.fstec.ru/ , (in Russian).
[16] Vulnerability CVE-2023-0386. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2023-0386
[17] Vulnerability CVE-2023-4911. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2023-4911
[18] Vulnerability CVE-2021-4034. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-4034
[19] A. M. Kanner, “Linux: process life cycle and access control,” *Information Security Questions*, vol. 4, pp. 37–40, 2014, (in Russian).