



VISVESVARAYA NATIONAL INSTITUTE OF TECHNOLOGY



TREASURE HUNT

Writeup by- IronHawk717

CONTENTS

Exiftool.....	2
Hashcat/crackstation.....	3
python Script.....	4
fuzzing/dirb.....	5
hash crack (.rar).....	7
Volatility,pstree.....	8
Wireshark.....	9

Exiftool

Our first Question Starts with this Image and we have our first flag there, as well as hint shared with the challenge “Use Strings/ Exiftool”. Which clearly suggests us to use Exiftool to view the metadata of this Image and proceed further.



SHELL/Only if we understand, can we care. Only if we care, we will help. Only if we help, we shall be saved."/>

checking the file type using the `file` command, it's simple png file.

```
~/Downloads/Shell_Treasure_hunt
ls
flag1.png
~/Downloads/Shell_Treasure_hunt
file flag1.png
flag1.png: PNG image data, 1530 x 1080, 8-bit/color RGBA, non-interlaced
```

using `exiftool` command:

```
~/Downloads/Shell_Treasure_hunt
exiftool flag1.png
ExifTool Version Number      : 13.10
File Name                   : flag1.png
Directory                   : .
File Size                   : 2.7 MB
File Modification Date/Time : 2025:04:19 10:02:35-04:00
File Access Date/Time       : 2025:04:19 10:03:23-04:00
File Inode Change Date/Time: 2025:04:19 10:02:58-04:00
File Permissions            : -rw-rw-r--
File Type                   : PNG
File Type Extension         : png
MIME Type                   : image/png
Image Width                 : 1530
Image Height                : 1080
Bit Depth                   : 8
Color Type                  : RGB with Alpha
Compression                 : Deflate/Inflate
Filter                      : Adaptive
Interlace                   : Noninterlaced
Significant Bits           : 8 8 8
Software                     : gnome-screenshot
Creation Time               : Sun 16 Feb 2025 07:10:05 PM IST
Comment                      : YAY you are on your way! Take this and go on: https://shellpwn.super.site/birth
Warning                     : [minor] Trailer data after PNG IEND chunk
Image Size                  : 1530x1080
Megapixels                  : 1.7
```

we found url for our next flag- ‘<https://shellpwn.super.site/birth>’

On visiting the site we found our next Flag



SHELL/"A child is carefree until the world teaches them what to fear and what to desire."/

nd got link for next challenge

Good Job!

Proceed to the next level - <https://hastebin.com/share/etivoquguo.sql>

HASHCAT/CRACKSTATION

In this challenge we got a key in hash form and some base64 words, encrypted with XOR. So we need to crack the hash and find the key first.

```

1  HELLO
2
3
4
5
6
7
8  SEEKER
9
10
11
12
13
14
15
16 We see that you have come in quest of a flag.
17 But all we have are some random Base64 words. They were encrypted with XOR Cipher and a key.
18 You can have the key. The only problem is that it's hashed.
19 Find some way to crack the hash, get the key and unlock this random text.
20
21 ↗ b3380ac2ab5dbf1ed81e18163db6a9d8
22
23 📁 zc3NyuC+goOZypqYhY2Yi4fKj4SJhY6PmcqZhYePyp6Pkp7Kg4SehcqAn4eIho+0yp2FmI6ZxMq9j8qfmY+OyoOeyp6Fyo+E1YW0j8qZhYePnoKDHl3KmY+JmI+exMzK4L6Cj8qamIWNmTuHyoWfnpqfnnsqd1
24
25 HINT: https://gchq.github.io/CyberChef/

```

I'll be using Hashcat to crack the hash. first we will copy that hash in a file `hash.txt`

```

~/Downloads/Shell_Treasure_hunt
nano hash.txt

```

```
~/Downloads/Shell_Treasure_hunt
hashcat -m 0 hash.txt /usr/share/wordlists/rockyou.txt --show
b3380ac2ab5dbf1ed81e18163db6a9d8:earthling
```

as we can see our hash is successfully cracked using hashcat and our key is: **earthling**

now we will use CyberChef to decrypt XOR and base64.

The screenshot shows the CyberChef interface with the following configuration:

- Input:** A long base64 encoded string: `zc3Ny...qfjeqj5ieyo+LiYLK...56C...4S...`
- XOR:** Key is set to **earthling**.
- Output:** The decrypted text is a Python script:

```
def reverse_text(text):
    """Reverse the text."""
    return text[::-1]

# Original text
text = ????

# Apply obfuscation steps one over another
step1 = encode_ascii(text)          # Convert to ASCII values
step2 = shift_ascii_values(step1)    # Shift ASCII values
step3 = ascii_to_chars(step2)       # Convert back to characters
step4 = shuffle_text(step3)         # Shuffle characters
step5 = reverse_text(step4)         # Reverse the final string

# Print results
print(f"Original Text: {text}")
print(f"Step 1 - ASCII Encoding: {step1}")
print(f"Step 2 - Shifted ASCII: {step2}")
```

Python Script

we have decrypted the text and we can see it contain a python script which we simply need to reverse to get the output.

```
1- def unshuffle(text):
2-     half = (len(text) + 1) // 2
3-     evens = text[:half]
4-     odds = text[half:]
5-     result = []
6-     for i in range(len(text)):
7-         if i % 2 == 0:
8-             result.append(evens[i // 2])
9-         else:
10-             result.append(odds[i // 2])
11-     return "".join(result)
12
13- def decode(encrypted):
14-     reversed_text = encrypted[::-1]
15-     unshuffled_text = unshuffle(reversed_text)
16-     ascii_encoded = "-".join(str(ord(c)) for c in
17-                               unshuffled_text)
18-     shifted_back = "-".join(str(int(num) - 3) for num in
19-                               ascii_encoded.split("-"))
20-     original_text = "".join(chr(int(num)) for num in
21-                               shifted_back.split("-"))
22-     return original_text
23- encrypted = "|lllqsh2wvusvqsok2=swooevr vu h1hx1zohv2vwk"
24- flag = decode(encrypted)
25- print("Decoded Text:", flag)
```

Decoded Text: <https://shellpwn.super.site/responsibility>
== Code Execution Successful ==

after reversing we get our flag and link for next challenge:
<https://shellpwn.super.site/responsibility>



SHELL/“For most of history, man has had to fight nature to survive; in this century he is beginning to realize that, in order to survive, he must protect it.”

Directory Busting/Fuzzing

There we got our next challenge and hint that we need to fuzz the site url and find the hidden pages. it can be done by dirbuster,gobuster,ffuf and many other tools.we will be using ffuf in this chall. and we are also provided with wordlist which we can download and use.

Well Done!

Now it gets a little intense . . .

It seems there's a page on this site that no one has even opened.

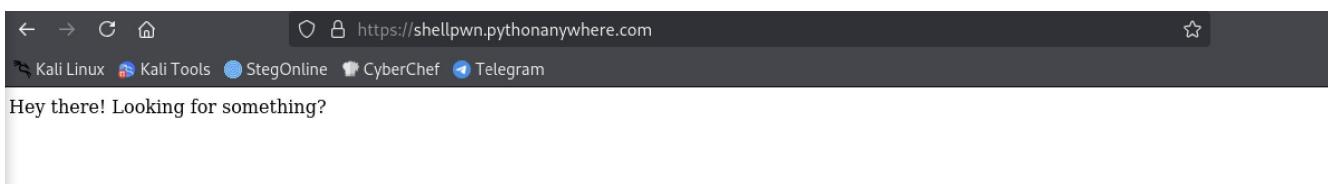
It's something like https://shellpwn.pythonanywhere.com/???????

Your job is to guess it.

`$> ffuf`

This list might help. You can learn more about it here

we will try to visit the site first



nothing interesting, now we will proceed for dirbusting.

here instead of given wordlist we used pre-provided wordlist in Kali Linux.

```
~/Downloads/Shell_Treasure_hunt
└─ ffuf -w /usr/share/wordlists/dirb/common.txt -u https://shellpwn.pythonanywhere.com/FUZZ
    / \ \ / \ \ / \ \ 
    ^ ^ ^ ^ ^ ^ ^ ^ 
    \| \| \| \| \| \| 
v2.1.0-dev

:: Method      : GET
:: URL         : https://shellpwn.pythonanywhere.com/FUZZ
:: Wordlist    : FUZZ: /usr/share/wordlists/dirb/common.txt
:: Follow redirects : false
:: Calibration   : false
:: Timeout       : 10
:: Threads       : 40
:: Matcher       : Response status: 200-299,301,302,307,401,403,405,500

[Status: 200, Size: 33, Words: 5, Lines: 1, Duration: 233ms]
secret      [Status: 200, Size: 582, Words: 93, Lines: 17, Duration: 212ms]
:: Progress: [4614/4614] :: Job [1/1] :: 189 req/sec :: Duration: [0:00:26] :: Errors: 0 ::
```

after successful iterations we found hidden directory `/secret`.We will open the final link: <https://shellpwn.pythonanywhere.com/secret> and claim our flag.

SHELL{“Many men go fishing all of their lives without knowing that it is not fish they are after.”}



SHELL{“Many men go fishing all of their lives without knowing that it is not fish they are after.”}

Hash crack(.rar)

and on the same page we got our next challenge with some hints

T Have some rest 🍷💻

Then go on to the next page <https://shellpwn.pythonanywhere.com/???>

But I forgot the last few characters :P

The last few characters were the process ID of the suspicious process that was running on [this](#) machine

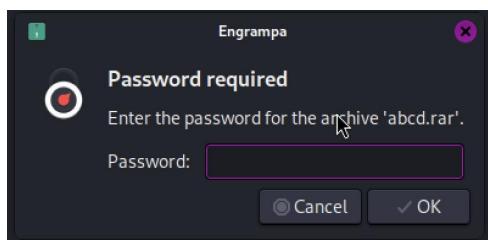
Hint:

Use Volatility imageinfo, pstree

Hint:

Use Volatility imageinfo, pstree

after downloading the .rar file when we go to extract the file it pops up with a password req field, means we need password to extract the file, but we don't have any passwords.



We will extract hash of this file and crack that hash to obtain our password using either hashcat or john the ripper. as we have already covered hashcat this time we will go with John the Ripper.

```
~/Downloads/Shell_Treasure_hunt
rar2john abcd.rar > hash2.txt

~/Downloads/Shell_Treasure_hunt
cat hash2.txt
abcd.rar:$RAR3$*0*a8dc2fea1b70d3d0*f3f2cc585e03b173caa3a7b81850b161:0::::abcd.rar
```

cracking hash using `rockyou.txt` wordlist

```
~/Downloads/Shell_Treasure_hunt
john --wordlist=/usr/share/wordlists/rockyou.txt hash2.txt
Using default input encoding: UTF-8
Loaded 1 password hash (rar, RAR3 [SHA1 128/128 ASIMD 4x AES])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
infected      (abcd.rar)
1g 0:00:01:12 DONE (2025-04-19 11:16) 0.01377g/s 343.7p/s 343.7c/s 343.7C/s apostol..24862486
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Found our password: `infected`

Volatility, pstree

now we will extract the file from abcd.rar
nd it is found to be: **0zapftis.vmem** which is a memory dump file
we need to find process id(PID) of sus process.

first we need to find what kind of os the memory belong to

```
~/Downloads/Shell_Treasure_hunt
python2 volatility/vol.py -f 0zapftis.vmem imageinfo
INFO : volatility.debug : Determining profile based on KDBG search ...
Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
    AS Layer1 : IA32PagedMemoryPae (Kernel AS)
    AS Layer2 : FileAddressSpace (/home/ironhawk717/Downloads/Shell_Treasure_hunt/0zapftis.vmem)
    PAE type : PAE
        DTB : 0x319000L
        KDBG : 0x80544ce0L
Number of Processors : 1
Image Type (Service Pack) : 2
    KPCR for CPU 0 : 0xffdff000L
    KUSER_SHARED_DATA : 0xffdf0000L
Image date and time : 2011-10-10 17:06:54 UTC+0000
Image local date and time : 2011-10-10 13:06:54 -0400
```

we found the profile **WinXPSP2x86** , we need this because volatility
need correct profile to parse memory structure correctly.

```
~/Downloads/Shell_Treasure_hunt
python2 volatility/vol.py -f 0zapftis.vmem --profile=WinXPSP2x86 pstree
Name          Pid  PPid  Thds  Hnds Time
0x819cc830:System      4     0   55   162 1970-01-01 00:00:00 UTC+0000
. 0x81945020:smss.exe 536    4    3   21 2011-10-10 17:03:56 UTC+0000
.. 0x816c6020:csrss.exe 601    536   11  355 2011-10-10 17:03:58 UTC+0000
.. 0x813a9020:winlogon.exe 632    536   24  533 2011-10-10 17:03:58 UTC+0000
... 0x816da020:services.exe 676    632   16  261 2011-10-10 17:03:58 UTC+0000
.... 0x817757f0:svchost.exe 916    676    9  217 2011-10-10 17:03:59 UTC+0000
.... 0x81772ca8:vmacthlp.exe 832    676    1  24 2011-10-10 17:03:59 UTC+0000
.... 0x816c6da0:svchost.exe 964    676   63  1058 2011-10-10 17:03:59 UTC+0000
..... 0x815c4da0:wscntfy.exe 1920   964    1  27 2011-10-10 17:04:39 UTC+0000
..... 0x815e7be0:wuauctl.exe 400    964    8  173 2011-10-10 17:04:46 UTC+0000
..... 0x8167e9d0:svchost.exe 848    676   20  194 2011-10-10 17:03:59 UTC+0000
..... 0x81754990:VMwareService.e 1444   676    3  145 2011-10-10 17:04:00 UTC+0000
..... 0x8136c5a0:alg.exe 1616   676    7  99 2011-10-10 17:04:01 UTC+0000
..... 0x813aeda0:svchost.exe 1148   676   12  187 2011-10-10 17:04:00 UTC+0000
..... 0x817937e0:spoolsv.exe 1260   676   13  140 2011-10-10 17:04:00 UTC+0000
.... 0x815daca8:svchost.exe 1020   676    5  58 2011-10-10 17:03:59 UTC+0000
... 0x813c4020:lsass.exe 688    632   23  336 2011-10-10 17:03:58 UTC+0000
0x813bcda0:explorer.exe 1956  1884   18  322 2011-10-10 17:04:39 UTC+0000
. 0x8180b478:VMwareUser.exe 192    1956   6  83 2011-10-10 17:04:41 UTC+0000
. 0x817a34b0:cmd.exe 544    1956   1  30 2011-10-10 17:06:42 UTC+0000
. 0x816d63d0:VMwareTray.exe 184    1956   1  28 2011-10-10 17:04:41 UTC+0000
. 0x818233c8:reader_sl.exe 228    1956   2  26 2011-10-10 17:04:41 UTC+0000
```

finally viewing parent-child relation of processes using pstree.
we found: **0x813bcda0:explorer.exe** as suspecious process(According to
me it is sus because it dosn't have any parent,nd its subprocess is
cmd.exe directly, feels like a malware)

finally completing our link: <https://shellpwn.pythonanywhere.com/1956>



SHELL/“Beliefs are dangerous. Beliefs allow the mind to stop functioning. A non-functioning mind is clinically dead.”

we got our flag and next chall as well.

WIRESHARK

I guess we are coming to an end now. Just one last stretch . . .
There are things hidden in [this](#) traffic.
Find them and move ahead

Hint1:

Use Wireshark

Hint2:

Use It's an HTTP GET Request (look for a url)

we download the file and open it using wireshark

No.	Time	Source	Destination	Protocol	Length	Info
29	0.680383374	127.0.0.53	127.0.0.1	DNS	125	Standard query response 0x1fd6 PTR 251.0.0.224.in-addr.arpa PTR mdns
30	0.682967175	127.0.0.53	172.25.139.16	DNS	157	Standard query response 0xac0 No such name PTR 3.136.25.172.in-addr.arpa PTR mdns
31	0.683458375	127.0.0.53	127.0.0.1	DNS	157	Standard query response 0xa135 No such name PTR 3.136.25.172.in-addr.arpa PTR mdns
32	0.764663837	172.25.135.5	172.25.143.255	NBNS	94	Name query NB SIDDHESH<1c>
33	0.925433968	172.25.139.16	35.173.69.207	HTTP	512	GET /7018 HTTP/1.1
34	1.142063656	172.25.136.3	224.0.0.251	MDNS	87	Standard query 0x0000 PTR _microsoft_mcc._tcp.local, "QM" question
35	1.142262286	fe80::59cf:5b09:7f8.. ff02::fb	MDNS	107	Standard query 0x0000 PTR _microsoft_mcc._tcp.local, "QM" question	
36	1.147554208	35.173.69.207	172.25.139.16	HTTP	723	HTTP/1.1 200 OK (text/html)
37	1.147678526	172.25.139.16	35.173.69.207	TCP	68	41356 - 80 [ACK] Seq=445 Ack=656 Win=491 Len=0 TStamp=1416339787 TSec=62 Who has 172.25.140.37? Tell 172.25.129.206
38	1.177776896	CompalInform_19:ec:..	ARP			
39	1.223756057	172.25.139.16	35.173.69.207	HTTP	523	GET /static/flag6.png HTTP/1.1
40	1.227614962	HP_3c:67:34	ARP			
41	1.253087904	172.25.136.147	172.25.143.255	UDP	596	61310 - 3490 Len=552
42	1.391599567	172.25.128.15	255.255.255.255	UDP	192	42446 - 7989 Len=148
43	1.446987481	35.173.69.207	172.25.139.16	HTTP	296	HTTP/1.1 304 NOT MODIFIED
44	1.447080720	172.25.139.16	35.173.69.207	TCP	68	41356 - 80 [ACK] Seq=900 Ack=884 Win=490 Len=0 TStamp=1416340086 TSec=62 Who has 172.25.140.37? Tell 172.25.130.13
45	1.519494974	HP_58:aa:6e	ARP			
46	1.528638746	172.25.135.5	172.25.143.255	NBNS	94	Name query NB SIDDHESH<1c>
47	1.552633631	ASIXElectron_10:da:..	ARP			
48	1.603229412	127.0.0.1	127.0.0.53	DNS	100	Standard query 0x6f03 PTR 159.129.25.172.in-addr.arpa OPT

we have network traffic, displaying packets with protocol, lengths etc.

according to hint we will filter out http requests

No.	Time	Source	Destination	Protocol	Length	Info
33	0.925433968	172.25.139.16	35.173.69.207	HTTP	512	GET /7018 HTTP/1.1
36	1.147554268	35.173.69.207	172.25.139.16	HTTP	723	HTTP/1.1 200 OK (text/html)
39	1.223756057	172.25.139.16	35.173.69.207	HTTP	523	GET /static/flag6.png HTTP/1.1
43	1.446987481	35.173.69.207	172.25.139.16	HTTP	296	HTTP/1.1 304 NOT MODIFIED
317	231.121070329	172.25.139.16	35.173.69.207	HTTP	512	GET /7018 HTTP/1.1

we can clearly see request 39. is GET request with /static/flag6.png

we will open this and analyze properly

```

▶ Frame 39: 523 bytes on wire (4184 bits), 523 bytes captured (4184 bits) on
▶ Linux cooked capture v1
▶ Internet Protocol Version 4, Src: 172.25.139.16, Dst: 35.173.69.207
▶ Transmission Control Protocol, Src Port: 41356, Dst Port: 80, Seq: 445, Ack
▼ Hypertext Transfer Protocol
  ▶ GET /static/flag6.png HTTP/1.1\r\n
    Host: shellpwn.pythonanywhere.com\r\n
    Connection: keep-alive\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, li
    DNT: 1\r\n
    Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8\r
    Referer: http://shellpwn.pythonanywhere.com/7018\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: en-US,en;q=0.9\r\n
    If-Modified-Since: Wed, 12 Feb 2025 06:27:01 GMT\r\n
\r\n
  [Response in frame: 43]
  [Full request URI: http://shellpwn.pythonanywhere.com/static/flag6.png]
```

very easily we found the full request URI nd visiting this will give us our FINAL flag.



SHELL/"The question is not, Can they reason? nor, Can they talk? but, Can they suffer?"/

..THANKYOU..