BITSCTF For - Black hole (10 points)

Author: Brandon Everhart
Date: Feb 2017

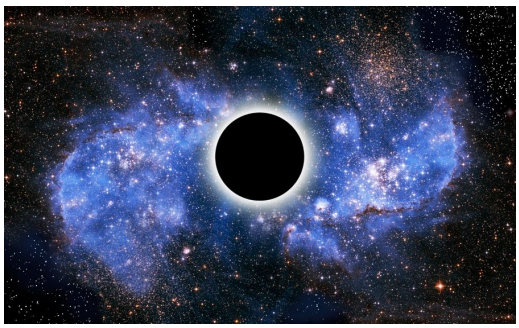
This was the first forensics challenge for BITSCTF. In a forensics challenge, we know that we are typically going to find the flag hidden in the data/information provided to us. Here is the challenge description:

"We are trying to study the Black Holes. One of the most controversial theory related to Black Holes is the "Information Loss Paradox". Calculations suggest that physical information could permanently disappear in a black hole. But this violates the quantum theory. To test the hypothesis, we sent our flag encoded in Base64 format towards a Black hole. With the help of our Hubble Space Telescope, we got some pictures of the Black Hole. See if you can recover the flag form this information."

In the description, the creator of the challenge has given us a very useful piece of information: "... we sent our flag encoded in Base64 format …". First approach: find the base64 encoded flag - seems pretty simple. I downloaded the file that was provided, black_hole.jpg. Next, I ran the 'file *filename*' command to get basic information about the file. (Yes, I know this file has the extension .jpg, but it is always wise to use other tools to check file information outside of just reading the extension).

```
brandon@BlueBerry:~/ctf/bitsctf$ file black_hole.jpg
black_hole.jpg: JPEG image data, JFIF standard 1.02, aspect ratio, density 1x1, segment length 16, baseline, precision 8, 900x564, fra
mes 3
```

This time the file is to be what it seems, a JPEG image. Well, I have a jpeg image, so a logical step would be to look at the image (left). Nothing helpful in the image either. So just like in the previous problem, I used the strings command to look at all the printable characters in the file. The right image is the last section of the output from 'strings *filename*'.



```
?1dRNz
}5?A
[F"x
2P_k
xmu7
M$Wh
UQklUQ1RGe1M1IDAwMTQrODF9
ZI;Z+
e!K]z
>}v#y=
&XSlP
7*qm
```

Looking through the strings output we see one line looks different than all the other lines, UQklUQ1RGe1M1IDAwMTQrODF9. The challenge mentioned the flag being base64 encoded

and this line looks a lot like base64. My prefered method to decode base64 is with python.

```
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> "UQklUQ1RGe1M1IDAwMTQrODF9".decode('base64')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/lib/python2.7/encodings/base64_codec.py", line 42, in base64_decode
    output = base64.decodestring(input)
  File "/usr/lib/python2.7/base64.py", line 328, in decodestring
    return binascii.a2b_base64(s)
binascii.Error: Incorrect padding
```

Since that didn't work let's see what else we can find about the file. The next command I used was 'xxd *filename*'. This command will print the hexdump of the file along with all the printable characters in line with their corresponding hex values.

```
000669b0: ab32 3f10 6a7a 75b6 91a8 daca ab3d 86b9  .2?.jzu......=..
000669c0: 25c4 114a 9e6d b999 5a36 0cf0 3663 7fba  %..J.m..Z6..6c..
000669d0: 010c a432 fca4 1181 4515 b6f4 97a9 e055  ...2....E......U
000669e0: 516b 6c55 5131 5247 6531 4d31 4944 4177  QklUQ1RGe1M1IDAw
000669f0: 4d54 5172 4f44 4639 0000 0000 0000 0000  MTQrODF9........
00066a00: 94a1 5a49 3b5a 2bf2 47ad fc65 f0f7 87a0  ..ZI;Z+.G..e....
00066a10: f07f ecdf e20d 3342 d374 4d5b e217 c30b  ......3B.tM[....
00066a20: bbcf 184f a3c6 d0c5 7779 06b7 7f69 1ce2  ...O....wy...i..
00066a30: df71 8a16 16f6 d029 10a2 2b15 2eca 5d9d  .q.....)..+...].
```

In the output from xxd we again see the same string of characters that were of interest earlier. But this time the leading 'U' is on a line by itself; maybe it's not part of the string? Base64 encoded strings must have a length that is a multiple of 4, and if we remove the 'U' we now have a string that is a multiple of 4. Let's try decoding with python again.

```
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> "QklUQ1RGe1M1IDAwMTQrODF9".decode('base64')
'BITCTF{S5 0014+81}'
```

There's our flag! **BITCTF{S5 0014+81}**