ATOTC 1

50

Our network analysts found exfil using PowerPoint presentations (.pptx). One example of what they found is this encrypted presentation. What is the password?

Fun fact: A Tale of Two Cities (ATOTC) is actually three books, freely downloadable on the Internet.

File: an analyst zipped up the encrypted presentation file; you can download it here.

Download the file and extract the contents to reveal a password protected PPT. We need to run office2john to extract the hash to crack.

```
kali@kali:~$ /usr/share/john/office2john.py Downloads/A_Tale_of_Two_Cities.zip
Downloads/A_Tale_of_Two_Cities.zip : zip container found, file is unencrypted?, invalid OLE file!
kali@kali:~$ /usr/share/john/office2john.py 5ctf/ATOTC/A_Tale_of_Two_Cities.pptx
A_Tale_of_Two_Cities.pptx:$office$*2013*100000*256*16*152fcc24f5f60b2b514e7aa234333f2f*4ea3f42e0dfc5dde0200318075f45cc0*0529c715ee217945d2338d1fc7a7abd00ceca118a6362066466cd574fb09c90a
```

Hash:
A_Tale_of_Two_Cities.pptx:$office$*2013*100000*256*16*152fcc24f5f60b2b514e7aa234333f2f*4ea3f42e0dfc5dde0200318075f45cc0*0529c715ee217945d2338d1fc7a7abd00ceca118a6362066466cd574fb09c90a

```
kali@kali:~/5ctf/ATOTC$ hashcat -a 0 -m 9600 ppt.hash /usr/share/wordlists/rockyou.txt --force
hashcat (v5.1.0) starting ...

OpenCL Platform #1: The pocl project
====================================
* Device #1: pthread-Intel(R) Core(TM) i5-5300U CPU @ 2.30GHz, 1024/3163 MB allocatable, 4MCU

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1
```

Next we need to find a copy of ATOTC in PDF to download to make a wordlist.

https://www.gutenberg.org/files/98/old/2city12p.pdf

The next thing we need is a way to extract the text out of the pdf we just downloaded.

There is a tool called pdftotext that we can use to extract this information.

```
kali@kali:~/5ctf/ATOTC$ pdftotext
pdftotext version 0.71.0
Copyright 2005-2018 The Poppler Developers - http://poppler.freedesktop.org
Copyright 1996-2011 Glyph & Cog, LLC
Usage: pdftotext [options] <PDF-file> [<text-file>]
  -f <int>             : first page to convert
  -l <int>             : last page to convert
  -r <fp>              : resolution, in DPI (default is 72)
  -x <int>             : x-coordinate of the crop area top left corner
  -y <int>             : y-coordinate of the crop area top left corner
  -W <int>             : width of crop area in pixels (default is 0)
  -H <int>             : height of crop area in pixels (default is 0)
  -layout              : maintain original physical layout
  -fixed <fp>          : assume fixed-pitch (or tabular) text
  -raw                 : keep strings in content stream order
  -htmlmeta            : generate a simple HTML file, including the meta information
  -enc <string>        : output text encoding name
  -listenc             : list available encodings
  -eol <string>        : output end-of-line convention (unix, dos, or mac)
  -nopgbrk             : don't insert page breaks between pages
  -bbox                : output bounding box for each word and page size to html.  Sets -htmlmeta
  -bbox-layout         : like -bbox but with extra layout bounding box data.  Sets -htmlmeta
  -opw <string>        : owner password (for encrypted files)
  -upw <string>        : user password (for encrypted files)
  -q                   : don't print any messages or errors
  -v                   : print copyright and version info
  -h                   : print usage information
  -help                : print usage information
  --help               : print usage information
  -?                   : print usage information
```

```
kali@kali:~/5ctf/ATOTC$ pdftotext /home/kali/5ctf/ATOTC/2city12p.pdf out.txt
kali@kali:~/5ctf/ATOTC$ ls -al
total 109400
drwxr-xr-x  2 kali kali      4096 May 28 09:56 .
drwxr-xr-x 10 kali kali      4096 May 28 09:42 ..
-rw-r--r--  1 kali kali   1316140 May 28 09:28 2city12p.pdf
-rw-r--r--  1 kali kali    779109 May 28 09:56 2city12p.txt
-rw-r--r--  1 kali kali 109128192 May 21 15:56 A_Tale_of_Two_Cities.pptx
-rw-r--r--  1 kali kali    779109 May 28 09:56 out.txt
-rw-r--r--  1 kali kali       159 May 27 16:26 ppt.hash
kali@kali:~/5ctf/ATOTC$
```

Now we need to read the file and split the words, then sort and get only the unique words.

```
$ cat out.txt | tr ' ' '\n' | sort -u > wordlist.txt
```

Now that we have a wordlist, we can run this hash through either hashcat or john the ripper.

JtR: /usr/lib/john/john-base-omp --format=office --wordlist=wordlist.txt ppt.hash

Hashcat: hashcat -a 0 -m 9600 -w 3 ppt.hash wordlist.txt –force

```
kali@kali:~/5ctf/ATOTC$ /usr/lib/john/john-base-omp --format=office --wordlist=wordlist.txt ppt.hash
Using default input encoding: UTF-8
Loaded 1 password hash (Office, 2007/2010/2013 [SHA1 128/128 SSE2 4x / SHA512 128/128 SSE2 2x AES])
Cost 1 (MS Office version) is 2013 for all loaded hashes
Cost 2 (iteration count) is 100000 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:18 1.28% (ETA: 10:24:24) 0g/s 22.51p/s 22.51c/s 22.51C/s accessible..accomplices
0g 0:00:02:12 12.91% (ETA: 10:17:55) 0g/s 20.13p/s 20.13c/s 20.13C/s carousing..carrion.
0g 0:00:02:59 16.50% (ETA: 10:18:57) 0g/s 18.65p/s 18.65c/s 18.65C/s comforted,..commenced
Monseigneur      (?)
1g 0:00:08:10 DONE (2020-05-28 10:09) 0.002037g/s 22.32p/s 22.32c/s 22.32C/s Monseigneur..monsieur?'
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

```
$office$*2013*100000*256*16*152fcc24f5f60b2b514e7aa234333f2f*4ea3f42e0dfc5dde0200318075f45cc0*0529c715ee217945d2338d1fc7a7abd00ceca118a6362066466cd574fb09c90a:Monseigneur

Session..........: hashcat
Status...........: Cracked
Hash.Type........: MS Office 2013
Hash.Target......: $office$*2013*100000*256*16*152fcc24f5f60b2b514e7aa ... 09c90a
Time.Started.....: Thu May 28 10:03:06 2020 (11 mins, 8 secs)
Time.Estimated...: Thu May 28 10:14:14 2020 (0 secs)
Guess.Base.......: File (wordlist.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:       18 H/s (42.16ms) @ Accel:1024 Loops:64 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 12288/19399 (63.34%)
Rejected.........: 0/12288 (0.00%)
Restore.Point....: 8192/19399 (42.23%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:99968-100000
Candidates.#1....: here." → patriots;
```

Flag: Monseigneur

ATOTC 2

50

One of our host analysts found an unencrypted version of the file on a compromised host. This version may contain more hidden data. Investigate it in order to answer the rest of this set of questions. What is the flag that is linked?

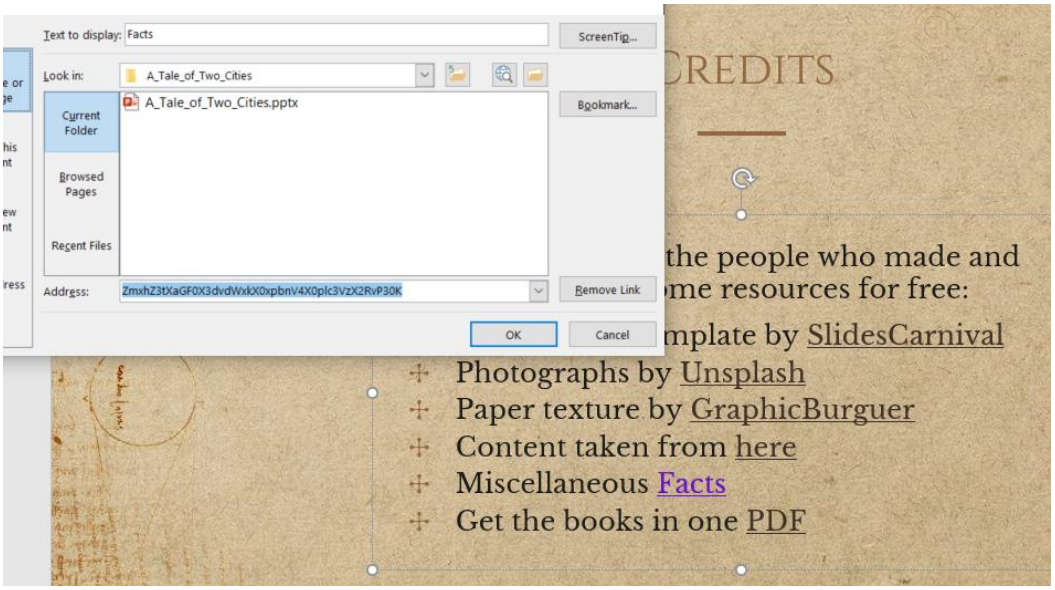
Fun fact: there is an unbelievably large number of ways to hide information in PowerPoint presentations.

File: an analyst zipped up the unencrypted presentation file; you can download it here.

NOTE1: file contents in MS OOXML documents (.pptx in this case) can be finicky to updates or saves which may cause some hidden data to permanently disappear (i.e. never be found again); this may also be true when saving "elements" out of a presentation.

NOTE2: this presentation will work with Microsoft PowerPoint 2016, LibreOffice Impress 5.3, and OpenOffice 4.1.7; it should also work with other versions that supports .pptx.

On the last slide we see a link to facts, when we inspect the link it is base64 encoded text.



In cyberchef we get the following decoding

```
ZmxhZ3tXaGF0X3dvdWxkX0xpbnV4X0plc3VzX2RvP30K
```

Output

```
flag{What_would_Linux_Jesus_do?}
```

Flag: flag{What_would_Linux_Jesus_do?}

ATOTC 3

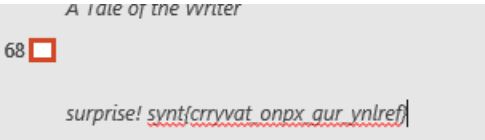
60

Find the layered flag.

Fun fact: it may be a happy time of the year for you, although some grouches consider it ROTTEN.

NOTE: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.

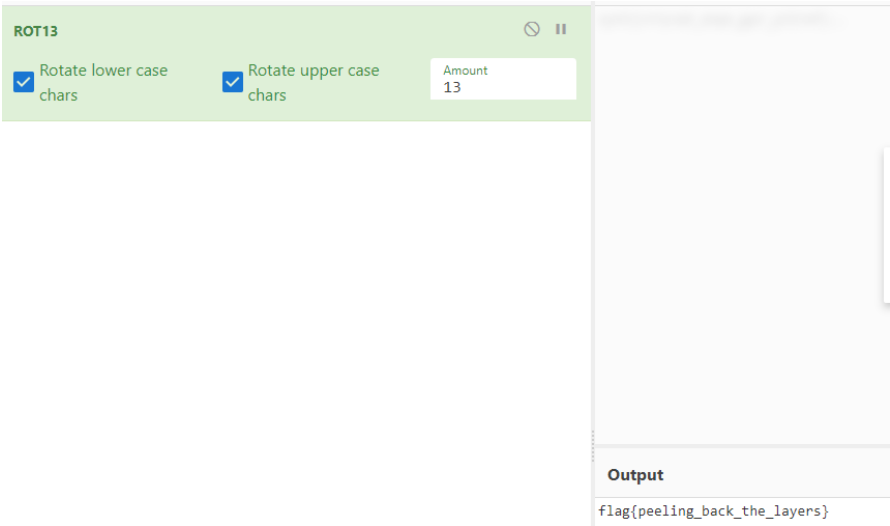
On slide 68 we see this in the outline



This is hidden behind the images

synt{crryvat_onpx_gur_ynlref}

Rot13 this in cyberchef:



flag: flag{peeling_back_the_layers}

ATOTC 4

50

Find the flag that is attributable to the APT.

Fun fact: this is not a real APT, or is it?

NOTE: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.

Look at the author of the document.

Flag: flag{APT_Carl_Toddson}

ATOTC 5

100

Find the hidden flag, and remember that every page counts.

Fun fact: ATOTC is about Paris and London.

NOTE1: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.

NOTE2: be sure to add flag{} around the found flag.

The clue here is that every page counts. We can see that not every slide has a number and the pattern is not even/odd pages. What if we treat this as 1/0 where a page count means 1 and no page count is a 0?

Here are the pages that have number:

2 3 4 6 8 10 14 15 18 19 21 22 23 24 26 27 28 30 32 34 35 37 38 39 42 43 46 50 53 54 56 58 59 62 64 67 72

So, we should have the following:

011101010100011001101111011101010110111001100100010011010110010100100001

Flag: flag{uFoundMe!}

ATOTC 6

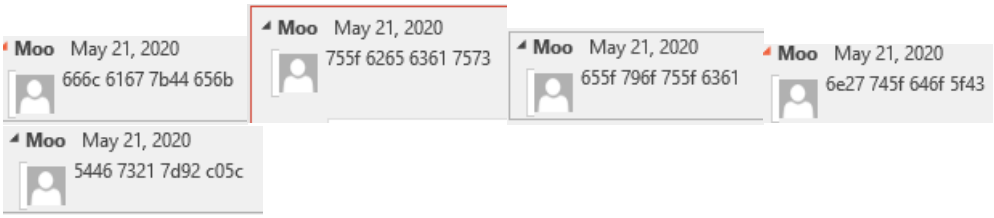
50

Find the hidden flag that can be easily traced.

Fun fact: I plead the fifth… all I'll say is "cows go Moo!"

NOTE: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.

In the comments we see hex characters on slides 5, 10, 15, 20, 25



666c 6167 7b44 656b 755f 6265 6361 7573 655f 796f 755f 6361 6e27 745f 646f 5f43 5446 7321 7d92 c05c

Flag: flag{Deku_because_you_can't_do_CTFs!}

ATOTC 7

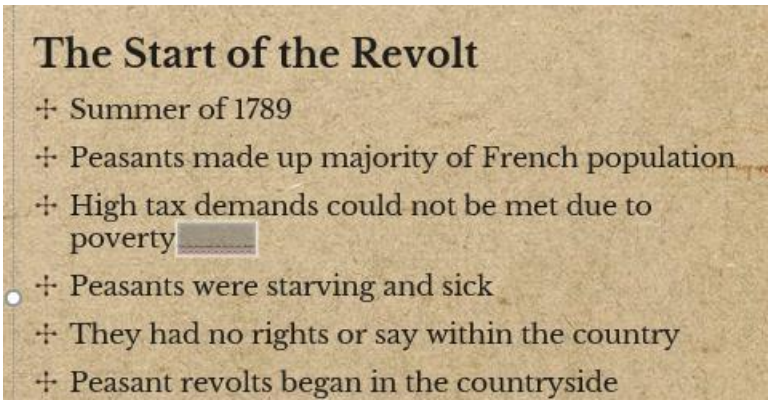
50

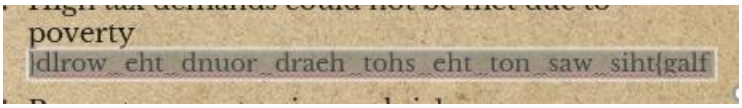
Find the very small flag.

Fun fact: this one didn't start with a shot heard around the world!

NOTE: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.

This one can be found on slide 56, there is text that is in 1pt font, we blow it up and it is a flag in reverse.





Flag: flag{this_was_not_the_shot_heard_round_the_world}

ATOTC 8

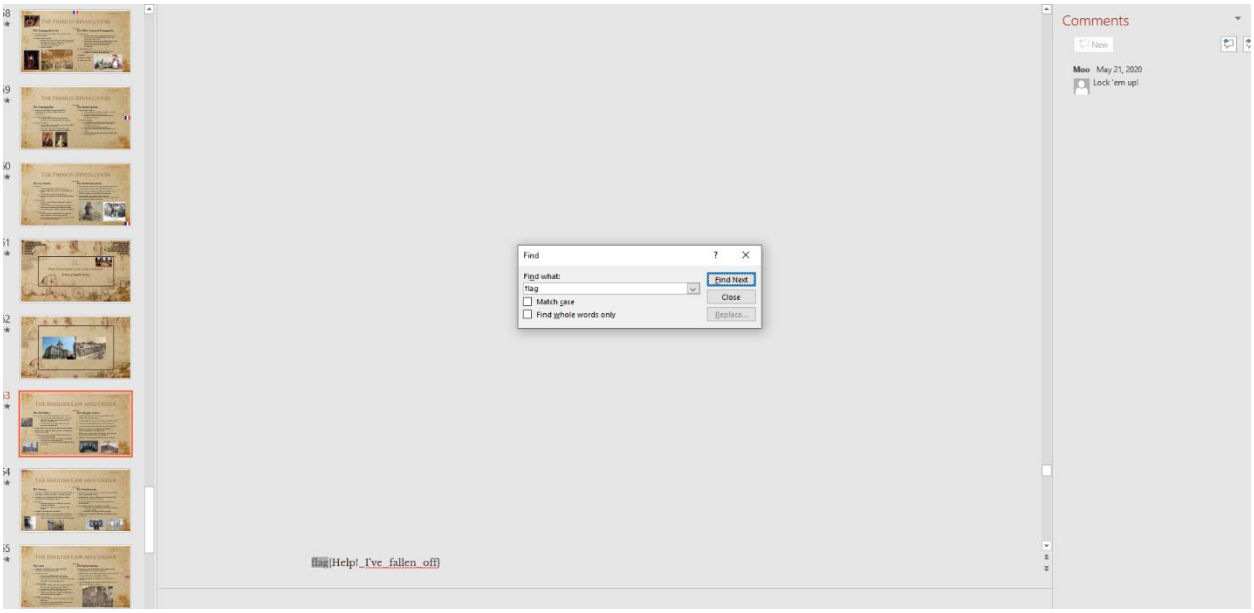
50

Find the flag that has fallen off.

Fun fact: law and order are a must in a civil society.

NOTE: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.

Do a search for the words flag we get 3 hits and the third one is located off the slide on slide 63



Flag: flag{Help!_I've_fallen_off}

ATOTC 9

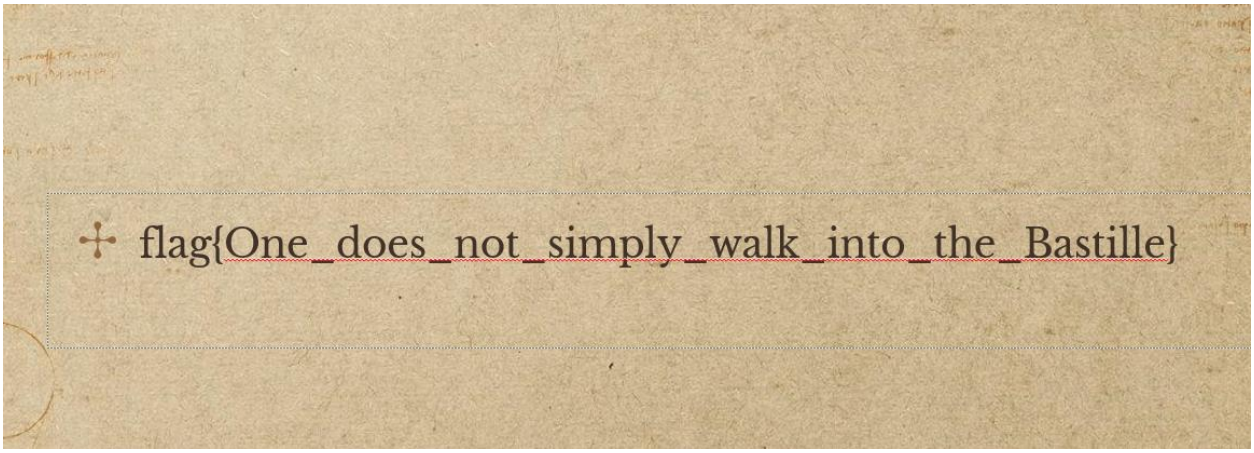
60

Find the one flag to rule them all.

Fun fact: this PowerPoint template was found online and provides a good foundation for this presentation.

NOTE: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.

Looking at the master slides we see this flag:



Flag: flag{One_does_not_simply_walk_into_the_Bastille}

ATOTC 10

50

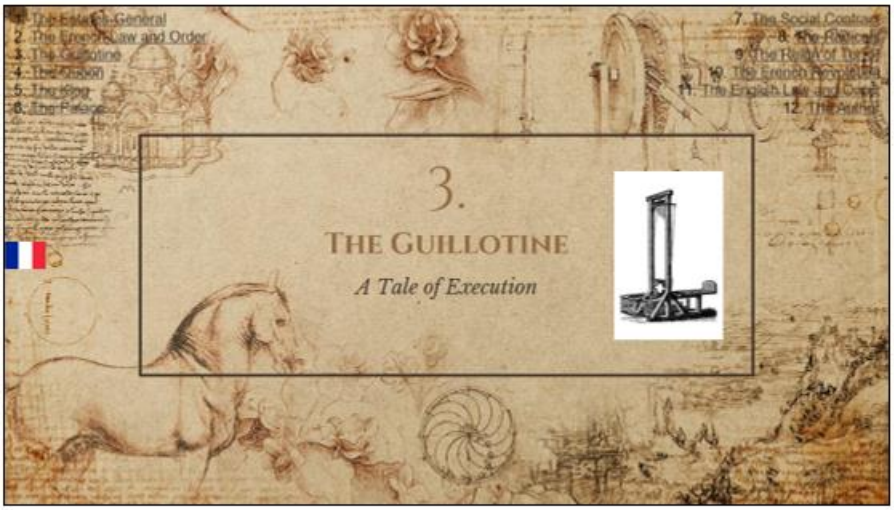
Find the flag that is easily noted.

Fun fact: guillotines were considered humane and painless.

NOTE1: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.

NOTE2: also, if using LibreOffice Impress, you might want to use OpenOffice Impress (or PowerPoint) instead for this question.

Looking at the notes slides we see the following on slide 15



Click to add notes

flag{Note:_may_the_guillotine_be_ever_sharp!}

flag{Note:_may_the_guillotine_be_ever_sharp!}

ATOTC 11

50

Find the flag that is not selected.

Fun fact: well-written subtitles can create a better watching experience, otherwise how do you know what you're missing?

NOTE: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.

Looking at the outline view we find on slide 10 this flag:

Flag: flag{A_Tale_of_French_Justice}

ATOTC 12

100

Find the flag that is bright.

Fun fact: some facts are just ramblings, while others may be purposeful… just read between the lines, so to speak.

NOTE: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.

For this one we think of the brightest space in the slide, which happens to be on slide 2 with the White space with the excerpt from the book.

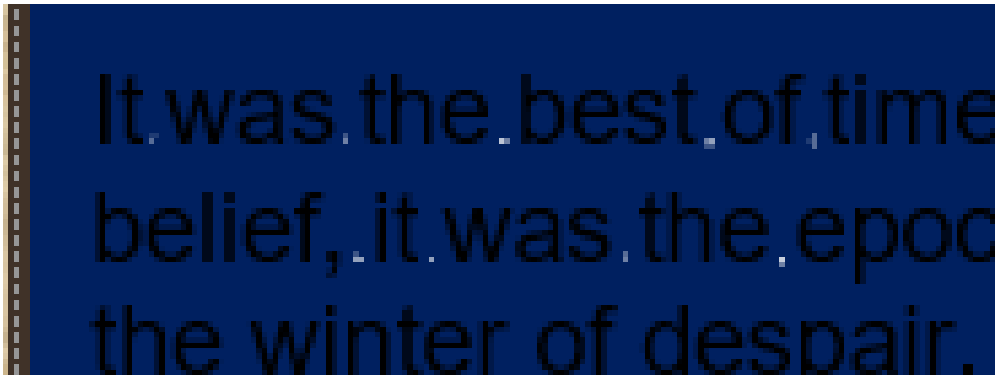
It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way—in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.

If we copy that paragraph to cyber

```
It    f    was    l    the    a    best    g    of    {    times,    t    it    h    was    1    the    s    worst    _
of    w    times,    h    it    1    was    t    the    e    age    s    of    p    wisdom,    a    it    c    was    e
the    _    age    s    of    u    foolishness,    r    it    e    was    _    the    1    epoch    s    of    _    belief,
b    it    r    was    1    the    g    epoch    h    of    t    incredulity,    ,    it    _    was    d    the    o
season    n    of    '    Light,    t    it    _    was    y    the    o    season    u    of    _    Darkness,    a    it
g    was    r    the    e    spring    e    of    ?    hope,    }    it was the winter of despair, we had everything before
us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way—in short, the
```

chef we can see something interesting.

The other option to find this one would be to change the background on the white textbox to a dark color like black or dark blue. You can see white specs as they are 1pt font.



Select the text and enlarge it and the flag sticks out.

It f was l the a best g of { times, t it h
was 1 the s worst _ of w times, h it 1
was t the e age s of p wisdom, a it c
was e the _ age s of u foolishness, r it e
was _ the 1 epoch s of _ belief, b it r
was 1 the g epoch h of t incredulity, , it _
was d the o season n of ' Light, t it _
was y the o season u of _ Darkness, a it
g was r the e spring e of ? hope, } it was
the winter of despair, we had everything before us, we had

Flag: flag{th1s_wh1tespace_sure_1s_br1ght,_don't_you_agree?}

ATOTC 13

150

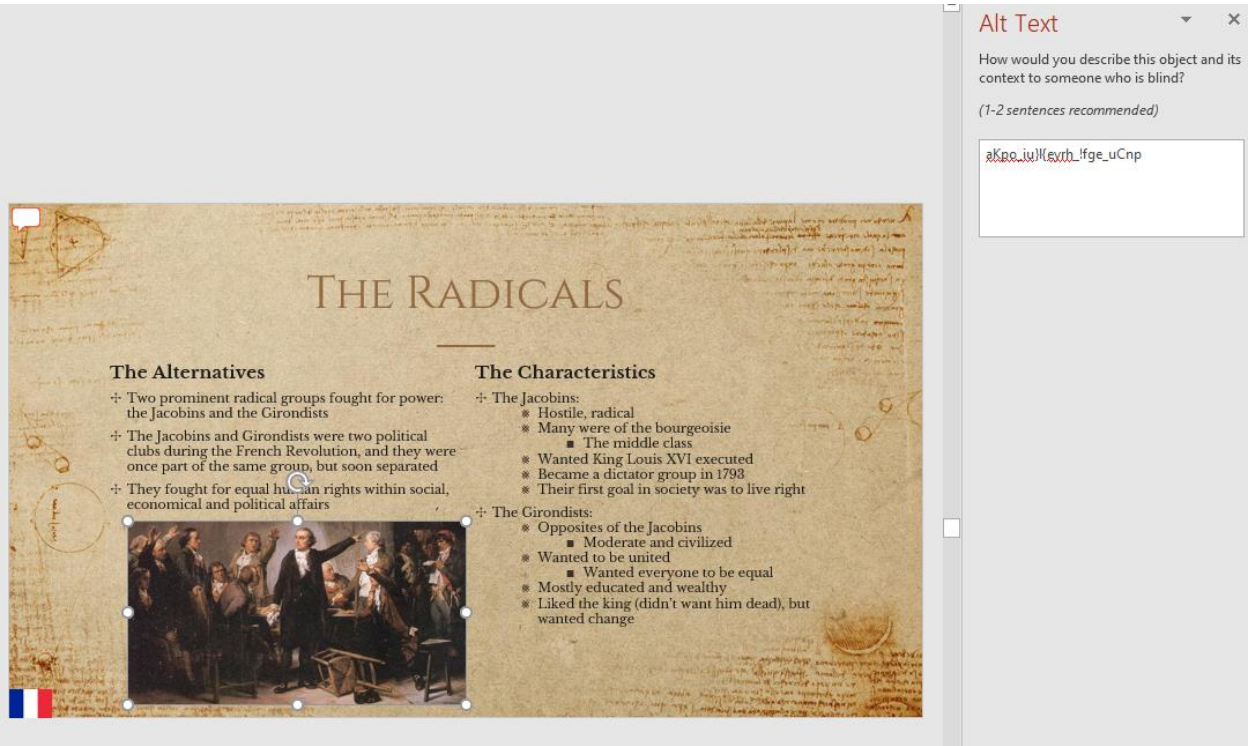
Find the text leading to the object with the alternative flag.

Fun fact: it may lean to the left if rotated, but it starts with a width that is always the bullet count.

NOTE1: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.

NOTE2: finding the flag can NOT be completed in LibreOffice/OpenOffice Impress alone, you will need to use other tools as well in order to actually see it; however, the flag can be found and viewed with just PowerPoint.

Looking at ALT text for pictures, we come across this which looks like a flag.



aKpo_iu}l{eyrh_!fge_uCnp

This flag has been rotated in an unusual way where it was split up in an 8x3 pattern then rotated and put together backwards.

flag{Keep_your_Chin_up!}

	3	2	1
1	f	l	a
2	g	{	K
3	e	e	p
4	_	y	o
5	u	r	_
6	C	h	i
7	n	_	u
8	p	!	}

```
$flag = "aKpo_iu}l{eyrh_!fge_uCnp"
$out = ""

for ($i = 0 ; $i -le ($flag.Length/3) ; $i += 1){
    $out += $flag[$i+16]
    $out += $flag[$i+8]
    $out += $flag[$i]
}

$out
```

Flag: flag{Keep_your_Chin_up!}

ATOTC 14

80

Find the flag that is out in the open but would take a radical to recognize.

Fun fact: this one is not quite acrostic, but in some way more so.

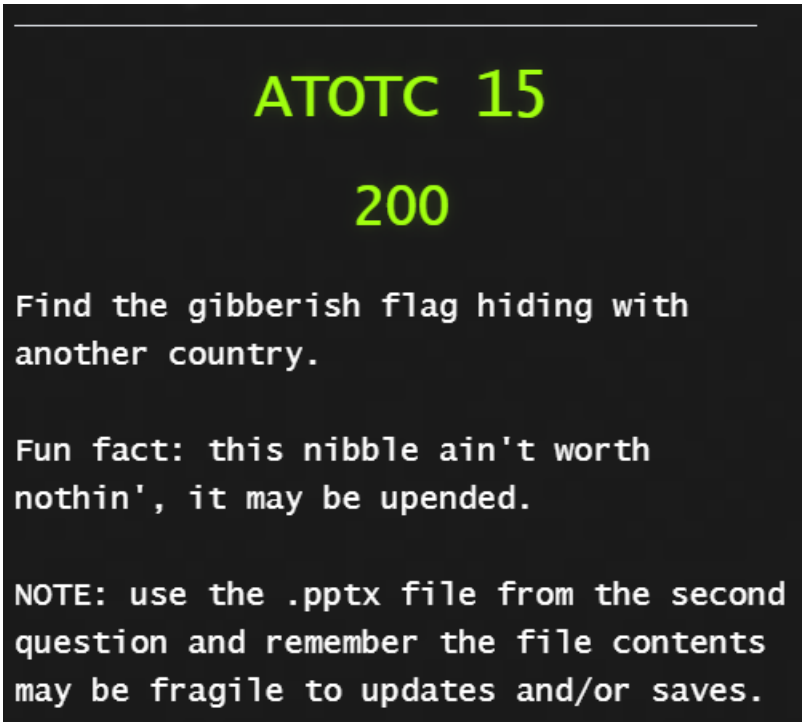
NOTE: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.

Key word here is Radicals, and if we go to that section slide 44 has text that seems out of place with extra {}.

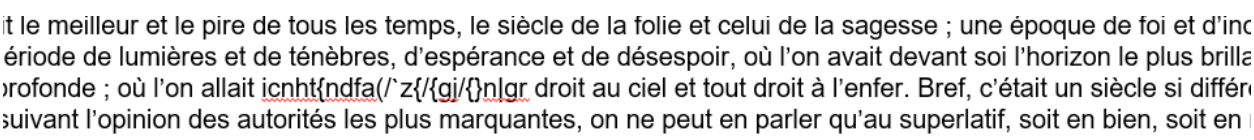
If we take the first letter of the words, we find a flag

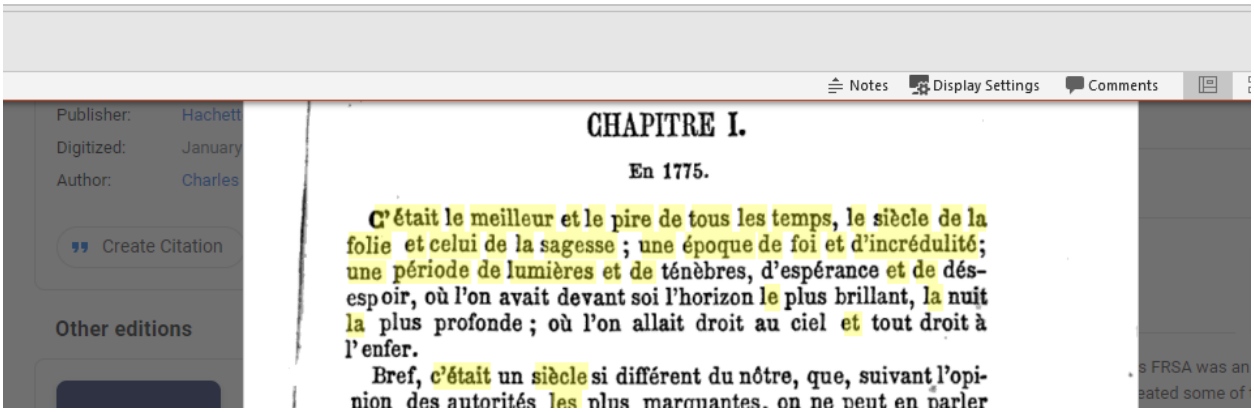
Tflag{hostileRadicals}kmutgwo

flag{hostileRadicals}



On the first page there is some interesting characters in the French section and when compared to the original, we see the differences.





This does not belong here: icnht{ndfa(/`z{/{gj/{}n|gr

If we look at the clues in the question it talks about nibbles being upended. So, here is a script that converts the 2nd nibble 's 1's to 0's and vice versa

```
$in = "icnht{ndfa(/`z{/{gj/{}n|gr"
$bin = [system.Text.Encoding]::Default.GetBytes($in) |
%{[System.Convert]::ToString($_,2).PadLeft(8,'0') }
$out = ""
$text = ""
for ($a = 0 ; $a -lt $bin.Length; $a++){
    $nibble = ""
    for ($b=4; $b -le 7; $b++){
        if ($bin[$a][$b] -eq "1"){
            $nibble += "0"
        } else {
            $nibble += "1"
        }
        #$nibble += $bin[$a][$b]
    }
    $static = ($bin[$a][0..3]) -join ""
    "Bin " + $bin[$a]
    "Mod " + $static + $nibble
    $comb = $static + $nibble
    "Letter " +  [char]([convert]::ToInt32("$comb",2))
    $out += $static + $nibble + " "
    $text += [char]([convert]::ToInt32("$comb",2))
}
$out
$text
```

Flag : flag{takin' out the trash}

01100110 01101100 01100001 01100111 01111011 01110100 01100001 01101011 01101001
01101110 00100111 00100000 01101111 01110101 01110100 00100000 01110100 01101000
01100101 00100000 01110100 01110010 01100001 01110011 01101000 01111101 00001010

icnht{ndfa(/`z{/{gj/{}n|gr

01101001 01100011 01101110 01101000 01110100 01111011 01101110 01100100 01100110
01100001 00101000 00101111 01100000 01111010 01111011 00101111 01111011 01100111
01101010 00101111 01111011 01111101 01101110 01111100 01100111 01110010 00000101

Flag : flag{takin' out the trash}

ATOTC 16

80

Find the descriptive image flag.

Fun fact: rusty guillotines are not good for one's health.

NOTE: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.

For this one we know that there is a picture of a rusty Guillotine.

We need to extract this image from the ppt and look at its properties closer.

We need to get the raw files out of the ppt slide, and in order to do that we need to convert this from a ppt to a zip file. Change the extension or add on .zip

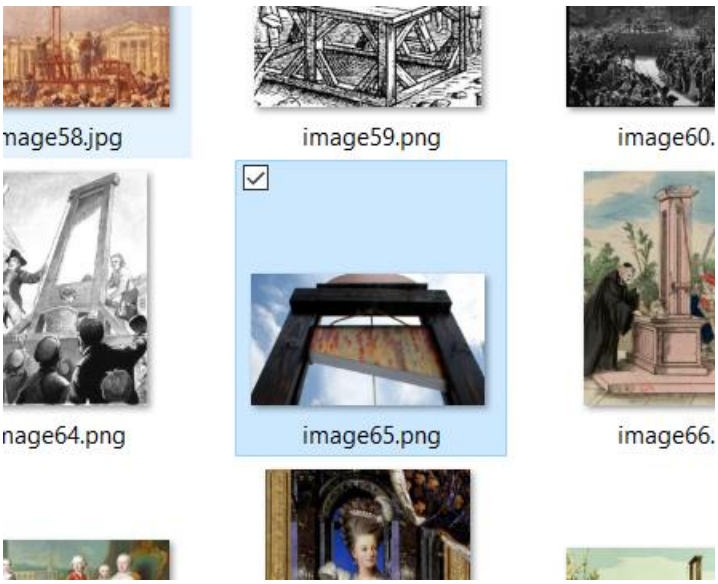
Extract the files to make it easier to look at the files.

A_Tale_of_Two_Cities-unencrypted	5/29/2020 11:03 AM	File folder	
A_Tale_of_Two_Cities-unencrypted.zip	5/28/2020 10:40 AM	Compressed (zipp...	105,797 KB

We are going to look at the media folder that has all the images and audio that have been added to the presentation.

A_Tale_of_Two_Cities-unencrypted › A_Tale_of_Two_Cities-unencrypted › ppt › media

We are looking for Img65



We can strings this file to see what the description of the file is.

```
1Kf5
,tEXtDescription
'flag{When_you_gotta_tinkle...}'
IEND
PS C:\Users\John\Downloads\CTF> .\sysinternals\strings.exe .\may\ATOTC\A_Tale_of_Two_Cities-unencrypted\A_Tale_of_Two_Ci
ties-unencrypted\ppt\media\image65.png
```

Flag : flag{When_you_gotta_tinkle...}

ATOTC 17

80

Find the flag that talks about a file that itself is commentary on things.

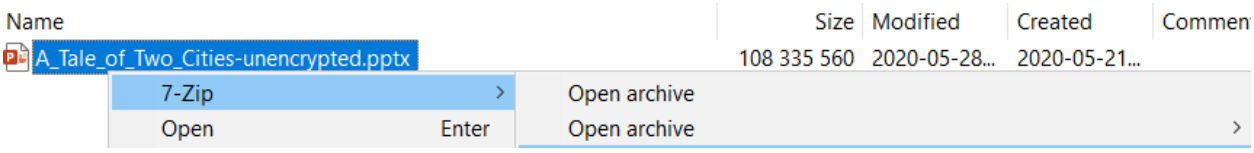
Fun fact: modern Office documents follow the Open Office XML standard, which if you really think about it is just one big container with its own properties.

NOTE: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.

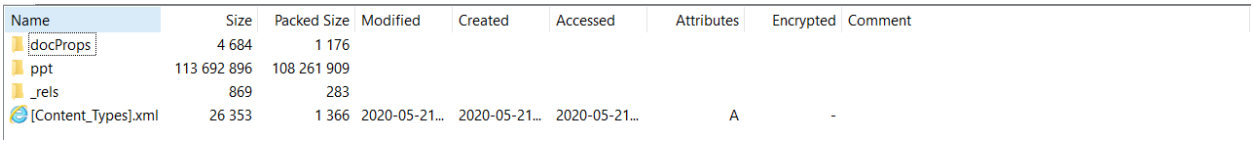
For this flag we are looking into the pptx file itself (or the archive of the pptx).

There are two ways to find this flag.

First up is 7zip, open the folder containing the pptx file and right-click -> 7-zip -> open archive



In here we see a column called Comment we can browse through the folders to see if anything pops up there.



Name	Size	Packed Size	Modified	Created	Accessed	Attributes	Encrypted	Comment
docProps	4 684	1 176						
ppt	113 692 896	108 261 909						
_rels	869	283						
[Content_Types].xml	26 353	1 366	2020-05-21...	2020-05-21...	2020-05-21...	A	-	

Name	Size	Packed Size	Modified	Created	Accessed	Attributes	Encrypted	Comment
comments	17 877	9 890						
fonts	198 628	198 628						
media	112 371 235	107 702 411						
notesMasters	8 995	1 568						
notesSlides	190 054	80 009						
slideLayouts	38 152	8 806						
slideMasters	28 796	2 204						
slides	800 204	251 577						
theme	13 957	2 979						
_rels	11 521	790						
commentAuthors....	579	332	1980-01-01...				-	
presentation.xml	8 116	1 275	1980-01-01...				-	
presProps.xml	1 283	541	1980-01-01...				-	
tableStyles.xml	2 714	523	1980-01-01...				-	
viewProps.xml	785	376	1980-01-01...				-	

Name	Size	Packed Size	Modified	Created	Accessed	Attributes	Encrypted	Comment
comment1.xml	597	356	1980-01-01...				-	ZmxhZ3tBbGxfdGhydXN0X2FuZF9ub192ZWN0b3IhfQo
comment2.xml	996	453	1980-01-01...				-	
comment3.xml	609	364	1980-01-01...				-	
comment4.xml	740	446	1980-01-01...				-	
comment5.xml	597	357	1980-01-01...				-	

There is our flag in base64 encoding

The second option is to do this on Linux with the zipnote tool

Just run zipnote on the pptx and you may need to | to more (or less) to see the flag at the top.

```
@ docProps/app.xml
@ (comment above this line)
@ docProps/core.xml
@ (comment above this line)
@ ppt/commentAuthors.xml
@ (comment above this line)
@ ppt/comments/comment1.xml
ZmxhZ3tBbGxfdGhydXN0X2FuZF9ub192ZWN0b3IhfQo
```

```
kali@kali:~/5ctf/ATOTC$ zipnote A_Tale_of_Two_Cities-unencrypted.pptx  | less
kali@kali:~/5ctf/ATOTC$ echo ZmxhZ3tBbGxfdGhydXN0X2FuZF9ub192ZWN0b3IhfQo | base64 -d
flag{All_thrust_and_no_vector!}
base64: invalid input
```

Flag : flag{All_thrust_and_no_vector!}

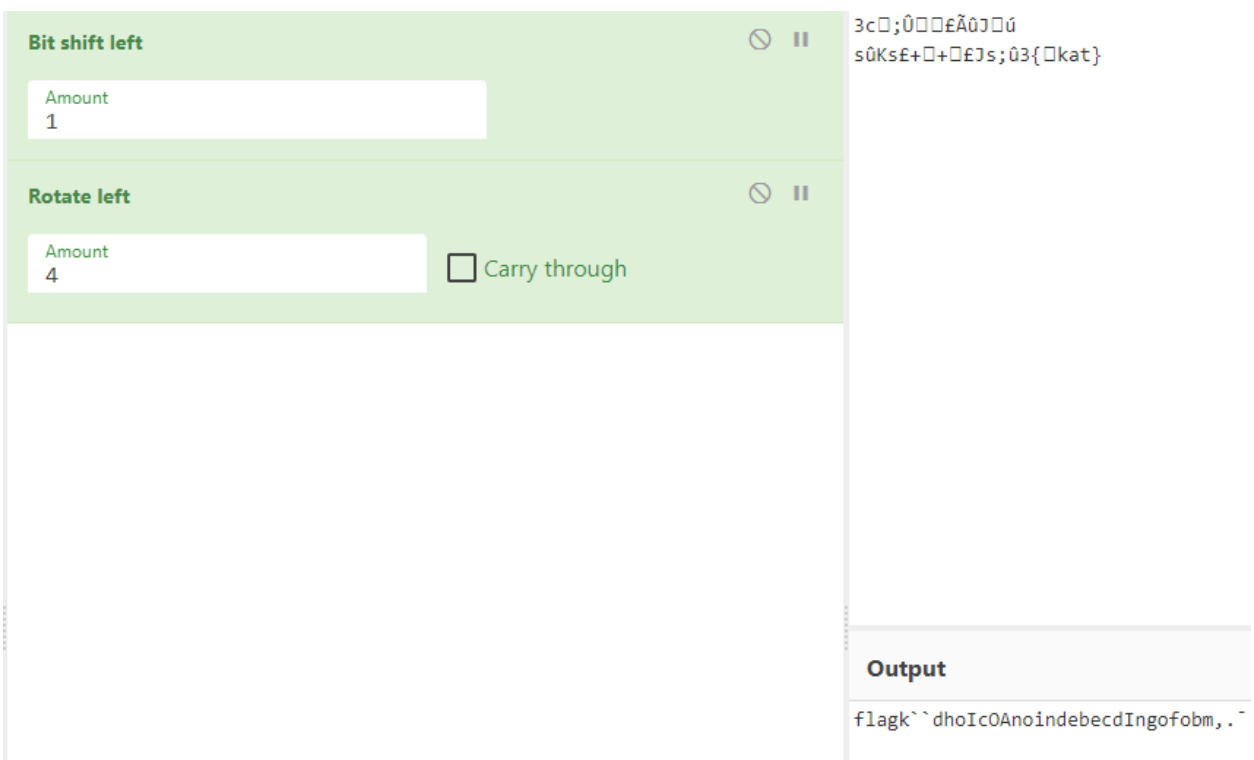
ATOTC 18

150

Find the text that is being shifty and hanging out with other files.

Fun fact: some would say that 7-zip is the most popular freeware file archive tool for Windows; it gets my vote but I do rotate through my three favorite archivers (7-zip, WinZip, and WinRar).

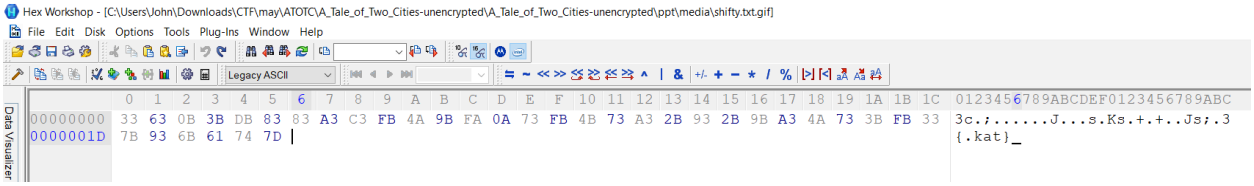
NOTE: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.



Since we can see flag in cyberchef we know the bit shift and rotate of the string potentially.

CyberChef only has 8bit shifts so we need another tool that can allow us to do more shifting.

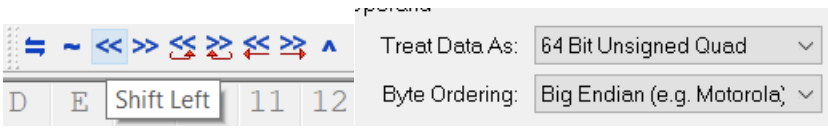
I came across Hex Workshop when looking around and this seemed like the best tool around for this.

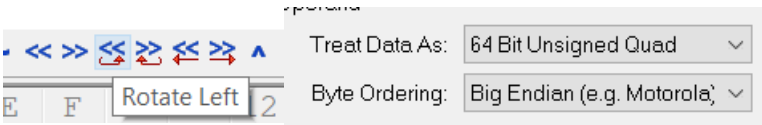


Make sure to add the bit operators to the toolbar to save lots of time.

This tool allows us to do 8,16,32 and 64 bit shifts in both Big Endian and Little Endian. Good thing we know the counts of this or this could take a long time.

To start off i went with 64 big e across the board.





(x4)

```
lag{pptf.iS.AN.hnterestiNg.fo
rmiat}_
```

That is looking like we are close to the answer, just need to keep messing around now until we get the right answer.

The right answer to get the flag was 32bit Little Endian and 2x 32bit little endian rotate right.

```
flaG{ppTx_iS_an_intEresTing_f
orMat}
```

Flag : flag{pptx_is_an_interesting_format}

ATOTC 19

80

Find the secret message in an image hidden among other files.

Fun fact: this one requires something very short.

NOTE: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.

We need to get the raw files out of the ppt slide, and in order to do that we need to convert this from a ppt to a zip file. Change the extension or add on .zip

Extract the files to make it easier to look at the files.

A_Tale_of_Two_Cities-unencrypted	5/29/2020 11:03 AM	File folder	
A_Tale_of_Two_Cities-unencrypted.zip	5/28/2020 10:40 AM	Compressed (zipp...	105,797 KB

We are going to look at the media folder that has all the images and audio that have been added to the presentation.

A_Tale_of_Two_Cities-unencrypted › A_Tale_of_Two_Cities-unencrypted › ppt › media

At the bottom of this folder we see secret.jpg with a picture of Napoleon.

secret.jpg


```
flag{Le'capitaine,_do_you_need_some_books_to_stand_on?}
```

Flag : flag{Le'capitaine,_do_you_need_some_books_to_stand_on?}

ATOTC 20

60

There is another "talking" flag that comments on the presentation and may take more digging.

Fun fact: XML 1.0 became a thing in February of 1998.

NOTE: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.

HINT: There may be other ways to look at the presentation other than using PowerPoint or LibreOffice Impress.

Let ‘s Zero in on the XML portion of this challenge. Moving the unzipped pptx document to kali we can run grep looking for a comment in any file. The comment for XML starts with < !

```
kali@kali:~/5ctf/ATOTC/A_Tale_of_Two_Cities-unencrypted$ grep -nr "<!" | grep -v Binary
[Content_Types].xml:2:<!--ZmxhZ3tOb3dfeW91X2tub3dfbW9yZV93YXlzX3RvX2hpZGVfaW5mb190aGF0X3lvdV9ldmVyX2NhcmVkX3RvfQo-->
kali@kali:~/5ctf/ATOTC/A_Tale_of_Two_Cities-unencrypted$
```

Base64 decode the string to find our flag

```
kali@kali:~/5ctf/ATOTC/A_Tale_of_Two_Cities-unencrypted$ echo ZmxhZ3tOb3dfeW91X2tub3dfbW9yZV93YXlzX3RvX2hpZGVfaW5mb190aGF0X3lvdV9ldmVyX2NhcmVkX3RvfQo | base64 -d
flag{Now_you_know_more_ways_to_hide_info_that_you_ever_cared_to}
```

Flag : flag{Now_you_know_more_ways_to_hide_info_that_you_ever_cared_to}

ATOTC 21

100

Find the trimmed flag.

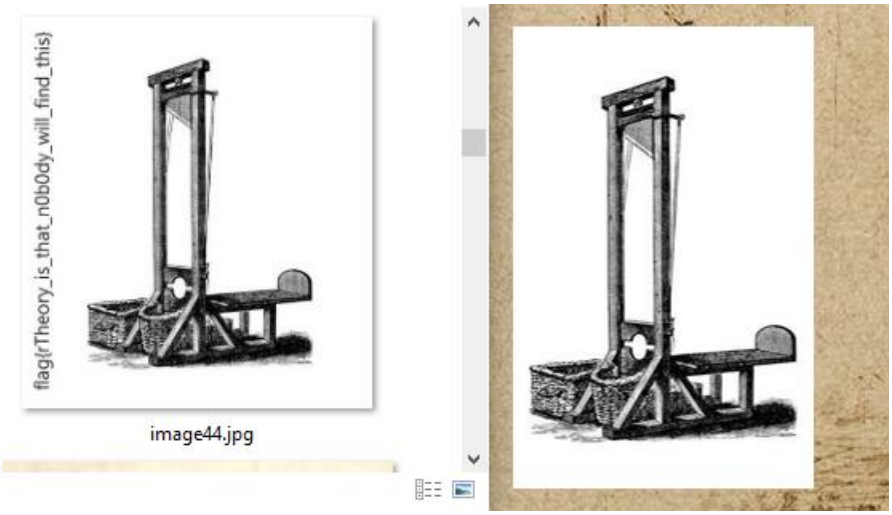
Fun fact: this one might be chopped up.

NOTE: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.

This flag talks about chopping up and trimming.

When you crop an image, the full file is still saved in the document.

To find this one just have the pptx up and the images up in explorer from the unzipped view and compare images together.



Flag: flag{rTheory_is_that_n0b0dy_will_find_this}

ATOTC 22

125

Find the flag that hides behind speech.

Fun fact: Styx "The Best of Times" rose to #3 on the Pop Singles chart in 1981.

NOTE: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.

We need to get the raw files out of the ppt slide, and in order to do that we need to convert this from a ppt to a zip file. Change the extension or add on .zip

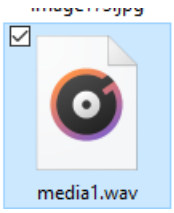
Extract the files to make it easier to look at the files.

A_Tale_of_Two_Cities-unencrypted	5/29/2020 11:03 AM	File folder	
A_Tale_of_Two_Cities-unencrypted.zip	5/28/2020 10:40 AM	Compressed (zipp...	105,797 KB

We are going to look at the media folder that has all the images and audio that have been added to the presentation.

A_Tale_of_Two_Cities-unencrypted › A_Tale_of_Two_Cities-unencrypted › ppt › media

At the bottom of this folder we see media1.wav as the only item that would be « Talking «



Run ExifTool on the file :

```
kali@kali:~/5ctf/ATOTC$ exiftool media1.wav
ExifTool Version Number         : 11.99
File Name                       : media1.wav
Directory                       : .
File Size                       : 8.1 MB
File Modification Date/Time     : 2020:05:29 12:02:45-04:00
File Access Date/Time           : 2020:06:01 13:48:28-04:00
File Inode Change Date/Time     : 2020:05:29 14:55:09-04:00
File Permissions                : rwxrw-rw-
File Type                       : WAV
File Type Extension             : wav
MIME Type                       : audio/x-wav
Encoding                        : Microsoft PCM
Num Channels                    : 2
Sample Rate                     : 44100
Avg Bytes Per Sec               : 176400
Bits Per Sample                 : 16
Duration                        : 0:00:48
```

Run file on the WAV file :

```
kali@kali:~/5ctf/ATOTC$ file media1.wav
media1.wav: RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, stereo 44100 Hz
```

Doing a quick search on RIFF LSB Stego the third result looked interesting.

RIFF LSB stego

All | Images | Shopping | Videos | News | More | Settings

About 48,200 results (0.42 seconds)

pdfs.semanticscholar.org › ...

LSB Audio Steganography Approach - Semantic Scholar

LSB Audio **Steganography** Approach ... The next fields define **RIFF** file which contains Wave ... information from an **LSB** encoded audio file (**stego** object),.

by KU Singh - Cited by 1 - Related articles

www.iasj.net › iasj ▾ PDF

Secret Message Hiding in WAVE PCM Sound File

In this paper a scheme of **steganography** system for hiding secret text message in ... using Low Bit Encoding (**LSB**) Least Significant Bit substitution techniques. ... just a **RIFF** file with a single "WAVE" chunk which consists of two sub-chunks—a.

by R Salih - 2015 - Related articles

github.com › ragibson › Steganography ▾

ragibson/Steganography: Least Significant Bit ... - GitHub

Least Significant Bit **Steganography** for bitmap images (.bmp and .png), WAV sound files, and byte sequences. Simple **LSB** Steganalysis (**LSB** extraction) for ...

I stumbled across wavsteg as an option to unhide stego in the WAV/RIFF file

```
git clone https://github.com/ragibson/Steganography
cd Steganography
python3 setup.py install
```

How to use

WavSteg requires Python 3

Run WavSteg with the following command line arguments:

```
Command Line Arguments:
 -h, --hide                To hide data in a sound file
 -r, --recover             To recover data from a sound file
 -i, --input TEXT          Path to a .wav file
 -s, --secret TEXT         Path to a file to hide in the sound file
 -o, --output TEXT         Path to an output file
 -n, --lsb-count INTEGER   How many LSBs to use  [default: 2]
 -b, --bytes INTEGER       How many bytes to recover from the sound file
 --help                    Show this message and exit.
```

Example:

```
$ stegolsb wavsteg -h -i sound.wav -s file.txt -o sound_steg.wav -n 1
# OR
$ stegolsb wavsteg -r -i sound_steg.wav -o output.txt -n 1 -b 1000
```

1 LSB does not return anything of interest so let's go to the default of 2.

```
kali@kali:~/5ctf/ATOTC$ stegolsb wavsteg -r -i media1.wav -o out.txt -n 1 -b 1000
Files read                    in 0.00s
Recovered 1000 bytes          in 0.00s
Written output file           in 0.00s
kali@kali:~/5ctf/ATOTC$ cat out.txt
```

Switch to LSBits count of 2 we get some interesting results:

```
 stegolsb wavsteg -r -i media1.wav -o out.txt -n 2 -b 1000
Files read                    in 0.00s
Recovered 1000 bytes          in 0.00s
Written output file           in 0.00s
```

```
kali@kali:~/5ctf/ATOTC$ cat out.txt
flag{It_was_the_best_of_times}
```

Flag : flag{It_was_the_best_of_times}

ATOTC 23

80

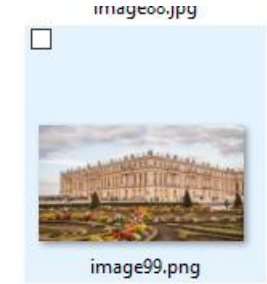
This flag is a hidden gem.

Fun fact: the Palace of Versailles gardens cover over 800 hectares of land.

NOTE: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.

To start off we need to look for gardens of Versailles and gems, we can find this on slide 35

We need to pull the picture in the lower right corner and examine that closer.



We need to verify that the png file is a png file

```
kali@kali:~/5ctf/ATOTC$ pngcheck image99.png
OK: image99.png (648×365, 32-bit RGB+alpha, non-interlaced, 47.4%).
```

Running a stego detection tool we find our flag.

```
kali@kali:~/5ctf/ATOTC$ zsteg image99.png
imagedata           .. file: apollo a88k COFF executable not stripped - version 1277
b1,bgr,msb,xy       .. file: 370 XA sysV pure executable not stripped - 5.2 format
b1,abgr,msb,xy      .. text: "=ywww33="
b2,r,lsb,xy         .. text: "UPTUUUUUQ"
b2,rgba,lsb,xy      .. text: "flag{So, you wanna_be_a_logistician?}\nOO"
b2,abgr,msb,xy      .. text: "ssswwwww"
b3,g,lsb,xy         .. text: "SI'.nImi"
b4,r,lsb,xy         .. text: "wwwuTC\"\""
b4,g,lsb,xy         .. text: "vTExgeTD33333233"
b4,b,lsb,xy         .. text: "3EgUDD#4D"
b4,rgba,msb,xy      .. text: "r{2323z{z<I"
```

Flag : flag{So, you wanna_be_a_logistician?}

ATOTC 24

80

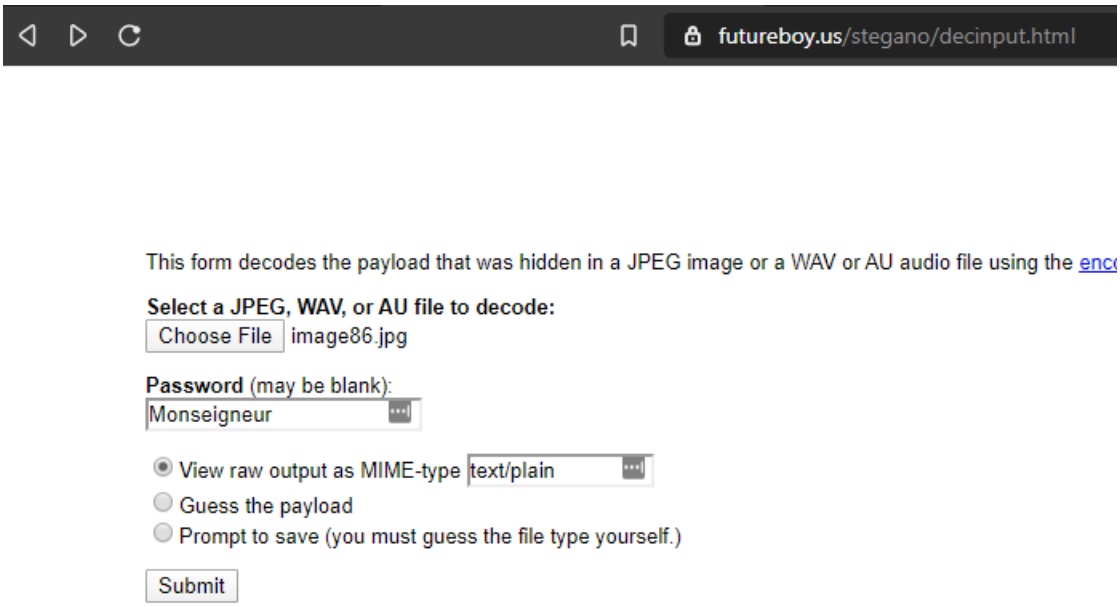
Find the biggest flag of them all.

Fun fact: background checks are used to ensure clearances are not flagged, but you may need your creds first.

NOTE: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.

Clues here are Background, and creds. So, we will attempt to replicate a couple of the previous flags on different images with the password we unlocked the pptx with.

The first three background images and the questions page did not produce any results. But the one with the image of Versailles did give us some good output.



```
flag{Thou_shalt_not_pass!}
```

Flag : flag{Thou_shalt_not_pass!}

ATOTC 25

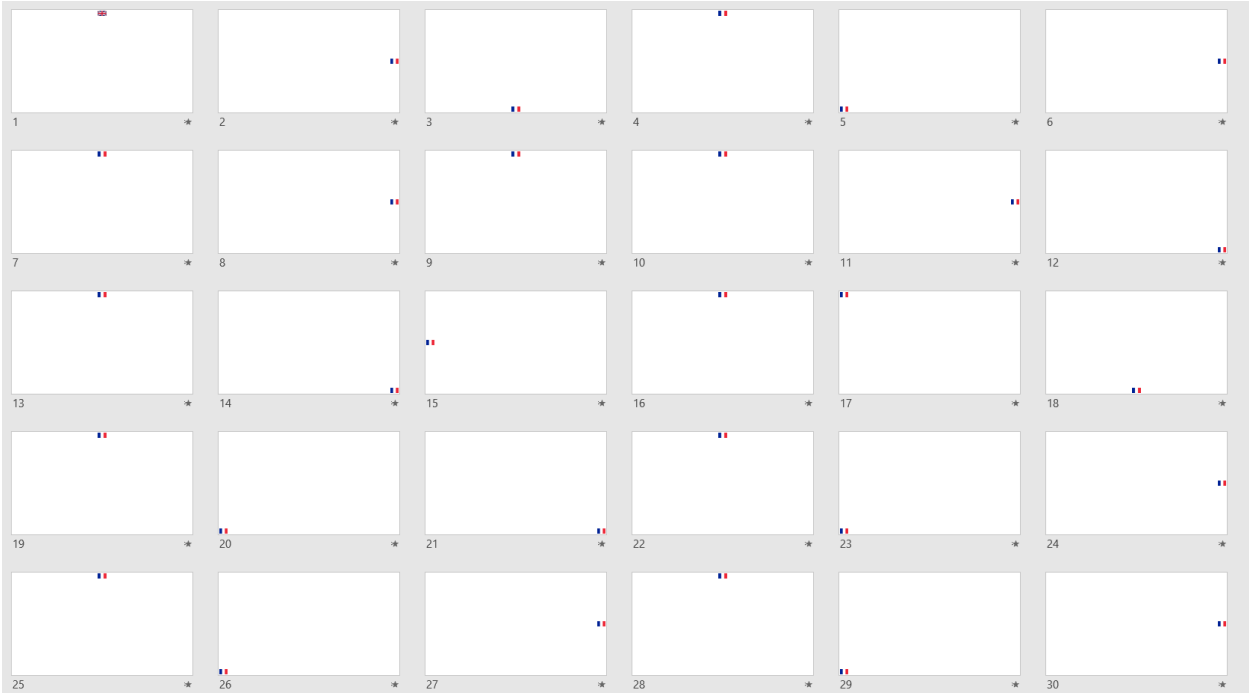
175

Find the moving flags.

Fun fact: there are eight main settings in ATOTC: [0] the Bastille in Paris, [1] the Tribunals of the Republic, [2] the Defarge's wine shop, [3] Dover, [4] the French chateau, [5] a courtroom in London, [6] the Manette's house in soho, and [7] Tellson's Bank.

NOTE1: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.

NOTE2: there is an error in the obfuscated flag; the answer will take the decoded wrong flag or the actual intended flag; bonus points if you enter the intended flag (not really!).



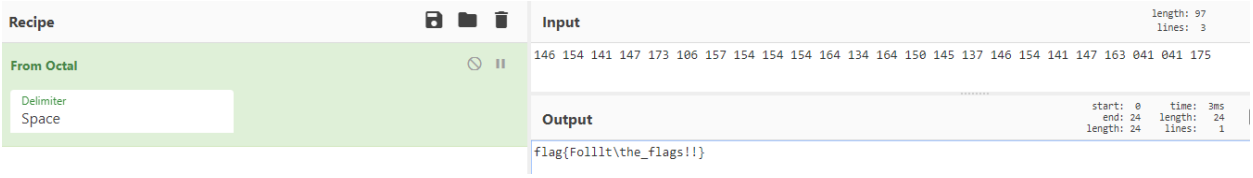
0	1	2
3		4
5	6	7

146154141147173106157154154154164134164150145137146154141147163041041175

For this we need to make it octal

UTF-8		
Binary	Octal	Hexadecimal
00100100	044	24
11000010 10100010	302 242	C2 A2
11100000 10100100 10111001	340 244 271	E0 A4 B9
11100010 10000010 10101100	342 202 254	E2 82 AC
11101101 10010101 10011100	355 225 234	ED 95 9C
11110000 10010000 10001101 10001000	360 220 215 210	F0 90 8D 88

146 154 141 147 173 106 157 154 154 154 164 134 164 150 145 137 146 154 141 147 163 041 041 175



Flag : flag{Folllt\the_flags!!}

ATOTC 26

300

Find the compressed flag.

Fun fact: the analyst was deflated when he saw the number of slides in the presentation to analyze; to analyze this file methodically would probably take five analysts.

NOTE: use the .pptx file from the second question and remember the file contents may be fragile to updates and/or saves.

Name	Size	Attributes	Method
_rels	93 150		
slide1.xml	3 222	A	Deflate:Fastest
slide2.xml	57 202	A	Deflate:Maximum
slide3.xml	9 065	A	Deflate:Fastest
slide4.xml	3 954	A	Deflate
slide5.xml	7 733	A	Store
slide6.xml	10 115	A	Deflate:Fast
slide7.xml	12 501	A	Deflate
slide8.xml	15 299	A	Deflate:Maximum
slide9.xml	12 244	A	Store
slide10.xml	9 193	A	Deflate:Fast
slide11.xml	2 673	A	Store
slide12.xml	14 559	A	Deflate:Maximum
slide13.xml	10 486	A	Deflate
slide14.xml	11 326	A	Deflate:Fastest
slide15.xml	9 027	A	Deflate
slide16.xml	2 971	A	Deflate:Maximum
slide17.xml	12 041	A	Store
slide18.xml	14 363	A	Store
slide19.xml	11 782	A	Deflate:Fast
slide20.xml	9 992	A	Deflate:Fastest
slide21.xml	8 997	A	Deflate:Fastest
slide22.xml	3 848	A	Deflate:Fastest
slide23.xml	10 546	A	Deflate:Maximum
slide24.xml	9 865	A	Deflate
slide25.xml	8 718	A	Deflate

One method for this would be to manually create a spreadsheet and fill in the slide number and method.

Using formulas to generate the rest of the output we need:

Conversion Column cells have this formula =IF(B2=L$3,0,IF(B2=L$2,1,IF(B2=L$5,2,IF(B2=L$4,3,4))))

Basically, a nested if statement for each option (L$2 is important to anchor the reference points)

Then you can copy the column of conversion and paste into text editor (as plain text) and remove all the newlines to get a single base 5 result

3231401240421312440333211142400022113041323403004234214403231323322201 04

Slide	Method	Conversion		Methods
1	Deflate:Fastest	3	1	Deflate
2	Deflate:Maximum	2	0	Deflate:Fast
3	Deflate:Fastest	3	3	Deflate:Fastest
4	Deflate	1	2	Deflate:Maximum
5	Store	4	4	Store
6	Deflate:Fast	0		
7	Deflate	1		
8	Deflate:Maximum	2		
9	Store	4		
10	Deflate:Fast	0		
11	Store	4		
12	Deflate:Maximum	2		
13	Deflate	1		

There is also the automated approach which involves creating a script off the 7zip display.

7zip does not send out nice objects, but rather line based strings.

If summary the script will list all the fields of the entire PPTX document, but we only want the slides and the compression method.

The script will read line by line and if it does not start with Path or Method it skips it.

If the line matches Path then the file name is captured

Then skip any line unless it matches Method, convert the method to a number from 0-4 (like the manual method).

I am also skipping anything that is not slide##.xml as we do not need those.

I am then adding these to a custom PowerShell object to make it easier to handle.

I then take the converted numbers sorted by the slide number (Very important to get the slides in order by number as an int, other wise it will treat them as string and will go 1,11,12,13,14, etc.).

Finally, I am reading the method out to a single string in order to give use the base 5 conversion.

Must install 7zip, and be in the folder with the pptx to run the following script.

Script:

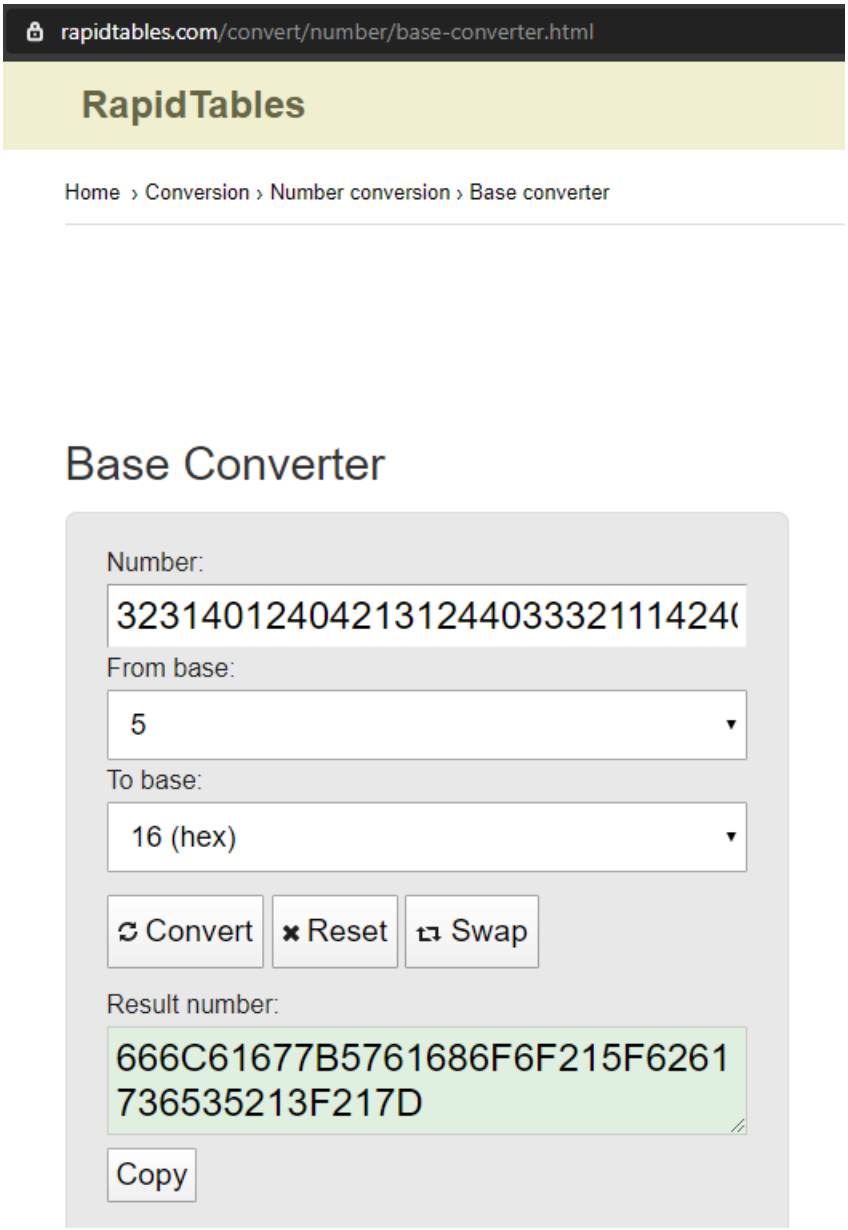
```
$zip1 =& 'C:\Program Files\7-Zip\7z.exe' l A_Tale_of_Two_Cities-unencrypted.pptx -slt
| ConvertFrom-String -Delimiter "=" -PropertyNames key,value
$out = @()
$record = @()
foreach ($line in $zip1){
    if ($line -match "Path"){
        $file = (($line -split "\\")[2] -split ".xml")[0]
    }elseif ($line -match "Method"){
        if ($file -notmatch "Layout" -and $file -notmatch "Master" -and $file -
notmatch "notes"){
            if ($line.value -match "Store") {
                    $method = "4"
            }elseif ($line.value -match "Deflate") {
                $def = $line.value -split ":"
                if ($def[1] -match "Fastest"){
                    $method = "3"
                }elseif ($def[1] -eq "Fast") {
                    $method = "0"
```

```
                }elseif ($def[1] -eq "Maximum") {
                    $method = "2"
                }else{
                    $method = "1"
                }
            }else{
                $method="no match"
            }
            $record = New-Object PSObject -Property @{
                "file_name" = [int]($file -split "ide")[1]
                "Method_orig" = $line.value
                "method" = [int]$method
            }
        }
    }else{
        Continue
    }
    if($record.file_name -ne "0" -and $line.key -ne "Path"){
        $out += $record
    }
}
$post = $out | sort file_name -Unique
$answer = ""
foreach ($a in $post.method){
    $answer += $a
}
$answer
```

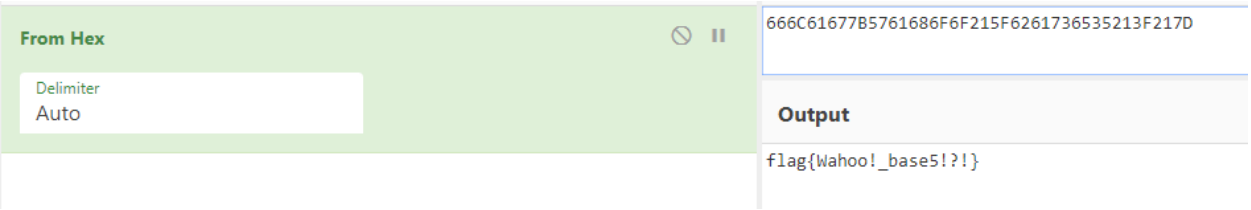
```
    $answer += $a
}
$answer

323140124042131244033321114240002211304132340300423421440323132332220104
```

We need to convert this from base5 to hex and from hex to ASCII

https://www.rapidtables.com/convert/number/base-converter.html



Take the Hex and copy it over to cyberchef to get the flag



Flag: flag{Wahoo!_base5!?!}