Ciphertext CTF 2020

Cryptography & Steganography

thousand_and_one_codings

Description:

A bunch of encodings and compression were used to encode the flag, can you decode it? Hint: if you stuck, remember how binary looks like and fix what should be fixed?

Files:

thousand and one encodings Size: 1.78 KB MD5: 4e18dbb55972d774908d05a5581e0be9

Solution:

Lets start by determining the type of the file, it is zip, decompressing it gives us a bzip2, decompressing it gives us gzip compressed file, decompressing it gives us ASCII text file:

```
© Command Prompt

C:\Users\venom\Desktop\CTCTF\crypto\thousand_and_one_codings>file thousand_and_one_encodings
thousand_and_one_encodings: Zip archive data, at least v2.0 to extract

C:\Users\venom\Desktop\CTCTF\crypto\thousand_and_one_codings>unzip thousand_and_one_encodings
Archive: thousand_and_one_encodings
inflating: DD

C:\Users\venom\Desktop\CTCTF\crypto\thousand_and_one_codings>mv out

C:\Users\venom\Desktop\CTCTF\crypto\thousand_and_one_codings>file out
out: bzip2 compressed data, block size = 900k

C:\Users\venom\Desktop\CTCTF\crypto\thousand_and_one_codings>pzip2 -d out
bzip2: Can't guess original name for out -- using out.out
out.out: gzip compressed data, was "\z40", last modified: Thu Apr 16 14:40:06 2020, max speed, from FAT filesystem (MS-D
OS, OS/2, NT), original size modulo 2^32 37736

C:\Users\venom\Desktop\CTCTF\crypto\thousand_and_one_codings>mv out.out out.gz
C:\Users\venom\Desktop\CTCTF\crypto\thousand_and_one_codings>pzip -d out.gz

C:\Users\venom\Desktop\CTCTF\crypto\thousand_and_one_codings>ls
out thousand_and_one_encodings

C:\Users\venom\Desktop\CTCTF\crypto\thousand_and_one_codings>ls
out thousand_and_one_encodings

C:\Users\venom\Desktop\CTCTF\crypto\thousand_and_one_codings>file out
out: ASCII text, with very long lines, with no line terminators

C:\Users\venom\Desktop\CTCTF\crypto\thousand_and_one_codings>

C:\Users\venom\Desktop\CTCTF\crypto\thousand_and_one_codings>
```

MUdoJEYxYyRwSTFHaCRGMWMlIUUxR2gkRjFjJHQiMUdoJEYxYyUhQ...

Decoding it gives us **ASCII85** encoded data:

1*Gh*\$*F*1*c*\$*pI*1*Gh*\$*F*1*c*%!*E*1*Gh*\$*F*1*c*\$*t*"1*Gh*\$*F*1*c*%!*C*1*Gh*\$*F*1*c*%\$*E*1*Gh*\$*F*...

Decoding it gives us hexadecimal encoded data:

34714338347143523471434e34714350347143613471433834771434...

Decoding it gives us **Base64** encoded data:

4qC84qCR4qCN4qCP4qCa4qC84qCa4qCD4qC84qCK4q...

Decoding it gives us **Braille** encoded data:

Decoding it gives us **Base32-crockford** encoded data:

5MPJOB9E5RG2W81E5MPJOB9D40PJWBH05RPJOBHD5MG2TB9...

Decoding it gives us **Morse** code:

^{*}Note that all the following decoding are made on <u>dcode.fr</u>, other tools may give slightly different outputs. The contents of the text file seem to be **Base64** encoded data:

Decoding it gives us the following **Base64** encoded data:

MDEWMDAWMTEGMDEWMTAXMDAGMDEWMDAWMTEGMDEWMTAXMDAGMDEWMDAXMTAGMDEXMTE WMTEGMDEWMDAWMDAGMDEXMDAGMDAXMTAWMDAGMDEXMTAXMDAGMDEWMTEXMTEGMDE XMDEXMTEGMDEXMDAXMTAGMDEWMTEXMTEGMDAXMDAXMDAGMDAXMTAWMTEGMDEXMDAWMTEG MDEXMTAWMTAGMDAXMTAWMTEGMDAXMTAXMTEGMDEWMTEXMTEGMDEXMDAXMDAGMDEXMDAWM DEGMDEXMTAGMDAXMTAXMTAGMDEXMTAXMDEGMDEXMDAWMDEGMDEXMDAXMTAGMDAXMTA WMTEGMDAXMDAXMDAGMDEXMTEXMDE=

Decoding it will give a strange binary data with missing chunks:

What could be the problem? The hint says to remember how binary looks like and to fix something. We know that decoding from Morse can result in a sting that is either all upper-case or all lower-case, maybe that's the problem here, lets encode some binary data and see how it looks when Base64 encoded:

Let's fix the capitalization now by replacing all capital X,W,G,O with lowercase:

```
C:\python
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>> x='MDEWNDAWMTEGMDEWMTAXMDAGMDEWMDAWMTEGMDEWMTAXMDAGMDEWMDAXMTAGMDEXMTEGMDEWMDAWMTAGMDEXMTAWMTEGMDEXMDAGMDEXMTAWMTAGMDEXMTAXMDAGMDEXMTAXMDAGMDEXMTAWMTEGMDEXMTAWMTAGMDEXMTAWMTEGMDAXMTAWMTEGMDAXMTAWMTEGMDAXMTAWMTEGMDAXMTAWMTEGMDAXMTAWMTEGMDAXMTAWMTEGMDAXMTAWMTEGMDAXMTAXMTAGMDEXMTAXMDEGMDEXMDAMMDAGMDEXMTAWMTEGMDAXMTAWMTEGMDAXMTAWMTEGMDAXMTAWMTEGMDAXMTAMMTEGMDAXMTAXMDAGMDEXMTEXMDE='
>>> for i in 'XWGO':
... x=x.replace(i,i.lower())
...

'MDEWNDAWMTEGMDEWMTAXMDAGMDEWMDAWMTEGMDEWMTAXMDAGMDEWMDAXMTAGMDEXMTEWMTEGMDEXMDAMMDAGMDEXMTAWMDAGMDEXMTAXMDAGMDEXMTAXMDAGMDEXMTAWMTEGMDEXMTAWMTAGMDEXMTAWMTAGMDEXMTAXMTEGMDEXMTAWMTAGMDEXMTAWMTEGMDEXMTAWMTAGMDEXMTAWMTEGMDEXMTAWMTAGMDEXMTAWMTEGMDEXMTAWMTAGMDEXMTAWMTAGMDEXMTAXMTEGMDEXMTAWMTAGMDEXMTAWMTAGMDEXMTAWMTAGMDEXMTAXMTEGMDEXMTAXMTEGMDEXMDAMMDAGMDEXMDAXMTAWMTAGMDAXMTAWMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMDAMMDAGMDEXMDAXMTAWMTAGMDAXMTAWMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMTAXMTAGMDEXMT
```

Now it is ready for decoding:

```
>>> encoded=b64decode(x).split()
>>> encoded
[b'01000011', b'01010100', b'01000011', b'01010100', b'01000110', b'01111011', b'01000000', b'01101100', b'00110000', b'
01110100', b'01011111', b'01101111', b'01100110', b'01011111', b'00100100', b'00110011', b'01100011', b'01110010', b'01
10011', b'00110111', b'01011111', b'01101100', b'01100001', b'01101110', b'00110110', b'01110101', b'01100001', b'011001
11', b'00110011', b'00100100', b'01111101']
>>> for i in encoded:
... print(chr(int(i,2)),end='')
...
CTCTF{@l0t_of_$3cr37_lan6uag3$}>>> _
```

1001 encodings in total, not counting the compression :D The flag is CTCTF{@l0t_of_\$3cr37_lan6uag3\$}.