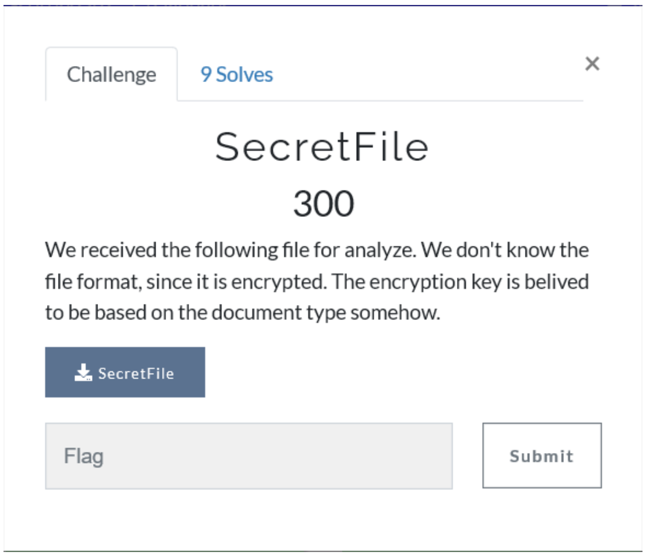
300


Challenge
9 Solves

SecretFile

300

We received the following file for analyze. We don't know the file format, since it is encrypted. The encryption key is belived to be based on the document type somehow.

SecretFile

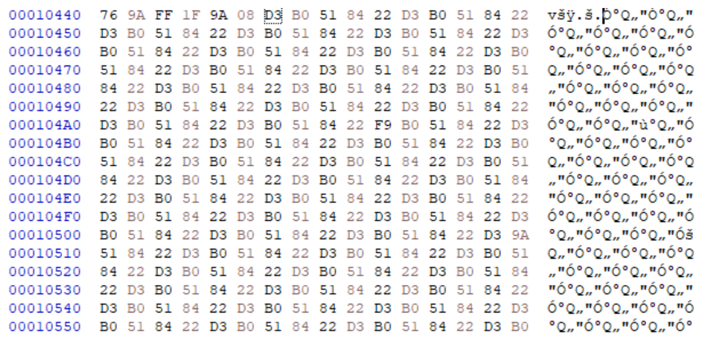
Flag

Submit

This challenge we got a file that was encrypted.

Binwalk, file or strings did not give any meaningful information.

When checking the file in the hex editor the following stood out:

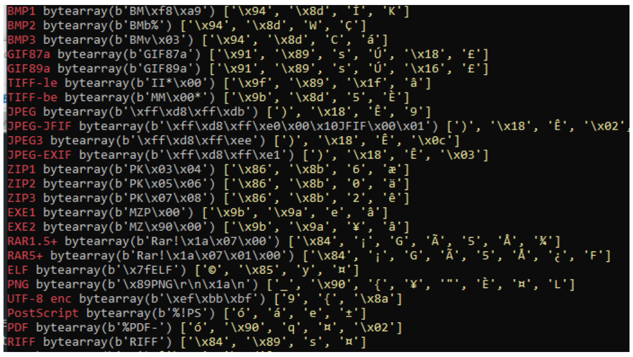

```
00010440  76 9A FF 1F 9A 08 D3 B0 51 84 22 D3 B0 51 84 22  všÿ.š.Ó°Q„"Ó°Q„"
00010450  D3 B0 51 84 22 D3 B0 51 84 22 D3 B0 51 84 22 D3  Ó°Q„"Ó°Q„"Ó°Q„"Ó
00010460  B0 51 84 22 D3 B0 51 84 22 D3 B0 51 84 22 D3 B0  °Q„"Ó°Q„"Ó°Q„"Ó°
00010470  51 84 22 D3 B0 51 84 22 D3 B0 51 84 22 D3 B0 51  Q„"Ó°Q„"Ó°Q„"Ó°Q
00010480  84 22 D3 B0 51 84 22 D3 B0 51 84 22 D3 B0 51 84  „"Ó°Q„"Ó°Q„"Ó°Q„
00010490  22 D3 B0 51 84 22 D3 B0 51 84 22 D3 B0 51 84 22  "Ó°Q„"Ó°Q„"Ó°Q„"
000104A0  D3 B0 51 84 22 D3 B0 51 84 22 F9 B0 51 84 22 D3  Ó°Q„"Ó°Q„"ù°Q„"Ó
000104B0  B0 51 84 22 D3 B0 51 84 22 D3 B0 51 84 22 D3 B0  °Q„"Ó°Q„"Ó°Q„"Ó°
000104C0  51 84 22 D3 B0 51 84 22 D3 B0 51 84 22 D3 B0 51  Q„"Ó°Q„"Ó°Q„"Ó°Q
000104D0  84 22 D3 B0 51 84 22 D3 B0 51 84 22 D3 B0 51 84  „"Ó°Q„"Ó°Q„"Ó°Q„
000104E0  22 D3 B0 51 84 22 D3 B0 51 84 22 D3 B0 51 84 22  "Ó°Q„"Ó°Q„"Ó°Q„"
000104F0  D3 B0 51 84 22 D3 B0 51 84 22 D3 B0 51 84 22 D3  Ó°Q„"Ó°Q„"Ó°Q„"Ó
00010500  B0 51 84 22 D3 B0 51 84 22 D3 B0 51 84 22 D3 9A  °Q„"Ó°Q„"Ó°Q„"Óš
00010510  51 84 22 D3 B0 51 84 22 D3 B0 51 84 22 D3 B0 51  Q„"Ó°Q„"Ó°Q„"Ó°Q
00010520  84 22 D3 B0 51 84 22 D3 B0 51 84 22 D3 B0 51 84  „"Ó°Q„"Ó°Q„"Ó°Q„
00010530  22 D3 B0 51 84 22 D3 B0 51 84 22 D3 B0 51 84 22  "Ó°Q„"Ó°Q„"Ó°Q„"
00010540  D3 B0 51 84 22 D3 B0 51 84 22 D3 B0 51 84 22 D3  Ó°Q„"Ó°Q„"Ó°Q„"Ó
00010550  B0 51 84 22 D3 B0 51 84 22 D3 B0 51 84 22 D3 B0  °Q„"Ó°Q„"Ó°Q„"Ó°
```

Lots of bytes repeating with only a few bytes changing every here and there.

I tried XORing the bytes that had changed in the pattern with the bytes that was commonly in the same position and all bytes became “*”. This somehow indicated to me that XOR was in play somehow here. But when I tried XORing the rest of the file with “*” I got gibberish as result.

I now went down a very very deep rabbit hole trying to figure out the encryption key. assuming it had something to do with the file type of the original file.

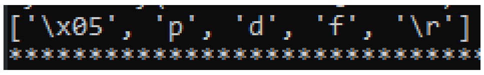
I XORed the first few bytes with the magic bytes of each file type I could think of but all gave me gibberish.



```
BMP1 bytearray(b'BM\xf8\xa9') ['\x94', '\x8d', 'İ', 'K']
BMP2 bytearray(b'BMb%') ['\x94', '\x8d', 'W', 'Ç']
BMP3 bytearray(b'BMv\x03') ['\x94', '\x8d', 'C', 'á']
GIF87a bytearray(b'GIF87a') ['\x91', '\x89', 's', 'Ú', '\x18', '£']
GIF89a bytearray(b'GIF89a') ['\x91', '\x89', 's', 'Ú', '\x16', '£']
TIFF-le bytearray(b'II*\x00') ['\x9f', '\x89', '\x1f', 'â']
TIFF-be bytearray(b'MM\x00*') ['\x9b', '\x8d', '5', 'È']
JPEG bytearray(b'\xff\xd8\xff\xdb') [')', '\x18', 'Ê', '9']
JPEG-JFIF bytearray(b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01') [')', '\x18', 'Ê', '\x02',
JPEG3 bytearray(b'\xff\xd8\xff\xee') [')', '\x18', 'Ê', '\x0c']
JPEG-EXIF bytearray(b'\xff\xd8\xff\xe1') [')', '\x18', 'Ê', '\x03']
ZIP1 bytearray(b'PK\x03\x04') ['\x86', '\x8b', '6', 'æ']
ZIP2 bytearray(b'PK\x05\x06') ['\x86', '\x8b', '0', 'ä']
ZIP3 bytearray(b'PK\x07\x08') ['\x86', '\x8b', '2', 'ê']
EXE1 bytearray(b'MZP\x00') ['\x9b', '\x9a', 'e', 'â']
EXE2 bytearray(b'MZ\x90\x00') ['\x9b', '\x9a', '¥', 'â']
RAR1.5+ bytearray(b'Rar!\x1a\x07\x00') ['\x84', '¡', 'G', 'Ã', '5', 'Å', '¾']
RAR5+ bytearray(b'Rar!\x1a\x07\x01\x00') ['\x84', '¡', 'G', 'Ã', '5', 'Å', '¿', 'F']
ELF bytearray(b'\x7fELF') ['©', '\x85', 'y', '¤']
PNG bytearray(b'\x89PNG\r\n\x1a\n') ['_', '\x90', '{', '¥', '"', 'È', '¤', 'L']
UTF-8 enc bytearray(b'\xef\xbb\xbf') ['9', '{', '\x8a']
PostScript bytearray(b'%!PS') ['ó', 'á', 'e', '±']
PDF bytearray(b'%PDF-') ['ó', '\x90', 'q', '¤', '\x02']
RIFF bytearray(b'RIFF') ['\x84', '\x89', 's', '¤']
```

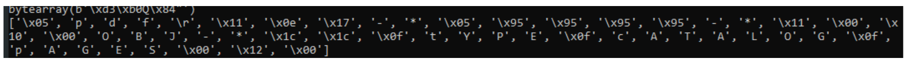
What I tried then was to see if the bytes hidden in the repeatable pattern was 0x00 bytes.

When XORing the first few bytes in the file with the bytes from the file "D3 B0 51 84 22", I noticed a very pleasant surprise:



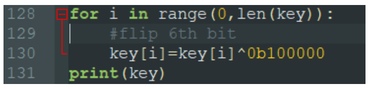
```
['\x05', 'p', 'd', 'f', '\r']
********************************
```

This was not entirely gibberish! Repeating the same key over a few times gave the following result:



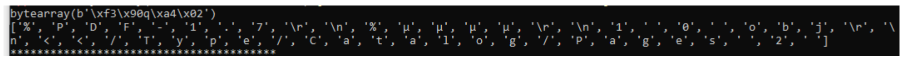
```
bytearray(b'\xd3\xb0Q\x84"')
['\x05', 'p', 'd', 'f', '\r', '\x11', '\x0e', '\x17', '-', '*', '\x05', '\x95', '\x95', '\x95', '\x95', '-', '*', '\x11', '\x00', '\x
10', '\x00', 'O', 'B', 'J', '-', '*', '\x1c', '\x1c', '\x0f', 't', 'Y', 'P', 'E', '\x0f', 'c', 'A', 'T', 'A', 'L', 'O', 'G', '\x0f',
'p', 'A', 'G', 'E', 'S', '\x00', '\x12', '\x00']
```

The bytes seemed "almost" correct. Checking what the PDF header should look like showed that all bytes had the 6th bit wrong:



```python
for i in range(0,len(key)):
    #flip 6th bit
    key[i]=key[i]^0b100000
print(key)
```

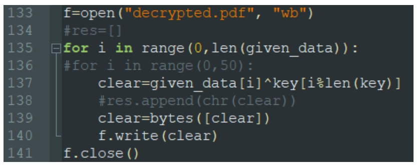
Correcting the key and we get the following result:



```
bytearray(b'\xf3\x90q\xa4\x02')
['%', 'P', 'D', 'F', '-', '1', '.', '7', '\r', '\n', '%', 'µ', 'µ', 'µ', 'µ', '\r', '\n', '1', ' ', '0', ' ', 'o', 'b', 'j', '\r', '\
n', '<', '<', '/', 'T', 'y', 'p', 'e', '/', 'C', 'a', 't', 'a', 'l', 'o', 'g', '/', 'P', 'a', 'g', 'e', 's', ' ', '2', ' ']
*******************************************
```

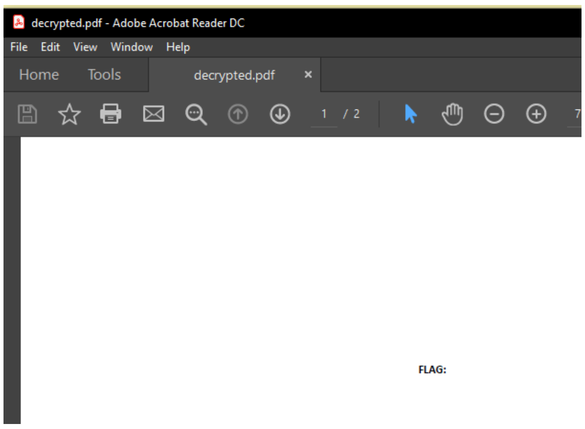
Much better!

Now write it to a file:



```python
f=open("decrypted.pdf", "wb")
#res=[]
for i in range(0,len(given_data)):
#for i in range(0,50):
    clear=given_data[i]^key[i%len(key)]
    #res.append(chr(clear))
    clear=bytes([clear])
    f.write(clear)
f.close()
```

And we get our flag! 😊


decrypted.pdf - Adobe Acrobat Reader DC
File Edit View Window Help
Home Tools decrypted.pdf
1 / 2

FLAG: