A Crazy Dev

Auteur : Delionor CTF FlagMalo 2022

Bonjour à tous, ce WU portera sur la résolution du challenge « A Crazy Dev » du CTF FlagMalo 2022
L'énoncé nous indique que le fichier fournit possède sûrement des fonctionnalités cachées
On commence donc par télécharger ce fichier .pyc ... Si on veut réellement voir ce qui a l'intérieur, il faut le décompiler :)
On peut alors utiliser uncompyle6 ou d'autres outils de même type... et on tombe sur le code suivant :

```
'''
@Author Dede

You're right, PI is not what we're here for
I wrote different functions in this file, and I forgot which ones I used to encode the flag. I only know I wrote a reminder somehow about h
'''
import base64
import binascii
import math


#Must be my reminder but what does that mean.
myreminder = "PokerFace ^ ZumbaCafeo = \n\x1a\x06\x07\x13%\x00\x05\x00. What an interesting equation isn't it ? By the way, do you like purp

#AND THIS IS NOT THE FLAG DON'T PAY ATTENTION
encodedflag = "011100100111001010001100010011001000001010011010100100100010001010111010010100100101010010101110011000100110001001110100111

def gettingPIFromUser():
    nice = input("give me PI with it's 15 decimals")
    return nice

def comparePIWithInput(thisisaparameter):
    nice = thisisaparameter.encode('ascii')
    pii = str(math.pi).encode('ascii')
    if pii == nice:
        return "Oh right, it's PI !"
    else:
        return "You fail my challenge :c"

take1t = ""
def f14gM4l0():
    thisisnotbinary = "01100100 01110101 00100000 01110011 01110101 01100011 01101000 01110011 01110100 00100000 01101110 01101001 01100011
    return thisisnotbinary
```

Figure 1 – Début du code décompilé

Et maintenant, c'est parti pour un peu de lecture ! Le nouvel énoncé nous dit que la variable encodedflag a été obtenue grâce à certaines des fonctions suivantes. Le développeur nous informe également qu'il s'était écrit un petit mémo pour se rappeler quelles fonctions il avait utilisées et dans quel ordre.

Le mémo en question nous dit ceci : `PokerFace ^ ZumbaCafeo = \n\x1a\x06\x07\x13%\x00\x05\x00.` Il faut savoir ici que le symbole ^ représente la fonction XOR en python.
Mais comment l'utiliser ? Et bien après une petite lecture, on voit très bien une fonction xor utilisé dans la méthode « followDeliodraws(arg) ».
Mais comment l'exploiter ? Il semble que l'argument de cette fonction est xoré avec la chaine de caractères suivante :

```
\x07\n\x07\tR\"\x0e\r\x00\x00\x7f\x01z,\x02\x17A.\x00\x02\x00OII\x00~rgs3Caw \x0f\x01w \x04\x06EF\x1d\
x00\x01\x01\x03E;E\x06\x07\x0c\nGM\x1cQ\x05\x13\x1b\x0e\x1d\x0fMI\x07\x01\x07\x11N\n\x1bDZ\x006\x11ET\
```

```
x0e\tA\x10@X\x0cu\x17\nRY\x06\x1bP\x19\x1dK\x12A\x03U\x11\x1f\x02\x13EMT2\x01\x00\r\x01\x00\x05\x1d\
x00\ x07EO\x13D\t\x1dT\x14\x002\x1b\x14E\x16&D\x16\x03]\tLWK\x00X\x03\x0b\x02\x1f\n[\t\x1f\x1ec\x06\
x0f<\ x08OE>\x0c\x02\x00&DE1\x17\x07\x1f\n\x1c\x06\x07.
```

Il faut donc réussir à trouver une autre chaine de caractère à passer en argument de cette fonction. Même si c'est redondant, il va falloir essayer toutes les chaines de caractères présentes dans le fichier.
Pour cela, lançons le programme **sans fermer le terminal** après l'exécution. Pour cela, on utilise la commande suivante :

```
top>python -i chalencoding.py
```

Notre invite de commande nous demande ensuite les fameuses décimales de PI. Honnêtement, elles ne nous intéressent plus du tout :) Rentrez ce que vous voulez, puis remarquez que votre terminal change d'affichage: il vous affiche désormais '>>>'

```
give me PI with it's 15 decimals555
You fail my challenge :c
>>>
```

Cela signifie que vous êtes toujours ' à l'intérieur ' du programme python, même si vous avez fini de l'exécuter. Par exemple, vous pouvez print n'importe quelle variable accessible :

```
>>> print(myreminder)
PokerFace ^ ZumbaCafeo =
→♠‼% ♣ . What an interesting equation isn't it ? By the way, do you like purple ? We do quite a lot. And do you like sna
kes ? Cobra, viper, python..?
>>>
```

Nous pouvons également utiliser les fonctions présentes dans le fichier. Comme prévu, essayons de xorer des chaines avec la fonction folloDeliodraws(arg) :

```
>>> followDeliodraws(myreminder)
"Well done !! You made it that far !! Here's how I used my program to encode the flag!! User input was converted in this
 order : superIdol() > expecto() > imNiceHereTheAnswer()"
>>>
```

Chance pour nous, la première chaîne du programme (myreminder) semble renvoyer quelque chose d'intéressant :)
Nous avons maintenant l'ordre et les fonctions utilisées pour encoder le flag.

Nous devons maintenant écrire un programme qui inversera le procédé :
Le programme finit par utiliser imNiceHereTheAnswer(arg). Cela signifie que nous devons commencer par défaire cette fonction. Analysons la :
Elle prend un paramètre d'entré auquel elle ajoute la chaîne `01001100 01101111 01110010 01100101 01101101 00100000 01101001 01110000 01110011 01110101 01101101`
puis elle print tous les caractères de cette nouvelle chaine, puis elle la retourne pour que le dernier caractère se retrouve en premier -c'est l'effet de la ligne `return takeit[::-1]`-

En résumé, elle ne fait pas grand-chose de très intéressant : il suffit de ré-inverser la chaîne pour pouvoir passer aux fonctions suivantes

```
>>> followDeliodraws(myreminder)
"Well done !! You made it that far !! Here's how I used my program to encode the flag!! User input was converted in this
 order : superIdol() > expecto() > imNiceHereTheAnswer()"
>>> step1 = encodedflag[::-1]
>>> print(step1)
0b1111010101100101101011001011000010101100111000010100100100011100101110010001100100011001110101001010100100101001011101
0100010001001001010110010100000100110010001100010100111001001110
>>>
```

Nous pouvons ici remarquer que les deux premiers caractères sont ‘0b’… On retrouve une chaîne en binaire ?
Cela tombe bien, car à la lecture de la fonction qui a précédé followDeliodraws(), on comprend vite que celle ci réalise une simple conversion binaire.
Nous devons donc écrire une fonction pour repasser de binaire à ascii.
Ou pas ! Il se trouve qu’il y a une autre fonction maniant du binaire dans le fichier : forYouZumba(). Si vous avez pris le temps de la lire… peut être savez vous déjà ce qu’elle va faire ? :)

```
>>> step2 = forYouZumba(step1)
>>> print(step2)
b'=Yk,+8RG.FFu*J]DIYA21NN'
>>>
```

On a vraiment bien avancé : il ne reste qu’une seule fonction à défaire ‘superIdol()’. Cette fonction prend un argument qui sera ensuite codé en a85. Autrement dit… nous pouvons la décoder et la repasser à l’état d’origine ?

```
>>> step2 = forYouZumba(step1)
>>> print(step2)
b'=Yk,+8RG.FFu*J]DIYA21NN'
>>> step3 = base64.a85decode(step2)
>>> print(step3)
b'Y3s_I_L0v3_3nc0d3s'
```

Et voici notre flag tant attendu ! Plus qu’à l’écrire au format FMCTF{Y3s_I_L0v3_3nc0d3s}