24/06/2023 – PicoCTF – Transformation

Link: <https://play.picoctf.org/practice/challenge/104>

O arquivo representado a seguir é um programa em linguagem Python. Ele foi comentado visando ser auto explicativo e demonstrando como este CTF foi resolvido. O CTF entrega um arquivo “enc” e uma função de encriptação “''.join([chr((ord(flag[i]) << 8) + ord(flag[i + 1])) for i in range(0, len(flag), 2)])”, feita em linguagem Python. O raciocínio para resolver tal desafio está descrito no código abaixo.

*# armazenando o resultado da função de encriptação em uma variável*

result = "灩捯䍔䙻ㄶ形楴獟楮獴㌴摟潦弸弰㑣〷㘰摽"

print("Resultado da função de encriptação: ", result)

*# exemplo*

flag = "Ahmad."

print(*f*"Para uma entrada {flag}: {''.join([chr((ord(flag[i]) << 8) + ord(flag[i + 1])) for i in range(0, len(flag), 2)])}") *# verificado que o tamanho da flag deve ser um número par*

*# dicionário contendo os valores de cada caractere unicode de um teclado convencional (usado posteriormente)*

ku = {

    'A': 65,

    'B': 66,

    'C': 67,

    'D': 68,

    'E': 69,

    'F': 70,

    'G': 71,

    'H': 72,

    'I': 73,

    'J': 74,

    'K': 75,

    'L': 76,

    'M': 77,

    'N': 78,

    'O': 79,

    'P': 80,

    'Q': 81,

    'R': 82,

    'S': 83,

    'T': 84,

    'U': 85,

    'V': 86,

    'W': 87,

    'X': 88,

    'Y': 89,

    'Z': 90,

    'a': 97,

    'b': 98,

    'c': 99,

    'd': 100,

    'e': 101,

    'f': 102,

    'g': 103,

    'h': 104,

    'i': 105,

    'j': 106,

    'k': 107,

    'l': 108,

    'm': 109,

    'n': 110,

    'o': 111,

    'p': 112,

    'q': 113,

    'r': 114,

    's': 115,

    't': 116,

    'u': 117,

    'v': 118,

    'w': 119,

    'x': 120,

    'y': 121,

    'z': 122,

    '0': 48,

    '1': 49,

    '2': 50,

    '3': 51,

    '4': 52,

    '5': 53,

    '6': 54,

    '7': 55,

    '8': 56,

    '9': 57,

    ' ': 32,

    '!': 33,

    '@': 64,

    '#': 35,

    '$': 36,

    '%': 37,

    '^': 94,

    '&': 38,

    '\*': 42,

    '(': 40,

    ')': 41,

    '-': 45,

    '\_': 95,

    '=': 61,

    '+': 43,

    '[': 91,

    ']': 93,

    '{': 123,

    '}': 125,

    '|': 124,

    '\\': 92,

    ';': 59,

    ':': 58,

    "'": 39,

    '"': 34,

    ',': 44,

    '<': 60,

    '.': 46,

    '>': 62,

    '/': 47,

    '?': 63,

    '`': 96,

    '~': 126

}

*# invertendo o dicionário de keyboard\_unicode*

kui = *dict*(map(reversed, ku.items()))

*# descobrindo o tamanho da flag*

tamanhos\_possiveis = []

tamanho = 0

nao\_encontrou = True

acabou\_as\_possibilidades = False

while(not acabou\_as\_possibilidades):

    if(len(result) == len(range(0, tamanho, 2))):

        nao\_encontrou = False

        tamanhos\_possiveis.append(tamanho)

    elif(nao\_encontrou == False):

        acabou\_as\_possibilidades = True

    tamanho += 1

print("Tamanhos possíveis para a flag buscada: ", tamanhos\_possiveis)

*# tamanho = 37/38, mas o tamanho da flag deve ser par, então o tamanho é 38*

*# analisando a função de encriptação*

"""

灩 -> uso do caractere 0 e 1 da flag

ord(flag[0]) -> A

ord(flag[1]) -> B

灩 = chr((ord(flag[0]) << 8) + ord(flag[1]))

灩 = A\*2^8 + B

"""

*# tentando encontrar os valores possíveis para um único caractere 灩*

c = ord('灩')

unicode\_valores = ku.values()

a = -1

b = -1

for i in unicode\_valores:

    for j in unicode\_valores:

        if((i << 8) + j == c):

            a = i

            b = j

print(*f*"Para o caractere '灩' de unicode {c}, foram encontrados os caracteres '{kui[a]}' e '{kui[b]}' de unicode {a} e {b}, respectivamente.")

*# foram encontrados os caracteres "p" e "i", que devem originar "picoCTF", dando um indício de que esta linha de raciocínio pode estar correta*

*# fazendo por bruteforce para todos os 38 caracteres:*

flag\_posicoes = *list*(range(len(range(0, 38, 2))))

flag\_final = *list*()

for p in flag\_posicoes:

    a = -1

    b = -1

    c = ord(result[p])

    for i in unicode\_valores:

        for j in unicode\_valores:

            if((i << 8) + j == c):

                a = i

                b = j

    if(a != -1 and b != -1):

        flag\_final.append(kui[a])

        flag\_final.append(kui[b])

    else:

        print("ERRO")

        flag\_final.append(" ")

        flag\_final.append(" ")

*# exibindo a flag final*

print(''.join(flag\_final))

A seguir está representado o que aparece no terminal ao se rodar o arquivo acima:

**Resultado da função de encriptação: 灩捯䍔䙻ㄶ形楴獟楮獴㌴摟潦弸弰㑣〷㘰摽**

**Para uma entrada Ahmad.: 䅨浡搮**

**Tamanhos possíveis para a flag buscada: [37, 38]**

**Para o caractere '灩' de unicode 28777, foram encontrados os caracteres 'p' e 'i' de unicode 112 e 105, respectivamente.**

**$flag**