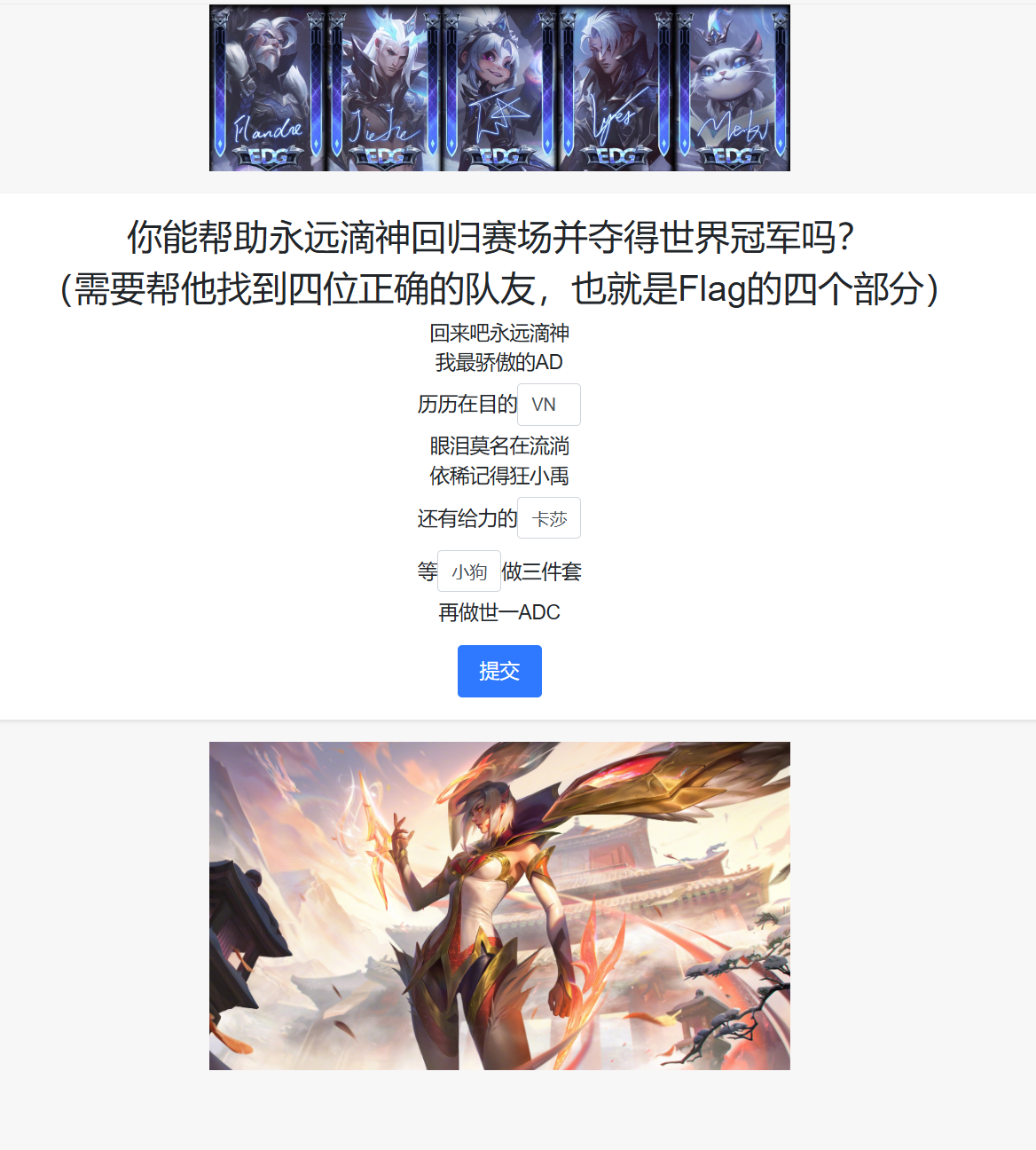
# ISCC2024 WriteUp

AkyOI 李承达 [3433778745@qq.com](mailto:3433778745@qq.com)

## Web+回来吧永远滴神

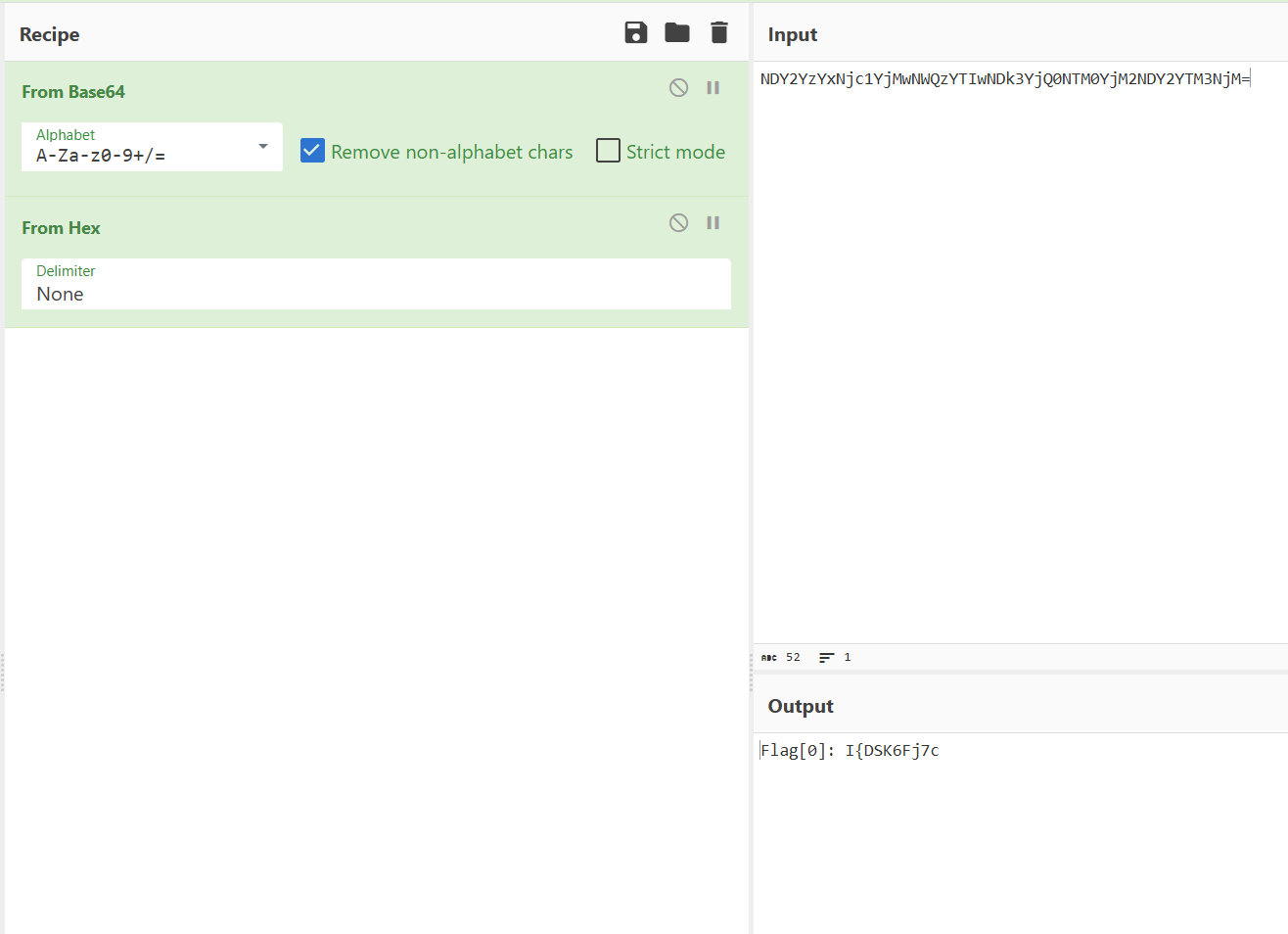
## 解题思路

提交答案进入隐藏关卡



源码有base64

解密之后就是一段flag内容

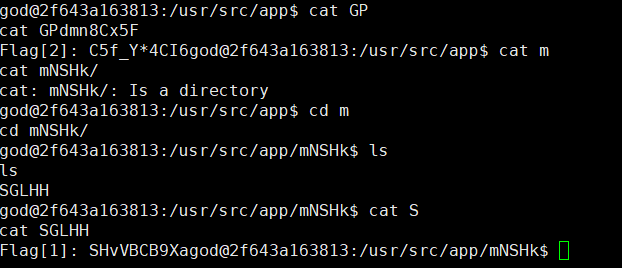


猜测应该是ssti吧，于是直接尝试。



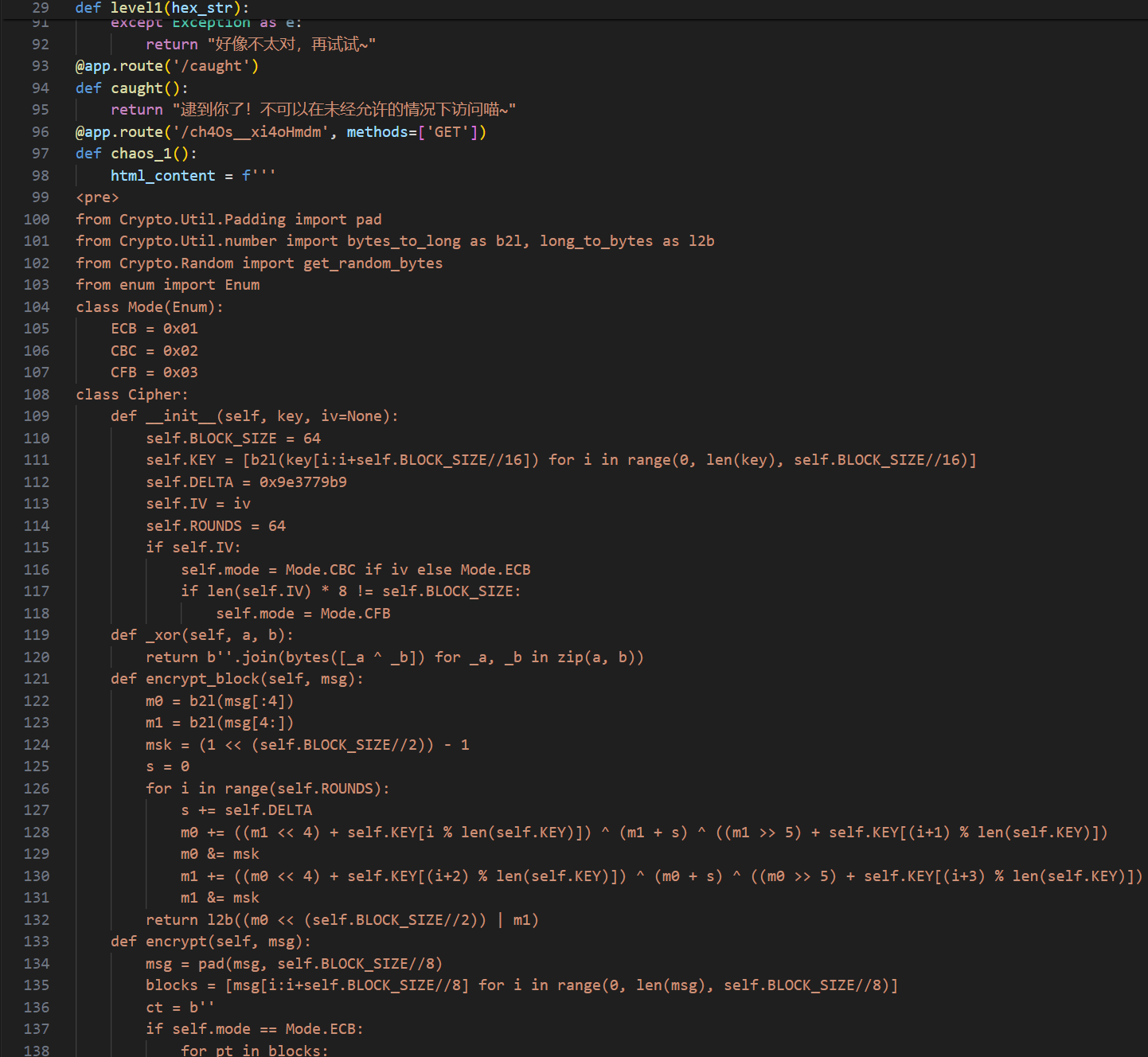
还真是

Ssti直接跑脚本反弹shell



找到flag的另外部分

app.py中找到加密逻辑



按照逻辑写解密脚本得到最后一部分flag，

其实只要在原本代码基础上写一个解密函数就行了

之后拼接起来跑一次栅栏密码就是ISCC开头的flag了



## Exp

解密脚本

from Crypto.Util.Padding import pad

from Crypto.Util.number import bytes\_to\_long as b2l, long\_to\_bytes as l2b

from Crypto.Random import get\_random\_bytes

from enum import Enum

class Mode(Enum):

ECB = 0x01

CBC = 0x02

CFB = 0x03

class Cipher:

def \_\_init\_\_(self, key, iv=None):

self.BLOCK\_SIZE = 64

self.KEY = [b2l(key[i:i+self.BLOCK\_SIZE//16]) for i in range(0, len(key), self.BLOCK\_SIZE//16)]

self.DELTA = 0x9e3779b9

self.IV = iv

self.ROUNDS = 64

if self.IV:

self.mode = Mode.CBC if iv else Mode.ECB

if len(self.IV) \* 8 != self.BLOCK\_SIZE:

self.mode = Mode.CFB

def \_xor(self, a, b):

return b''.join(bytes([\_a ^ \_b]) for \_a, \_b in zip(a, b))

def encrypt\_block(self, msg):

m0 = b2l(msg[:4])

m1 = b2l(msg[4:])

msk = (1 << (self.BLOCK\_SIZE//2)) - 1

s = 0

for i in range(self.ROUNDS):

s += self.DELTA

m0 += ((m1 << 4) + self.KEY[i % len(self.KEY)]) ^ (m1 + s) ^ ((m1 >> 5) + self.KEY[(i+1) % len(self.KEY)])

m0 &= msk

m1 += ((m0 << 4) + self.KEY[(i+2) % len(self.KEY)]) ^ (m0 + s) ^ ((m0 >> 5) + self.KEY[(i+3) % len(self.KEY)])

m1 &= msk

return l2b((m0 << (self.BLOCK\_SIZE//2)) | m1)

def decrypt(self, val0000000000000000):

blocks = [val0000000000000000[i:i+self.BLOCK\_SIZE//8] for i in range(0, len(val0000000000000000), self.BLOCK\_SIZE//8)]

msg = b''

if self.mode == Mode.ECB:

for block in blocks:

msg += self.decrypt\_block(block)

elif self.mode == Mode.CBC:

X = self.IV

for block in blocks:

pt = self.\_xor(X, self.decrypt\_block(block))

msg += pt

X = block

elif self.mode == Mode.CFB:

X = self.IV

for block in blocks:

output = self.encrypt\_block(X)

pt = self.\_xor(output, block)

msg += pt

X = block

return pad(msg, self.BLOCK\_SIZE//8)

if \_\_name\_\_ == '\_\_main\_\_':

KEY = bytes.fromhex('3362623866656338306539313238353733373566366338383563666264386133')

IV = bytes.fromhex('64343537373337663034346462393931')

chitessst = Cipher(KEY, IV)

val0000000000000000 = bytes.fromhex('1cb8db8cabe8edbbddb236d5eb6f0cdeb610e9af855b52d3')

decrypted\_message = chitessst.decrypt(val0000000000000000)

print(decrypted\_message)