XYCTF2024 别管 Writeup

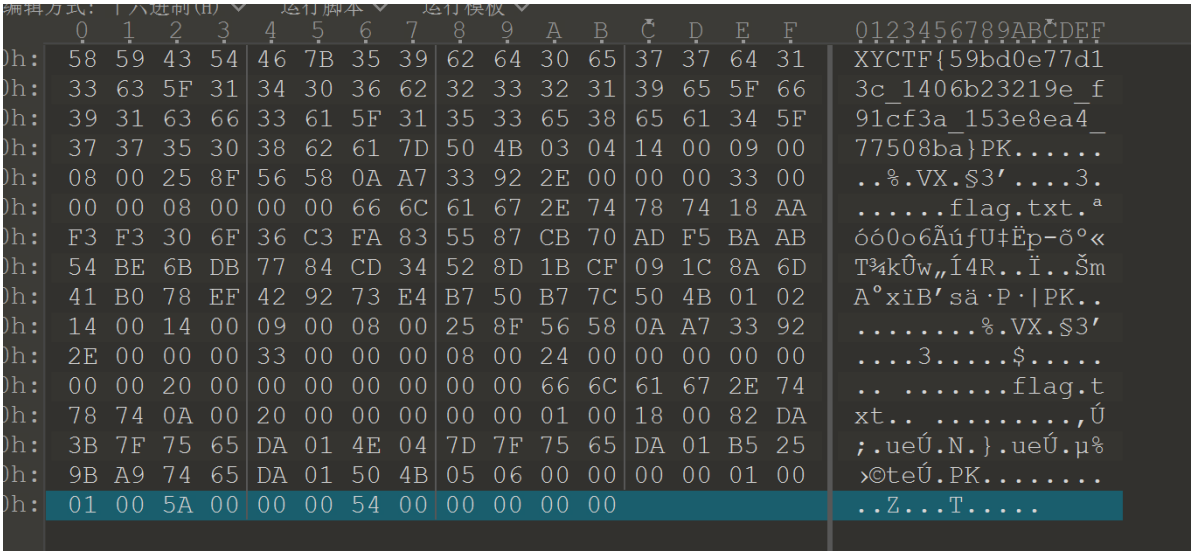
Misc

签到

扫码输入XYCTF2024，启动启动启动！！！得

真>签到

打开附件压缩包得



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0h:	58	59	43	54	46	7B	35	39	62	64	30	65	37	37	64	31	XYCTF{59bd0e77d1
0h:	33	63	5F	31	34	30	36	62	32	33	32	31	39	65	5F	66	3c_1406b23219e_f
0h:	39	31	63	66	33	61	5F	31	35	33	65	38	65	61	34	5F	91cf3a_153e8ea4_
0h:	37	37	35	30	38	62	61	7D	50	4B	03	04	14	00	09	00	77508ba}PK.....
0h:	08	00	25	8F	56	58	0A	A7	33	92	2E	00	00	00	33	00	..%.VX.§3'....3.
0h:	00	00	08	00	00	00	66	6C	61	67	2E	74	78	74	18	AA	......flag.txt.ª
0h:	F3	F3	30	6F	36	C3	FA	83	55	87	CB	70	AD	F5	BA	AB	óó0o6ÃúƒU‡Ëp-õº«
0h:	54	BE	6B	DB	77	84	CD	34	52	8D	1B	CF	09	1C	8A	6D	T¾kÛw„Í4R..Ï..Šm
0h:	41	B0	78	EF	42	92	73	E4	B7	50	B7	7C	50	4B	01	02	A°xïB'sä·P·|PK..
0h:	14	00	14	00	09	00	08	00	25	8F	56	58	0A	A7	33	92	........%.VX.§3'
0h:	2E	00	00	00	33	00	00	00	08	00	24	00	00	00	00	00	....3.....$.....
0h:	00	00	20	00	00	00	00	00	00	00	66	6C	61	67	2E	74	.. ..flag.t
0h:	78	74	0A	00	20	00	00	00	00	00	01	00	18	00	82	DA	xt.. ....,Ú
0h:	3B	7F	75	65	DA	01	4E	04	7D	7F	75	65	DA	01	B5	25	;.ueÚ.N.}.ueÚ.µ%
0h:	9B	A9	74	65	DA	01	50	4B	05	06	00	00	00	00	01	00	›©teÚ.PK........
0h:	01	00	5A	00	00	00	54	00	00	00	00	00					..Z...T.....

XYCTF{59bd0e77d13c_1406b23219e_f91cf3a_153e8ea4_77508ba}

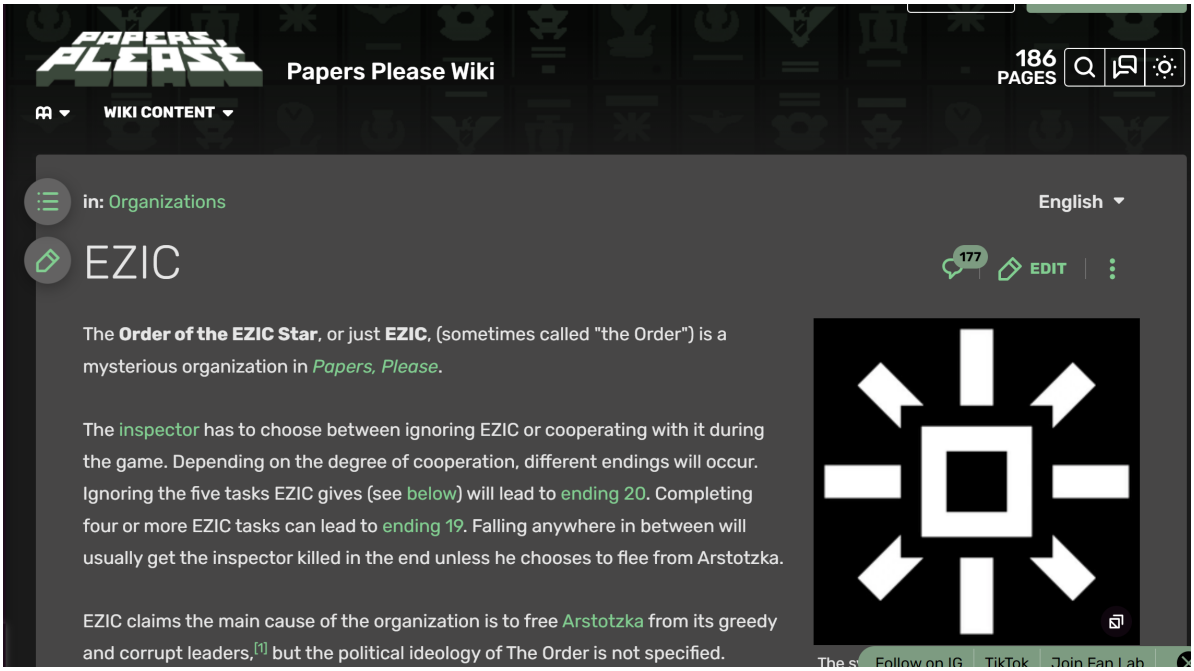
熊博士

得到一张小纸条，内容CBXGU{ORF_BV_NVR_BLF_CRZL_QQ}，随波一把梭

atbash解码，没听过思密达。

game

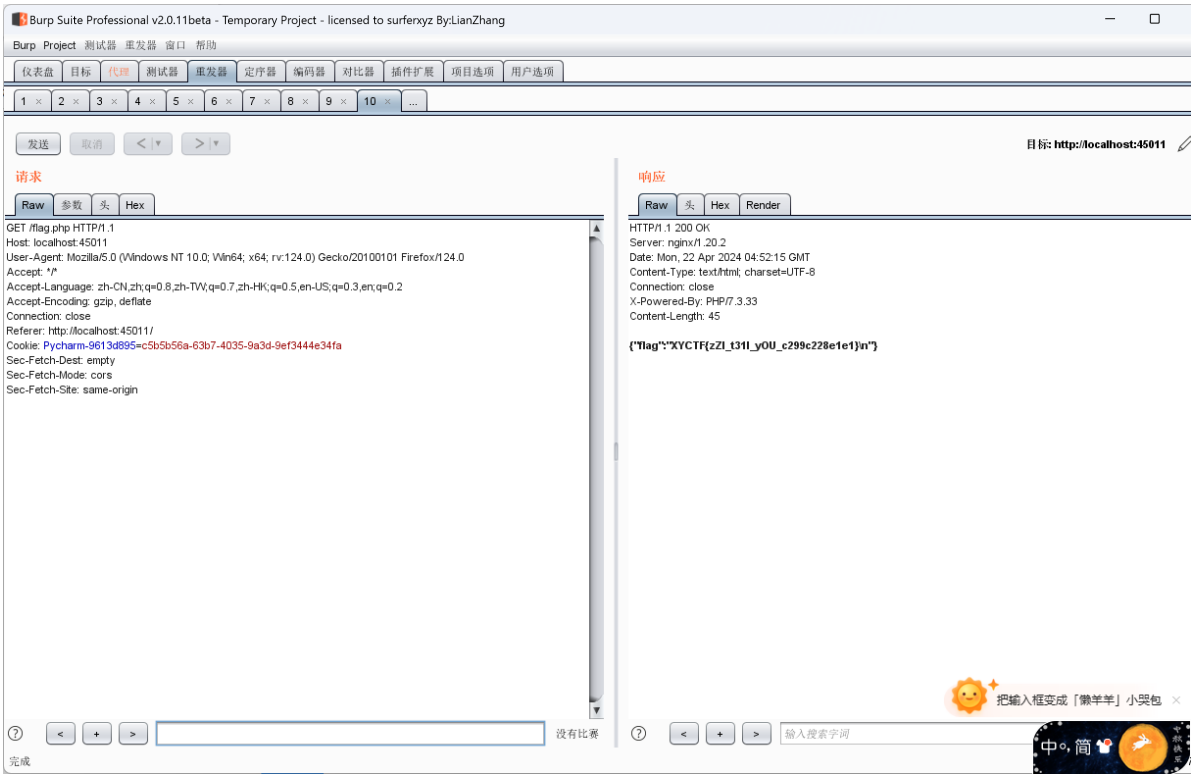
google识图得到游戏名字



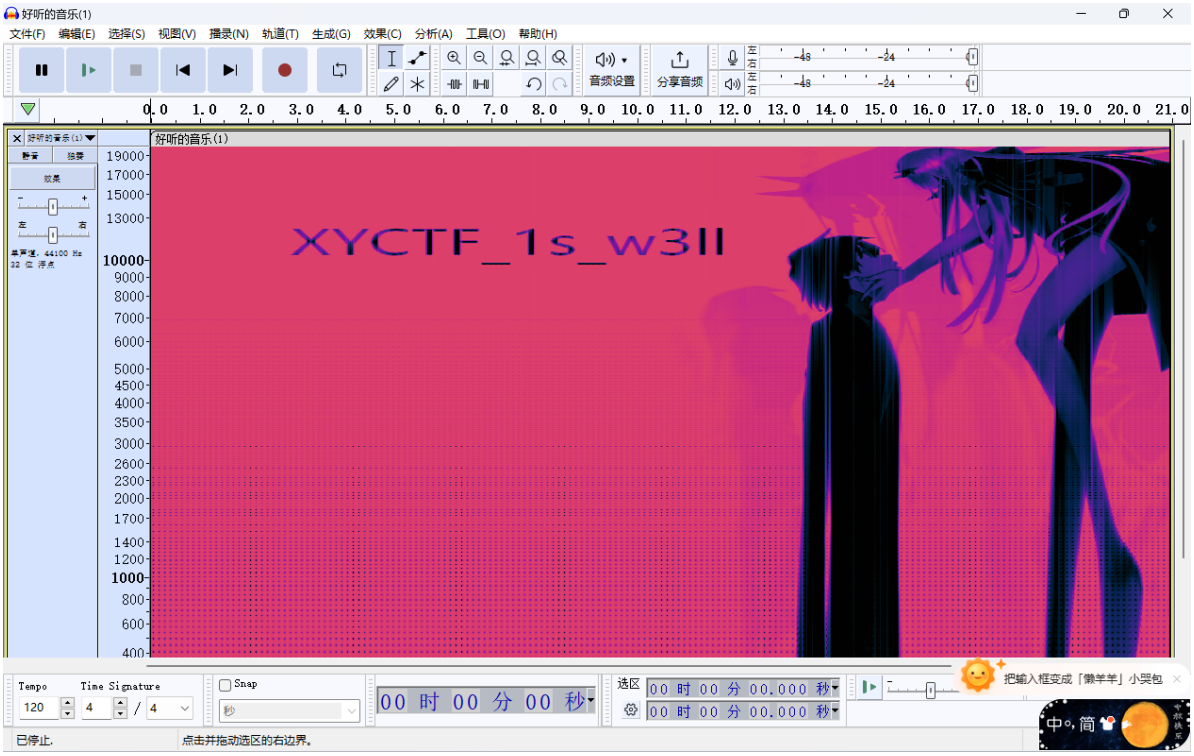
有关于游戏的简略介绍

zzl的护理小课堂

burp废包后得到flag.php请求后得到flag，我也不知道原理，为啥放包不可以，但是废包后就判定我过了……



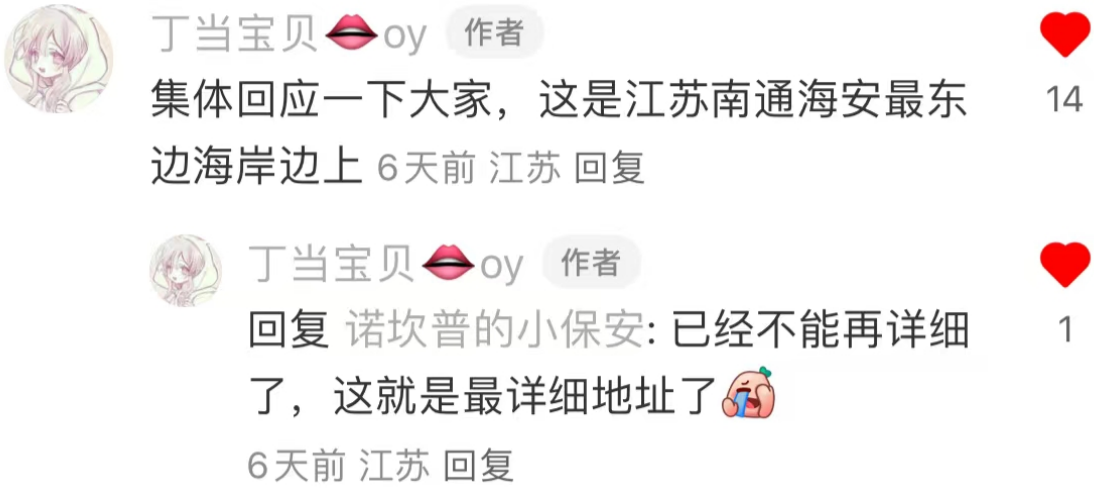
美妙的歌声



Audacity得到密钥XYCTF_1s_w3ll，既然有密钥又是音频文件，容易想到用deepsound查看隐藏文件，最后得到隐藏的flag文件

Osint1

直接社工善良的南通人民（bushi



找到东边海岸上把周围的几条路都试了一下，最后试出来是滨江东路，旁边自然是黄海。

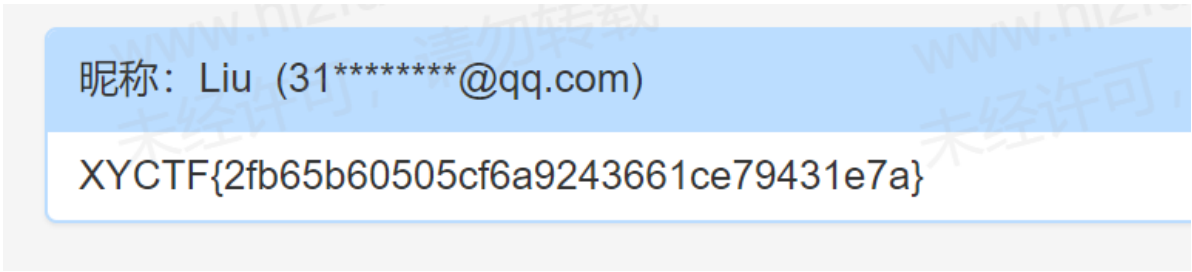
Osint2

根据高铁站的告示牌信息此高铁从洛阳龙门开往泸州，用12306查一下就得到了列车车次，景区名搜一下洛阳几个著名的景点，一个一个试就行。

Ez_osint

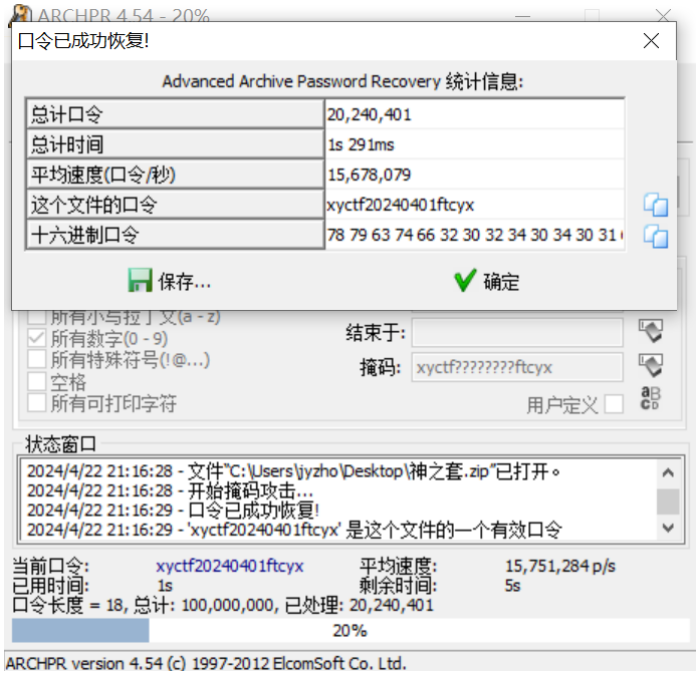
附件是一张图片，仔细看可以看到背景有淡淡的网站水印，为了更清楚点放到steg里查看一下

登录该网站，可以发现这是个时光邮局，根据图片首行内容可以知道这是封两年前的信，在网址栏修改page参数可以快速跳转到以往信件页面，找到2022/2/17时间的信就可以找到这封信，点开评论区得到flag

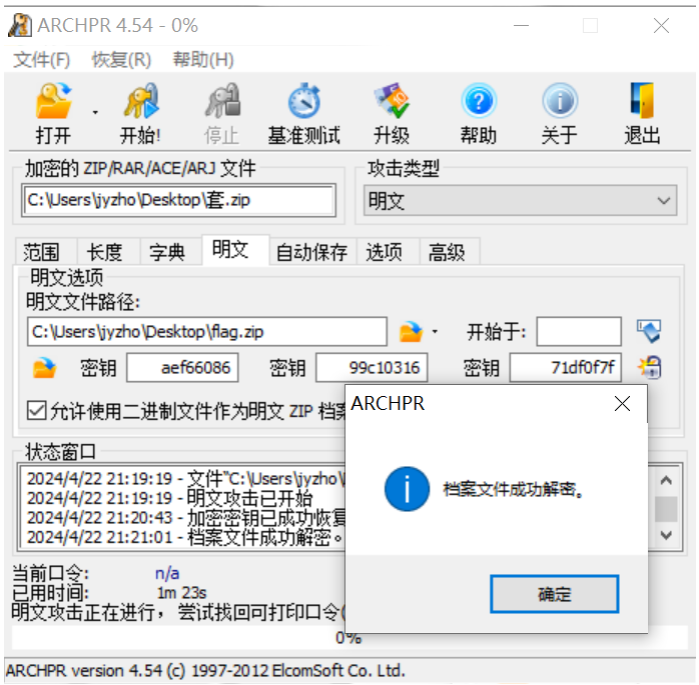


ez_隐写

下载一个压缩包，没有任何关于密码的线索，盲猜伪加密，改一下识别位



成功解压，但是解压出的hint.png好像有点问题，丢进010，发现在图片的宽高那边有一个较明显的错误



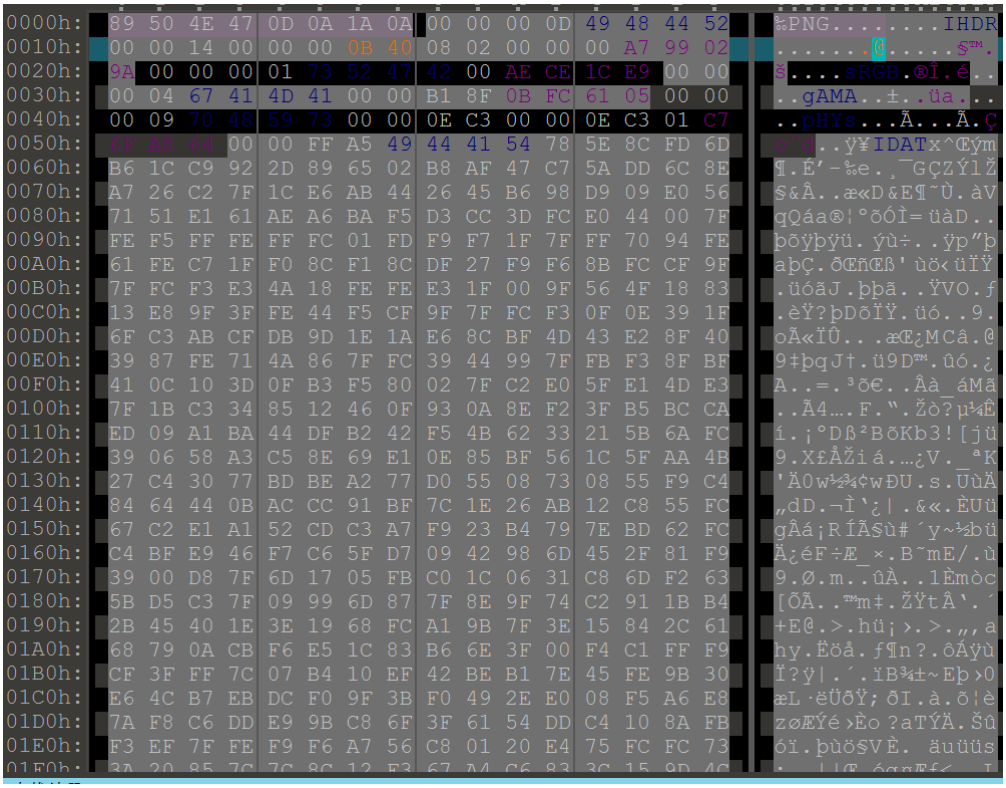
但是图片的宽高肉眼可见地有问题，放进linux更是显示crc校验有问题，多半是宽高的问题，爆破一下

脚本：

```
import binascii
import struct
crcbp = open("hint.png", "rb").read()     #填入图片名
crc32frombp = int(crcbp[29:33].hex(),16)      #读取图片中的CRC校验值
print(crc32frombp)

for i in range(6000):                          #宽度1-4000进行枚举
    for j in range(6000):                      #高度1-4000进行枚举
        data = crcbp[12:16] + \
            struct.pack('>i', i)+struct.pack('>i', j)+crcbp[24:29]
        crc32 = binascii.crc32(data) & 0xffffffff
```

```
        # print(crc32)
        if(crc32 == crc32frombp):
            print(i, j)
            print('hex:', hex(i), hex(j))
            exit(0)
```

```
2811822746
5120 2880
hex: 0x1400 0xb40
```

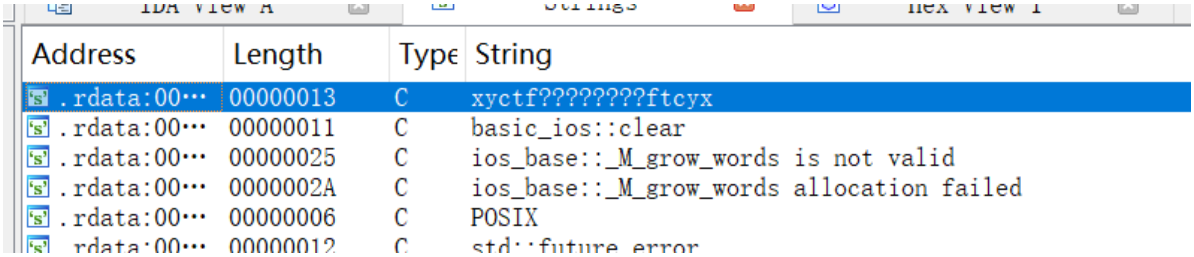


看到hint，压缩包密码20240401

根据压缩包名称WATERMARK，得知是盲水印。又因为只有一张图片，所以用watermark.exe提取即可

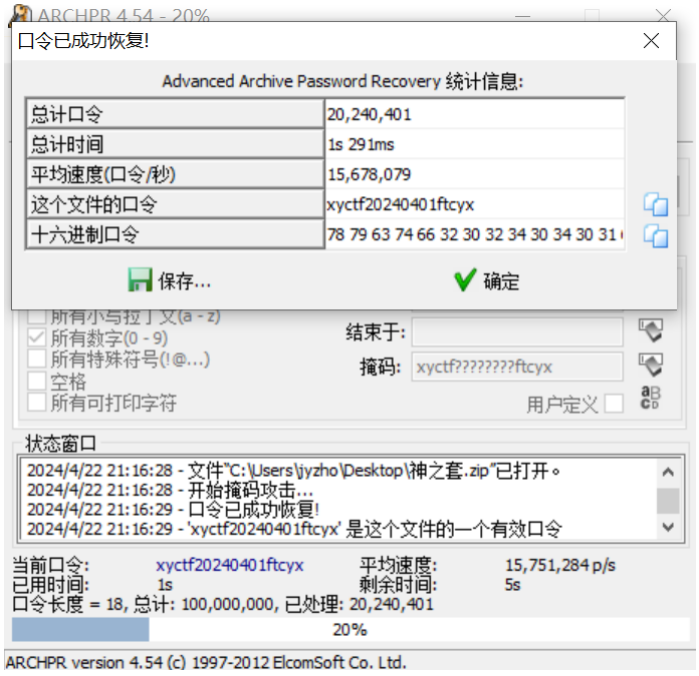
ZIP神之套

解压得到一个加密的压缩包和一个exe，将exe丢进IDA看到一个字符串，看来是提示使用掩码攻击

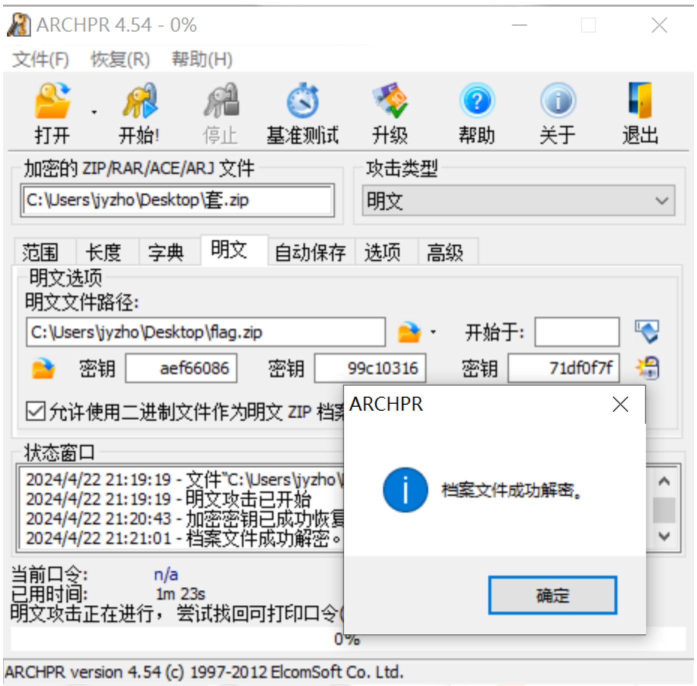


Address	Length	Type	String
.rdata:00…	00000013	C	xyctf????????ftcyx
.rdata:00…	00000011	C	basic_ios::clear
.rdata:00…	00000025	C	ios_base::_M_grow_words is not valid
.rdata:00…	0000002A	C	ios_base::_M_grow_words allocation failed
.rdata:00…	00000006	C	POSIX

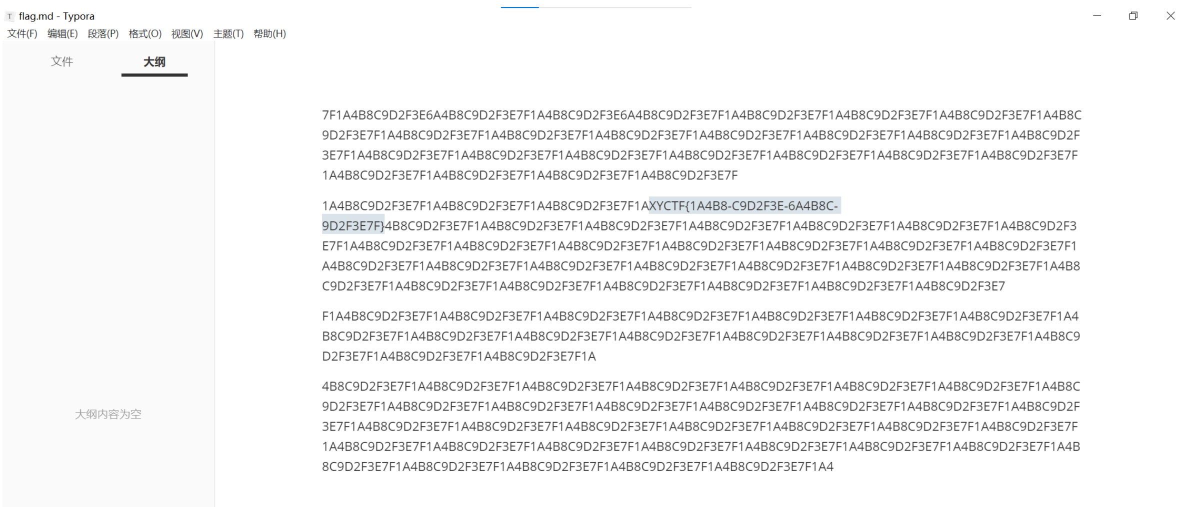
成功恢复密码



给了两个只有一个文件之差的压缩包，明文攻击



打开flag.md，找到flag



出题有点烦

下载下来一个压缩包，仍然没有任何关于密码的提示，010看了一下又不是伪加密，那么就直接硬干爆破密码

解压flag.7z，里面有五张图片，最终在5.png里面找到了flag。

binwalk分解一下5.png，得到一个压缩包

```
┌──(root㉿DESKTOP-LQMRD0K)-[~]
└─# binwalk -e 5.png --run-as=root

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
28            0x1C            TIFF image data, big-endian, offset of first image directory: 8
3068          0xBFC           Copyright string: "Copyright (c) 1998 Hewlett-Packard Company"
224265        0x36C09         Zip archive data, encrypted at least v2.0 to extract, compressed size: 69, uncompressed size: 54, name: xy@ct$f
224460        0x36CCC         End of Zip archive, footer length: 22
```

还是需要密码，盲猜xyctf（也可以爆破，就是时间长一点），对了。

打开就送flag

EZ_Base1024*2

在github上面找到了可以用于decode的代码https://github.com/qntm/base2048

exp:

```
import { encode, decode } from 'base2048'
function uint8ArrayToString(uint8Array) {
    let result = '';
```

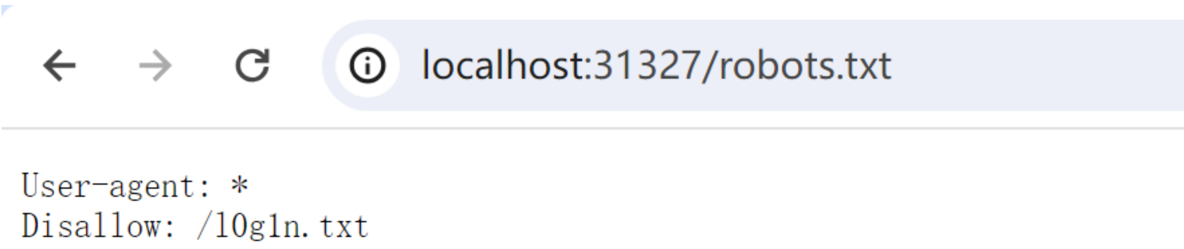
```
    for (let i = 0; i < uint8Array.length; i++) {
        result += String.fromCharCode(uint8Array[i]);
    }
    return result;
}
function uint8ArrayListToStringArray(uint8ArrayList) {
    let resultArray = [];
    for (let i = 0; i < uint8ArrayList.length; i++) {
        resultArray.push(uint8ArrayToString(uint8ArrayList[i]));
    }
    return resultArray;
}
const str='nထ़ٍيጠⱡअϒپᎧᏥپ੍hᏗ०ٺЭШڙnƫ∀ॡʅ؋□ಜ□ϵष≥5'
const uint8Array2 = decode(str)
const stringArray = uint8ArrayListToStringArray([uint8Array2]);
console.log(stringArray);
```

```
> (1) ['XYCTF{84ca3a6e-3508-4e34-a5e0-7d0f03084181}']
```

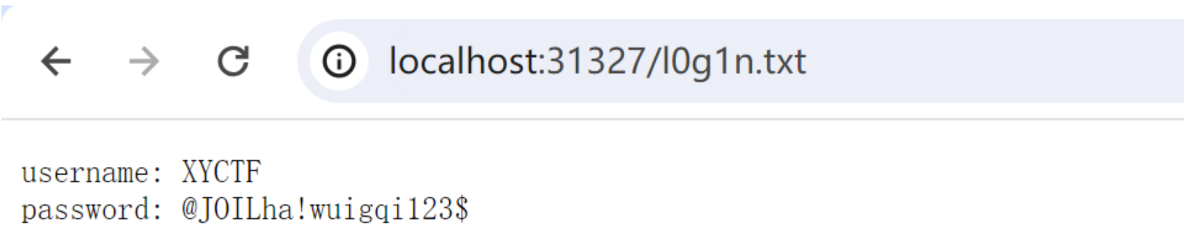
Web

ezhttp

打开是个登陆界面，先尝试一些常用的后台地址，在robots.txt中发现

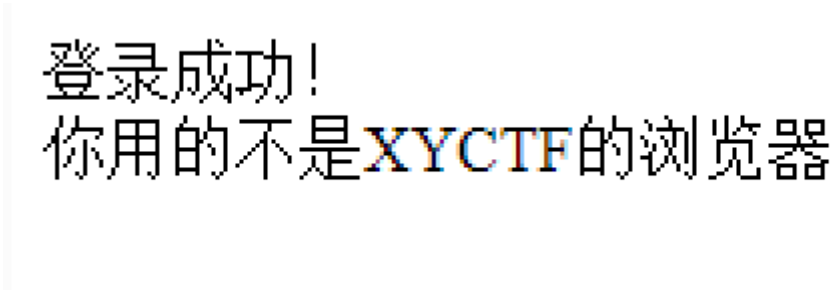


再打开/l0g1n.txt，发现了用户名和密码

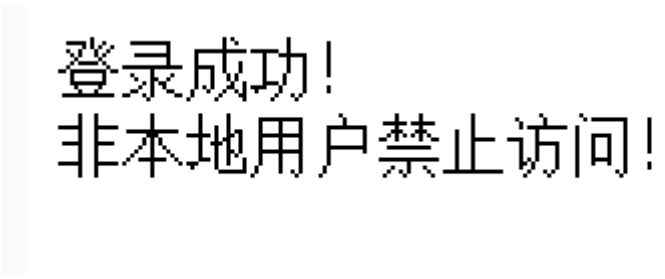


返回到最初的登录页面，输入账号密码

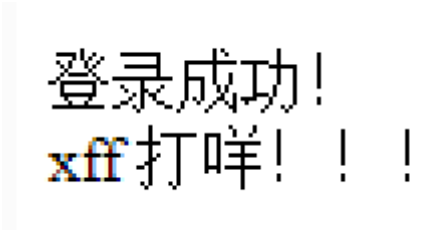
要从yuanshen.com来，把请求头Refer修改一下即可



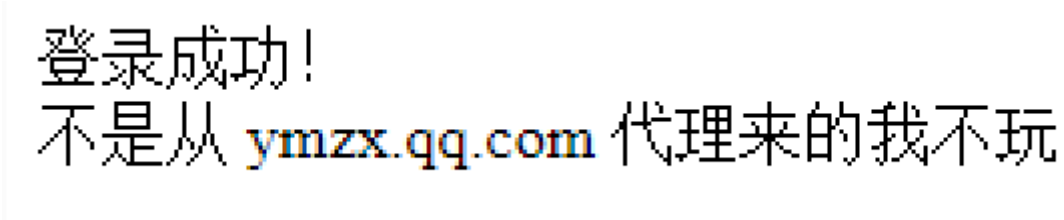
用的不是XYCTF浏览器，把User-Agent修改为XYCTF



非本地用户禁止访问，xff伪造一下



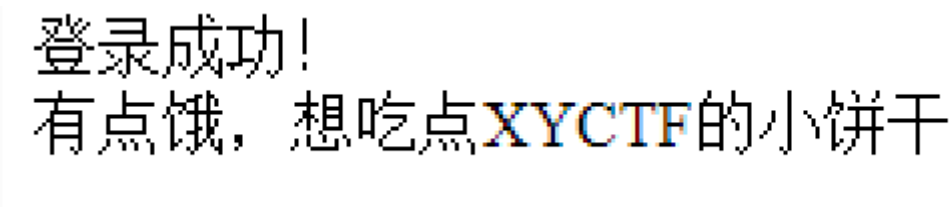
被识破了，得用其他伪造ip的办法，尝试后发现Client-IP可以



要从ymzx.qq.com代理，查询资料得知要通过Via代理

Via	通知中间网关或代理服务器地址，通讯协议	Via: 1.0 fred, 1.1 nowhere.com (Apache/1.1)

https://blog.csdn.net/qq_63701832/article/details/128684100?ops_request_misc=%257B%2522request%255Fid%2522%253A%252217137968231680019703509 9%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=171379682316800197035099&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_click~default-1-128684100-null-null.142（附上常见的http知识点)



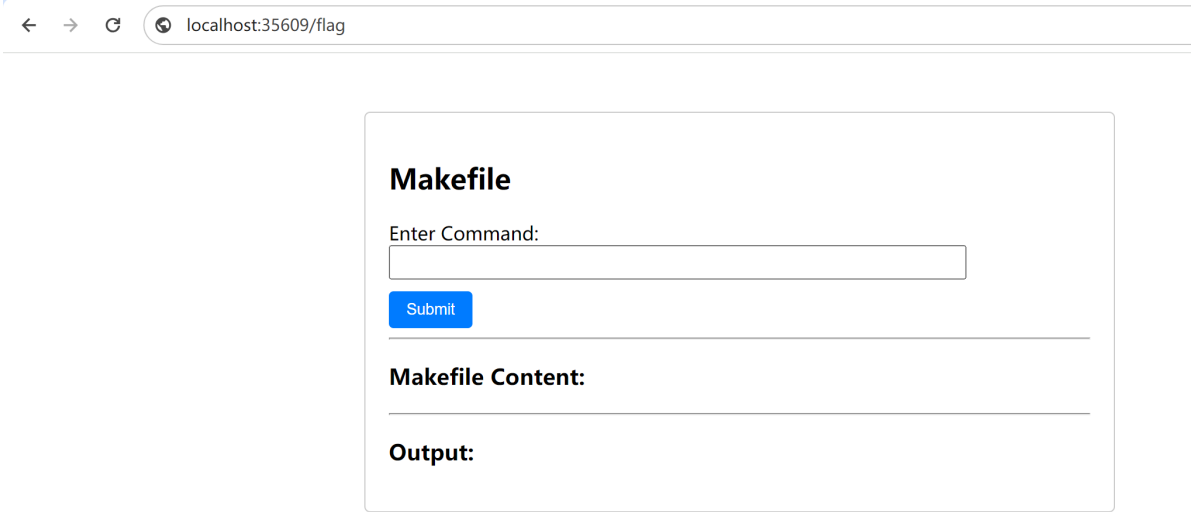
修改cookie值为XYCTF，得到flag

登录成功!
恭喜你拿到flag!
XYCTF{7053646a-1f85-421b-a18a-64402af28054}

最终的payload：

```
POST /index.php HTTP/1.1
Host: localhost:31327
User-Agent: XYCTF
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 42
Origin: http://localhost:31327
Connection: close
Referer: yuanshen.com
Cookie: Pycharm-9613d895=c5b5b56a-63b7-4035-9a3d-9ef3444e34fa
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Client-IP: 127.0.0.1
Via: ymzx.qq.com

username=XYCTF&password=@JOILha!wuigqi123$
```

ezMake



直接输入/flag即可……就挺突然的。

XYCTF{a4312d6b-a1e7-4b3b-8eb1-48e5af5bde73}

ezmd5

题目意思就是传入两张能被识别为图片的文件，两张图片要不相同，但是他们的md5值要相同，尝试随便用一个文件再经过fastcoll生成两个md5相同的文件，但提交后回显只能是jpg文件，看来程序会识别jpg的文件头尾，于是我们去查询一下jpg文件的文件头

JPEG (jpg)，　　文件头：FFD8FF　　文件尾：FF D9

https://blog.csdn.net/m0_73766321/article/details/131725808?ops_request_misc=%257B%2522request%255Fid%2522%253A%252217137978441680022655434 4%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=171379784416800226554344&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-1-131725808-null-null.142 (总结一下常见的文件头尾)

随便生成一个文件并将他的文件头尾人工修改一下，再将文件拖动到fastcoll_v1.0.0.5.exe的地方，即可自动生成两个md5相同的文件。

{"areEqual":true,"md5Equal":true,"md5_1":"7d79e151e2bd67c2fff6087a6b38f691","md5_2":"7d79e151e2bd67c2fff6087a6b38f691"}XYCTF{d8255d97-7894-4f75-a9de-17e64b7f6653}

warm up

第一关源码如下：

```php
<?php
include 'next.php';
highlight_file(__FILE__);
$XYCTF = "Warm up";
extract($_GET);

if (isset($_GET['val1']) && isset($_GET['val2']) && $_GET['val1']  != $_GET['val2'] && md5($_GET['val1']) == md5($_GET['val2'])) {
echo "ez" . "<br>";
} else {
die("什么情况,这么基础的md5做不来");
}

if (isset($md5) && $md5 == md5($md5)) {
echo "ezez" . "<br>";
} else {
die("什么情况,这么基础的md5做不来");
}

if ($XY == $XYCTF) {
    if ($XY != "XYCTF_550102591" && md5($XY) ==  md5("XYCTF_550102591")) {
        echo $level2;
} else {
    die("什么情况,这么基础的md5做不来");
    }
} else {
    die("学这么久,传参不会传?");
}
```

第一关payload：

?val1=QNKCDZO&val2=240610708&md5=0e215962017&XYCTF=QNKCDZO&XY=QNKCDZO

这里用到了变量覆盖的操作，先把XYCTF赋值为一个md5值开头是0e的字符串，然后再使得XY赋值为相同字符串，这样可以满足代码中$XY == $XYCTF的条件，同时也满足了XY不等于XYCTF_550102591并且

可用0e绕过，其md5加密值为0e937920457786991080577371025051，进入第二关LLeeevvveeelll222.php，

第二关源码如下：

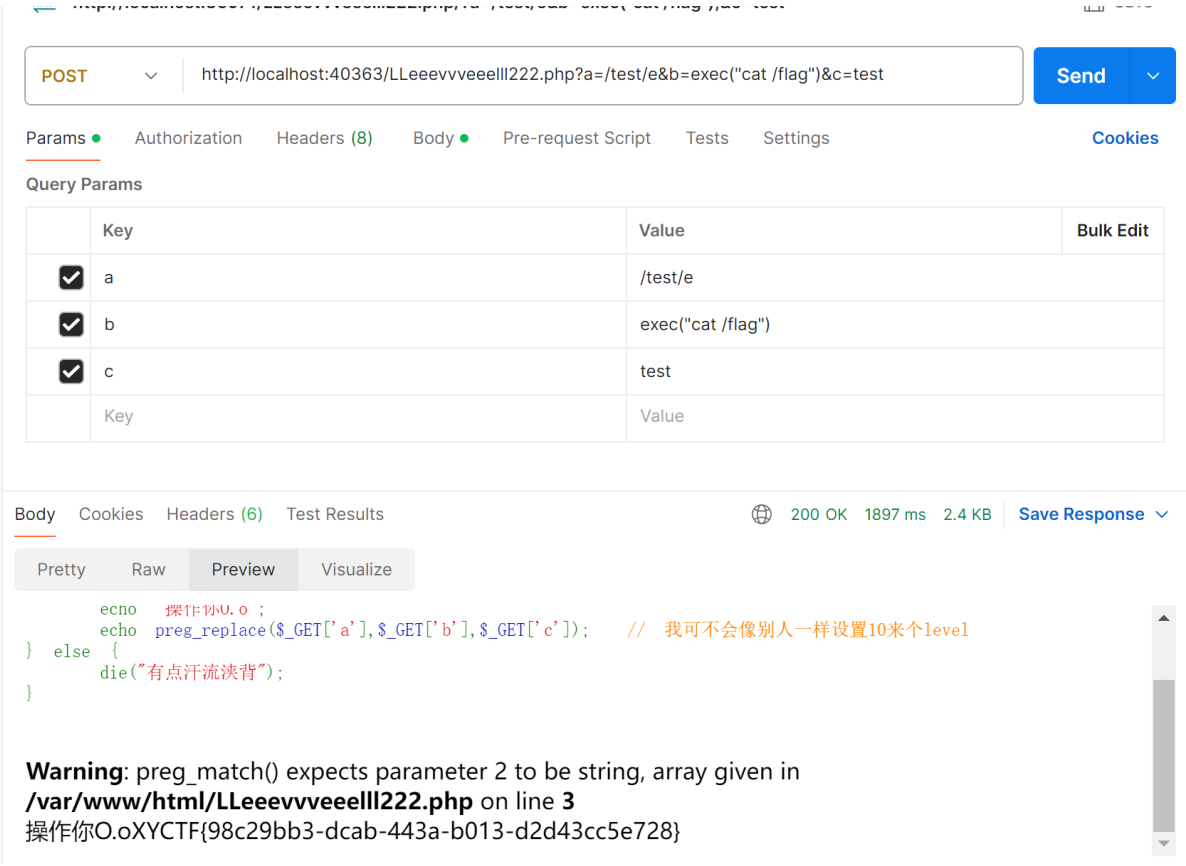
```
<?php
highlight_file(__FILE__);
if (isset($_POST['a']) && !preg_match('/[0-9]/', $_POST['a']) &&
intval($_POST['a'])) {
    echo "操作你O.o";
    echo preg_replace($_GET['a'],$_GET['b'],$_GET['c']);  // 我可不会像别人一样设置
10来个level
} else {
    die("有点汗流浃背");
}
```

先是用一个数组绕过，给a post一个数组即a[]=1

Warning: preg_match() expects parameter 2 to be string, array given in **/var/www/html/LLeeevvveeelll222.php** on line **3**
操作你O.o
Warning: preg_replace(): Empty regular expression in **/var/www/html/LLeeevvveeelll222.php** on line **5**

```
if (isset($_POST['a']) && !preg_match('/[0-9]/', $_POST['a']) &&
intval($_POST['a']))
```

第一层if判断就被我绕过了，再看第二层，是一个preg_replace绕过，当第一个参数结尾是/e时，函数将会执行第二个参数的命令，以达到getshell的效果。



Crypto

factor1

题目

```
import gmpy2
import hashlib
from Crypto.Util.number import *

p = getPrime(512)
q = getPrime(512)
d = getPrime(512)
e = gmpy2.invert(d, (p**3 - 1) * (q**3 - 1))
flag = "XYCTF{" + hashlib.md5(str(p + q).encode()).hexdigest() + "}"
print(e)
print(p * q)
#
172005065945326769176157335849432320425605083524943730546805772515111751580759726759492349719668775270727323745284785341119685198468883978645793770975366048506237371435027612758232099414404389043740306443065413069994232238075194102578269859784981454218948784071599231415554297361219709787507633404217550013282713899284609273532223781487419770338416653260109238572639243087280632577902857385265070736208291583497988891353312351322545840742380550393294960815728021248513046077985900158814037534487146730483099151396746751774427787635287611736111679074330407715700153025952858666841328055071403960165321273972935204988906850585454805923440635864200149694398767776539993952528995717480620593326867245714074205285828967234591508039849777840636255379730281105670496110061909219669860172557450779495125345533232776767292561378244884362014224844319802810586344516400297830227894063759083198761120293919537342405893653545157892446163
#
990751853894430780083272143283287477923851538838365997530969714123773668658262540335342938860348288042190374662461755263470140458118525319945375203030631139854860630224449727612765314225386949150301594209894012800120252491291118716498311850478202364173856932854614200401343138335719490907576358066589581937 93
```

e这么大，多半是用维纳攻击。注意到这里算出e的phi是(p^3 - 1) * (q^3 - 1)，所以对e和n^3进行攻击，得到d。

接下来就是根据n,e,d求一下p,q，爆破一下即可

exp:

```
import gmpy2
import libnum
import hashlib
from Crypto.Util.number import *
import random
def transform(x,y):
    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]:
```

```
        denominator,numerator=numerator,i*numerator+denominator
    return denominator,numerator



def sub_fraction(x,y):
    res=transform(x,y)
    res=list(map(continued_fraction,(res[0:i] for i in range(1,len(res)))))
    return res

def get_pq(a,b,c):
    par=gmpy2.isqrt(b*b-4*a*c)
    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):
        if k==0:
            continue
        if (e*d-1)%k!=0:
            continue

        phi=(e*d-1)//k
        px,qy=get_pq(1,n-phi+1,n)
        if px*qy==n:
            p,q=abs(int(px)),abs(int(qy))
            d=gmpy2.invert(e,(p-1)*(q-1))
            return d
    print("该方法不适用")
def getpq(n,e,d):
    while True:
        k = e * d - 1
        g = random.randint(0, n)
        while k%2==0:
            k=k//2
            temp=gmpy2.powmod(g,k,n)-1
            if gmpy2.gcd(temp,n)>1 and temp!=0:
                return gmpy2.gcd(temp,n)
e=172005065945326769176157335849432320425605083524943730546805772515111751580759
726759492349719668775270727323745284785341119685198468883978645793770975366048 50
623737143502761275823209941440438904374030644306541306999423223807519410257826 98
597849814542189487840715992314155542973612197097875076334042175500132827138992 84
609273532223781487419770338416653260109238572639243087280632577902857385265070 73
620829158349798889135331235132254584074238055039329496081572802124851304607798 59
001588140375344871467304830991513967467517744277876352876117361116790743304077 15
700153025952858666841328055071403960165321273972935204988906850585454805923440 63
586420014969439876777653999395252899571748062059332686724571407420528582896723 45
915080398497778406362553797302811056704961100619092196698601725574507794951253 45
533232776767292561378244884362014224844319802810586344516400297830227894063759 08
319876112029391953734240589365354515789244616 3
n=990751853894430780083272143283287477923851538838365997530969714123773668658262
540335342938860348288042190374662461755263470140458118525319945375203030631139 85
486063022444972761276531422538694915030159420989401280012025249129111871649831 18
504782023641738569328546142004013431383357194909075763580665895819379 3
d=wienerAttack(e,n**3)
p=getpq(n,e,d)
q=n//p
flag = "XYCTF{" + hashlib.md5(str(p + q).encode()).hexdigest() + "}"
```

```
print(flag)
```

```
XYCTF{a83211a70e18145a59671c08ddc67ba4}
```

happy_to_solve1

题目

```
from Crypto.Util.number import *
import sympy
from secrets import flag


def get_happy_prime():
    p = getPrime(512)
    q = sympy.nextprime(p ^ ((1 << 512) - 1))
    return p, q


m = bytes_to_long(flag)
p, q = get_happy_prime()
n = p * q
e = 65537
print(n)
print(pow(m, e, n))
#
24852206647750545040640868093921252282805229864862413863025873203291042799096787
78928846142655571678528828649253019490113004294027910959807195801230317982364515
16377591035587371262714356366577672727039083848025283660908716530241923213987850
17073393201385586868836278447340624427705360349350604325533927890879
#
14767985399473111932544176852718061186100743117407141435994374261886396781040934
63211060821948214046567126995818084988609749165310593936839571659641335256300502
78675465851911032146507908847207296011715176156202021835340219876181468622605586
24458833387692782722514796407503120297235224234298891794056695442287
```

生成p，q的方式太过抽象导致的(

p ^ ((1 << 512) - 1)就相当于2^512-1-p，2^512-1-p与它的下一个素数相差的值与p和q相比很小，可忽略不计，看做p+q=2^512，又因为n=pq，所以代入下面的式子就可以得到p-q

$$n = (\frac{p+q}{2})^2 - (\frac{p-q}{2})^2$$

通过p+q和p-q我们很容易能得到p的近似值，稍稍向后爆破一下即可找到正确的p值

exp:

```
from Crypto.Util.number import *
import gmpy2
n=24852206647750545040640868093921252282805229864862413863025873203291042799096787
77892884614265557167852882864925301949011300429402791095980719580123031798236451
51637759103558737126271435636657767272703908384802528366090871653024192321398785
01707339320138558686883627844734062442770536034935060432553392789087 9
```

```
c=14767985399473111932544176852718061186100743117407141435994374261886396781040934632110608219482140465671269958180849886097491653105939368395716596413352563005027867546585191103214650790884720729601171517615620202183534021987618146862260558624458833387692782722514796407503120297235224234298891794056695442287
e=65537
t1=1<<512
p=(2**512+gmpy2.iroot((2**512)**2-4*n,2)[0])//2
p=int(p)
while n%p!=0:
    p=gmpy2.next_prime(p)
q=n//p
phi=(p-1)*(q-1)
d=gmpy2.invert(e,phi)
m=pow(c,d,n)
print(long_to_bytes(m))
```

```
b'XYCTF{3f22f4efe3bbbc71bbcc999a0a622a1a23303cdc}'
```

Sign1n[签到]

题目

```
from Crypto.Util.number import *
from tqdm import *
import gmpy2
flag=b'XYCTF{uuid}'
flag=bytes_to_long(flag)
leak=bin(int(flag))
while 1:
    leak += "0"
    if len(leak) == 514:
        break

def swap_bits(input_str):
    input_list = list(input_str[2:])
    length = len(input_list)

    for i in range(length // 2):
        temp = input_list[i]
        input_list[i] = input_list[length - 1 - i]
        input_list[length - 1 - i] = temp

    return ''.join(input_list)

input_str = leak
result = swap_bits(input_str)
a=result

def custom_add(input_str):
    input_list = list(input_str)
    length = len(input_list)

    for i in range(length):
        input_list[i] = str((int(input_list[i]) + i + 1) % 10)

    result = ''.join(input_list)
```

```
    return result


input_str = a
result = custom_add(input_str)
b=result
print(b)
#12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789
01234567890123456789012345678901234567890123456789012345678901234567890123456789012345678
01234567891134567799013445689902335567801124457889023356679911344567801223566790
01244578891123566789013445778012235578800123557889123356678902344568890233557880
01334678891223566799023445679912235578800133457889113346779901245577890233557880
01344578890123566780013445689912234577801134557889023346778001245578801233467789
11235577991223457799023355678011 3
```

仔细看swapbits就是一个花里胡哨的操作，用原函数就能逆回去。对于custom_add，其实可以通过操作结果得到原来那一位的数字（0或1）。对于偶数位上的0，其结果必为奇数；对于偶数位上的1，其结果必为偶数；对于奇数位上的0，其结果必为偶数；对于奇数位上的1，其结果必为奇数。

exp:

```
from Crypto.Util.number import *
b='12345678901234567890123456789012345678901234567890123456789012345678901234567
89012345678901234567890123456789012345678901234567890123456789012345678901234567
89012345678911345677990134456899023355678011244578890233566799113445678012235667
90012445788911235667890134457780122355788001235578891233566789023445688902335578
80013346788912235667990234456799122355788001334578891133467799012455778902335578
80013445788901235667800134456899122345778011345578890233467780012455788012334677
89112355779912234577990233556780113'
s=''
for i in range(len(b)):
    if i%2==0:
        if int(b[i])%2==0:
            s+='1'
        else:
            s+='0'
    else:
        if int(b[i])%2==0:
            s+='0'
        else:
            s+='1'
def swap_bits(input_str):
    input_list = list(input_str[2:])
    length = len(input_list)

    for i in range(length // 2):
        temp = input_list[i]
        input_list[i] = input_list[length - 1 - i]
        input_list[length - 1 - i] = temp

    return ''.join(input_list)
s=swap_bits(s)
print(long_to_bytes(int(s,2)>>167))
```

```
b'XYCTF{bf7d6203-aae1-4939-b874-901238562af1}'
```

babyRSAMAX

题目

```
from Crypto.Util.number import *
from gmpy2 import *
from random import choice

flag = b'XYCTF{******}'
e = '?'
def getBabyPrime(nbits):
    while True:
        p = 1
        while p.bit_length() <= nbits:
            p *= choice(sieve_base)

        if isPrime(p+1):
            return p+1

p = getBabyPrime(512)
q = getBabyPrime(512)
n = p*q
gift1 = (pow(p,e,n)-pow(q,e,n)) % n
gift2 = pow(p+q,e,n)

t = 65537
x = bytes_to_long(e)
y = pow(x, t, n)

m = bytes_to_long(flag)
c = powmod(m, e, n)

print(f'n = {n}')
print(f'gift1 = {gift1}')
print(f'gift2 = {gift2}')
print(f'c = {c}')
print(f'y = {y}')

'''
n =
393324238727402107832460690308559462441049823811571668439775997802339111831585609013773599254350923266533039642615501586585515186260140487834352454715369598448740365169315424447195499979714826449055234594077753927022110861492794737847962020202819097067233804725718627920036874237915765300857477167064752205323 21
gift1 =
454940244474633832734900723581818779395028510509172616757355241267841675969466016695678275563144727166210856408438209856299995022870830090220157158341911629993226447838119703440233848187293757617219720251977078245834360606054469460885284422840045723210090421706291404734266353413866849032840002265181659736731 0
gift2 =
111061215998959709920736448050860427855012026815376672067601244053580566359594802604251992986382187891022583247997994146019970445247509119719411310760491983876636264003942870756402328634092146799825005835867245563420135253048223898334460067523975023732153230791136870324302259127159852763634051238811969161011462
```

```
c =
16938927825234407267026017561045490265698491840814929432152839745035946118743714
56662331503380268100901769552637439737034398436099790316584259141420319718494658
84703557289849125220407446919748196301181639762592469415790636878579941933095541
29816268931672391946592680578681270693589911021465752454315629283033043
y =
18136500012709677098413064912977169089694252488885109851093818812703627550313855
64927869313112540534780853966341044526856705589020295048473305762088786992446350
06002488111774104126039140596281718267442171523919721127466845094766639459412176
43337941383084421241147448921641558942563269616051374792860829645202 17

'''
```

gift2进行二项式定理展开其实就等价于(pow(p,e,n)+pow(q,e,n)) % n，gift2-gift1与n求最大公因数即可得到p，正常RSA解密得到e的值为4096。

这导致了e，phi肯定不互素，因此用AMM算法求解。

根据flag来看是个rabin算法，没听过，之后研究一下，，

```
from gmpy2 import gcd, invert, iroot
from Crypto.Util.number import long_to_bytes

n =
39332423872740210783246069030855946244104982381157166843977599780233911183158560
90137735992543509232665330396426155015865855151862601404878343524547153695984487
40365169315424447195499979714826449055234594077753927022110861492794737847962020
20281909706723380472571862792003687423791576530085747716706475220532321
gift1 =
45494024447463383273490072358181877939502851050917261675735524126784167596946601
66956782755631447271662108564084382098562999950228708300902201571583419116299932
26447838119703440233848187293757617219720251977078245834360606054469460885284422
84004572321009042170629140473426635341386684903284000226518165973673 10
gift2 =
11106121599895970992073644805086042785501202681537667206760124405358056635959480
26042519929863821878910225832479979941460199704452475091197194113107604919838766
36264003942870756402328634092146799825005835867245563420135253048223898334460067
52397502373215323079113687032430225912715985276363405123881196916101146 2
c =
16938927825234407267026017561045490265698491840814929432152839745035946118743714
56662331503380268100901769552637439737034398436099790316584259141420319718494658
84703557289849125220407446919748196301181639762592469415790636878579941933095541
29816268931672391946592680578681270693589911021465752454315629283033043
y =
18136500012709677098413064912977169089694252488885109851093818812703627550313855
64927869313112540534780853966341044526856705589020295048473305762088786992446350
06002488111774104126039140596281718267442171523919721127466845094766639459412176
43337941383084421241147448921641558942563269616051374792860829645202 17
p = gcd(gift2 - gift1, n)
q = n // p
phi = (p - 1) * (q - 1)
d = invert(65537, phi)
x = pow(y, d, n)  # XYCTF{e==4096}
e = 4096
R.<a> = Zmod(p)[]
f = a ^ e - c
f = f.monic()
```

```
res1 = f.roots()
R.<b> = Zmod(q)[]
f = b ^ e - c
f = f.monic()
res2 = f.roots()
for i in res1:
    for j in res2:
        m =crt([int(i[0]),int(j[0])],[p,q])
        flag = long_to_bytes(m)
        if flag.startswith(b'XYCTF'):
            print(flag)
```

b'XYCTF{Rabin_is_so_biggggg!}'

Sign1n_Revenge

题目

```
from Crypto.Util.number import *
from tqdm import *
import gmpy2
flag=b'XYCTF{uuid}'
flag=bytes_to_long(flag)
leak=bin(int(flag))
while 1:
    leak += "0"
    if len(leak) == 514:
        break

def swap_bits(input_str):
    input_list = list(input_str[2:])
    length = len(input_list)

    for i in range(length // 2):
        temp = input_list[i]
        input_list[i] = input_list[length - 1 - i]
        input_list[length - 1 - i] = temp

    return ''.join(input_list)

input_str = leak
result = swap_bits(input_str)
a=result

def custom_add(input_str):
    input_list = list(input_str)
    length = len(input_list)

    for i in range(length):
        input_list[i] = str((int(input_list[i]) + i + 1) % 10)

    result = ''.join(input_list)
    return result


input_str = a
```

```
result = custom_add(input_str)
b=result
print(b)
#12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789
01234567890123456789012345678901234567890123456789012345678901234567890123456789
01234567890123456799123455788902334577900124556899023346779902344568990233566780
11334678890223467780012455678012235667900133566889113356679911245568991223456890
01335668890133566789023445779912234577900133566889113356678001344578891223456890
01335668890124566799013445778912234677900133456899112456679911245578991223566890
11345568991123467789122455689902 3
```

抽象题目，用前面那个签到题的脚本都能出，，

可能是用来测试你的nc能力的吧，，

```
from Crypto.Util.number import *
b='12345678901234567890123456789012345678901234567890123456789012345678901234567
89012345678901234567890123456789012345678901234567890123456789012345678901234567
89012345678901234567991234557889023345779001245568990233467799023445689902335667
80113346788902234677800124556780122356679001335668891133566799112455689912234568
90013356688901335667890234457799122345779001335668891133566780013445788912234568
90013356688901245667990134457789122346779001334568991124566799112455789912235668
90113455689911234677891224556899023'
s=''
for i in range(len(b)):
    if i%2==0:
        if int(b[i])%2==0:
            s+='1'
        else:
            s+='0'
    else:
        if int(b[i])%2==0:
            s+='0'
        else:
            s+='1'
def swap_bits(input_str):
    input_list = list(input_str[2:])
    length = len(input_list)

    for i in range(length // 2):
        temp = input_list[i]
        input_list[i] = input_list[length - 1 - i]
        input_list[length - 1 - i] = temp

    return ''.join(input_list)
s=swap_bits(s)
print(long_to_bytes(int(s,2)>>167))
```

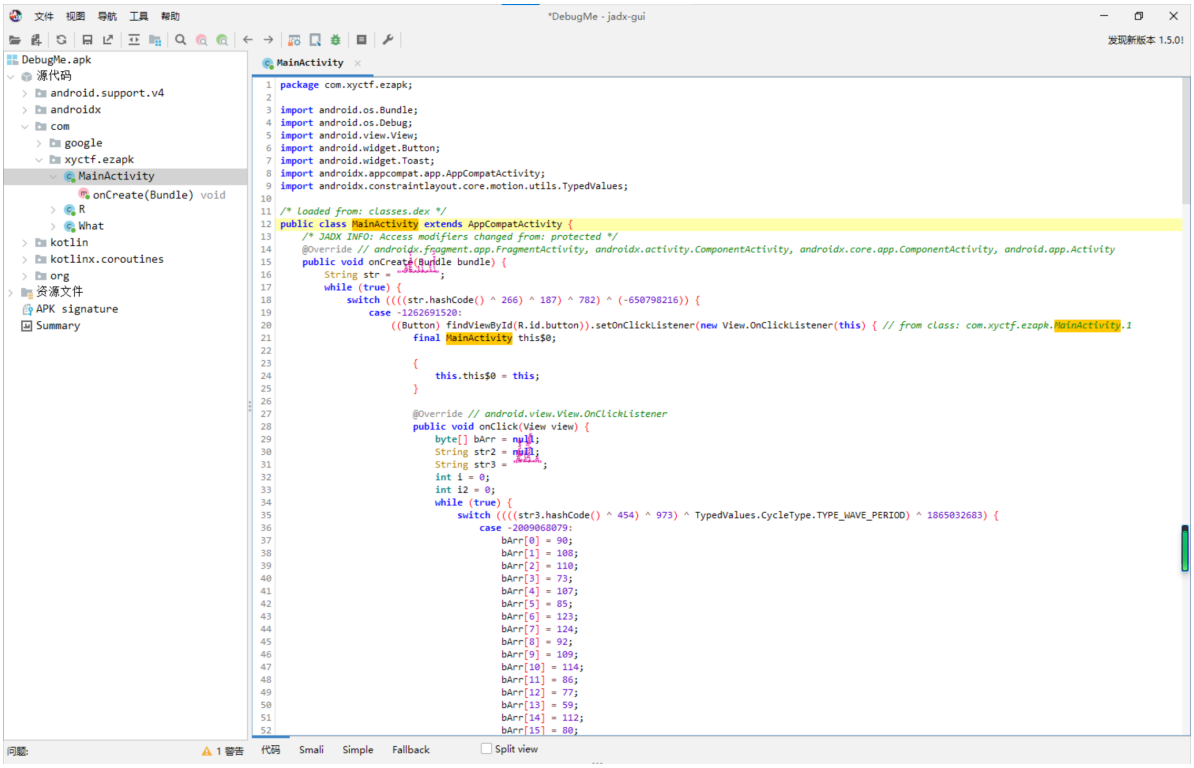
```
b'flag{7e9a6118-8325-4584-86e5-38bb9cf9ff4c}\x00'
```

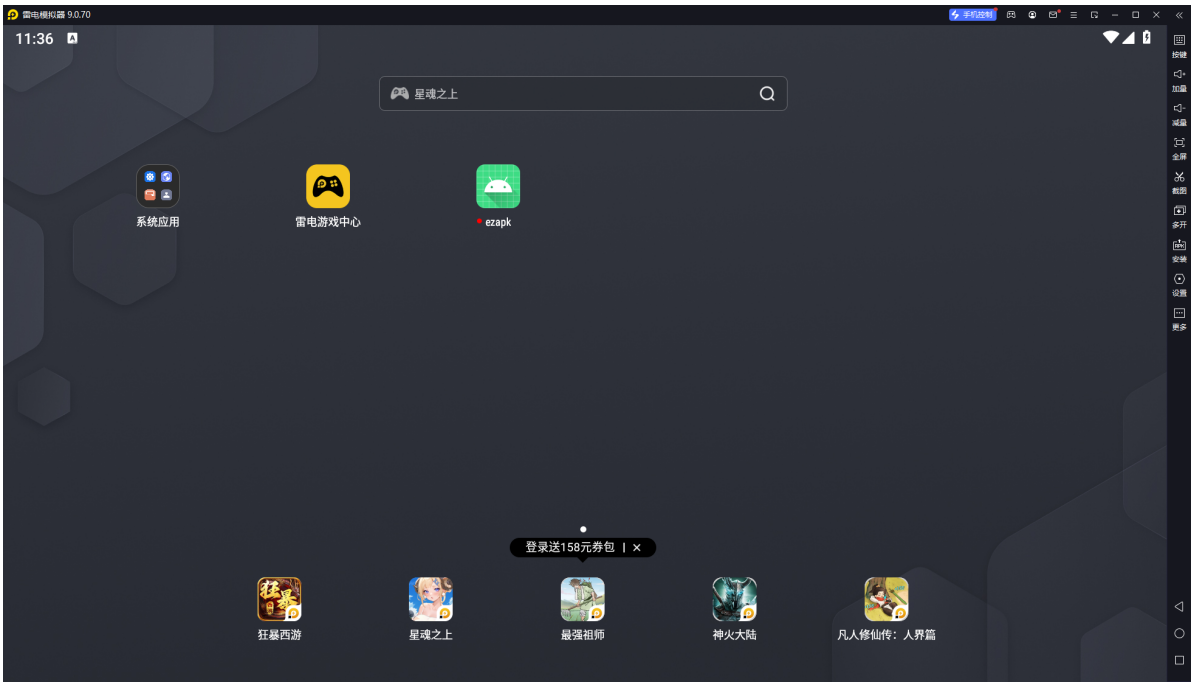
Reverse

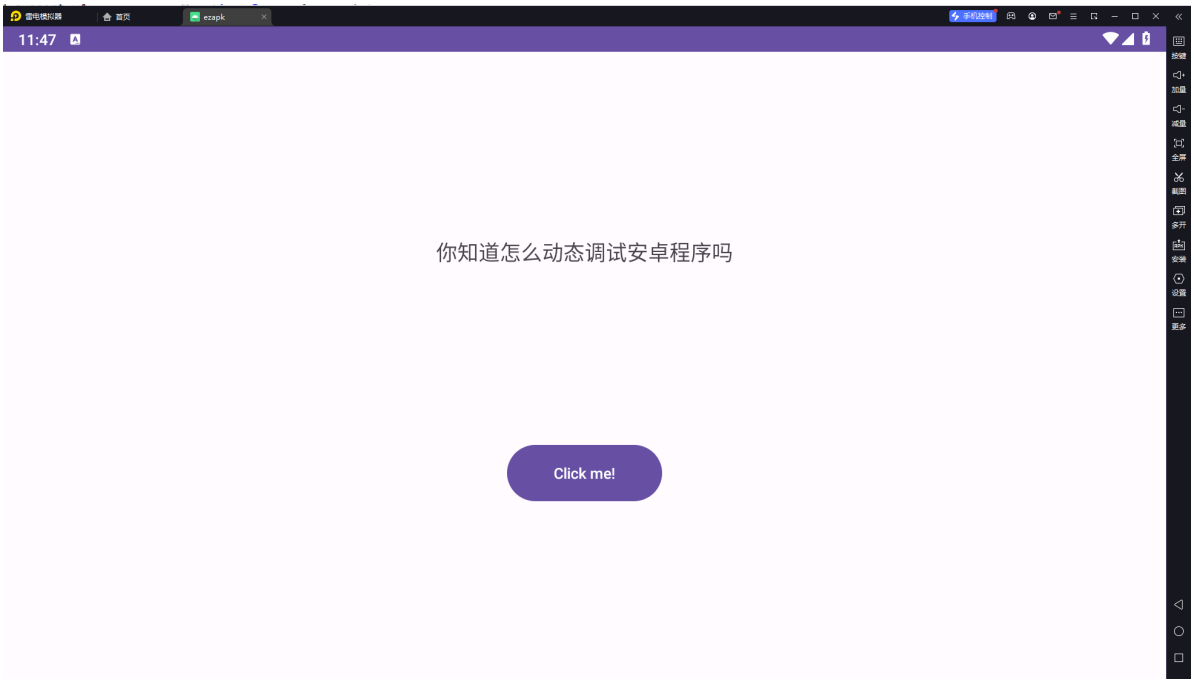
DebugMe

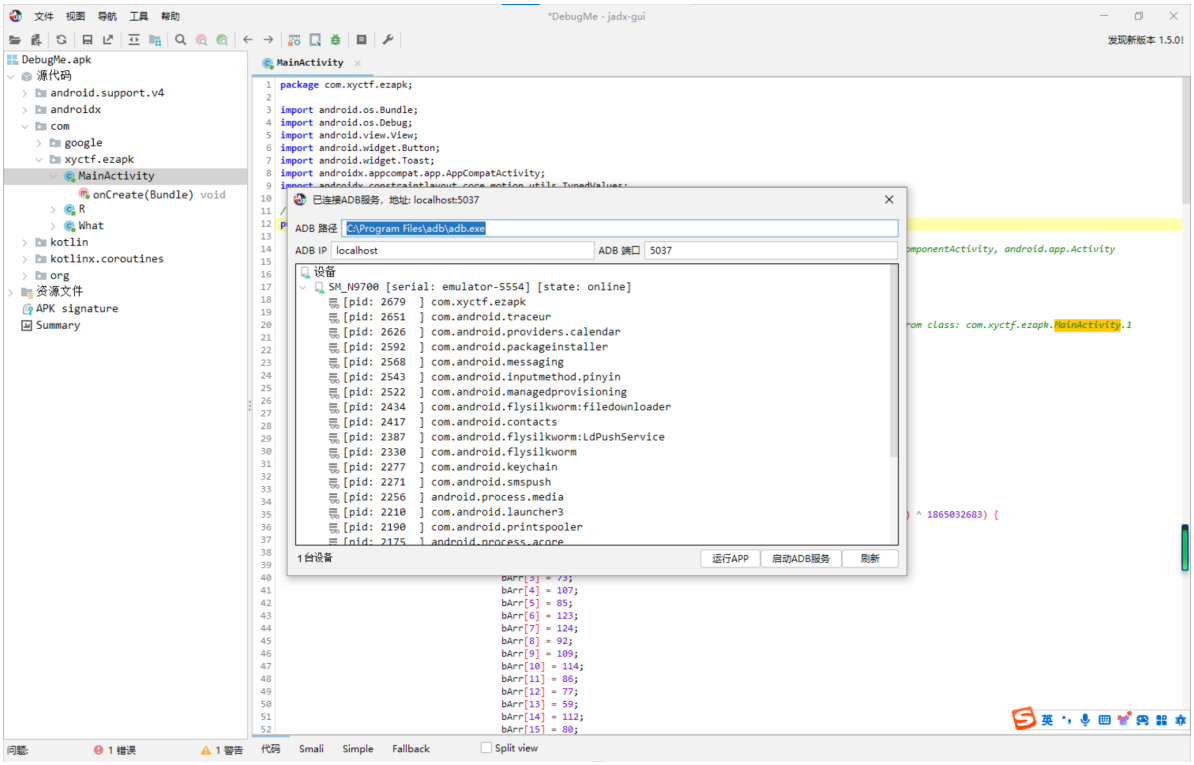
一个apk文件，只要检测到调试器运行就给flag

扔进jadx，再在雷电模拟器上打开，连接到调试器

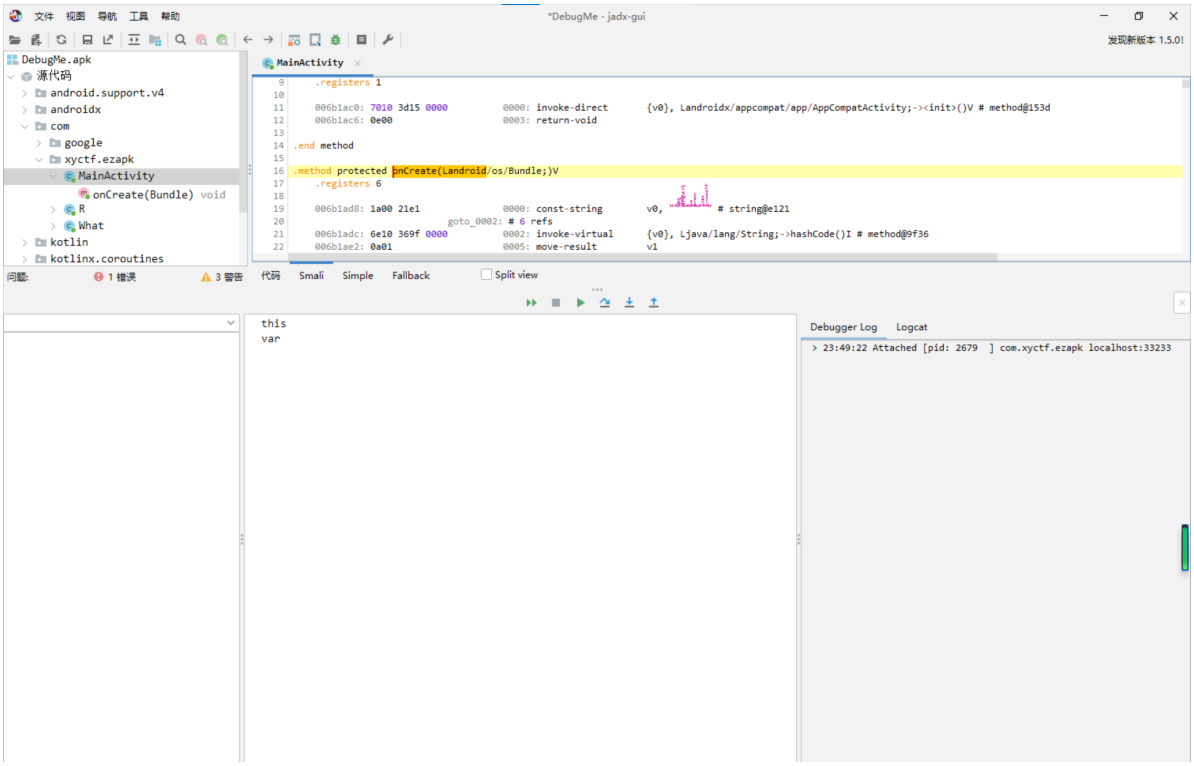




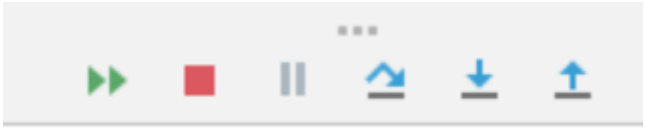

11:47
你知道怎么动态调试安卓程序吗
Click me!

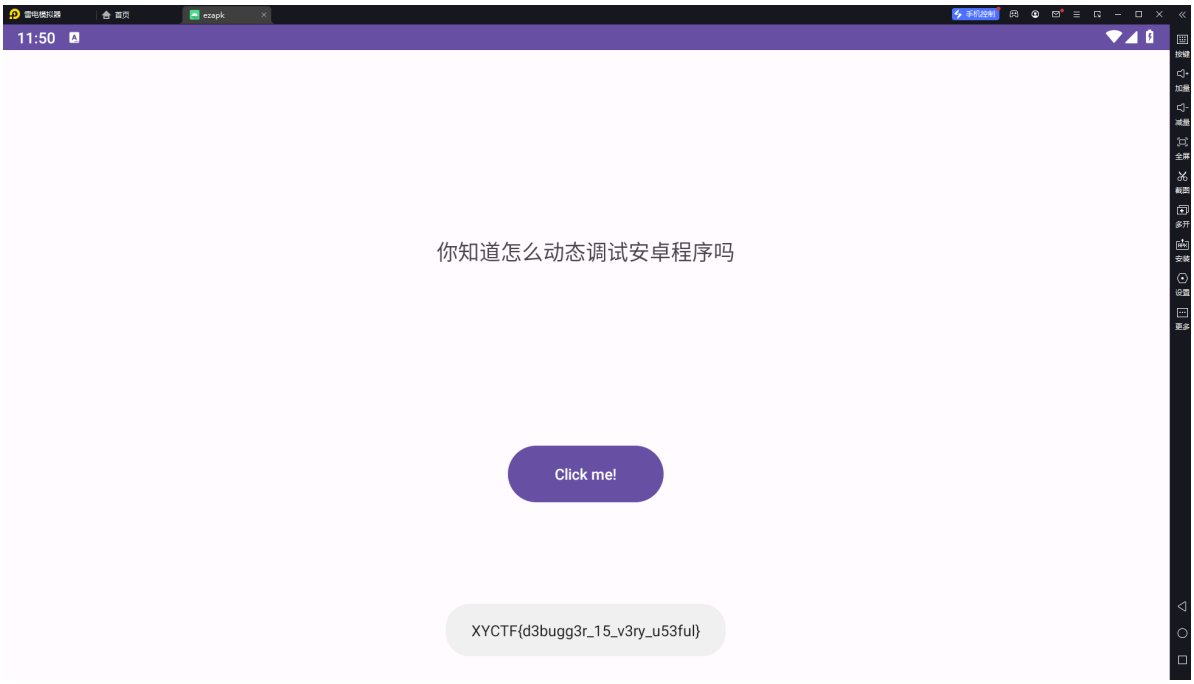

DebugMe - jadx-gui
DebugMe.apk
源代码
android.support.v4
androidx
com
google
xyctf.ezapk
MainActivity
onCreate(Bundle) void
R
What
kotlin
kotlinx.coroutines
org
资源文件
APK signature
Summary
已连接ADB服务，地址: localhost:5037
ADB 路径 C:\Program Files\adb\adb.exe
ADB IP localhost
ADB 端口 5037
设备
SM_N9700 [serial: emulator-5554] [state: online]
[pid: 2679] com.xyctf.ezapk
[pid: 2651] com.android.traceur
[pid: 2626] com.android.providers.calendar
[pid: 2592] com.android.packageinstaller
[pid: 2568] com.android.messaging
[pid: 2543] com.android.inputmethod.pinyin
[pid: 2522] com.android.managedprovisioning
[pid: 2434] com.android.flysilkworm:filedownloader
[pid: 2417] com.android.contacts
[pid: 2387] com.android.flysilkworm:LdPushService
[pid: 2330] com.android.flysilkworm
[pid: 2277] com.android.keychain
[pid: 2271] com.android.smspush
[pid: 2256] android.process.media
[pid: 2210] com.android.launcher3
[pid: 2190] com.android.printspooler
1台设备
运行APP
启动ADB服务
刷新

进入ezapk的进程



点击开始调试的按钮，然后回到模拟器点一下clickme的按钮就可以看到flag





你是真的大学生吗?

不知道预期解是怎么做，反正我是硬读汇编代码读出来的。。。其实就是一个简单的异或

exp:

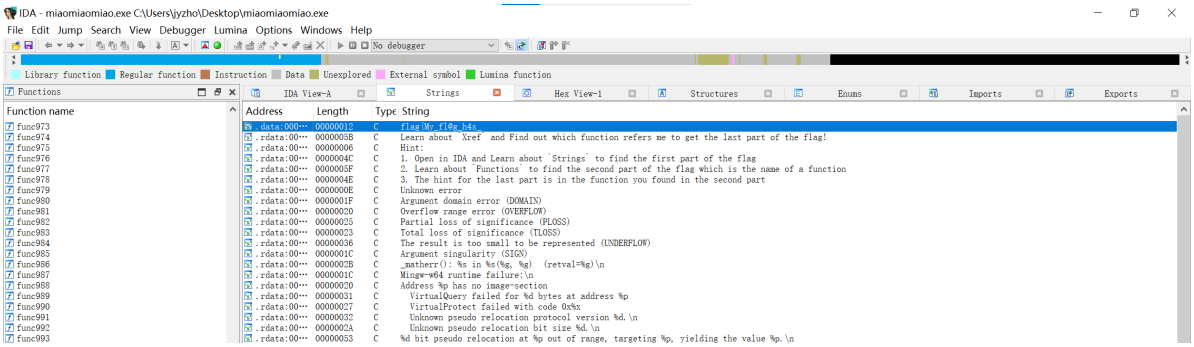
```
def decrypt_string(expected_output):

    encryption_key = ord('/')
```

```
    decrypted_string = ""
    for char in expected_output:
        decrypted_char = chr(ord(char) ^ encryption_key)
        decrypted_string += decrypted_char
        encryption_key = ord(char)
    return decrypted_string[::-1]

def main():

    ls = [
    0x76, 0x0E, 0x77, 0x14, 0x60, 0x06, 0x7D, 0x04,
    0x6B, 0x1E, 0x41, 0x2A, 0x44, 0x2B, 0x5C, 0x03,
    0x3B, 0x0B, 0x33, 0x05,0x15
    ]
    expected_output=""
    for i in ls:
        expected_output+=chr(i)

    decrypted_input = decrypt_string(expected_output)

    print(decrypted_input[::-1])

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

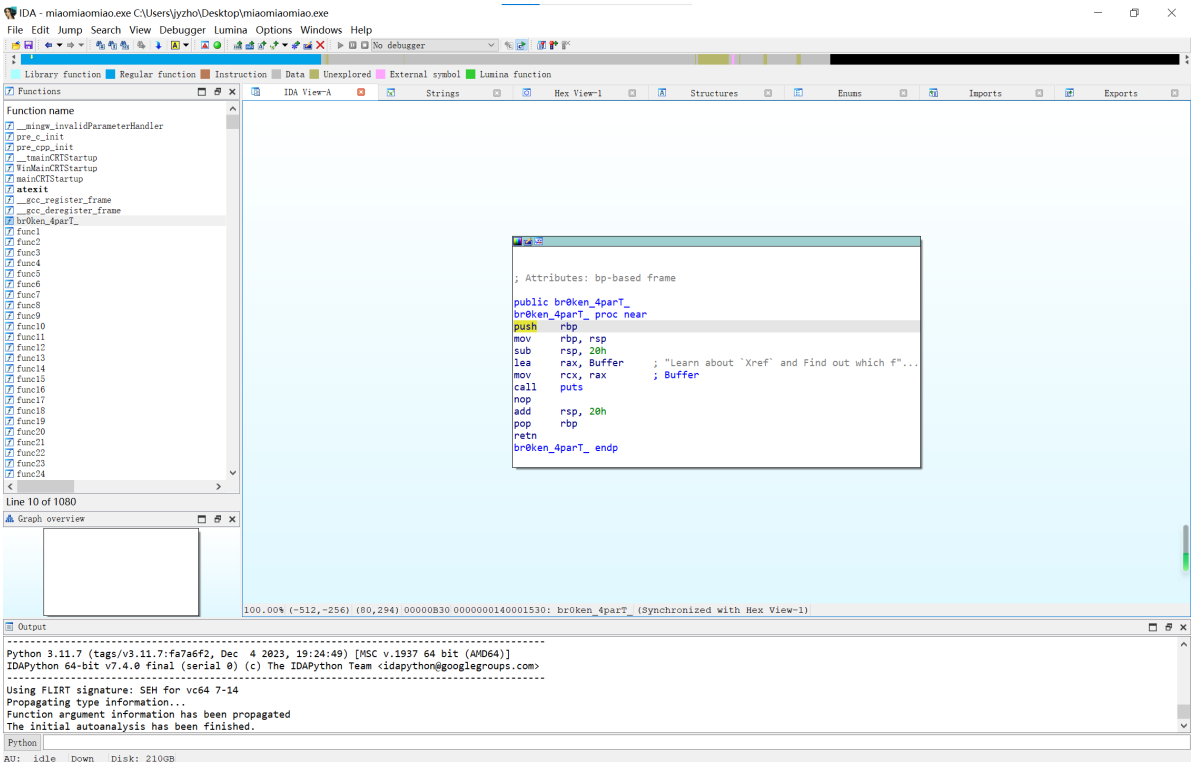
```
Yxyctf{you_know_8086
```

喵喵喵的flag碎了一地

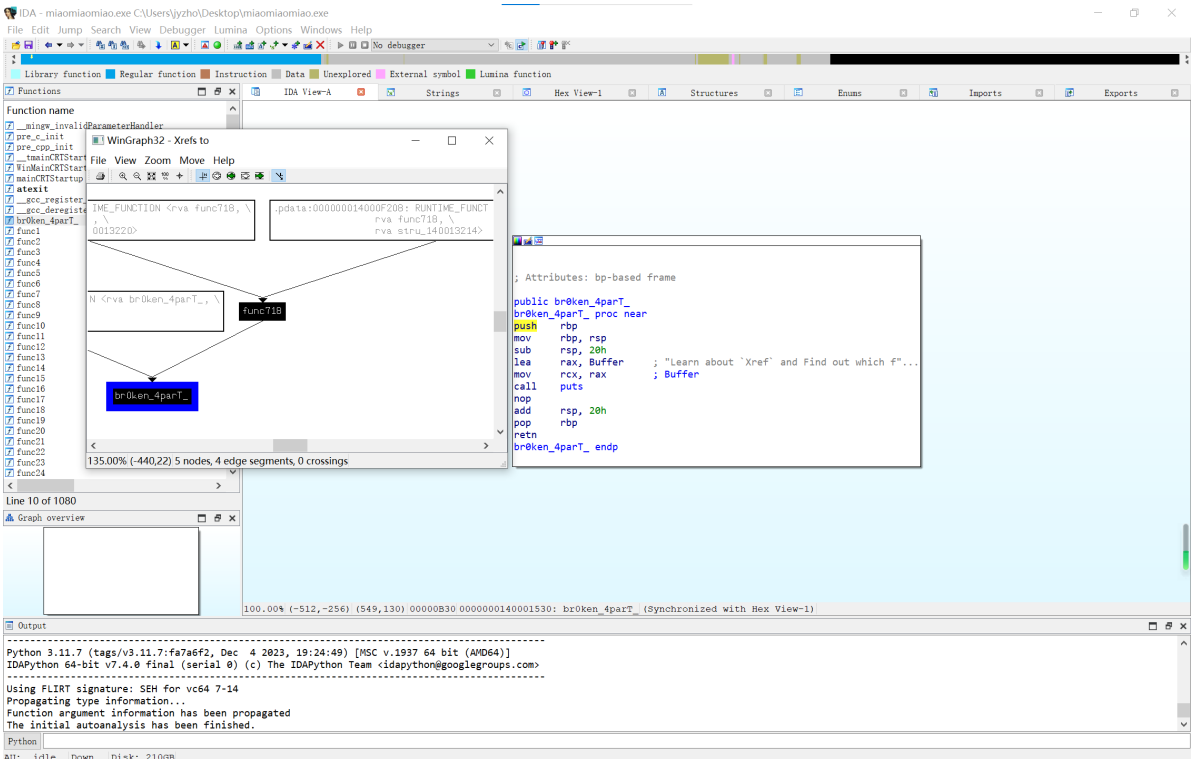
丢进IDA，strings看到第一段flag和hint



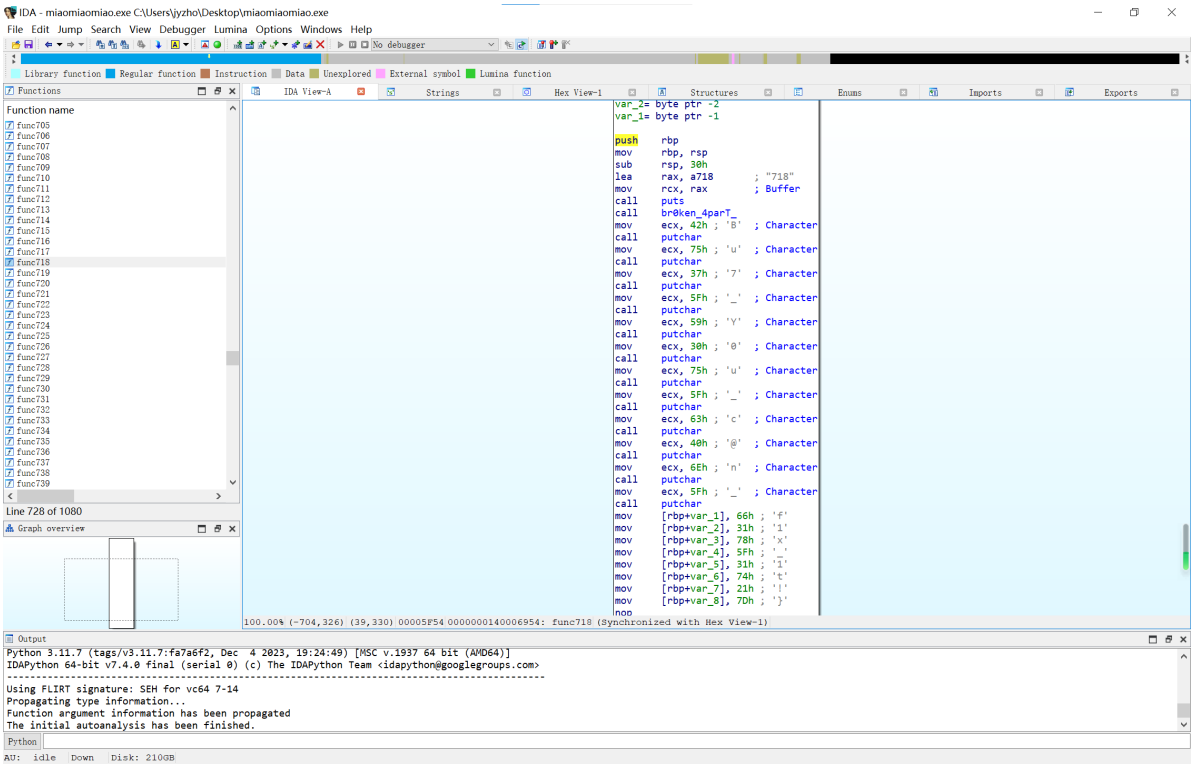
在函数中找到第二段flag和hint



查看xref to，据此查看func718

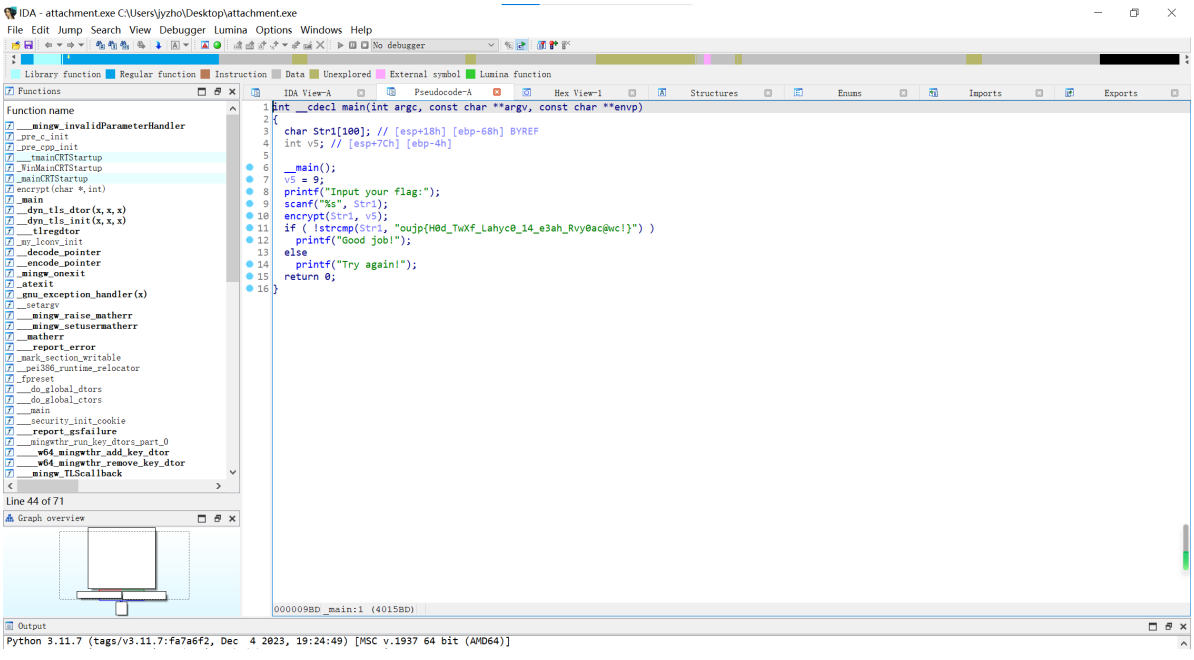


在其中找到最后一段flag

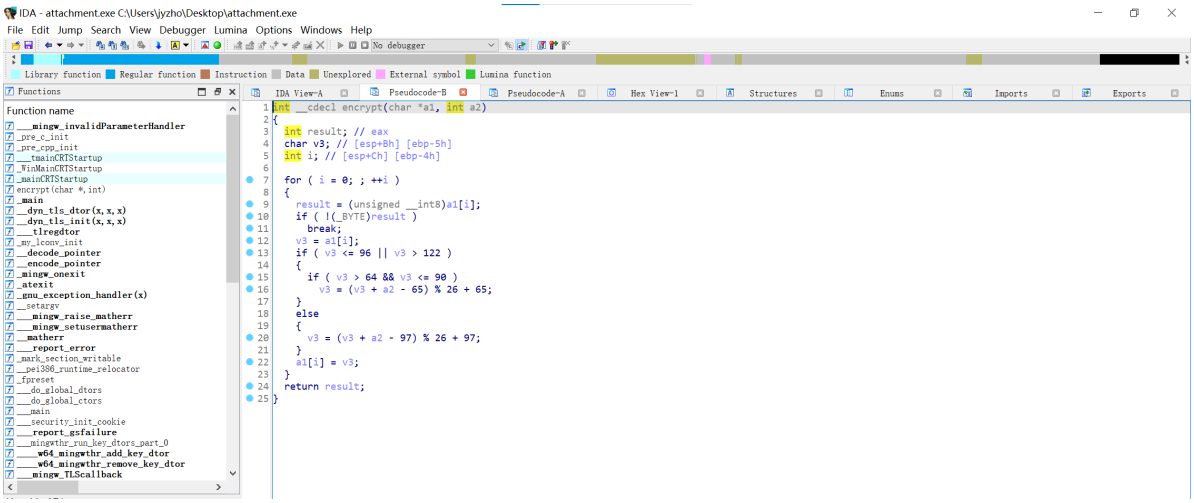


聪明的信使

用IDA打开，生成伪代码



看一下encrypt函数



经典的凯撒加密，密钥为9

转换前：

oujp{H0d_TwXf_Lahyc0_14_e3ah_Rvy0ac@wc!}

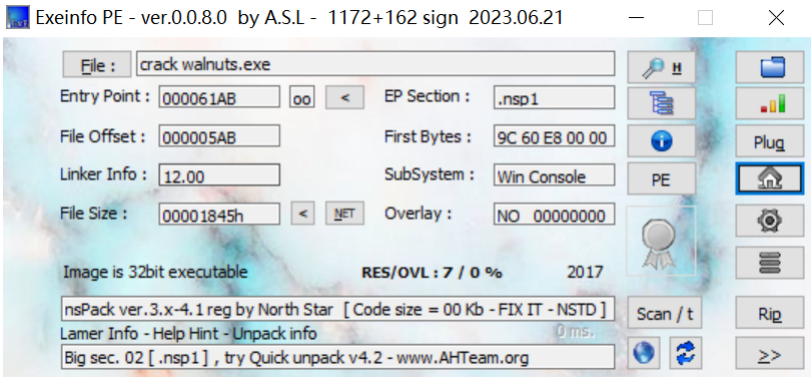
加密位移： 9 加密> 解密>

转换后：

flag{Y0u_KnOw_Crypt0_14_v3ry_Imp0rt@nt!}

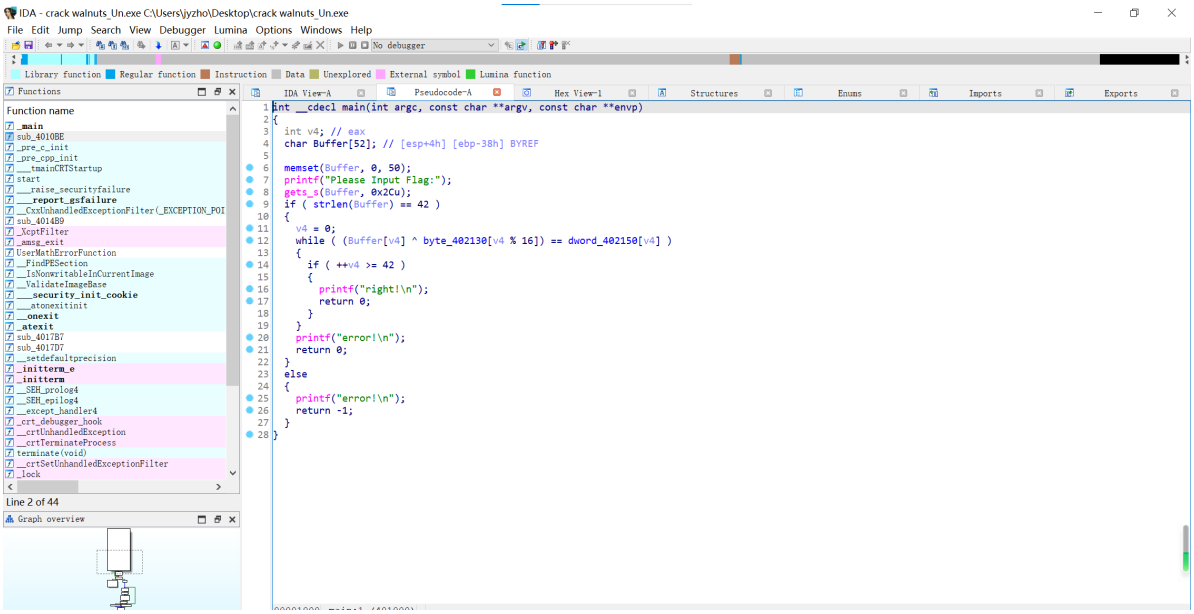
砸核桃

查壳，发现用nspack加了壳



用工具脱壳

放进IDA，生成伪代码看一眼



大概就是某种花里胡哨的异或，看一下结果和用于异或的key

```
.nsp0:00402130 byte_402130     db 74h                  ; DATA XREF: _main:loc_40107F↑r
.nsp0:00402131 aHisIsNotFlag   db 'his_is_not_flag',0
.nsp0:00402141                 align 10h
.nsp0:00402150 ; int dword_402150[]
.nsp0:00402150 dword_402150    dd 12h                  ; DATA XREF: _main+8D↑r
.nsp0:00402154                 dd 4, 8, 14h, 24h, 5Ch, 4Ah, 3Dh, 56h, 0Ah, 10h, 67h, 0
.nsp0:00402184                 dd 41h, 0
.nsp0:0040218C                 dd 1, 46h, 5Ah, 44h, 42h, 6Eh, 0Ch, 44h, 72h, 0Ch, 0Dh
.nsp0:004021B8                 dd 40h, 3Eh, 4Bh, 5Fh, 2, 1, 4Ch, 5Eh, 5Bh, 17h, 6Eh, 0Ch
.nsp0:004021E8                 dd 16h, 68h, 5Bh, 12h, 2 dup(0)
.nsp0:00402200                 dd 48h, 0Eh dup(0)
```

写脚本吧

```
ls = [
    0x12, 0x4, 0x8, 0x14, 0x24, 0x5C, 0x4A, 0x3D, 0x56, 0x0A, 0x10, 0x67,0x0,
    0x41, 0x0, 0x1, 0x46, 0x5A, 0x44, 0x42, 0x6E, 0x0C, 0x44, 0x72, 0x0C,
    0x0D, 0x40, 0x3E, 0x4B, 0x5F, 0x2, 0x1, 0x4C, 0x5E, 0x5B, 0x17, 0x6E,
    0x0C, 0x16, 0x68, 0x5B, 0x12
]
print(len(ls))
key=[0x74, 0x68, 0x69, 0x73, 0x5f, 0x69, 0x73, 0x5f, 0x6e, 0x6f, 0x74, 0x5f,
0x66, 0x6c, 0x61, 0x67]
s=''
for i in range(len(ls)):
    s+=chr(ls[i]^key[i%16])
print(s)
```

flag{59b8ed8f-af22-11e7-bb4a-3cf862d1ee75}