虎符2022RE复现

原创

Pvr1sC 已于 2022-04-10 22:11:08 修改 2896 收藏

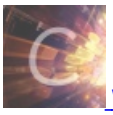
分类专栏： WP 文章标签： 逆向 Re

于 2022-03-26 19:00:29 首次发布

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/m0_51911432/article/details/123746809

版权


WP 专栏收录该内容

7 篇文章 0 订阅

订阅专栏

很痛苦，感觉自己是个废物

鸽了好久。- 。

fpbe

在反编译文件中找到特定的函数属于libbpf.h

然后就一直在看LLVM eBPF编程，经过队友的提示，需要把ebpf提取出，解方程即可。

发现在fpbe_bpf__create_skeleton可以看到初始化skeleton时也初始化了BPF字节码和BPF程序，所以BPF字节码在0x4F4018，长度为1648。

所以先binwalk提取`binwalk -D=elf fpbe`将ebpf字节码提取出，F4018多出好多东西，删一删，然后用llvm-objdump -d F4018反编译，但是不行

```
F4018: file format elf64-bpf
error: unable to get target for 'bpfel--', see --version and --triple.
```

类型不支持，好像是llvm需要加什么东西。。。LLVM 后端实践笔记 9：ELF 文件支持

手搓eBPF？

开玩笑怎么可能，最后在github上找到了eBPF_processor相当于支持IDA反编译ebpf，好牛。

```
uprobe_func:0000000000000008 uprobe:
uprobe_func:0000000000000008                ldxdw         r2, [r1+0x68]
uprobe_func:0000000000000010                lsh           r2, 0x20
uprobe_func:0000000000000018                rsh           r2, 0x20
uprobe_func:0000000000000020                ldxdw         r3, [r1+0x70]
uprobe_func:0000000000000028                lsh           r3, 0x20
uprobe_func:0000000000000030                rsh           r3, 0x20
uprobe_func:0000000000000038                mov           r4, r3
uprobe_func:0000000000000040                mul           r4, 28096
uprobe_func:0000000000000048                mov           r5, r2
uprobe_func:0000000000000050                mul           r5, 64392
uprobe_func:0000000000000058                add           r5, r4
uprobe_func:0000000000000060                ldxdw         r4, [r1+0x60]
uprobe_func:0000000000000068                lsh           r4, 0x20
uprobe_func:0000000000000070                rsh           r4, 0x20
uprobe_func:0000000000000078                mov           r0, r4
uprobe_func:0000000000000080                mul           r0, 29179
uprobe_func:0000000000000088                add           r5, r0
uprobe_func:0000000000000090                ldxdw         r1, [r1+0x58]
uprobe_func:0000000000000098                mov           r0, 0
uprobe_func:00000000000000A0                stxb          [r10-8], r0
```

```
uprobe_func:00000000000000A8    stxdw    [r10-0x10], r0
uprobe_func:00000000000000B0    stxdw    [r10-0x18], r0
uprobe_func:00000000000000B8    lsh      r1, 0x20
uprobe_func:00000000000000C0    rsh      r1, 0x20
uprobe_func:00000000000000C8    mov      r0, r1
uprobe_func:00000000000000D0    mul      r0, 0xCC8E
uprobe_func:00000000000000D8    add      r5, r0
uprobe_func:00000000000000E0    mov      r6, 1
uprobe_func:00000000000000E8    lddw     r0, 0xBE18A1735995
uprobe_func:00000000000000F8    jne      r5, r0, LBB0_5
uprobe_func:0000000000000100    mov      r5, r3
uprobe_func:0000000000000108    mul      r5, 0xF1BF
uprobe_func:0000000000000110    mov      r0, r2
uprobe_func:0000000000000118    mul      r0, 0x6AE5
uprobe_func:0000000000000120    add      r0, r5
uprobe_func:0000000000000128    mov      r5, r4
uprobe_func:0000000000000130    mul      r5, 0xADD3
uprobe_func:0000000000000138    add      r0, r5
uprobe_func:0000000000000140    mov      r5, r1
uprobe_func:0000000000000148    mul      r5, 0x9284
uprobe_func:0000000000000150    add      r0, r5
uprobe_func:0000000000000158    lddw     r5, 0xA556E5540340
uprobe_func:0000000000000168    jne      r0, r5, LBB0_5
uprobe_func:0000000000000170    mov      r5, r3
uprobe_func:0000000000000178    mul      r5, 0xDD85
uprobe_func:0000000000000180    mov      r0, r2
uprobe_func:0000000000000188    mul      r0, 0x8028
uprobe_func:0000000000000190    add      r0, r5
uprobe_func:0000000000000198    mov      r5, r4
uprobe_func:00000000000001A0    mul      r5, 0x652D
uprobe_func:00000000000001A8    add      r0, r5
uprobe_func:00000000000001B0    mov      r5, r1
uprobe_func:00000000000001B8    mul      r5, 0xE712
uprobe_func:00000000000001C0    add      r0, r5
uprobe_func:00000000000001C8    lddw     r5, 0xA6F374484DA3
uprobe_func:00000000000001D8    jne      r0, r5, LBB0_5
uprobe_func:00000000000001E0    mov      r5, r3
uprobe_func:00000000000001E8    mul      r5, 0x822C
uprobe_func:00000000000001F0    mov      r0, r2
uprobe_func:00000000000001F8    mul      r0, 0xCA43
uprobe_func:0000000000000200    add      r0, r5
uprobe_func:0000000000000208    mov      r5, r4
uprobe_func:0000000000000210    mul      r5, 0x7C8E
uprobe_func:0000000000000218    add      r0, r5
uprobe_func:0000000000000220    mov      r5, r1
uprobe_func:0000000000000228    mul      r5, 0xF23A
uprobe_func:0000000000000230    add      r0, r5
uprobe_func:0000000000000238    lddw     r5, 0xB99C485A7277
uprobe_func:0000000000000248    jne      r0, r5, LBB0_5
uprobe_func:0000000000000250    stxw     [r10-0xC], r1
uprobe_func:0000000000000258    stxw     [r10-0x10], r4
uprobe_func:0000000000000260    stxw     [r10-0x14], r2
uprobe_func:0000000000000268    stxw     [r10-0x18], r3
uprobe_func:0000000000000270    lddw     r1, 755886917287302211
uprobe_func:0000000000000280    stxdw    [r10-0x28], r1
uprobe_func:0000000000000288    lddw     r1, 5064333215653776454
uprobe_func:0000000000000298    stxdw    [r10-0x30], r1
uprobe_func:00000000000002A0    lddw     r1, 2329017756590022981
uprobe_func:00000000000002B0    stxdw    [r10-0x38], r1
```

```
uprobe_func:00000000000002B8                lddw            r1, 5642803763628229975
uprobe_func:00000000000002C8                stxdw           [r10-0x40], r1
uprobe_func:00000000000002D0                mov             r6, 0
uprobe_func:00000000000002D8                stxb            [r10-0x20], r6
uprobe_func:00000000000002E0                mov             r1, r10
uprobe_func:00000000000002E8                add             r1, -0x40
uprobe_func:00000000000002F0                mov             r3, r10
uprobe_func:00000000000002F8                add             r3, -0x18
uprobe_func:0000000000000300                mov             r2, 0x21
uprobe_func:0000000000000308                call            6       ; long bpf_trace_printk(const char *fmt, __
u32 fmt_size, ...)
uprobe_func:0000000000000310
uprobe_func:0000000000000310 LBB0_5:                                ; CODE XREF: uprobe+F0↑j
uprobe_func:0000000000000310                                        ; uprobe+160↑j ...
uprobe_func:0000000000000310                mov             r0, r6
uprobe_func:0000000000000318                ret
```

z3脚本

uprobe_func函数r1,r2,r3,r4应当满足方程组

28096*r1+64392*r2+29179*r3+52366*r4 == 209012997183893

61887*r1+27365*r2+44499*r3+37508*r4 == 181792633258816

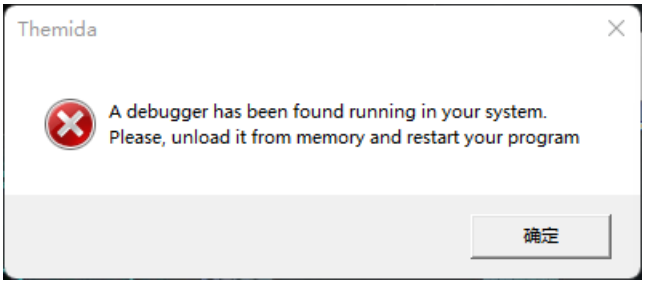
56709*r1+32808*r2+25901*r3+59154*r4 == 183564558159267

33324*r1+51779*r2+31886*r3+62010*r4 == 204080879923831

```python
from z3 import *
from Crypto.Util.number import *
r1 = Int('r1')
r2 = Int('r2')
r3 = Int('r3')
r4 = Int('r4')
s = Solver()
s.add(28096*r1+64392*r2+29179*r3+52366*r4 == 209012997183893)
s.add(61887*r1+27365*r2+44499*r3+37508*r4 == 181792633258816)
s.add(56709*r1+32808*r2+25901*r3+59154*r4 == 183564558159267)
s.add(33324*r1+51779*r2+31886*r3+62010*r4 == 204080879923831)
if s.check() == sat:
 flag = b""
 m = s.model()
 for i in [r1, r2, r3, r4]:
  flag += long_to_bytes(m[i].as_long())[::-1]
 print(flag)
# 0vR3sAlbs8pD2h53
```

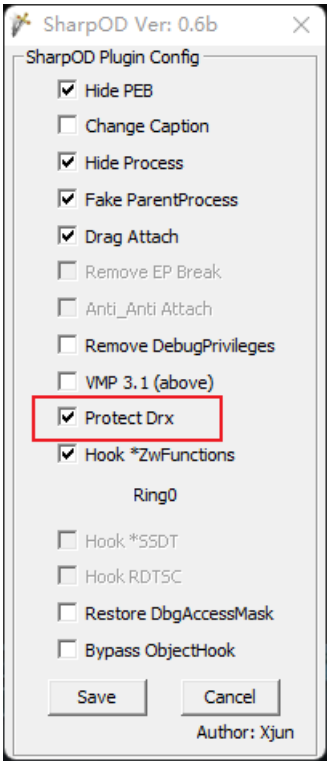
the_shellcode

运行发现warring



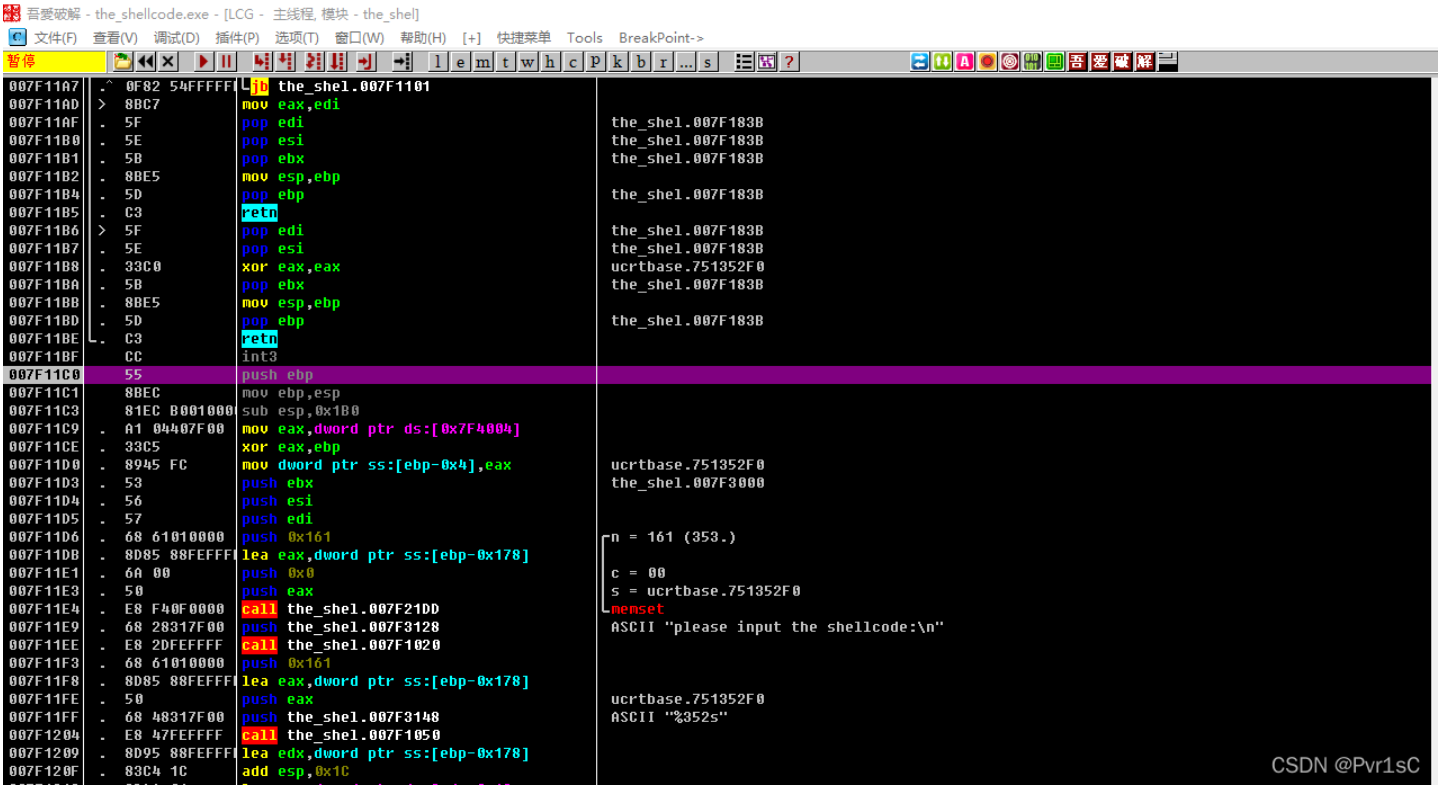
可知为强壳 Themida ， TMD壳

需要插件sharpOD的Protect Drx，但本人使用Drx Protect却不行，不知道为什么

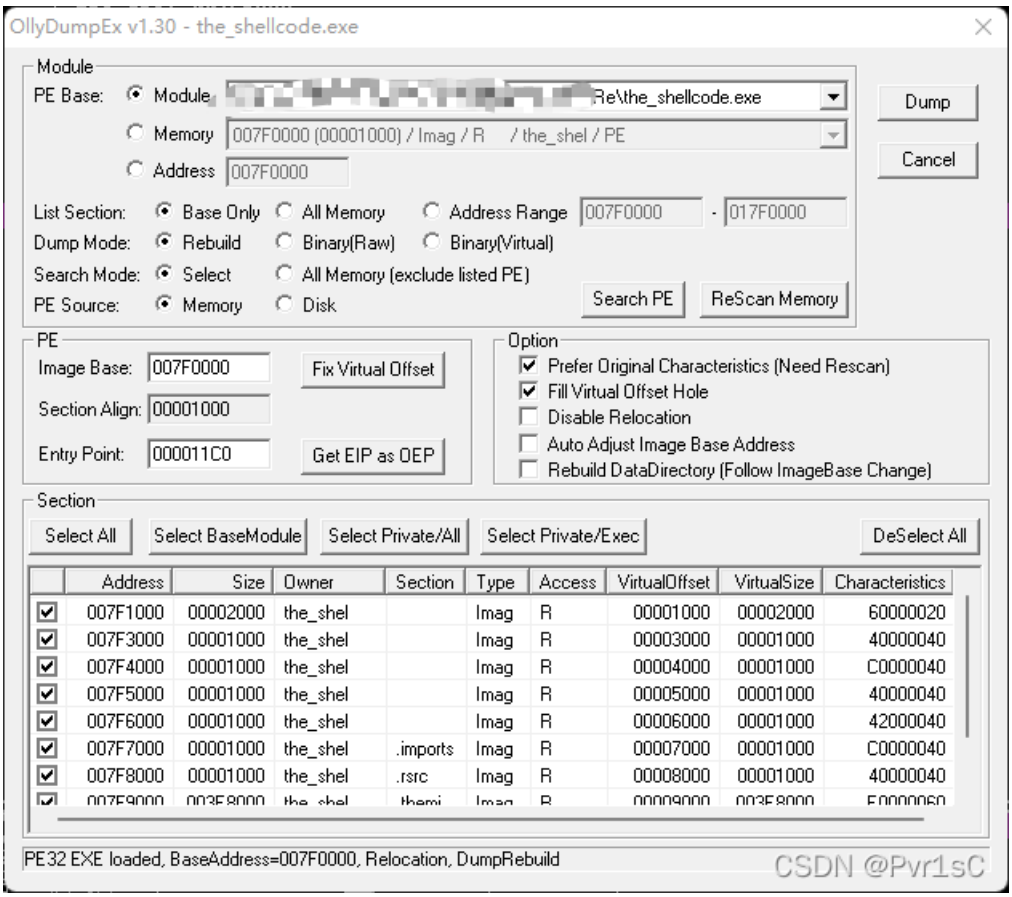


继续动调，F9运行

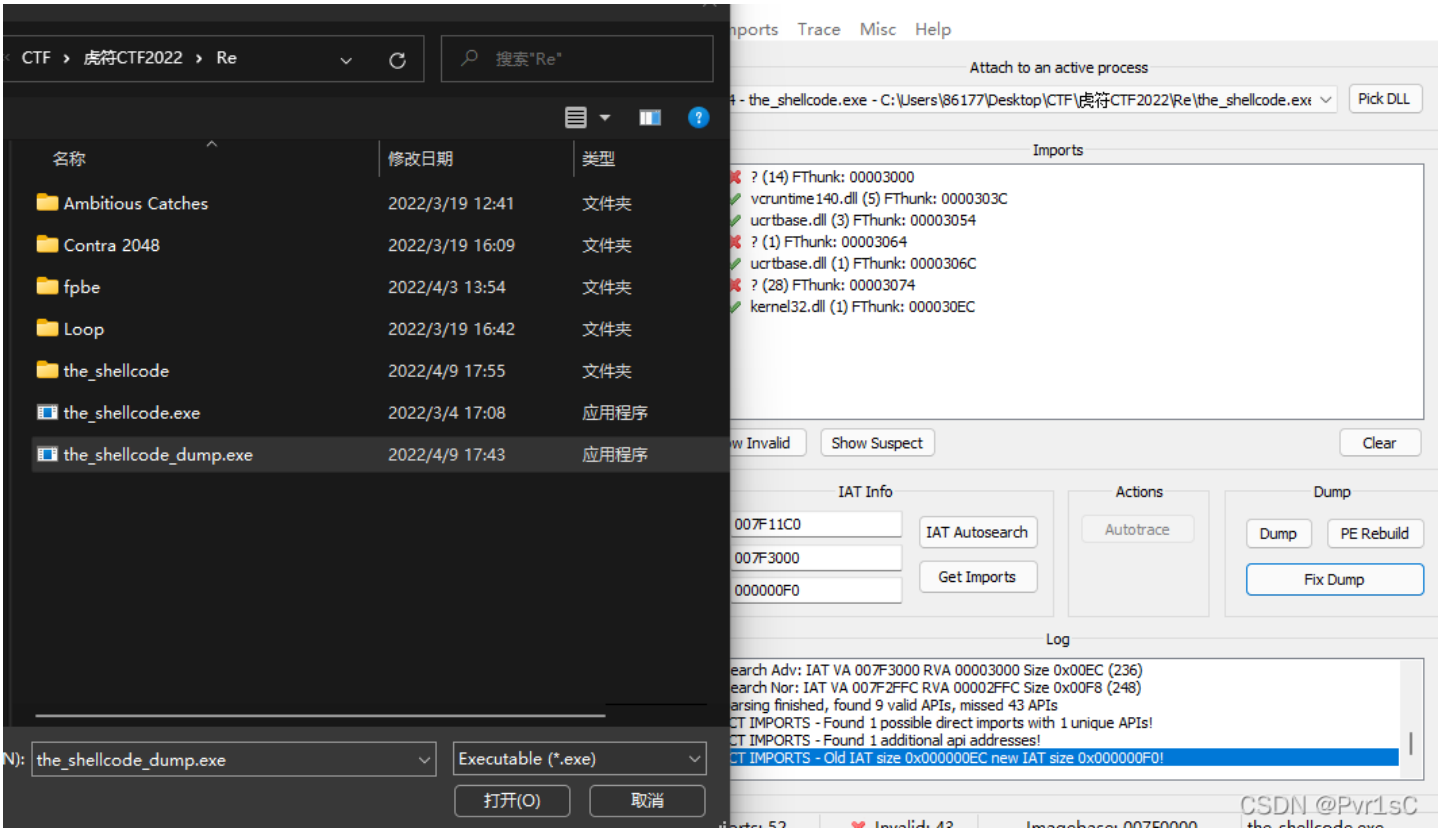
在shellcode模块中寻找，找到oep，下硬件访问断点，



重新运行断到该位置(7F11c0)，dump出来



使用scylla修复IAT表，fix dump导入之前dump的文件



已成功脱壳

使用IDA反编译发现有些系统函数显示不出，存在`__24(aPause);`之类奇怪的函数，可能没有完全修复IAT表，不过也能看

需要输入的shellcode

sub_231090函数为base64加密

```
  memset(v42, 0, 0x161u);
  sub_231020(aPleaseInputThe);                  // please input the shellcode:
  sub_231050("%352s", v42);
  if ( strlen(v42) != 352 )
  {
    sub_231020(aShellcodeLengt);
LABEL_3:
    __24(aPause);
    return 0;
  }
  Size = 352;
  Src = (void *)sub_231090(&Size);
  if ( (Size & 3) != 0 )
    v1 = 4 * (Size >> 2) + 4;
  else
    v1 = Size;
  v2 = (__m128i *)malloc(v1);
  v3 = v2;
  v31 = v2;
  if ( !v2 )
    return 0;
  memset(v2, 0, v1);
  v4 = Size;
  if ( Size )
  {
    if ( Src && v1 >= Size )
    {
      memcpy(v3, Src, Size);
00000630 start:57 (231230)
```

CSDN @Pvr1sC

XXTEA加密，魔改了z>>6

```
108  v32 = v6 - 1;
109  do
110  {
111    v10 = 0;
112    v37 = v9 - 1640531527;
113    v34 = ((unsigned int)(v9 - 1640531527) >> 2) & 3;
114    if ( v8 )
115    {
116      do
117      {
118        v3->m128i_i32[v10] += ((v37 ^ v3->m128i_i32[v10 + 1]) + (v40 ^ v41[v34 ^ v10 & 3])) ^ (((16 * v40) ^ ((unsig
119                                                                                              + ((v7 >> 6) ^ (4 * v3-
120        v7 = v3->m128i_u32[v10++];
121        v40 = v7;
122      }
123      while ( v10 < v32 );
124    }
125    v8 = v32;
126    *v36 += ((v37 ^ v3->m128i_i32[0]) + (v40 ^ v41[v34 ^ v10 & 3])) ^ (((16 * v40) ^ ((unsigned __int32)v3->m128i_i3
127                                                                      + ((v7 >> 6) ^ (4 * v3->m128i_i32[0])));
128    v11 = v33-- == 1;
129    v7 = *v36;
130    v9 -= 1640531527;
131    v40 = *v36;
132  }
```

CSDN @Pvr1sC

v41数组为key

v41[0] = 0x74;

v41[1] = 0x6F;

v41[2] = 0x72;

v41[3] = 0x61;

delta为0x9e3779b9

脚本求shellcode

```
#include <stdio.h>
#include <stdint.h>
#define delta 0x9e3779b9
#define MX (((z >> 6 ^ y << 2) + (y >> 3 ^ z << 4)) ^ ((sum ^ y) + (key[(p & 3) ^ e] ^ z)))
```

```
int ROL(int n)
{
    return (n >> 3) | ((n & 7) << 5);
}
void btea(uint32_t *v, int n, uint32_t const key[])
{
    uint32_t y, z, sum;
    unsigned p, rounds, e;
    if (n > 1) /* Coding Part */
    {
        rounds = 6 + 52 / n;
        sum = 0;
        z = v[n - 1];
        do
        {
            sum += delta;
            e = (sum >> 2) & 3;
            for (p = 0; p < n - 1; p++)
            {
                y = v[p + 1];
                z = v[p] += MX;
            }
            y = v[0];
            z = v[n - 1] += MX;
        } while (--rounds);
    }
    else if (n < -1) /* Decoding Part */
    {
        n = -n;
        rounds = 6 + 52 / n;
        sum = rounds * delta;
        y = v[0];
        do
        {
            e = (sum >> 2) & 3;
            for (p = n - 1; p > 0; p--)
            {
                z = v[p - 1];
                y = v[p] -= MX;
            }
            z = v[n - 1];
            y = v[0] -= MX;
        } while ((sum -= delta) != 0);
    }
}
int main()
{
    uint32_t v[] = {1265338785, 1958827091, 1083351150, 1117457415, 1076371076, 2338014409, 1727968123, 10144742
43, 2042988845, 226155159, 491891286, 47503107, 1336223418, 855299658, 202334353, 1445688723, 3684359527, 198117
5139, 2784465685, 988518685, 448209364, 2865601836, 2187078439, 1990686234, 3019923224, 293549923, 1361888576, 3
314852207, 3504492428, 2627986153, 178045653, 1177151005, 1668360675, 3394144983, 4125077361, 1196320939, 141241
4522, 3597932055, 2903358437, 1660402659, 3369503751, 2282657038, 4161742266, 1987716684, 2591875092, 1552665070
, 1570699220, 3113856222, 3001847315, 3999802157, 2457624526, 3387140189, 2893329771, 1119282962, 3441989850, 42
43659864, 4117788142, 1349969528, 3570821685, 2500876985, 464536579, 2990586201, 2068719130, 600910296, 41255779
33, 1319604080};
    uint32_t k[] = {0x74, 0x6F, 0x72, 0x61};
    int n = 66;
    btea(v, -n, k);
    for (size_t i = 0; i < 264; i++)
```

```
        printf("0x%02x ", ROL(*((char *)v + i) & 0xff));
    return 0;
}
// 0x60, 0xfc, 0x68, 0x4c, 0x77, 0x26, 0x7, 0x33, 0xd2, 0x64, 0x8b, 0x52, 0x30, 0x8b, 0x52, 0xc, 0x8b, 0x52, 0x1
4, 0x8b, 0x72, 0x28, 0xf, 0xb7, 0x4a, 0x26, 0x33, 0xff, 0x33, 0xc0, 0xac, 0x3c, 0x61, 0x7c, 0x2, 0x2c, 0x20, 0xc
1, 0xcf, 0xd, 0x3, 0xf8, 0xe2, 0xf0, 0x52, 0x57, 0x8b, 0x52, 0x10, 0x8b, 0x42, 0x3c, 0x3, 0xc2, 0x8b, 0x40, 0x78
, 0x85, 0xc0, 0xf, 0x84, 0xbe, 0x0, 0x0, 0x0, 0x3, 0xc2, 0x50, 0x8b, 0x48, 0x18, 0x8b, 0x58, 0x20, 0x3, 0xda, 0x
83, 0xf9, 0x0, 0xf, 0x84, 0xa9, 0x0, 0x0, 0x0, 0x49, 0x8b, 0x34, 0x8b, 0x3, 0xf2, 0x33, 0xff, 0x33, 0xc0, 0xac,
0xc1, 0xcf, 0xd, 0x3, 0xf8, 0x3a, 0xc4, 0x75, 0xf4, 0x3, 0x7c, 0x24, 0x4, 0x3b, 0x7c, 0x24, 0xc, 0x75, 0xd9, 0x3
3, 0xff, 0x33, 0xc9, 0x83, 0xc2, 0x50, 0xf, 0xb6, 0x4, 0xa, 0xc1, 0xcf, 0xd, 0x3, 0xf8, 0x41, 0x83, 0xf9, 0xe, 0
x75, 0xf1, 0xc1, 0xcf, 0xd, 0x57, 0x33, 0xff, 0x33, 0xc9, 0x8b, 0x54, 0x24, 0x3c, 0x52, 0xf, 0xb6, 0x1c, 0xe, 0x
b8, 0x67, 0x66, 0x66, 0x66, 0xf7, 0xeb, 0xd1, 0xfa, 0x8b, 0xc2, 0xc1, 0xe8, 0x1f, 0x3, 0xc2, 0x8d, 0x4, 0x80, 0x
2b, 0xd8, 0x5a, 0xf, 0xb6, 0x4, 0xa, 0x2b, 0xc3, 0xc1, 0xcf, 0xd, 0x3, 0xf8, 0x41, 0x83, 0xf9, 0xe, 0x75, 0xd4,
0xc1, 0xcf, 0xd, 0x3b, 0x3c, 0x24, 0x74, 0x16, 0x68, 0x25, 0x73, 0x0, 0x0, 0x8b, 0xc4, 0x68, 0x6e, 0x6f, 0x0, 0x
0, 0x54, 0x50, 0x8b, 0x5c, 0x24, 0x48, 0xff, 0xd3, 0xeb, 0x14, 0x68, 0x25, 0x73, 0x0, 0x0, 0x8b, 0xc4, 0x68, 0x7
9, 0x65, 0x73, 0x0, 0x54, 0x50, 0x8b, 0x5c, 0x24, 0x48, 0xff, 0xd3, 0x58, 0x58, 0x58, 0x58, 0x58, 0x58, 0x58, 0x
58, 0x58, 0x61, 0xc3, 0x58, 0x5f, 0x5a, 0x8b, 0x12, 0xe9, 0xb, 0xff, 0xff, 0xff
```

然后base64就行了，也没变表直接解密

`YPxoTHcmBzPSZItSMItSDItSFItyKA+3SiYz/zPArDxhfAIsIMHPDQP44vBSV4tSEItCPAPCi0B4hcAPhL4AAAADwlCLSBiLWCAD2oP5AA+EqQAAAEmLNIsD8jP/M8Cswc8NA/g6xHX0A3wkBDt8JAx12TP/M8mDwlAPtgQKwc8NA/hBg/kOdfHBzw1XM/8zyYtUJDxSD7YcDrhnZmZm9+vR+ovCwegfA8KNBIAr2FoPtgQKK8PBzw0D+EGD+Q511MHPDTs8JHQWaCVzAACLxGhubwAAVFCLXCRI/9PrFGglcwAAi8RoeWVzAFRQi1wkSP/TWFhYWFhYYYcNYX1qLEukLhhcHAwkFAA0MDN4MDgzeDg8ODg6AGDdVYPxoTHcmBzPSZItSMItSDItSFItyKA+3SiYz/zPArDxhfAIsIMHPDQP44vBSV4tSEItCPAPCi0B4hcAPhL4AAAADwlCLSBiLWCAD2oP5AA+EqQAAAEmLNIsD8jP/M8Cswc8NA/g6xHX0A3wkBDt8JAx12TP/M8mDwlAPtgQKwc8NA/hBg/kOdfHBzw1XM/8zyYtUJDxSD7YcDrhnZmZm9+vR+ovCwegfA8KNBIAr2FoPtgQKK8PBzw0D+EGD+Q511MHPDTs8JHQWaCVzAACLxGhubwAAVFCLXCRI/9PrFGglcwAAi8RoeWVzAFRQi1wkSP/TWFhYWFhYYYcNYX1qLEukL`
接下来是找flag
静态看不出来，上动调，脱壳有点问题运行中断，还是调试加壳的吧
flag: `hhcHAwkFAA0MDN`？
然后不对，得反调试
调了半天

脚本

```python
k = [1,1,2,0,1,0,3,4,2,4,1,4,0,0]
a = 'is program can'
for i in range(len(k)):
    print(chr(ord(a[i])+k[i]),end='')
#jt"psojvcq!gan
```

Contra 2048

得要点10下，进程序只有个helloworld，随便瞎点没反应

```java
protected void onCreate(Bundle arg2) {
        super.onCreate(arg2);
        this.setContentView(0x7F09001C);  // layout:activity_main
        this.imageButton = (ImageButton)this.findViewById(0x7F070046);  // id:imageButton
        this.imageButton.setOnClickListener(new View.OnClickListener() {
            @Override  // android.view.View$OnClickListener
            public void onClick(View arg3) {
                ++MainActivity.this.cnt;
                if(MainActivity.this.cnt > 10) {
                    MainActivity.this.cnt = 0;
                    Intent v3 = new Intent();
                    v3.setClass(MainActivity.this, TestActivity.class);
                    MainActivity.this.startActivity(v3);
```

在activity_main查看页面布局

```
<?xml version="1.0" encoding="UTF-8"?>
<androidx.constraintlayout.widget.ConstraintLayout android:background="#ffffff" android:layout_height="-1" andro
id:layout_width="-1" xmlns:android="http://schemas.android.com/apk/res/android" xmlns:app="http://schemas.androi
d.com/apk/res-auto">
  <TextView android:id="@id/sample_text" android:layout_height="-2" android:layout_width="-2" android:text="Hell
o World!" app:layout_constraintBottom_toBottomOf="0" app:layout_constraintLeft_toLeftOf="0" app:layout_constrain
tRight_toRightOf="0" app:layout_constraintTop_toTopOf="0"/>
  <ImageButton android:background="#ffffff" android:id="@id/imageButton" android:layout_height="47.0dp" android:
layout_width="52.0dp" app:layout_constraintBottom_toBottomOf="0" app:layout_constraintEnd_toEndOf="0" app:layout
_constraintHorizontal_bias="1.0" app:layout_constraintStart_toStartOf="0" app:layout_constraintTop_toTopOf="0" a
pp:layout_constraintVertical_bias="1.0" app:srcCompat="@android:color/background_light"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

按钮位置在android:layout_height="47.0dp" android:layout_width="52.0dp",点一下还会发出声音，不知道是什么原因。
然后就是webview封装的2048，`this.webView.addJavascriptInterface(this, "gameManager");`在assert/web/js下课以找到`game_manager`
然后看一下一行，格式化一下，js混淆。。。
去混淆
发现关键函数

```
206 GameManager.prototype["checkseq"] = function () {
207     if (seq.length < 100) {
208     }
209     for (Value_fl = 0; Value_fl < Math.floor(seq.length / 4); Value_fl++) {}
210     console.log("Cheat code: " + _0x2603b4);
211     var _0x382931 = kzlso("", "");
212     window.gameManager.pre("Hello from 2048!");
213     bbb = fromByteArray(string2Bin(_0x382931));
214     res = window.gameManager.docheck(bbb);
215     console.log(res);
216     seq = new Array();
217     if (String(res) == "Bingo!") {
218     }
219     return false;
220 };
```

CSDN @Pvr1sC

混淆去的不彻底，在kzlso函数中使用了XTEA加密
fromByteArray函数是base64
在so文件中不找到check函数，猜测做了隐藏
frida dump so文件，出不来（可能是我的操作问题）
`一定要认真看代码啊，在libnative-lib文件在发现了对frida的反调试，然后本人花了一下午时间在折腾frida`
用unidbg，不会搞，瞄了眼emtanling大佬的wp，得到偏移值为1970，libnative-lib跳转到sub_1970

```
__int64 __fastcall sub_1970(__int64 *a1, __int64 a2, __int64 a3)
{
  __int64 v5; // x0
  __int64 v6; // x0
  __int64 v7; // x1
  int v8; // w0
  __int64 v9; // x8
  __int64 v10; // x21
  _BYTE v12[256]; // [xsp+0h] [xbp-1A0h] BYREF
  __int128 v13[6]; // [xsp+100h] [xbp-A0h] BYREF
  int v14; // [xsp+160h] [xbp-40h]
  __int64 v15; // [xsp+168h] [xbp-38h]

  v15 = *(_QWORD *)(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
  v14 = 0;
  memset(v13, 0, sizeof(v13));
  v5 = sub_1418();                              // anti_debug
  v6 = sub_15B0(v5);
  sub_1724(v6, v7);
  sub_150C((__int64)v12, 256LL, (__int64)&byte_100F8);
  v8 = (*(__int64 (__fastcall **)(__int64 *, __int64))(*a1 + 1344))(a1, a3);
  v9 = *a1;
  if ( v8 == 64 )
  {
    v10 = (*(__int64 (__fastcall **)(__int64 *, __int64, _QWORD))(v9 + 1352))(a1, a3, 0LL);
    if ( (unsigned int)sub_1874(v10, v13) == 1 )// aes
    {
      sub_58E4(v13, 48LL);
      sub_150C((__int64)v12, 256LL, (__int64)&byte_100FC);
    }
    (*(void (__fastcall **)(__int64 *, __int64, __int64))(*a1 + 1360))(a1, a3, v10);
    v9 = *a1;
  }
  return (*(__int64 (__fastcall **)(__int64 *, _BYTE *))(v9 + 1336))(a1, v12);
}
```

sub_1418,sub_15B0,sub_1724都是anti_debug，
在这有些对字符串的混淆，就只是异或，分别
在`datadiv_decode14253863403468425951()`和`datadiv_decode6598209846029502604`写个IDA脚本就好或者在需要的时候异或一下

```
int8x16_t datadiv_decode6598209846029502604()
{
  int8x16_t v0; // q3
  int8x16_t v1; // q2
  int8x16_t v2; // q4
  int8x16_t v3; // q5
  int8x16_t v4; // q17
  int8x16_t result; // q0

  v0.n128_u64[0] = 0x6565656565656565LL;
  v0.n128_u64[1] = 0x6565656565656565LL;
  xmmword_100E0 = (__int128)veorq_s8((int8x16_t)xmmword_100E0, v0);
  *(int8x8_t *)a1W63R = veor_s8(*(int8x8_t *)a1W63R, (int8x8_t)0x5252525252525252LL);
  a1W63R[8] ^= 0x52u;
  a1W63R[9] ^= 0x52u;
  a1W63R[10] ^= 0x52u;
  a1W63R[11] ^= 0x52u;
  a1W63R[12] ^= 0x52u;
  a1W63R[13] ^= 0x52u;
  a1W63R[14] ^= 0x52u;
  *(int8x8_t *)aVpcagprkf = veor_s8(*(int8x8_t *)aVpcagprkf, (int8x8_t)0x2222222222222222LL);
  byte_10090 ^= 0xC6u;
  byte_10091 ^= 0xC6u;
  aVpcagprkf[8] ^= 0x22u;
  aVpcagprkf[9] ^= 0x22u;
  byte_100A4 ^= 0xA1u;
  byte_100A5 ^= 0xA1u;
  byte_100A6 ^= 0xA1u;
```

CSDN @Pvr1sC

sub_1874(v7, v10)里有aes加密
然后在sub_540C函数，OLLVM混淆，发现了md5，但重磅还是下面的sendto函数，因为我们需要找到发的pcap包进行的加密

```
50            else
51            {
52              v7 = malloc(0x11u);
53              *v7 = -1;
54              v8 = (char *)malloc(0x34u);
55              *(_DWORD *)v8 = dword_10330;
56              *((_DWORD *)v8 + 1) = 4;
57              *((_DWORD *)v8 + 2) = time(0LL);
58              sub_C004(v27);                        // md5
59              sub_C01C(v27, v7, 17LL);
60              sub_C098(v27, v26);
61              v9 = v7[16];
62              v10 = *(_OWORD *)v7;
63              *((_DWORD *)v8 + 3) = v26[0];
64              *((_DWORD *)v8 + 4) = 17;
65              v8[36] = v9;
66              *(_OWORD *)(v8 + 20) = v10;
67              sendto(fd, v8, 0x34u, 0, v22, 16);
68              close(fd);
69              v2 = 0;
70              v5 = 11889251;
71            }
72          }
```

CSDN @Pvr1sC

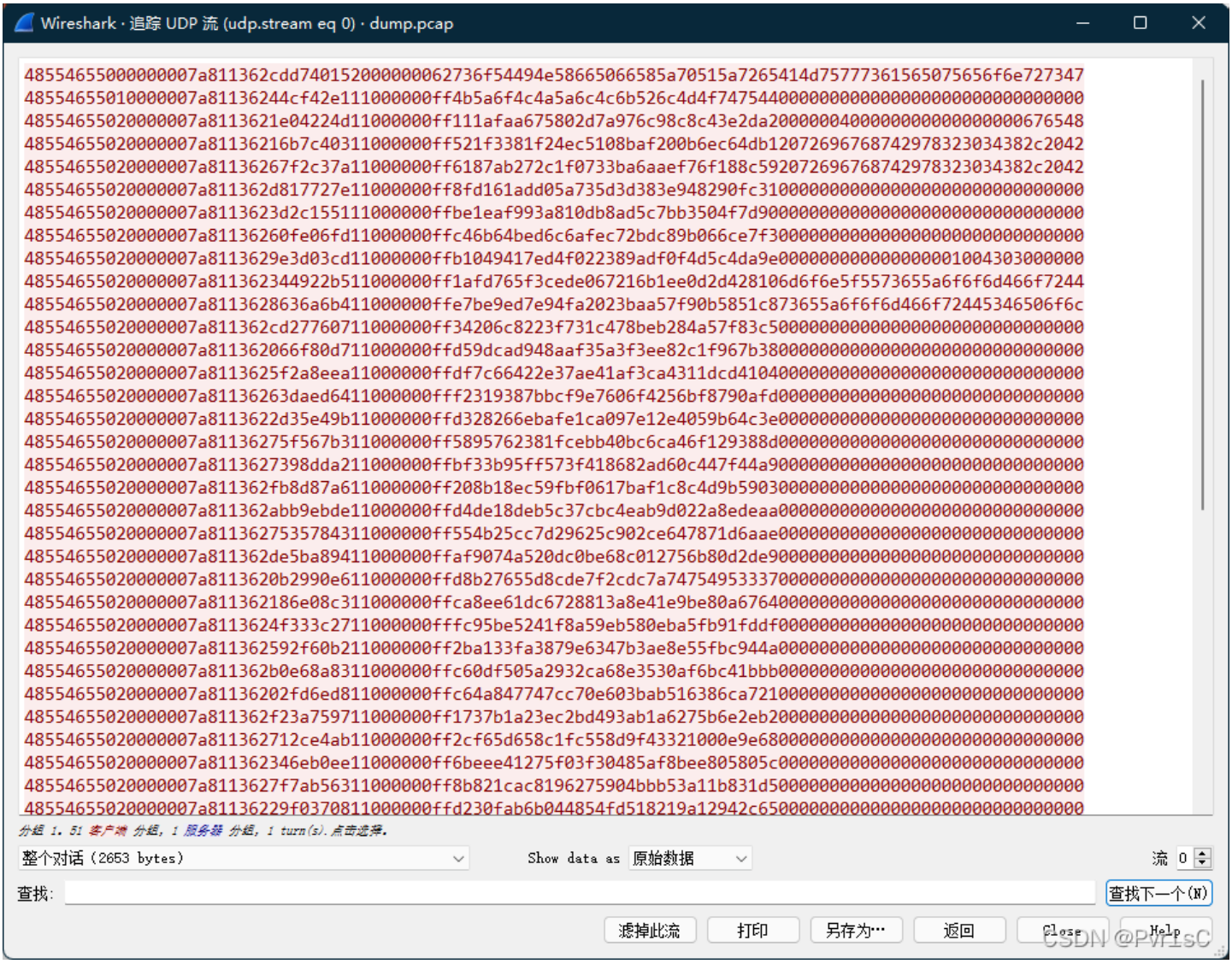
经过一系列瞎找，找到了
上边是AES加密

```
95    sub_A110(&aYhXhY, &v23 - 30, 128LL);          // aes
96    sub_5CD0(v18, v18, &v23 - 30, 128LL);
97    v19 = (char *)malloc(0x34u);                  // 包的大小
98    *(_DWORD *)v19 = dword_10330;                 // HUFU
99    *((_DWORD *)v19 + 1) = 2;                     // type
100   *((_DWORD *)v19 + 2) = time(0LL);             // time
101   md5_init((__int64)v31);                       // md5加密
102   md5_updata((__int64)v31, v17, 17LL);
103   md5_final(v31, v30);
104   v20 = v17[16];
105   v21 = *(_OWORD *)v17;
106   v22 = v24;
107   *((_DWORD *)v19 + 3) = v30[0];                // MD5
108   *((_DWORD *)v19 + 4) = 17;                    // length
109   v19[36] = v20;                                // data
110   *(_OWORD *)(v19 + 20) = v21;                  // data[0]==-1
111   result = sendto(v4, v19, 0x34u, 0, v22, 16);
112 }
113 return result;
```

CSDN @Pvr1sC

```python
# 得到key
s="yh]~xh^~Y`^"
for i in range(11):
    print(chr(ord(s[i])^0x32),end="")
# KZoLJZlLkRl
```

查看pcap，只提取AES加密的部分，且type为2



提取，因为AES加密key的长度为16所以，为AES-128加密，128位

```
48554655 02000000 7a811362 1e04224d 11000000 ff 111afaa675802d7a976c98c8c43e2da2 00000040000000000000000676548
HUFU       type     time     md5      length  -1  data
```

提取pcap信息的脚本

```python
from Crypto.Cipher import AES
import binascii
s="""
48554655020000007a8113621e04224d11000000ff111afaa675802d7a976c98c8c43e2da200000040000000000000000676548
48554655020000007a81136216b7c40311000000ff521f3381f24ec5108baf200b6ec64db120726967687429783230343820c2042
48554655020000007a81136267f2c37a11000000ff6187ab272c1f0733ba6aaef76f188c592072696768742978323034382c2042
48554655020000007a811362d817727e11000000ff8fd161add05a735d3d383e948290fc310000000000000000000000000000
48554655020000007a8113623d2c155111000000ffbe1eaf993a810db8ad5c7bb3504f7d900000000000000000000000000000
48554655020000007a81136260fe06fd11000000ffc46b64bed6c6afec72bdc89b066ce7f300000000000000000000000000000
```

```
4855465502000007a8113629e3d03cd11000000ffb1049417ed4f022389adf0f4d5c4da9e000000000000000001004303000000
4855465502000007a811362344922b511000000ff1afd765f3cede067216b1ee0d2d428106d6f6e5f5573655a6f6f6d466f7244
4855465502000007a8113628636a6b411000000ffe7be9ed7e94fa2023baa57f90b5851c873655a6f6f6d466f72445346506f6c
4855465502000007a811362cd27760711000000ff34206c8223f731c478beb284a57f83c500000000000000000000000000000000
4855465502000007a811362066f80d711000000ffd59dcad948aaf35a3f3ee82c1f967b380000000000000000000000000000000
4855465502000007a8113625f2a8eea11000000ffdf7c66422e37ae41af3ca4311dcd410400000000000000000000000000000000
4855465502000007a81136263daed6411000000fff2319387bbcf9e7606f4256bf8790afd000000000000000000000000000000000
4855465502000007a8113622d35e49b11000000ffd328266ebafe1ca097e12e4059b64c3e000000000000000000000000000000000
4855465502000007a81136275f567b311000000ff5895762381fcebb40bc6ca46f129388d000000000000000000000000000000000
4855465502000007a8113627398dda211000000ffbf33b95ff573f418682ad60c447f44a900000000000000000000000000000000
4855465502000007a811362fb8d87a611000000ff208b18ec59fbf0617baf1c8c4d9b590300000000000000000000000000000000
4855465502000007a811362abb9ebde11000000ffd4de18deb5c37cbc4eab9d022a8edeaa000000000000000000000000000000000
4855465502000007a81136275357843110000000ff554b25cc7d29625c902ce647871d6aae000000000000000000000000000000000
4855465502000007a811362de5ba89411000000ffaf9074a520dc0be68c012756b80d2de900000000000000000000000000000000
4855465502000007a8113620b2990e611000000ffd8b27655d8cde7f2cdc7a74754953337000000000000000000000000000000000
4855465502000007a811362186e08c311000000ffca8ee61dc6728813a8e41e9be80a676400000000000000000000000000000000
4855465502000007a8113624f333c2711000000ffffc95be5241f8a59eb580eba5fb91fddf000000000000000000000000000000000
4855465502000007a811362592f60b211000000ff2ba133fa3879e6347b3ae8e55fbc944a00000000000000000000000000000000
4855465502000007a811362b0e68a8311000000ffc60df505a2932ca68e3530af6bc41bbb00000000000000000000000000000000
4855465502000007a81136202fd6ed811000000ffc64a847747cc70e603bab516386ca721000000000000000000000000000000000
4855465502000007a811362f23a759711000000ff1737b1a23ec2bd493ab1a6275b6e2eb200000000000000000000000000000000
4855465502000007a811362712ce4ab11000000ff2cf65d658c1fc558d9f43321000e9e680000000000000000000000000000000
4855465502000007a811362346eb0ee11000000ff6beee41275f03f30485af8bee805805c000000000000000000000000000000000
4855465502000007a8113627f7ab56311000000ff8b821cac8196275904bbb53a11b831d500000000000000000000000000000000
4855465502000007a81136229f0370811000000ffd230fab6b044854fd518219a12942c650000000000000000000000000000000
4855465502000007a811362f5fd20f611000000ffa0435d523e66126d7e6fe850492319a200000000000000000000000000000000
4855465502000007a811362c5e03ec811000000ff0fe33dede172c8d2bca1f1cda054f6410000000000000000000000000000000
4855465502000007a811362a89e5d0511000000ffca4633933b0464fc3158e4ccb987fc8f000000000000000000000000000000000
4855465502000007a811362e7c61d5711000000ff6e3f1bc661417ef0f5d2aa23630fa113000000000000000000000000000000000
4855465502000007a8113627cea0abc11000000ff95729dfa6ee0629f5f7fb8b443bddf7d000000000000000000000000000000000
4855465502000007a81136224678bf611000000ffaee28a1366cc508ff196e160dd9e5ac400000000000000000000000000000000
4855465502000007a811362c4f86dd511000000ff9aa2b87453e0f0dabcf02667ba668c150000000000000000000000000000000000
4855465502000007a811362d424678711000000ffd1a2347614fa59ecfb2ac964c1198fa000000000000000000000000000000000
4855465502000007a811362825265cb11000000ffc1aff18b31c51a7ce47d3a2277515b550000000000000000000000000000000000
4855465502000007a811362a8591ee411000000ffac25f6f85a0668f26b53bda11d5f10b000000000000000000000000000000000
4855465502000007a8113624412ff9a11000000ffaf38d6d29ebfbbc44a85538b69adf05a000000000000000000000000000000000
4855465502000007a8113624ee147da11000000fff17f0bec7ade982167e8903b0bd553680000000000000000000000000000000000
4855465502000007a811362357c142a11000000ffaf6009e720b4d142d249c55afcc0892f000000000000000000000000000000000
4855465502000007a8113625b64481e11000000fff42702d51a7c62683ab60b3ecbd5180d000000000000000000000000000000000
4855465502000007a811362d1ac385611000000ff8969dcc7f808b88aff97be2d0942431c000000000000000000000000000000000
4855465502000007a811362230b107e11000000ffacadd756fd5a86401c88b0d0469c7152000000000000000000000000000000000
4855465502000007a811362e7ed93d511000000ff7b9772ccb2282ff68ed5d77132785c040000000000000000000000000000000000
""".split()
s1=[]

key = b"KZoLJZlLkRlMOtuD"
crypto = AES.new(key, mode=AES.MODE_ECB)
for i in range(len(s)):
    s1.append(s[i][42:42+32])
decode = []
for enc in s1:
 decode.append(crypto.decrypt(binascii.a2b_hex(enc))[4])
print(decode)
```

然后是调试，先中断了，之后搞