【逆向学习】 拯救地球 writeup

原创

charlie_heng 于 2017-12-20 20:42:43 发布 600 收藏

分类专栏： 二进制-逆向工程

本文链接：https://blog.csdn.net/charlie_heng/article/details/78857350

二进制-逆向工程 专栏收录该内容

34 篇文章 3 订阅

订阅专栏

这是16年的国赛的逆向，话说这名字感觉真有点中二。。。。

首先反编译了一下class文件，看了下，并没有什么重要的东西，只是调用了native函数

然后反编译下so文件

首先看了下JNI_Onload函数

```
int v7; // [sp+8h] [bp-10h]

v7 = a3;
v6 = 0;
if ( (*(int (**)(void))(*(_DWORD *)a1 + 24))()
  || (v3 = v6,
      (v4 = ((int (__fastcall *)(_JNIEnv *, const char *))v6->functions->FindClass)(
              v6,
              "com/itea/shield/PZApplication")) == 0)
  || ((int (__fastcall *)(_JNIEnv *, int, char **, signed int))v3->functions->RegisterNatives)(v3, v4, off_5004, 1) < 0 )
{
  result = -1;
}
else
{
  result = 65542;
}
return result;
}
```


看起来注册了一个native函数，进那个函数看下

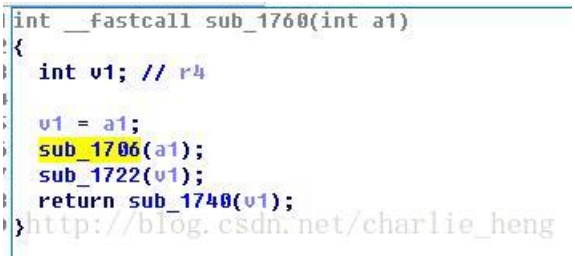
```
.pgtext:00002840 loc_2840                                ; DATA XREF: .data:0000500C↓o
.pgtext:00002840 ; __unwind {
.pgtext:00002840                 STCNE   p5, c11, [R6], {0xF8}
.pgtext:00002844                 LDCNE   p8, c4, [R7], {0x13}
.pgtext:00002844 ; ---------------------------------------------------------------------------
.pgtext:00002848                 DCB  0xD
.pgtext:00002849                 DCB 0x1C
.pgtext:0000284A                 DCB    0
.pgtext:0000284B                 DCB 0xF0
.pgtext:0000284C                 DCB 0x83
.pgtext:0000284D                 DCB 0xFC
.pgtext:0000284E                 DCB    4
.pgtext:0000284F                 DCB 0x1C
.pgtext:00002850                 DCB 0xFE
.pgtext:00002851                 DCB 0xF7
.pgtext:00002852                 DCB 0xC0
.pgtext:00002853                 DCB 0xFD
.pgtext:00002854                 DCB 0x39 ; 9
.pgtext:00002855                 DCB 0x1C
.pgtext:00002856                 DCB 0x20
.pgtext:00002857                 DCB 0x1C
.pgtext:00002858                 DCB    0
.pgtext:00002859                 DCB 0xF0
.pgtext:0000285A                 DCB 0x4A ; J
.pgtext:0000285B                 DCB 0xF8
.pgtext:0000285C                 DCB 0x31 ; 1
.pgtext:0000285D                 DCB 0x1C
.pgtext:0000285E                 DCB 0x20
.pgtext:0000285F                 DCB 0x1C
.pgtext:00002860                 DCB    0
.pgtext:00002861                 DCB 0xF0
.pgtext:00002862                 DCB 0x4D ; M
.pgtext:00002863                 DCB 0xF8
.pgtext:00002864                 DCB 0x29 ; )
.pgtext:00002865                 DCB 0x1C
.pgtext:00002866                 DCB 0x20
.pgtext:00002867                 DCB 0x1C
```


然后按alt+G把值改为1，设为code16模式

```
1 int __fastcall sub_2840(int a1, int a2, int a3)
2 {
3   int v3; // r6
4   int v4; // r7
5   int v5; // r5
6   int v6; // r4
7
8   v3 = a1;
9   v4 = a3;
10  v5 = a2;
11  v6 = operator new((unsigned int)&elf_hash_bucket[6]);
12  sub_13D4();
13  sub_28F0(v6, v4);
14  sub_28FE(v6, v3);
15  sub_2908(v6, v5);
16  sub_2B98(v6);
17  sub_2BE4(v6);
18  sub_290C(v6);
19  sub_2A16(v6);
20  sub_2B40(v6);
21  return sub_2FE4(v6);
22 }
```


看了下大部分都是各种反调试，还有提取资源

比较有用的是sub_2B40里的sub_1760



这三个函数的作用是把，encrypt_dex里面的0到999异或0x11，1000到2047异或0x22，2048到2999异或0x33，异或之后就很简单了

答案就是 yes,it is the answer的base64encode后的字符