实验吧 520app3 writeup

原创

charlie_heng 于 2018-01-25 19:04:59 发布 861 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/79165532

这道题在之前比赛的时候也遇到过，当时一脸懵逼……(虽然现在也是差不多

首先先看看怎么动态调试so，这是教程http://blog.csdn.net/feibabeibei_beibei/article/details/52740212

按着教程，把Debugger option的那三个选项给选上，之后也不用取消掉

然后断在linker

之后根据偏移找到.init_proc

本来想跟下去，看看怎么解密数据的，但是报了个error，我也懒得跟了。。。

按F9，然后断下来之后，所有数据都解密出来了，这个时候可以用dd 把现在的so给dump下来

本来是可以修复so，然后静态看出flag的，但是我也不会修。。。。。

所以直接在动态调试的代码上面看，因为在原始的so里面也知道JNI_Onload的偏移，所以这个时候可以根据偏移来找到JNI_Onload

```
v3 = 0;
if ( (*(int (__fastcall **)(int, _JNIEnv **, signed int))(*(_DWORD *)a1 + 24))(a1, &v3, 65540) )
  return -1;
v5 = ((int (__fastcall *)(_JNIEnv *, signed int))v3->functions->FindClass)(v3, 20256);
if ( !v5 )
  return -1;
v6 = ((int (__fastcall *)(_JNIEnv *, int, void *, signed int))v3->functions->RegisterNatives)(v3, v5, &unk_711C, 1);
if ( v6 )
  return -1;
MEMORY[0x4D98](&v4, 0, sub_F8C, 0);
return 65540;
}
```


JNI_Onload大概就是这个样子，这个也是大概的套路

然后红色那里其实是J_create，创建了一个线程，sub_F8C是一个函数，不断的接收东西，但是只接收，什么也不处理

所以关键就在unk_711C那里

```
:90CBC11C dword_90CBC11C DCD 0x90CB9EE8
:90CBC11C
:90CBC120 DCD 0x90CB9EF0
:90CBC124 DCD 0x90CB630D
```

这里说明一下，上面那个图片是自己修了一半的半成品。。。所以变成那样。。。下面这个图片是动态调试的，每个人的地址都可能不同

这里有三个地址，第一个地址是指向字符串upload 的，第二个是指向字符串(Ljava/lang/String;)I，第三个地址才是真正的地址

反编译之后会看到类似这样的东西，但是这里是逆向出来是connect之类的字符串

```
{
  for ( i = 0; i <= 5; ++i )
    byte_90CBC05C[i] ^= 0x6Bu;
  v12 = (void (__fastcall *)(signed int, signed int, _DWORD))sub_90CB9DA8(dword_90CBC13C, (int)byte_90CBC05C);
  for ( j = 0; j <= 5; ++j )
    byte_90CBC05C[j] ^= 0x6Bu;
```


真正比较可疑的地方是这里，这里可以解出来两个字符串，第一个大概是帐号，第二个有Phone的才是flag，然后顺便看了下面，大概是调用了别的so，然后用浏览器来传数据的样子

```
do
{
  *(&a62 + a11) = a11 ^ aHAhawfdDjpvlTm[a11];
  ++a11;
}
while ( a11 <= 11 );
a59 = 0;
a60 = 0;
a61 = 0;
j_memset_0(&a65, 0, 16);
LOWORD(a65) = 2;
for ( i = 0; i <= 8; ++i )
  *((_BYTE *)&a59 + i) = i ^ aHAhawfdDjpvlTm[i + 12];
HIWORD(a65) = 21282;
```


做题的话到这里就可以了，如果是真正逆向的话肯定就不止这样了