JNI_OnLoad与init_array下断方法整理

原创

PandaOS 于 2016-07-23 23:43:17 发布 4947 收藏 5

分类专栏： Android逆向 文章标签： jni 调试

本文链接：https://blog.csdn.net/PandaOS/article/details/52008037

JNI_OnLoad和init_array 在SO脱壳中占了很多比重，那么如何在它们执行前下断点呢？

考虑到SO加载的时机，在jdb附加之前并不能对我们想调试的SO下断点，即使是知晓了偏移位置。因为它们还没有被加载。据说这可以用脚本来解决，这不是这篇文章讨论的范围。本文章讨论的方法是从系统库中下断的方法来达到目的。

首先讨论init_array下断点的方法:

init_array 是在so加载后由linker负责调用，详细细节可以翻阅linker源码。

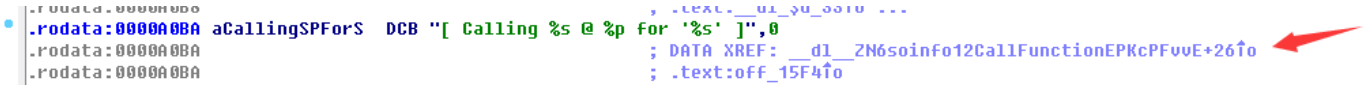
linker在调用init_array的时候会输出 “[Calling %s @ %p for ‘%s’]”

思路是可以通过定位该字符串来定位调用init_array 的位置。

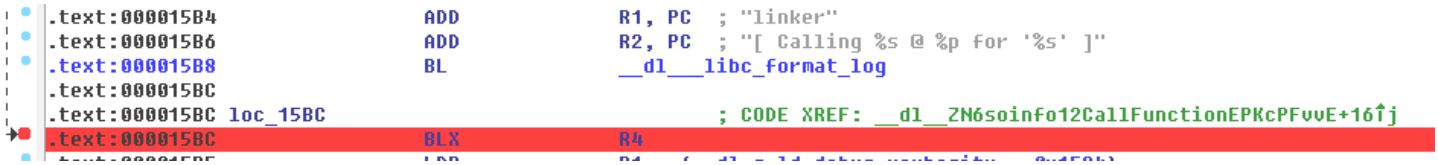
adb pull /system/bin/linker

下载手机中的linker，将下载来的linker拖入IDA中分析。等待IDA分析完后打开字符串表，搜索”[Calling %s @ %p for ‘%s’]”,为了方便记忆，直接搜索”call”也是可以的。

搜索出来后双击记录,下面是结果:



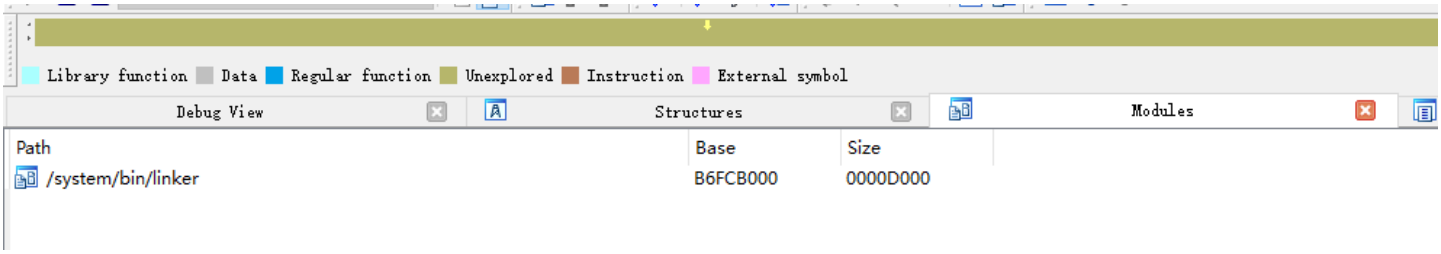
双击红色箭头处的 o 可以转到引用该字符串的地方。



BLX R4是调用代码Init_array的地方。 记录下此地偏移地址(0x15bc)，留作以为备用。

接下来在动态调试的时候，使用am以等待调试器附加的状态启动app，然后用ida附加。

附加后在IDA的module list中搜索linker这一项。



Base这一栏是linker在内存中加载的基地址，基地址=0xB6FCB000

所以调用init_array的地方blx r4在该内存中的偏移为Base+offset = 0xB6FCB000+0x15BC

在反汇编窗口按下G键，输入0xB6FCB000+0x15BC，即可跳转过去，并设置好断点。

如果跳过去后，目标并不是汇编代码，请机智的按下C键。

注意，这个时候系统库基本上都已经加载！注意，0x15BC这个偏移请牢记，下次可以直接用。

断点下好后，请用jdb附加，进程跑起来~

下面简介JNI_OnLoad函数下断方法。模仿init_array的思路，我们就想是不是可以在JNI_OnLoad的调用处下断点，这个思路完全正确。

首先，我们要确定JNI_OnLoad何时被调用。
JNI_OnLoad实际是在LoadNativeLibrary中调用的，详细的分析文章可以在看雪论坛找到。

LoadNativeLibrary通过dlsym查找JNI_OnLoad地址，所以也可以直接搜索”JNI_OnLoad”定位。

LoadNativeLibrary 函数在什么库？
Android 5.0 + libart.so
Android 5.0 - libdvm.so
请看官对号入座，6.0我不了解~

使用adb pull system/lib/libart.so 或 libdvm.so

将下载的文章载入到ida，在Export窗口搜索LoadNativeLibrary或字符串窗口搜索”JNI_OnLoad”。
注意LoadNativeLibrary是JavaVMExt类成员函数，所以搜出来有点怪，实际上就是它。

双击进入并F5出C源代码，LoadNativeLibrary代码量很大！
肉眼扫描 dlsym(handle, “JNI_OnLoad”);
第一个参数名可能不同，不过没关系。
如下面所述

```
v67 = dlsym(handle, "JNI_OnLoad");
if (v67)
{
    .......
     v85 = v67(v179, 0); //在此处按下Tab键，跳到汇编代码。
    ........
}
```

跳到汇编代码后:
.text:001DDE02 BLX R7
这里为调用处，记录下偏移，剩下的事情参考init_array。