linux so库反编译命令,使用IDA反编译.so文件并修改

转载

weixin_39640520 于 2021-05-04 23:24:10 发布 2149 收藏 2
文章标签： linux so库反编译命令
使用IDA反编译.so文件并修改

简述

之前一直在做应用层的开发很少接触底层，总感觉底层是一个很神秘的地方。最近各种原因之下有一些逆向的工作，但是好多应用的核心逻辑都是利用jni在c/c++层去实现的，这就给我们的逆向工作带来了很大的困难，所以了解底层知识还是比较重要的。逆向过程中java层面的逆向还是比较简单的，今天主要介绍一下如何逆向.so文件。

例子是我最近分析的一个项目，项目中数据是从.so库处理发出的。但是.so文件对手机的串号做了校验，也就是说只有固定的几部手机才能使用。

所以这篇博客最终的效果就是通过反编译.so文件，找到记录串号的Hex码修改为自己的串号，就可以绕过.so的串号校验了。

利用IDA反编译.so文件

IDA。是目前最棒的一个静态反编译软件，为众多0day世界的成员和ShellCode安全分析人士不可缺少的利器。IDA功能及其复杂，今天只是简单的介绍一些基本功能。

IDA的安装略过不讲，安装好之后打开IDA(Version 7.0.170914 Windows x64)第一次使用，选择直接选择Go即可，进入界面后左上角File—>Open选择要打开的so文件，我选择的是一个

armeabi-v7a指令集的.so文件。打开之后如下图设置，然后点击OK

1.png

打开后如下图

2.png

在左侧的function name中寻找main入口函数。找到后现在IDA View中显示的反编译出来的汇编语言看起来比较费劲，可以按F5转成伪代码

3.png

到这就可以看到逻辑是调用sub_32E4位置的方法然后执行死循环打印日志qcd err .....，这正是我启动失败logcat上显示的日志。现在基本能断定关键逻辑在sub_32E4方法中。

sub_32E4值得应该是方法的Hex码的起始位置是32E4，不过在IDA中可以直接鼠标双击该方法直接进入，如图

4.png

大致逻辑就是现获取手机的imei码然后和一个常量数组off_9D24里的已有串码进行比对，如果对不上就跳出，也就是执行了main方法里的打印错误日志的死循环。

所以关键的串号数组也就知道了，双击off_9D24找到数组位置，选中数组中某个条目，按 Q 就可以显示Hex码的位置，然后记住这个位置。

5.png

使用010 Editor修改.so文件

.so文件本质上已经编译成了二进制文件，使用010 Editor可以更方便的修改。

现在已经知道了需要修改的数组的位置，用010 Editor打开刚才分析的.so文件。

6.png

根据左侧的行号找到63F0这一行，这个时候就可以看到63F0这一行的第6个Hex码是38正好对应ASCII码的8，根据右侧的ASCII码对照表，正好是设备码数组的第二个元素的第一个字符。

找到了位置就可以把数组里的数据修改成任何你想要的设备码了，修改Hex码不太方便可以直接修改ASCII对照码表里的ASCII码。

7.png

现在我将8改成了9(当然设备码国内应该是86开头，这只是做个示范)，Hex码也自动改成了39。

好了现在就可以挑数组里的一个条目完全改成你自己的设备码，然后ctrl+S保存，将新的.so包放到工程里重新打包或者直接用adb放到手机的/data/user/0/com.xxx.xxx/lib下重启应用即可。经过尝试果然绕过了设备码限制，成功拿到数据。

后记

这次只是简单尝试，受困于不太熟悉汇编语言，只能修改静态资源值，如果熟悉汇编语言的话可以直接修改流程，比如修改if判断条件恒为true，或者置为相反条件即数组里的设备反而都不能用等，这是今后需要继续学习的地方