某货源app逆向抓取

所需设备和环境：

设备：安卓手机

抓包：fiddler+xposed+JustTrustme；

查壳：ApkScan-PKID

脱壳：frida-DEXDump；

反编译：jadx-gui；

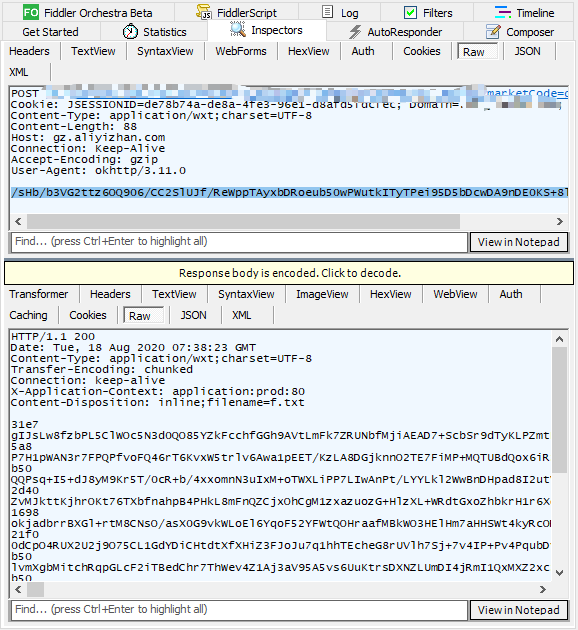
hook：frida;

**抓包**

手机安装app，设置好代理，打开fiddler先来抓个包，发现这个app做了证书验证，fiddler开启之后app提示连接不到服务器：



那就是app做了SSL pinning证书验证，解决这种问题一般都是安装xposed框架，里面有一个JustTrustme模块，它的原理就是hook，直接绕过证书验证类，安装方法大家百度吧。之后再打开app，可以看到成功抓到了包：



先简单分析一下，可以看到请求体中formdata是密文，响应内容也是密文，这个请求和响应中有用信息非常少，甚至都不知道在jadx-gui里怎么搜索，请求体中formdata是以两个等号结尾的，应该是个base64编码，其他一概不知，

**脱壳反编译**

那先来反编译，在这之前，通常是先用查壳工具检查一下app是否加固，打开ApkScan-PKID，把app拖入：



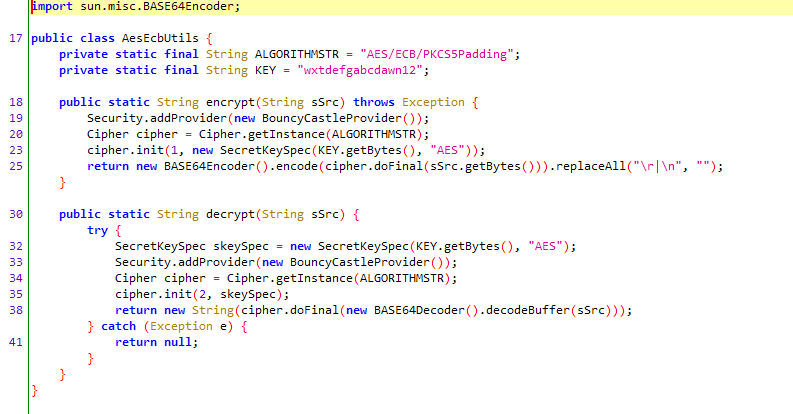
可以看到这个app使用了360加固，真是层层设限啊！！这里使用frida-DEXDump来脱壳，github上下载frida-DEXDump的源代码，完成之后打开项目所在文件夹，在当前位置打开命令行运行以下命令：

python main.py

等待脱壳完成，可以看到当前项目中生成了一个对应文件夹，里面有很多dex文件



下面用jadx-gui打开dex文件，一般先从最大的文件开始依次搜索关键字，我们知道java中使用base64是有BASE64Encoder关键字的，因为抓包得到的信息非常少，在这里就只能搜索这个关键字了，搜到第四个dex中，得到了疑似加密处：



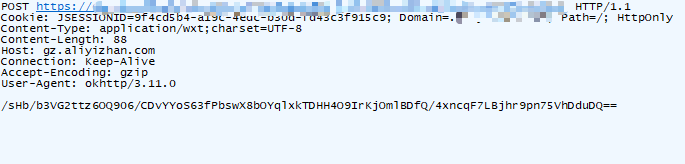
可以看到是使用了一个aes加密，密钥是固定的字符串。

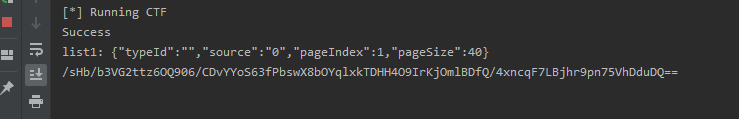
**Frida Hook**

Java不太熟，分析不来，直接使用frida来写一段hook代码看一看encrypt函数入参和出参的内容：

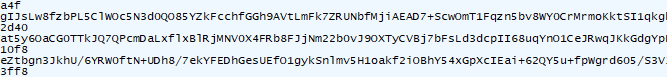


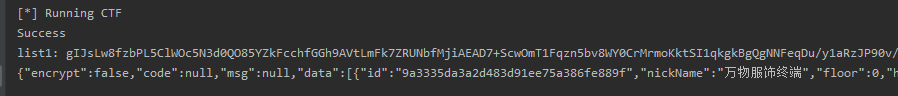
同时来抓包对比：





就得到了这里的请求data入参数据： pageIndex：当前页码，pageSize：当前页对应的数据条数， typeId和source是固定的， 接下来再来hook decrypt函数，对比抓包和hook结果：

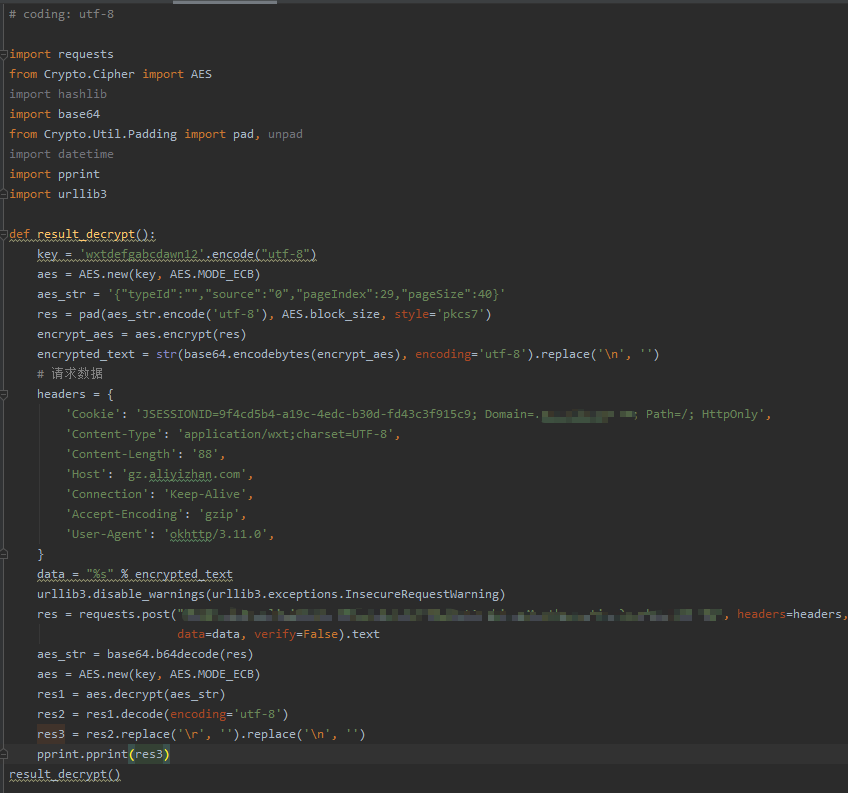




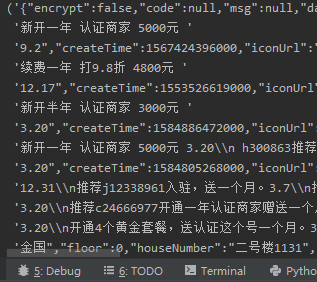
结果是一样的，至此，我们逆向分析就完成了。总结一下请求和响应过程，就是请求体中的data经过encrypt函数加密传参，改变pageIndex就可以得到每页数据，响应是经过decrypt函数加密显示，那我们只需要在python中实现这个aes加密解密过程就行了，从反编译的java代码中可以看出密钥是固定的：wxtdefgabcdawn12，没有iv偏移。

### 请求

直接上​代码：​



运行代码，成功拿到数据：



ok,以上就是逆向爬虫的全部内容，可以看到，现在数据加密已经很普遍了，随便一个很小的app都有好几道数据保护机制，这次只涉及到java层的加密，下次来讲讲native层加密的hook方法、frida-rpc主动调用以及逆向神器inspeckage的应用。 最后，以上内容仅供学习交流，大家不要搞事情啊！！