使用frida破解Native层加密

所需设备和环境：

设备：获取root权限的安卓手机

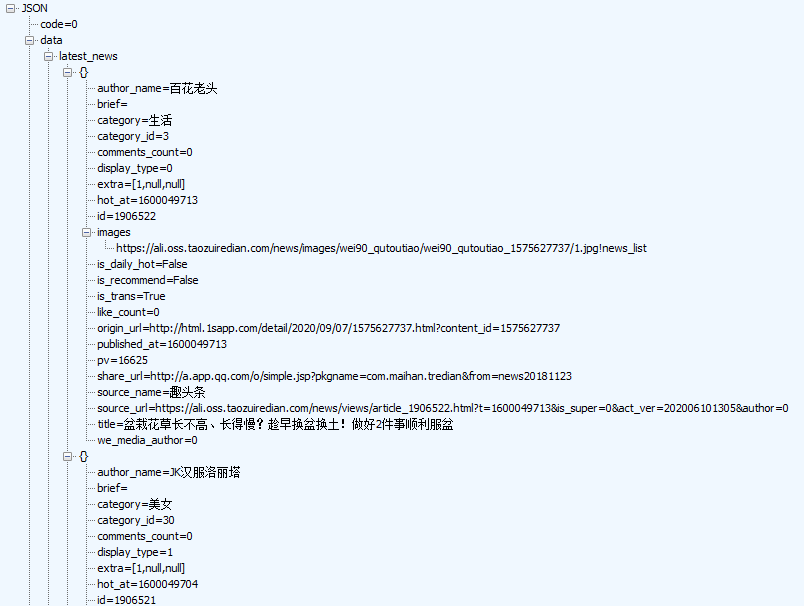
抓包：fiddler + proxyDroid

反编译：jadx-gui，ida

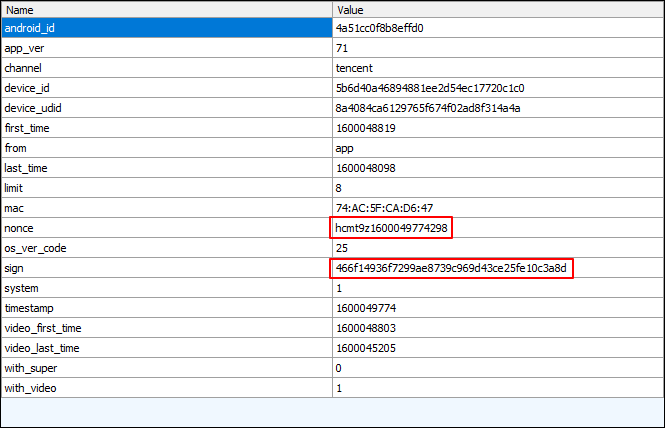
抓包

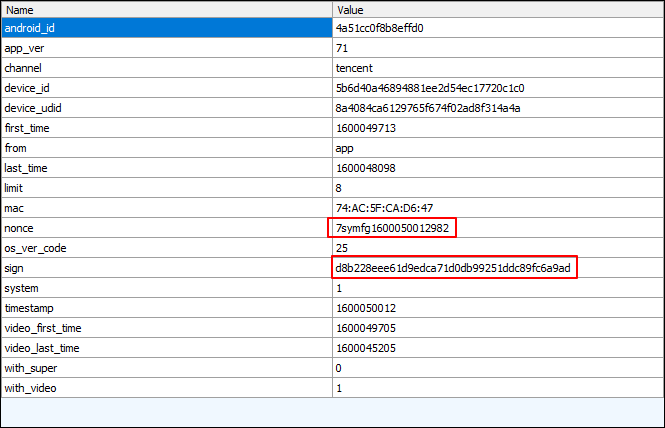
<https://zhuanlan.zhihu.com/p/46433599>

我们以这个app为例，手机获取root权限，安装proxyDroid强制全局代理app，打开fiddler先来抓个包：



成功抓到了列表详情页的数据，下拉刷新两次来看看请求参数有什么变化：





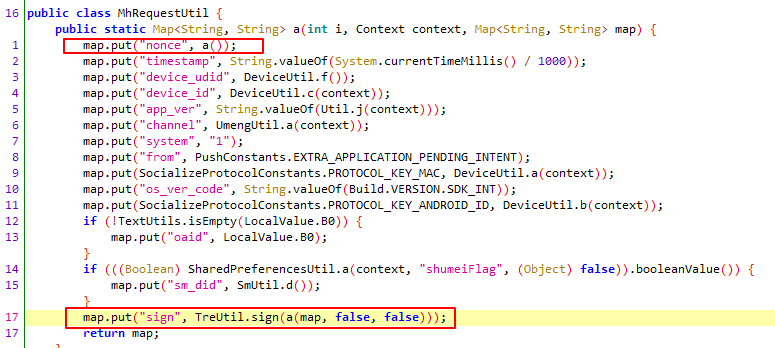
对比可得nonce和sign每次请求都在变，其他参数不是固定就是一些时间戳，所以我们只需破解这两个参数就可以拿到列表详情页的数据，打开jadx-gui来反编译app。

反编译

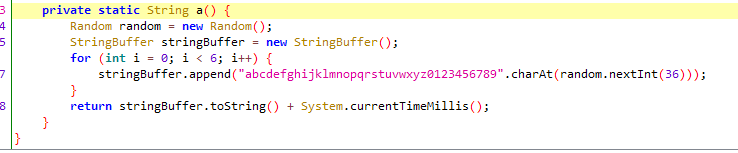
反编译完成之后先来全局搜索nonce：



点进去第一个看看：



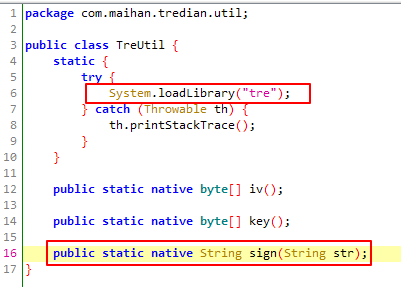
运气好像不错，这里定义了一个map，里面都是请求参数的生成方法，按住ctrl，鼠标点击这个a函数，看看nonce的具体内容：



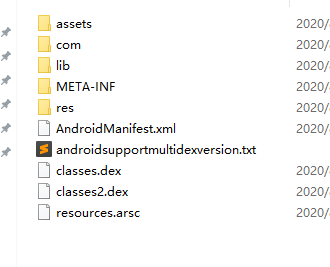
比较好容易理解，是从26个英文字母和10个阿拉伯数字中随机选择6个字符，然后拼接当前时间戳就形成了nonce参数，用python可以轻松实现，不过这里我们使用rpc主动调用来获取参数，以后如果遇到不理解的参数就可以直接主动调用：



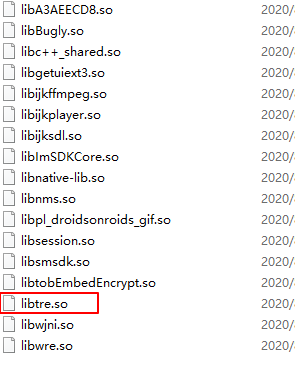
接下来看看sign，按住ctrl点击sign跳转到了如下页面：



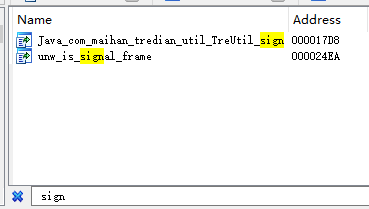
Sign的加密方法写到了native层去了，那我们只好去分析这个名为tre的so文件了，解压目标apk获取到了如下图这些文件：



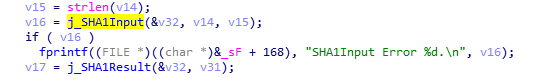
So文件就在这个lib文件夹里：

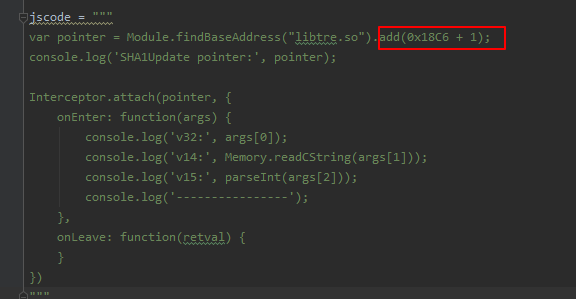


ida中打开这个文件，切换到Export选项卡，先来直接搜索sign：



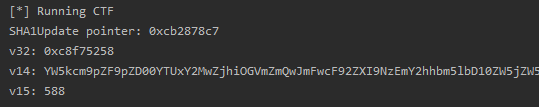
直接搜索到了，双击第一个点进去看看详情，都是汇编语言，按F5可以把汇编代码转换为伪C代码，下拉分析一波，看到这里有一个sha1的加密算法，因为在fiddler中看到 sign加密后的内容是40位，而sha1加密后也是40位，因此猜测它就是目标函数，那我们先来hook这个函数看看：





这里0x18C6即为j\_SHA1Input函数的地址，ida中使用tab键切换到汇编代码就可以获取到：

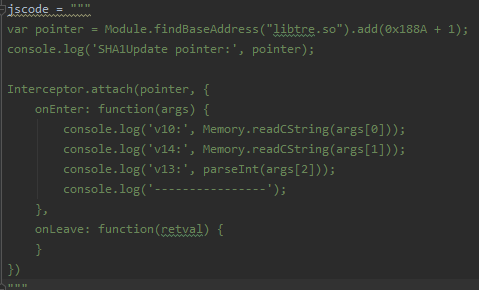
为什么要加1，这里不深究了，大神都是这样写的，我们也这样写，运行这个frida脚本来看看结果：



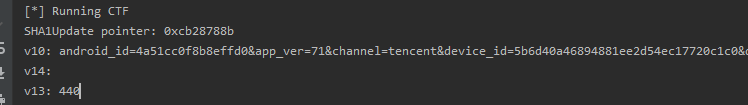
成功hook到了，但是这个结果是以”=”结尾的，应该是个base64，再来看看伪c代码：



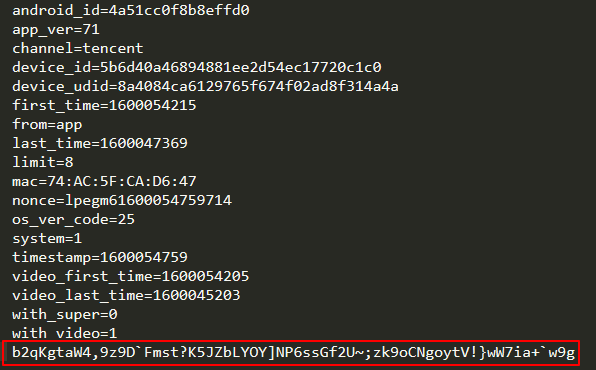
这里果然有一个base64编码，继续来hook这个函数：



运行脚本，得到了结果：



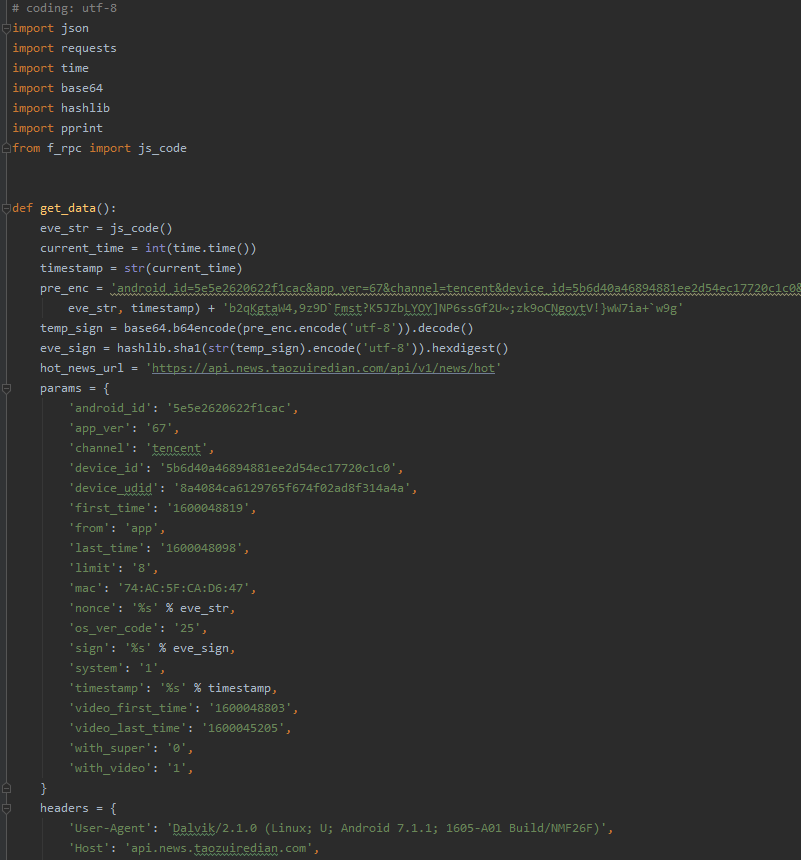
复制这个v10参数到sublime，使用正则表达式把”&”替换成换行符，来看看具体内容：



就是请求参数拼接了一个固定字符串，至此，我们逆向就全部完成了，总结一下，这个sign参数就是以上字符串先通过base64编码，然后sha1加密，使用python来改写。

请求

直接上代码：





运行一波，成功拿到数据：



总结

今天我们介绍了rpc和hook native层的脚本，可以看到frida是一个全能的hook框架，因此，想要做好逆向，必须学好frida，这里推荐两个个学习逆向的网站，一个是r0ysue大神的github：<https://github.com/r0ysue>，另一个是尼古拉斯赵四的公众号：编码美丽。掌握其中的内容，相信就能成功抓取绝大多数app的数据

Ok，以上就是今天的全部内容。