需求,逆向得物app首页推荐接口,

所需工具,

1. 一部root后的手机,
2. frida框架,
3. 将请求从socks层代理的app,DocksDroid,
4. 反编译so文件的软件ida64.
5. 反编译apk文件的jadx
6. 查看请求的软件charles

现在开始

1. 手机连接电脑,启动frida-server并设置端口转发



1. 手机启动DocksDroid,设置好ip和端口号,电脑打开charles.
2. 手机进入得物app,进入首页推荐,刷新,查看charles中类似的请求

图形用户界面, 文本, 应用程序, Word

描述已自动生成

看得出来是recommend/all/feed这个请求,下边分析该请求,请求体中newSign字段需要逆向,请求头中看似好几个字段都需要逆向, 多试几次,发现其他参数都可以不带,只有X-auth-token必须带,其实我们应该知道的是,android没有cookie机制,保存用户信息靠的是请求头中的token.该token是三段式的,代表用户的登录信息,短时间内多次请求,该字段是不变的

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

1. 所以先逆向newSign字段.反编译,直接搜,这里新版本应该是用了混淆,是搜不到的.我们使用老版本4.74.5,旧版本app使用豌豆荚下载.

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

一搜,感觉像是最后一个,可以hook试试,咱们就hook这个RequestUtils.c函数,看看该函数的参数和返回值,对于map类型的参数,js需要进行转换,不然看不到真实值

文本

描述已自动生成

很显然,这里就是入口,第一个map包含其他参数,这个函数生成newSign参数,第二个参数是

long currentTimeMillis = System.currentTimeMillis() - DuHttpConfig.d.getTimeOffset();通过查询,我们发现是当前时间戳

进入c函数,我们来看实现

文本

描述已自动生成

文本

描述已自动生成

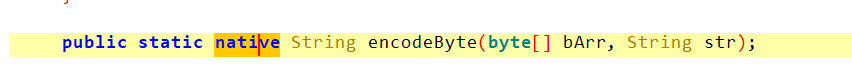
可以看到,把传进来的map字典传入几个参数,按照map的key的字符串顺序进行排序,最后把转换的字符串进行字符串转换,把这个字符串进行aes加密, DuHttpConfig.f15796c应该是密钥,最后执行a函数,a函数看得出来就是一个md5加密函数

现在需要知道的就是aes加密的模式,key和iv.进入encode函数,

文本

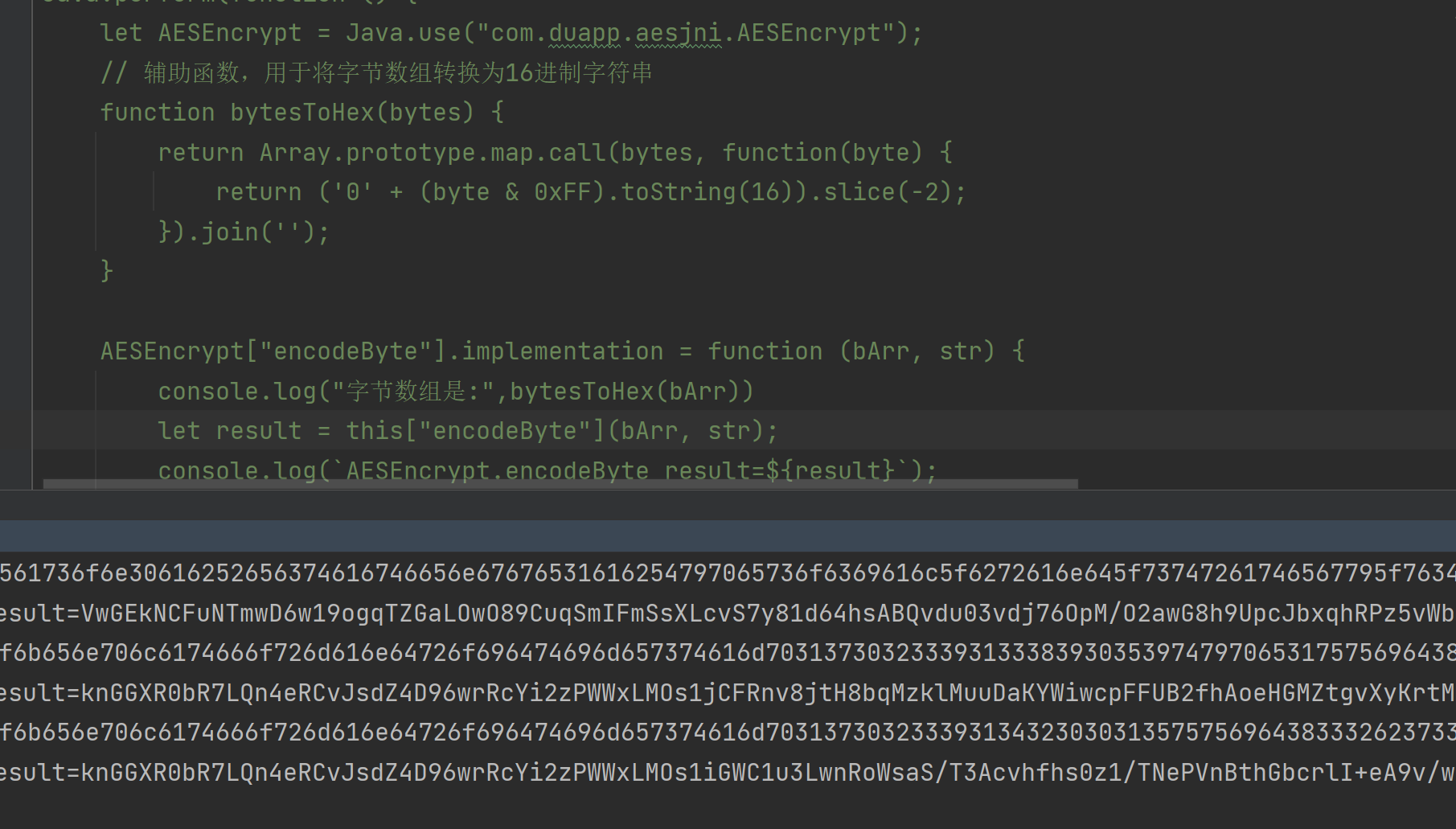
描述已自动生成

发现又进入另外一个encodeByte函数,点击进入,hook getvlaues函数发现返回的是一个定值,



发现encodeByte是用c语言写的函数.

现在先hook这个函数,看看我们的猜测对不对

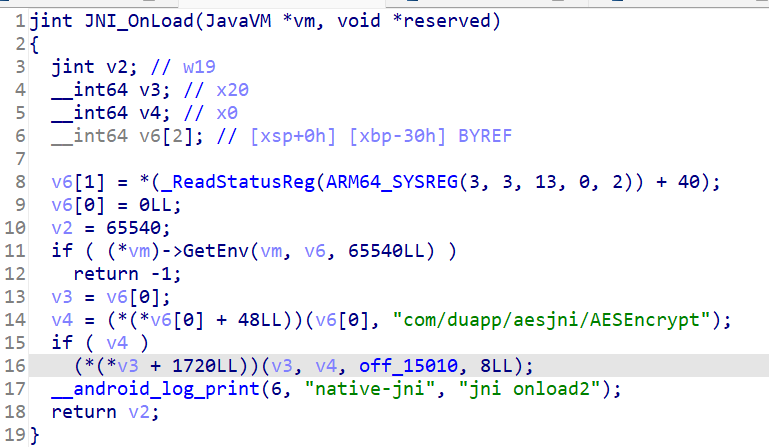


是这个没错了,既然是c语言,那么函数实现肯定在so文件中,具体在哪里.看上边

图形用户界面, 文本

描述已自动生成

显然在JNIEncrypt.so这个文件中.反编译之后, 发现是动态加载,进入jniload,找到关键参数,



这个off\_15010就记录着对应关系,



**进入找到encode函数,**

**图形用户界面, 文本, 应用程序

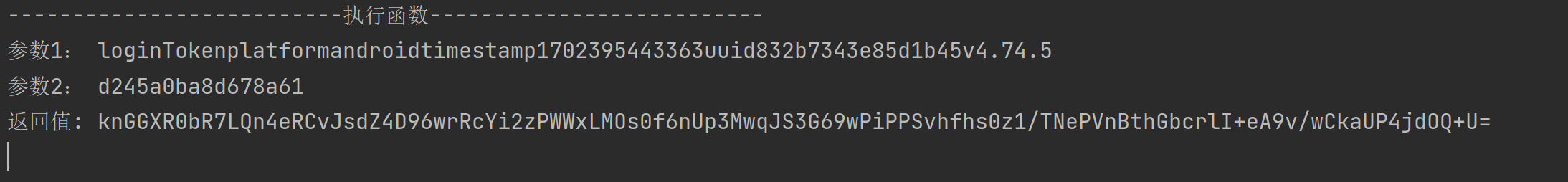
描述已自动生成**

**在这里可以看到是aes加密,加密模式为ecb,**

**图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成**

**我们就hook这个函数,** **d245a0ba8d678a61是密钥**

****

**显然参数2是密钥,参数1是要加密的字符串,显然经过aes加密之后进行base编码**

**至此,sign的加密还差下图几个字段**

****

**图片包含 文本

描述已自动生成**

**文本

描述已自动生成**

**Hook encode函数,我们发现上面几个参数都不变,至此,newSign部分逆向完成**

**下面开始逆向X-Auth-Token**

**文本

描述已自动生成**

**图片包含 文本

描述已自动生成**

**发现是抽象函数,没有实现.其实我们应该可以猜测是向服务器请求得到token**

**我们观察清理缓存之后的请求**

**图形用户界面, 应用程序, Word

描述已自动生成发现token在响应头中,至此逆向完成**