【悦跑圈】app签名算法逆向

2018年09月22日 17:54:31 sumousguo 阅读数：133 标签： 爬虫 逆向 安卓 xposed 更多

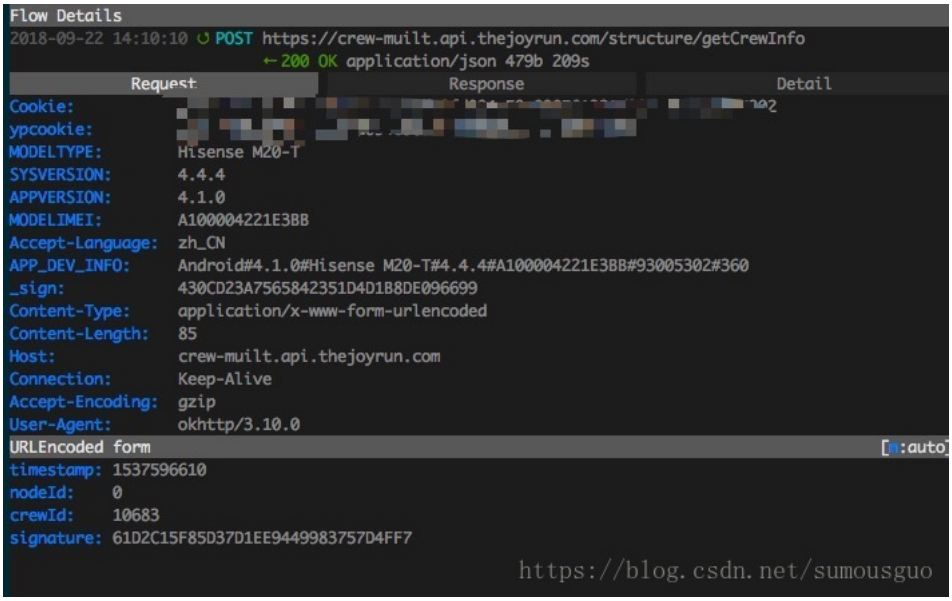
悦跑圈app签名算法逆向

悦跑圈

悦跑圈（http://www.thejoyrun.com/ ）
版本：安卓 4.1.0
下载链接: https://pan.baidu.com/s/1J9O6G4J1-fCmQug7hMC8FQ 提取码: nx6j

0x1 抓包分析

app抓包是常见的了，悦跑圈的app通信是https的，所以需要安装证书。这里我们用了mitmproxy作为抓包工具。
抓包如下：



这里可以看到，有两个签名参数，一个是_sign，另外一个是signature，我试着不带这两个参数，发现报错了。
因此，要想采集悦跑圈的数据，就必须逆向签名。从格式来看，加密算法猜测是md5的可能性比较大，也可能是sha1。

0x2 逆向dex

大部分小众的app，只要把classes.dex逆向出来，静态分析就能解决签名算法的逆向问题：

1. 把apk后缀改成zip，用解压软件打开，可以看到classes.dex;
2. 使用dex2jar把dex逆向成jar包， `./d2j-dex2jar.sh --force ../co.runner.app_4100/classes.dex` ，此时生成了classes-dex2jar.jar；
3. 用jd-gui打开我们对classes-dex2jar.jar包，就可以看到安卓java部分的源码了；

```
public native String getSignature(String paramString1, String paramString2, String paramString3);
public native String getSignatureV2(String paramString1, String paramString2, String paramString3);
```

0x3 逆向安卓so

现在我们知道，有关键词是getSignature，因此在co.runner.app_4100/lib/armeabi-v7a下找到我们要的so，就用grep去搜索查找字符串，结果得到

```
Downloads $ grep getSignature co.runner.app_4100/lib/armeabi-v7a/*
Binary file co.runner.app_4100/lib/armeabi-v7a/libjoyrun.so matches
```

通过文件名joyrun也能很明显得到这个是悦跑圈程序员写的c库，并且包含getSignature字符串。
这时候祭出我们的大器ida。

```
int __fastcall Java_co_runner_app_jni_NativeToolImpl_getSignature(int a1, int a2, int a3, int a4, int a5)
{
  return getSign(a1, a5, a3, a4, a5, "1fd6e28fd158406995f77727b35bf20a");
}
```



```
int __fastcall Java_co_runner_app_jni_NativeToolImpl_getSignatureV2(int a1, int a2, int a3, int a4, int a5)
{
  return getSign(a1, a5, a3, a4, a5, "9CCFDE35C19775C9A15D91E7C19BD4A9");
}
```


Java_co_runner_app_jni_NativeToolImpl_getSignatureV2和Java_co_runner_app_jni_NativeToolImpl_getSignature这两个明显的方法就是我们要找的
都调用getSign方法，只是有个32位的密钥不一样而已。

这里只要分析getSign方法，就可以得到全部加密算法了。具体的分析过程，就不详述了，提醒一点就是：getSign最后调用了java中co/runner/app/jni/
方法。getSign的流程，可以参考文章简单的使用jni调用java方法

加密算法：md5(paramString1 + saltKey + paramString2 + paramString3)，这里的saltKey是getSignature和getSignatureV2中的固定盐，在上面ida
到。

0x4 xposed hook技术

到目前为止，我们已经拿到了加密算法和加密中用到的盐，即将就接近终点了。但问题来了，paramString1、paramString2和paramString3这三个分
我们抓包的参数都不是固定参数个数的。
静态代码分析已经解决不了我们的困惑了，因此必须要上动态分析。
这里不详述怎么去做安卓的hook，具体可以看参考资料。 只讲几点：

- hook co.runner.jni.NativeToolImpl.class中等getSignature和getSignatureV2方法，把paramString1、paramString2和paramString3都log出来。
- log结果样例：
  paramString1 = crew_node_id0crewid10683limit_cnt3timestamp1537596610
  paramString2 = 93005302
  paramString3 = 7895e5723ccdc2fd094e50a898561321

0x5 参考代码

相关xposed代码和采集代码，均提交github

0x6 参考资料

1. 简单的使用jni调用java方法
2. Android Studio创建Xposed Module
3. Android Studio Xposed模块编写
4. anti_sign