拿到apk，国陆惯例，抓取请求

POST http://mobile.zztv.tv/kapp/m\_login.php?appid=39&appkey=O5PaX031BTwDbY6H6b5KuSxYFVygui89&client\_id\_android=a891d3737aa370941af5988e059e98be&device\_token=7f886d67b630fd32139daf28130862ae&\_member\_id=&version=1.0.4&app\_version=1.0.4&app\_version=1.0.4&package\_name=com.hoge.android.kanzhengzhou&system\_version=6.0&phone\_models=Nexus5 HTTP/1.1

X-API-TIMESTAMP: 1536742899112asJQht

X-API-SIGNATURE: Y2Y2MWE2N2MxNTYxYzM5ZjJjZjVmZDZlMTRhMjc3MzliYzY2Y2EzYQ==

User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; Nexus 5 Build/MRA58K) m2oSmartCity\_266 1.0.0

X-API-VERSION: 1.0.4

X-AUTH-TYPE: sha1

X-API-KEY: f7664060cc52bc6f3d620bcedc94a4b6

Content-Type: application/x-www-form-urlencoded

Content-Length: 129

Host: mobile.zztv.tv

Connection: Keep-Alive

Accept-Encoding: gzip

Cookie: SERVERID=fcbf3d63324e16d517214e160848ff32|1536742868|1536742798

client\_secret=111111&password=a123456&client\_id=111111&client\_id\_android=a891d3737aa370941af5988e059e98be&member\_name=18826645888

使用jadx反编译apk,搜索X-API-SIGNATURE



看到协议头中的时间戳X-API-TIMESTAMP，使用的方法

SecurityUtil.getRandomData(6); public static String[] chars\_array = new String[]{"0", "1", "2", "3", "4", "5", Constants.VIA\_SHARE\_TYPE\_INFO, "7", Constants.VIA\_SHARE\_TYPE\_PUBLISHVIDEO, "9", g.al, "b", "c", g.am, "e", "f", "g", "h", g.aq, "j", "k", "l", "m", "n", "o", "p", "q", "r", g.ap, "t", "u", MttTokenProvider.URL\_PARAM\_ENCRYPT\_VERSION, "w", "x", "y", "z", "A", "B", "C", QLog.TAG\_REPORTLEVEL\_DEVELOPER, QLog.TAG\_REPORTLEVEL\_USER, "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", NDEFRecord.TEXT\_WELL\_KNOWN\_TYPE, NDEFRecord.URI\_WELL\_KNOWN\_TYPE, "V", QLog.TAG\_REPORTLEVEL\_COLORUSER, "X", "Y", "Z"};

public static String getRandomData(int num) {

long microsecond = System.currentTimeMillis();

StringBuilder charsbuilder = new StringBuilder();

for (int i = 0; i < num; i++) {

charsbuilder.append(chars\_array[(int) (Math.random() \* ((double) chars\_array.length))]);

}

return microsecond + charsbuilder.toString();

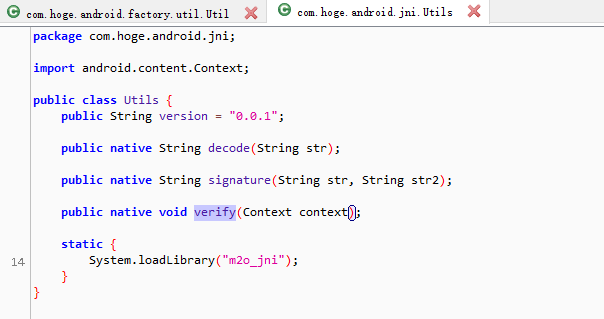
}

可以看出时间戳是在后面加上了6位的随机字符串

OK，回过头继续看协议头中的sign



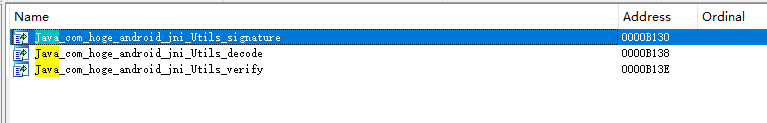
可以看到，它是反射调用了com.hoge.android.jni.Utils类中的signature方法，OK，去类中看一看



可以看到signature是native方法，在m2o\_jni这个so文件中

好，使用ida看看这个so

在ida中，查看导出函数表，搜索java



看到它是静态注册的verify方法，跟过去，并F5转换伪C看看



可以看到，它使用了sprintf函数进行了数据的拼装，再调用了sha1方法

但在函数的顶部，还有一条判断语句

if ( !is\_initialised )

exit(1);

下面并没有用到这个初始化的值，为了方便自己写个同包名apk调用测试这个so，还是把这个判断干掉吧



OK，改掉之后，自己写个同包名的apk，进行调用调试，查看它sha1之前的数据拼装

对了，通过分析，可以知道它带入signature的两个参数是，版本号和协议头中的时间戳

那么本次带入的两个参数就是

1.0.4

1536742899112asJQht

写apk调用修改过的so



打包apk，把修改过的so推送到这个包下的lib目标下



把抓取请求中的sign进行base64解码

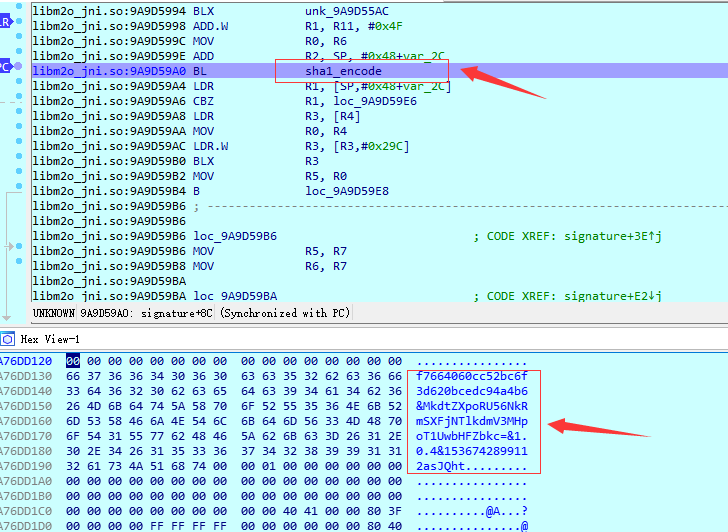
Y2Y2MWE2N2MxNTYxYzM5ZjJjZjVmZDZlMTRhMjc3MzliYzY2Y2EzYQ==

cf61a67c1561c39f2cf5fd6e14a27739bc66ca3a

可以看到与Toast弹出的结果一致

好了，数据对上，现在就开始调试

Ida调试so的过程就不写了，网上多的是，直接看结果



可以看到带入sha1的原文就是

f7664060cc52bc6f3d620bcedc94a4b6&MkdtZXpoRU56NkRmSXFjNTlkdmV3MHpoT1UwbHFZbkc=&1.0.4&1536742899112asJQht

本次学习结束，请用于学习用途，一切其它搞事，与我无关！