记一次frida实战——对某视频APP的脱壳、hook破解、模拟抓包、协议分析一条龙服务

看雪学院 前天

以下文章来源于三叶草小组Syclover ， 作者0x指纹

本文为看雪论坛精华文章

看雪论坛作者ID：0x指纹

一、前言

frida 是一个十分强大的工具，已经学习它有一段时间了，但也只是零零碎碎的练习与使用。

最近在对一个 APP 进行分析的过程中，使用 frida 完成了脱壳、hook 破解、模拟抓包、协议分析的操作，可以说是一条龙服务了， 感觉十分有意义，学到了很多，对 frida 的理解和掌握程度也提高了不少，记录下来这次实战分享给各位正在学习 frida 的看雪用户。（在看雪上论坛水了这么久，也该反馈些东西了，逃

frida 入门这里就不多说了，论坛已经有很多优秀的入门帖子了，我也是看着这些帖子一点一点学习的。不过提一下我在安装时候踩的坑，当时折腾很久也安装不上心态被搞得有点爆炸。

开始 python3.7 直接 pip install frida 和 pip install frida-tools一直卡在 Running setup.py install for frida ... – 了，最后的解决好的办法是到 https://pypi.org/project/frida/#files 下载 frida-xx.x.xx-py3.7-win-amd64.egg，并把它放到安装的python目录的 \Python37\Lib\site-packages 中。

然后找到对应的 frida-tools 版本pip3 install frida-tools执行即可安装，如果不对应，执行命令可能会把frida-xx.x.xx-py3.7-win-amd64.egg删掉又卡在Running setup.py install for frida ... –地方，可以根据发布日期来判断相应的 frida-tools 版本。

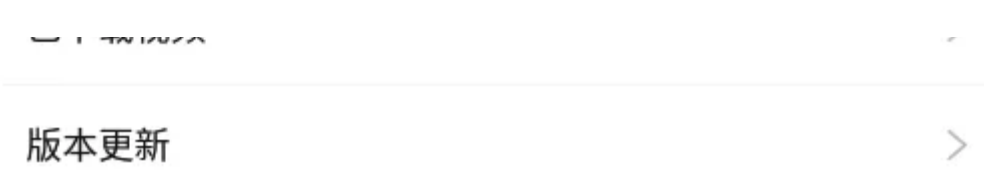
然后 python3.7 安装 easy_install，执行easy_install frida-xx.x.xx-py3.7-win-amd64.egg，即可在 python 中 import frida了。

二、App简单分析

这是一个视频播放的 APP，里面有着各种卫视和CCTV的在线播放，其他栏中是一些新闻栏目和电影。

这些都是免费播放的，但是我们注意到了右上角的“积分：0”字样，说明情况不简单，我们点开个人栏，查看发现“在线吧”、“在线吧2”里面需要积分消费才能进去。


个人
积分:0
获取积分
播放器
通知
设置
分享
电影轮播
腾讯视频
在线吧
在线吧2
成人台
自定义
已下载视频



而成人台里面又有一系列的栏目，每个进去都需要积分消费，点进去后会发现每个栏目里面都有一堆不可描述、胡里花哨、不符合核心价值观的影片或者直播分类，说明这是一个邪恶的APP。


成人台
直播
图片区新区1
图片区新区2
免费小说区
新区
新区2
新区3
新区4
在线视频1
在线视频2
在线视频3

在获取积分里面，可以购买 Vip 和 积分。

然后经过简单测试发现加了 360 壳，并且使用 Charles抓不到包，我们先使用 firda 脱壳拿到 DEX 文件，抓包问题后面再解决。

三、脱壳

既然是 frida 的一条龙服务，我们尝试用 frida 来进行脱壳，这里我们直接使用 frida-unpack（https://github.com/dstmath/frida-unpack）。

>>>> 1、关于frida-unpack

firda-unpack 原理是利用frida hook libart.so中的OpenMemory方法，拿到内存中dex的地址，计算出dex文件的大小，从内存中将dex导出，我们可以查看项目中的 OpenMemory.js 文件中的代码更清晰直观地了解。

```
'use strict';
/**
 * 此脚本在以下环境测试通过
```

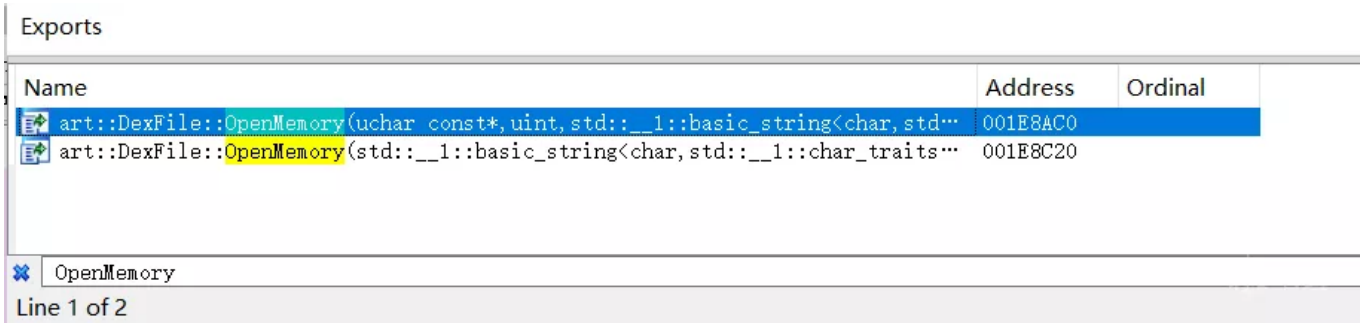
```
 * android os: 7.1.2 32bit  (64位可能要改OpenMemory的签名)
 * legu: libshella-2.8.so
 * 360:libjiagu.so
 */
Interceptor.attach(Module.findExportByName("libart.so",
"_ZN3art7DexFile10OpenMemoryEPKhjRKNSt3__112basic_stringIcNS3_11char_traitsIcEENS3_9allocatorIcEEEEjPNS_6MemMapEPKNS_10OatDexFileEPS9_"), {
    onEnter: function (args) {

        //dex起始位置
        var begin = args[1]
        //打印magic
        console.log("magic : " + Memory.readUtf8String(begin))
        //dex fileSize 地址
        var address = parseInt(begin,16) + 0x20
        //dex 大小
        var dex_size = Memory.readInt(ptr(address))

        console.log("dex_size :" + dex_size)
        //dump dex 到/data/data/pkg/目录下
        var file = new File("/data/data/xxx.xxx.xxx/" + dex_size + ".dex", "wb")
        file.write(Memory.readByteArray(begin, dex_size))
        file.flush()
        file.close()
    },
    onLeave: function (retval) {
        if (retval.toInt32() > 0) {
            /* do something */
        }
```

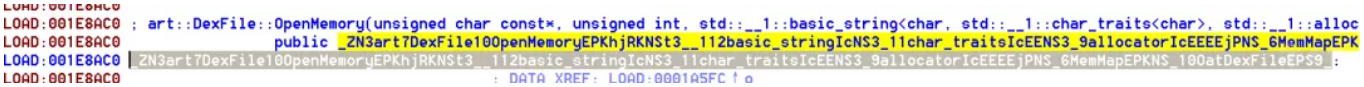
```
        }
    });
```

>>>> 2、frida-unpack使用报错及解决方案

使用

firda-unpack 的使用方法在项目的 README.md 中，其中查看OpenMemory的导出名称，我们在 /system/lib 中找到 libart.so 后，还可以拖进 IDA 然后在 Exports 窗口搜索到后点击查看。





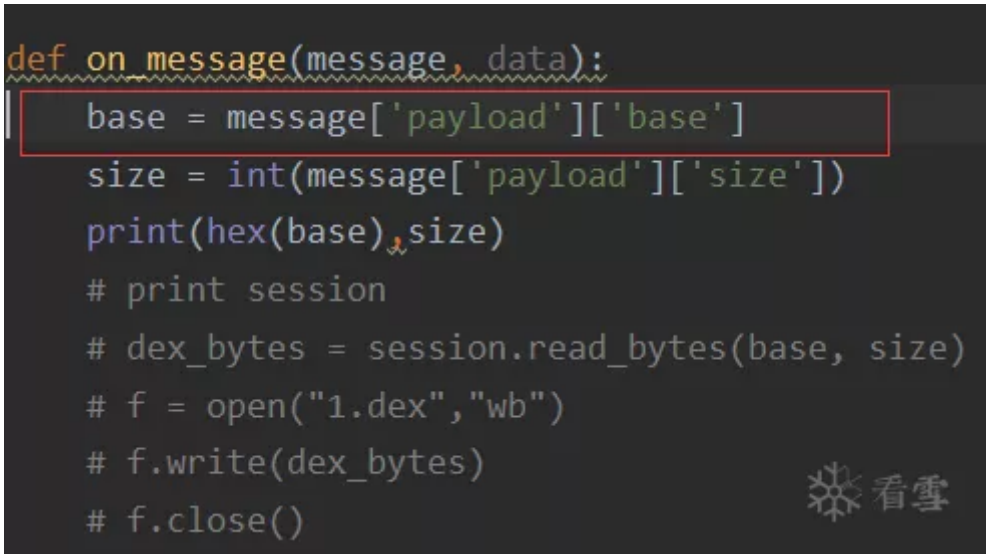
虽然怎么操作在项目中的 README.md 写得十分简单易懂，如果上手就能直接脱壳最好，但是我在使用的时候还是出现了错误。。

这里详细说下我的碰到的报错和解决方案。

KeyError 报错

首先我直接运行出现的错误是 KeyError: 'payload'，查看错误的代码行。

```
dex 导出目录为: /data/data/com.cz.babySister
Traceback (most recent call last):
  File "D:\Computer Science and Technology\IDE\Programming Environments\Python37\lib\site-packages\frida-12.6.11-py3.7-win-amd64.egg\frida\core.py", line 298, in _on_message
    callback(message, data)
  File "D:/Computer Science and Technology/Programming Code/PyCharm/frida-unpack-master1/frida_unpack.py", line 7, in on_message
    base = message['payload']['base']
KeyError: 'payload'
```



报错提示没有 'payload'这个 key，我们加上一句 print(mesaage) 把 message 值打印出来，发现 message 是一个字典，{'type': 'error', 'description': "TypeError: cannot read property 'readU8' of null", 'stack': "TypeError: cannot read property 'readU8' of null\n at [anon] (../../../frida-gum/bindings/gumjs/duktape.c:56648)\n at frida/runtime/core.js:386\n at /script1.js:29", 'fileName': '/_frida.js', 'lineNumber': 1480, 'columnNumber': 1}，里面确实没有 'payload' 这个key，并且可以看到类型是'error'。

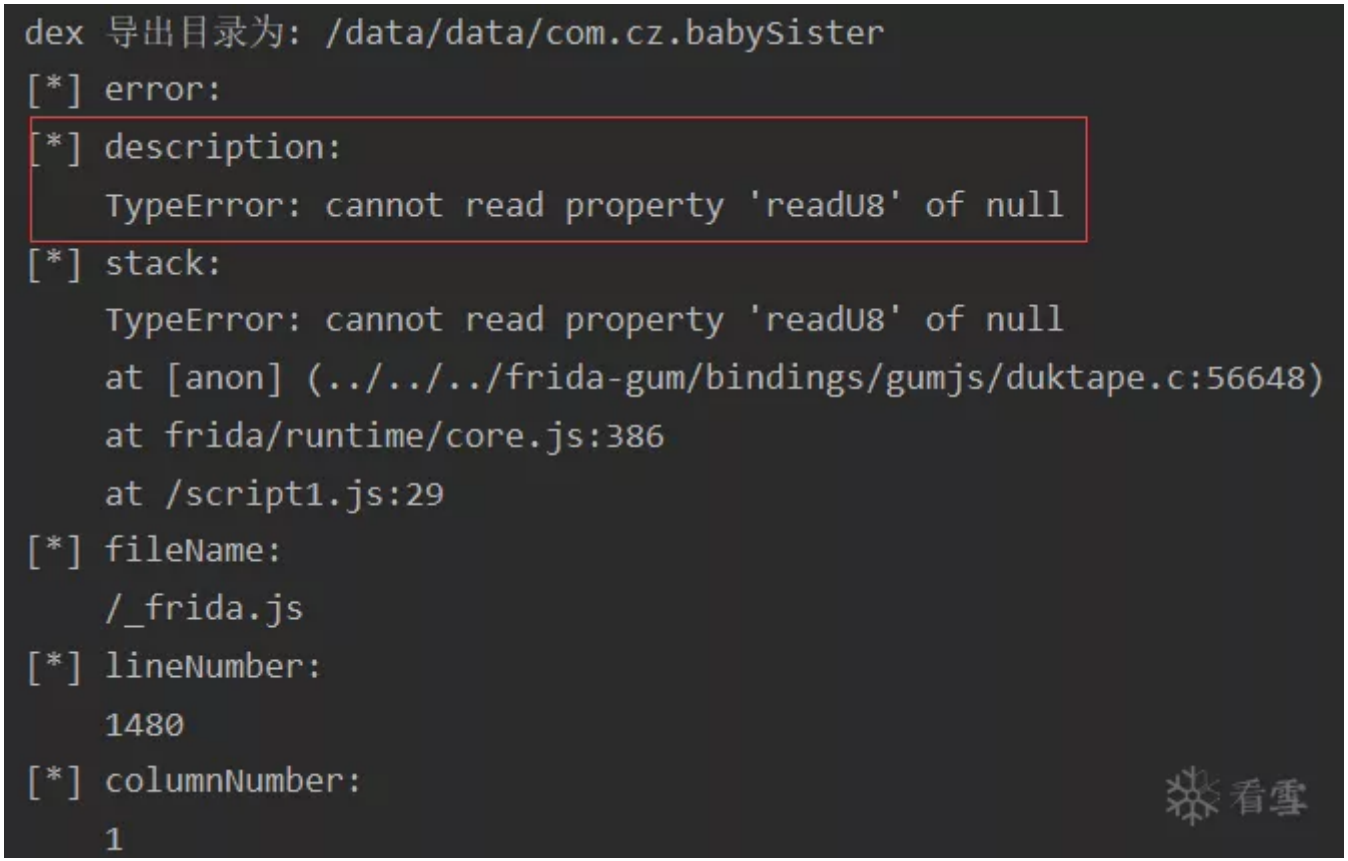
我们重新写一个 on_message 即可。

```python
def on_message(message, data):
    if message['type'] == 'send':
        base = message['payload']['base']
        size = int(message['payload']['size'])
        print(hex(base), size)
```

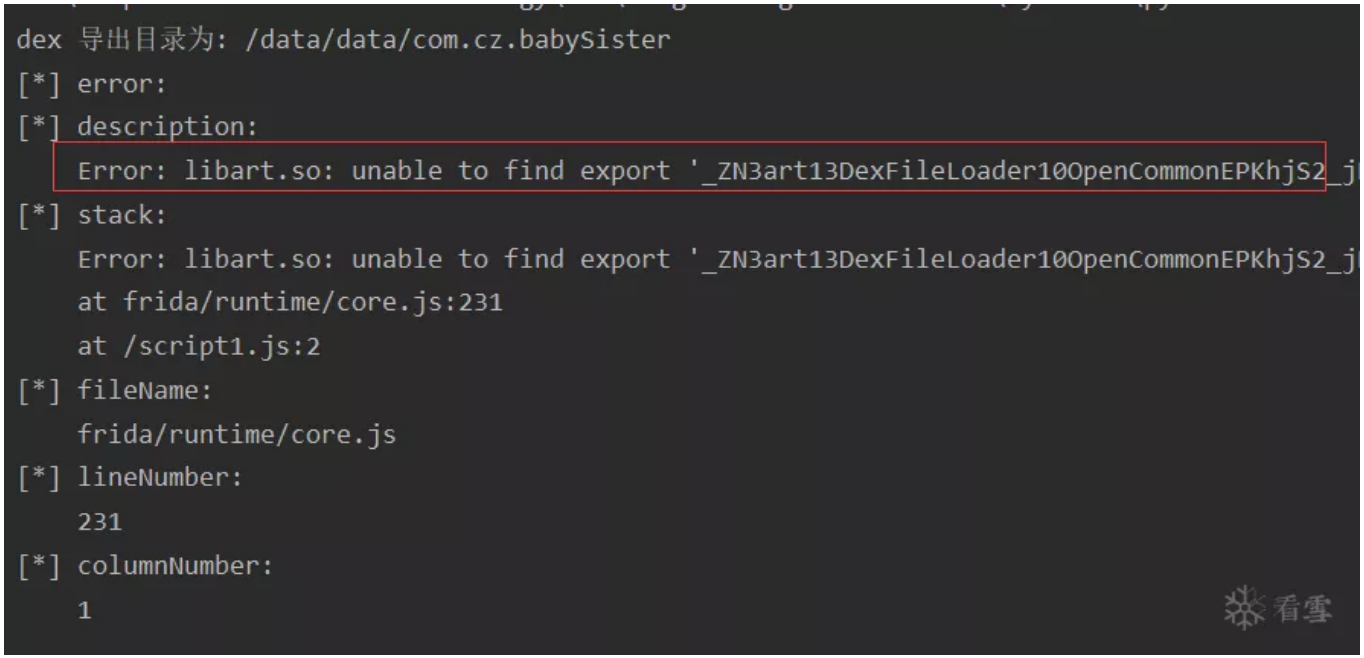
```
    elif message['type'] == 'error':
        for i in message:
            if i == "type":
                print("[*] %s" % "error:")
                continue
            if type(message[i]) is str:
                print("[*] %s" %
                      i + ":\n    {0}".format(message[i].replace('\t', '    ')))
            else:
                print("[*] %s" %
                      i + ":\n    {0}".format(message[i]))
    else:
        print(message)
```

readU8 报错

运行发现这个报错的问题解决了，但是又出现了新的问题。



不清楚是为什么，而且报错行数是 js 代码的最后一行，对报错进行搜索在 https://bbs.pediy.com/thread-250815.htm 这个帖子里面看到一个大佬说用Module.getExportByName替换Module.findExportByName就会得到具体的报错原因了，尝试在 js 代码中替换发现果然有更具体的报错原因了，感谢大佬。

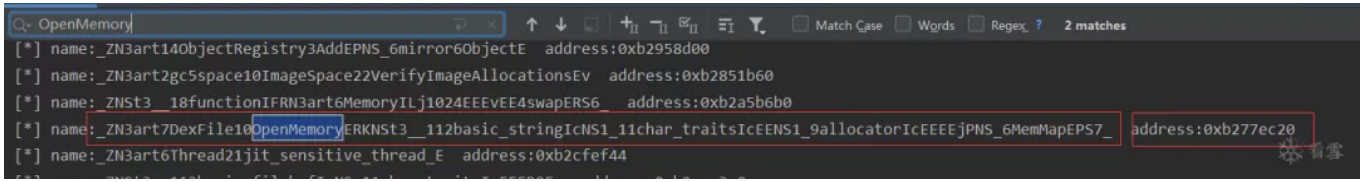


可以看到错误是因为在 libart.so 中不能通过 OpenMemory 的导出函数名找到它，这个十分奇怪，我们再写个 frida 脚本把内存中 liart.so 的导出函数名和地址都打印出来看看有没有 OpenMemory的。

```
import frida
import sys

jscode = """

  var exports = Module.enumerateExportsSync("libart.so");
    for(var i=0;i<exports.length;i++){
        send("name:"+exports[i].name+"  address:"+exports[i].address);
     }

"""
```

```
12
13 def on_message(message, data):
14     if message['type'] == 'send':
15         print("[*] {0}".format(message['payload']))
16
17 process = frida.get_usb_device().attach("com.cz.babySister")
18 script = process.create_script(jscode)
19 script.on('message', on_message)
20 script.load()
21 sys.stdin.read()
```

我们发现是有的，这就十分让人迷惑了，只能想想别的办法了解决。



修改代码

我们可以看到是由 OpenMemory 函数的地址的，因此想到尝试下直接 hook 这个地址会怎么样，需要先使用 new NativePointer 转换一下，试了是可行的，十分感动。

这里直接放出修改后的 OpenMemory.js 代码。

```
1 src = """
2 var exports = Module.enumerateExportsSync("libart.so");
```

```
    for(var i=0;i<exports.length;i++){
        if(exports[i].name ==
"_ZN3art7DexFile10OpenMemoryEPKhjRKNSt3__112basic_stringIcNS3_11char_traitsIcEENS3_9allocatorIcEEEEjPNS_6MemMapEPKNS_10OatDe
xFileEPS9_"){
            var openMemory = new NativePointer(exports[i].address);
            }
     }

Interceptor.attach(openMemory, {
    onEnter: function (args) {

        var begin = args[1]

        console.log("magic : " + Memory.readUtf8String(begin))

        var address = parseInt(begin,16) + 0x20

        var dex_size = Memory.readInt(ptr(address))

        console.log("dex_size :" + dex_size)

        var file = new File("/data/data/%s/" + dex_size + ".dex", "wb")
        file.write(Memory.readByteArray(begin, dex_size))
        file.flush()
        file.close()

        var send_data = {}
        send_data.base = parseInt(begin,16)
        send_data.size = dex_size
```

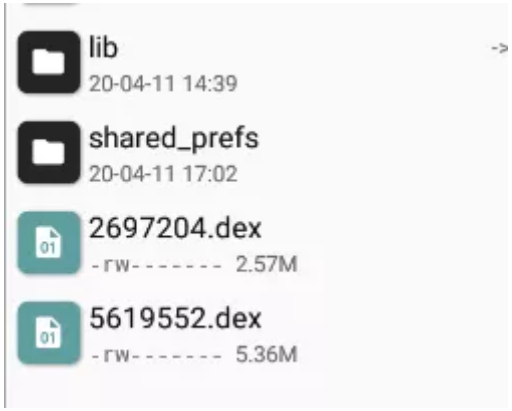
```
            send(send_data)
        },
        onLeave: function (retval) {
            if (retval.toInt32() > 0) {
            }
        }
    });
    """%(package)
```

>>>> 3、顺利脱壳

我们重新运行，非常激动地发现 frida-unpack 运行成功了。

```
dex 导出目录为: /data/data/com.cz.babySister
magic : dex
035
dex_size :2697204
0x9a6e5000 2697204
magic : dex
035
dex_size :5619552
0x99887000 5619552
magic : dex
035
dex_size :2697204
0x995f4000 2697204
```

去对应的文件目录查看，可以看到被脱掉后的两个 dex 文件。



都拖进 JEB 中看下，发现 APP 的逻辑代码都在 5619552.dex 中了，至此第一步脱壳完成。

四、Hook进行破解

有了反编译的代码后相信各位破解积分和 vip 充值是手到擒来了，不然都不好意思在看雪吱声了，这里只简单说下我一般的处理思路和方法，如果不行就要具体情况具体分析了。

我通常都会先在有使用到用户数据的 activity 中查看下代码，看有没有导入 UserInfo 这种类，然后如果有的话直接找到这个类，积分、vip、到期时间等等这些一般都会有相应的 getXxx（） 和 setXxx（）方法，然后就可以随意 hook了。

>>>> 1、积分

直接上相应的 js 代码，逻辑就是 hook 住 UserInfo 类中 getJifen（）方法，然后随便设置返回积分。

```
var userinfo = Java.use("com.cz.babySister.javabean.UserInfo");
userinfo.getJifen.implementation  = function(){
    return "100000";
}
```

值得一提的是，hook 后在测试过程中发现本地修改的数据会上传到服务器，因为发现换一个模拟器登录账号积分还是那么多，这个在后面的模拟抓包中会具体看下怎么回事。

>>>> 2、vip

破解 vip 时候发现尝试 hook getIsvip（） 和 setIsvip（）没有效果，猜测应该是内购之后向服务器进行请求成为 vip 用户，然后服务器返回自己账号 vip 用户数据，我们现在换下思路破解 vip，就是经常看到的破解内购。

破解内购应该也都是看雪用户的起手水平了，这个分析过程也不多说了，直接上 js 代码。

```
var pay = Java.use("com.cz.babySister.alipay.q");pay.b.implementation = function(){    return "9000"}
```

逻辑就是把支付时候下面的代码的支付失败流程变成支付成功流程，然后就会向服务器发送购买类型的请求。

```java
package com.cz.babySister.activity;

import android.os.Handler;
import android.os.Message;
import android.widget.Toast;
import com.cz.babySister.alipay.q;
import com.cz.babySister.utils.ParseJson;

class g extends Handler {
    g(BuyVipActivity arg1) {
        this.a = arg1;
        super();
    }

    public void handleMessage(Message arg3) {
        if(arg3.what != 1) {
        }
        else {
            q v0 = new q(arg3.obj);
            String v3 = v0.a();
            String v0_1 = v0.b();
            ParseJson.pasePay(v3);
            if(v0_1.equals("9000")) {
                Toast.makeText(this.a, "支付成功", 0).show();
                BuyVipActivity.a(this.a, BuyVipActivity.a(this.a), "");
            }
            else {
                Toast.makeText(this.a, "支付失败", 0).show();
            }
        }
    }
}
```

这样 hook 之后购买积分也是可以的。

>>>> 3、android_id

android_id 是在设备首次启动时，系统随机生成的一个64位的数字，并把这个数字以16进制字符串的形式保存下来，这个16进制的字符串就是android_id，当设备被wipe后该值会被重置。

但是为我们什么要 hook 这个呢，是因为因为我测试了两个账号都被封了。。没错，就是被封了，而且被封后我发现重新注册的号就不能看那些不可描述的视频和直播了，但是换一个模拟器登录新注册的账号就又可以看了。

因为使用的模拟器配置了 frida 和 Charles 环境及安装了别的分析工具，不想再换个模拟器重新配置了，就找了下为什么不能看了。在写出模拟抓包代码打印出注册账号的请求后，发现了原因，注册请求上传的一个参数是 memi1，找到对应 Java 代码赋值处溯源了下发现是获取的是 android_id，当账号被封后带有这个用户注册所用安卓机的 android_id 的请求都不会被处理，这也就是为什么重新注册账号后不能再看不可描述的东西了。

```
private String f() {
    String v1_1;
    String v0 = "0";
    try {
        v1_1 = Settings$Secure.getString(((Activity)this).getContentResolver(), "android_id");
        if(v1_1 != null) {
            if("".equals(v1_1)) {
                return v0;
            }

            return v1_1;
        }

        return v0;
    }
    catch(Exception v1) {
        v1.printStackTrace();
        return v0;
    }

    return v1_1;
}
```

这个时候 hook 了一下返回 android_id 字符串的函数，随便改一下，然后这个模拟器就能重新用了。

直接上 js 代码，要提一下的就是，Secure 类是 android.provider.Settings 的一个内部类，我们要 hook 的 getString（）方法在 Secure 类中，hook 类时候写成“ Java.use("android.provider.Settings$Secure"就不会报错。

```
var sec = Java.use("android.provider.Settings$Secure");
sec.getString.implementation = function(arg1,arg2){
    return "5c80b60fc1f73307";
}
```

运行之后，模拟器登录后就又可以重新看了。

>>>> 5.2、模拟抓包

使用 Charles 抓不到包怎么办呢，我们有 frida！可以通过 frida 来 hook 住 APP 构造网络请求和接收数据地方的代码，然后打印出来请求和返回数据，这样 APP 向服务器进行的网络请求和接收的数据便一览无余了。

一般网络请求和接收数据的代码都会写在一个类中，我们只要找到一个点来追踪去找到这个类就可以了。

我们将 LoginActivity 作为这个点，也就是登录界面的 activity 的代码中找到获取用户账号和密码的地方，然后通过 JEB 的交叉索引功能进行分析追踪，可以找到网络请求的地方都在一个类中，并且请求的方法有三个。

```java
public static String a(String arg2, String arg3) {
    try {
        URLConnection v2_1 = new URL(arg2).openConnection(Proxy.NO_PROXY);
        ((HttpURLConnection)v2_1).setRequestMethod("POST");
        ((HttpURLConnection)v2_1).setDoOutput(true);
        ((HttpURLConnection)v2_1).setDoInput(true);
        PrintWriter v0 = new PrintWriter(((HttpURLConnection)v2_1).getOutputStream());
        v0.write(arg3);
        v0.flush();
        ((HttpURLConnection)v2_1).connect();
        InputStream v2_2 = ((HttpURLConnection)v2_1).getInputStream();
        arg3 = a.a(v2_2);
        v2_2.close();
        return arg3;
    }
    catch(Exception v2) {
        v2.printStackTrace();
        return null;
    }
}
```

```java
public static String a(String arg1) {
    try {
        URLConnection v1_1 = new URL(arg1).openConnection(Proxy.NO_PROXY);
        ((HttpURLConnection)v1_1).connect();
        InputStream v1_2 = ((HttpURLConnection)v1_1).getInputStream();
        String v0 = a.a(v1_2);
        v1_2.close();
        return v0;
    }
    catch(Exception v1) {
        v1.printStackTrace();
        return null;
    }
}
```

```java
public static String b(String arg6) {
    KeyManager[] v0 = null;
    try {
        b v1 = new b();
        com.cz.babySister.c.a$a v2 = new com.cz.babySister.c.a$a();
        SSLContext v3 = SSLContext.getInstance("TLS");
        v3.init(v0, new X509TrustManager[]{v1}, new SecureRandom());
        HttpsURLConnection.setDefaultSSLSocketFactory(v3.getSocketFactory());
        HttpsURLConnection.setDefaultHostnameVerifier(((HostnameVerifier)v2));
        URLConnection v6_1 = new URL(arg6).openConnection(Proxy.NO_PROXY);
        ((HttpURLConnection)v6_1).connect();
        InputStream v6_2 = ((HttpURLConnection)v6_1).getInputStream();
        String v1_1 = a.a(v6_2);
        v6_2.close();
        return v1_1;
    }
    catch(Exception v6) {
        v6.printStackTrace();
        return ((String)v0);
    }
}
```

可以看到都使用了openConnection(Proxy.NO_PROXY)，Charles 当然抓不到包，每个方法传入的参数即是网络请求，返回的参数是接收的数据，然后我们通过 frida 来 hook 住这三个方法打印出来。

直接放上 js 代码。

```javascript
Java.perform(function(){

    /*
    * 模拟抓包
    */

    var client = Java.use("com.cz.babySister.c.a")
    client.a.overload("java.lang.String","java.lang.String").implementation = function(arg1,arg2){
        send("抓包**********************************************")
```

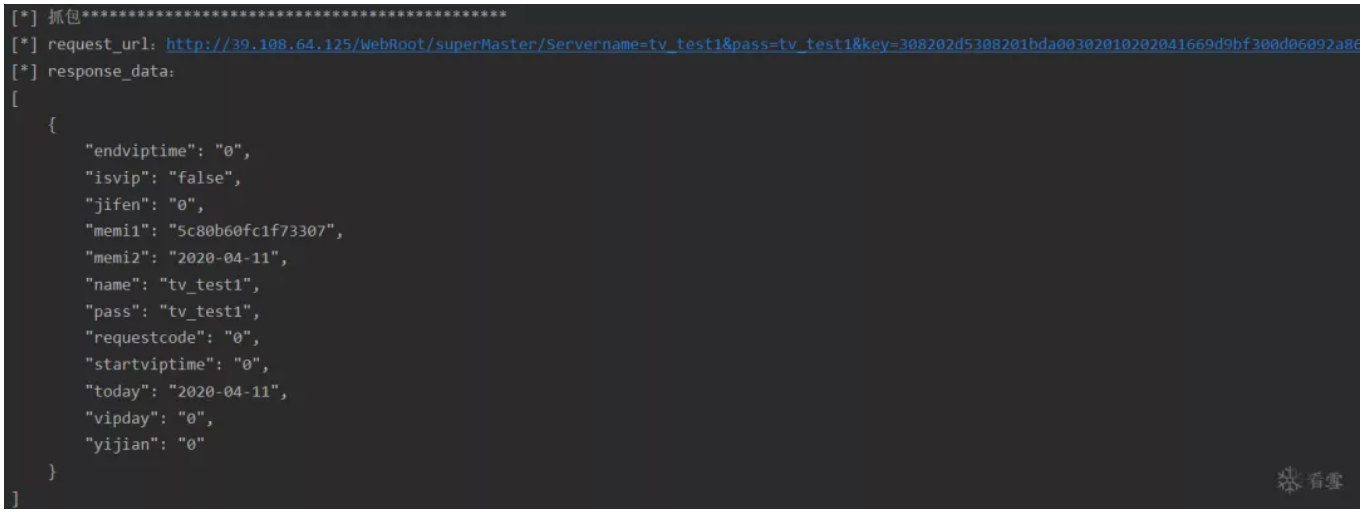
```
        send("request_url: "+arg1+arg2);
        var response_data1 = this.a(arg1,arg2);
        send("response_data: ");
        send(response_data1)
        return response_data1;
    }


    client.a.overload("java.lang.String").implementation = function(arg1){
        send("抓包**********************************************")
        send("request_url: "+arg1);
        var response_data2 = this.a(arg1);
        send("response_data: ");
        send(response_data2)
        return response_data2;
    }

     client.b.overload("java.lang.String").implementation = function(arg1){
        send("抓包**********************************************")
        send("request_url: "+arg1);
        var response_data3 = this.b(arg1);
        send("response_data: ");
        send(response_data3)
        return response_data3;
    }
```

需要提下的是有些返回的是一行 json 数据，我们可以在 on_message 函数里面解析一下把它优雅地打印出来，还有就是有些 json 解析会出错，on_message 函数定义如下。

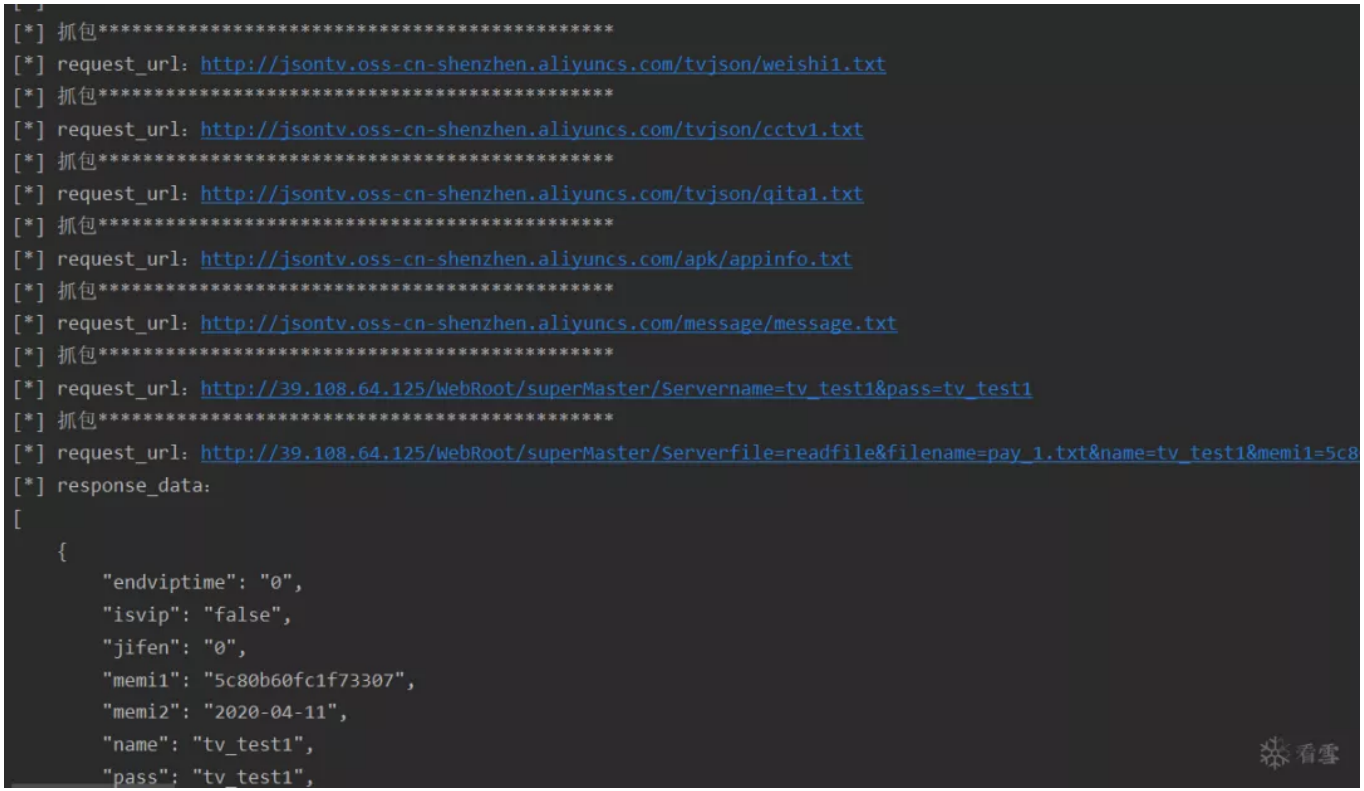
```
def on_message(message, data):
    if message['type'] == 'send':
        try:
           print(json.dumps(json.loads(message['payload'].encode('utf8')), sort_keys=True, indent=4, separators=(', ', ': '), ensure_ascii=False))
        except:
            print("[*] {0}".format(message['payload']))


    elif message['type'] == 'error':
        for i in message:
            if i == "type":
                print("[*] %s" % "error:")
                continue
            if type(message[i]) is str:
                print("[*] %s" %
                      i + ":\n    {0}".format(message[i].replace('\t', '    ')))
            else:
                print("[*] %s" %
                      i + ":\n    {0}".format(message[i]))
    else:
        print(message)
```

我们运行 frida 脚本，然后登录账号查看下效果。

这是登录时的网络请求和返回数据：



登录后 APP 初始化过程中又会进行一些网络请求来接收各大卫视和栏目的资源信息，以及关于 APP 的信息。

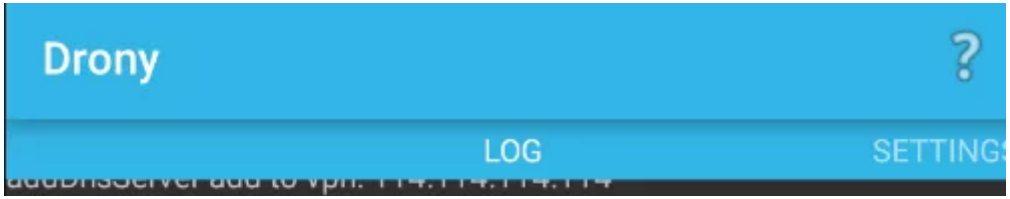

[*] 抓包**********
[*] request_url: http://jsontv.oss-cn-shenzhen.aliyuncs.com/tvjson/weishi1.txt
[*] 抓包**********
[*] request_url: http://jsontv.oss-cn-shenzhen.aliyuncs.com/tvjson/cctv1.txt
[*] 抓包**********
[*] request_url: http://jsontv.oss-cn-shenzhen.aliyuncs.com/tvjson/qita1.txt
[*] 抓包**********
[*] request_url: http://jsontv.oss-cn-shenzhen.aliyuncs.com/apk/appinfo.txt
[*] 抓包**********
[*] request_url: http://jsontv.oss-cn-shenzhen.aliyuncs.com/message/message.txt
[*] 抓包**********
[*] request_url: http://39.108.64.125/WebRoot/superMaster/Servername=tv_test1&pass=tv_test1
[*] 抓包**********
[*] request_url: http://39.108.64.125/WebRoot/superMaster/Serverfile=readfile&filename=pay_1.txt&name=tv_test1&memi1=5c8
[*] response_data:
[
{
"endviptime": "0",
"isvip": "false",
"jifen": "0",
"memi1": "5c80b60fc1f73307",
"memi2": "2020-04-11",
"name": "tv_test1",
"pass": "tv_test1",
看雪

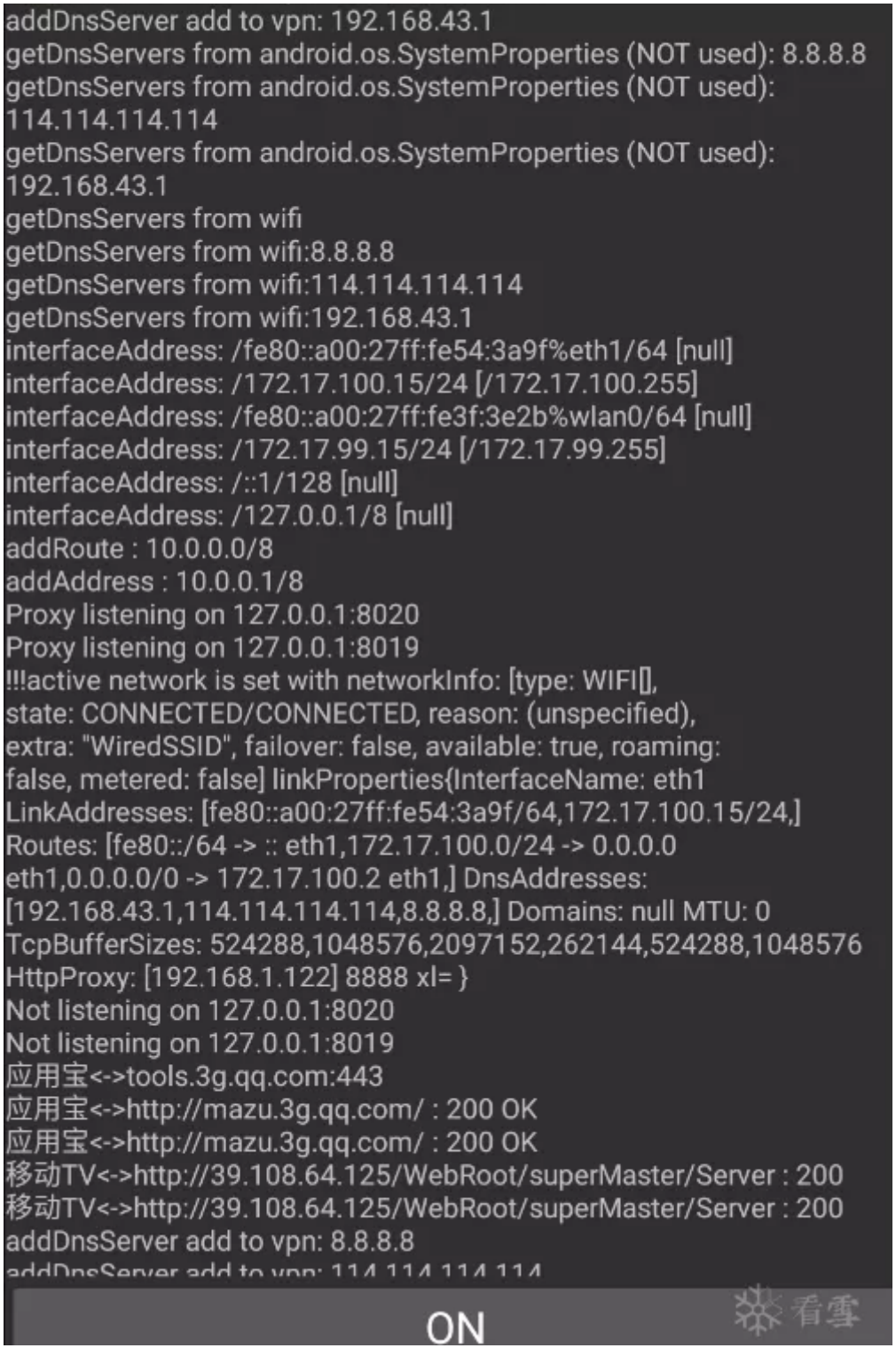
```
[*] request_url: http://39.108.64.125/WebRoot/superMaster/Servername=tv_test1&pass=tv_test1&key=308202d5308
[*] 抓包****************************************
[*] request_url: http://39.108.64.125/WebRoot/superMaster/Serverfile=readfile&filename=pay_1.txt&name=tv_te
[*] response_data:
{
    "describe": "替换新区1资源,无法更新或者更新后无法正常观看的请联系QQ
    "force": "0",
    "ischenge": "false",
    "url": "http://jsontv.oss-cn-shenzhen.aliyuncs.com/apk/movetv.apk",
    "version": "14.3"
}
[*] response_data:
[
    {
        "img": "http://jsontv.oss-cn-shenzhen.aliyuncs.com/icon/cctv.png",
        "name": "CCTV1综合",
        "url": "http://cctvalih5ca.v.myalicdn.com/live/cctv1_2/index.m3u8"
    },
    {
        "img": "http://jsontv.oss-cn-shenzhen.aliyuncs.com/icon/cctv.png",
        "name": "CCTV2-财经",
        "url": "http://cctvalih5ca.v.myalicdn.com/live/cctv2_2/index.m3u8"
    },
    {
        "img": "http://jsontv.oss-cn-shenzhen.aliyuncs.com/icon/cctv.png",
```


>>>> 5.3、使用Drony配合Charles抓包

我们可以看到 frida 完美地模拟了抓包分析的效果，就算抓不到包我们也不怕了。

其实针对由于 openConnection(Proxy.NO_PROXY) 引起的抓不到包问题，我们可以通过使用 Drony 来配合 Charles 来抓，Charles 和 Drony 配置的具体操作可以参考这两篇文章，https://www.jianshu.com/p/1d0360e50a01 和https://www.jianshu.com/p/75b3ad732183。




addDnsServer add to vpn: 192.168.43.1
getDnsServers from android.os.SystemProperties (NOT used): 8.8.8.8
getDnsServers from android.os.SystemProperties (NOT used):
114.114.114.114
getDnsServers from android.os.SystemProperties (NOT used):
192.168.43.1
getDnsServers from wifi
getDnsServers from wifi:8.8.8.8
getDnsServers from wifi:114.114.114.114
getDnsServers from wifi:192.168.43.1
interfaceAddress: /fe80::a00:27ff:fe54:3a9f%eth1/64 [null]
interfaceAddress: /172.17.100.15/24 [/172.17.100.255]
interfaceAddress: /fe80::a00:27ff:fe3f:3e2b%wlan0/64 [null]
interfaceAddress: /172.17.99.15/24 [/172.17.99.255]
interfaceAddress: /::1/128 [null]
interfaceAddress: /127.0.0.1/8 [null]
addRoute : 10.0.0.0/8
addAddress : 10.0.0.1/8
Proxy listening on 127.0.0.1:8020
Proxy listening on 127.0.0.1:8019
!!!active network is set with networkInfo: [type: WIFI[],
state: CONNECTED/CONNECTED, reason: (unspecified),
extra: "WiredSSID", failover: false, available: true, roaming:
false, metered: false] linkProperties{InterfaceName: eth1
LinkAddresses: [fe80::a00:27ff:fe54:3a9f/64,172.17.100.15/24,]
Routes: [fe80::/64 -> :: eth1,172.17.100.0/24 -> 0.0.0.0
eth1,0.0.0.0/0 -> 172.17.100.2 eth1,] DnsAddresses:
[192.168.43.1,114.114.114.114,8.8.8.8,] Domains: null MTU: 0
TcpBufferSizes: 524288,1048576,2097152,262144,524288,1048576
HttpProxy: [192.168.1.122] 8888 xl= }
Not listening on 127.0.0.1:8020
Not listening on 127.0.0.1:8019
应用宝<->tools.3g.qq.com:443
应用宝<->http://mazu.3g.qq.com/ : 200 OK
应用宝<->http://mazu.3g.qq.com/ : 200 OK
移动TV<->http://39.108.64.125/WebRoot/superMaster/Server : 200
移动TV<->http://39.108.64.125/WebRoot/superMaster/Server : 200
addDnsServer add to vpn: 8.8.8.8
ON
看雪

Drony 使用的话就是把最下面状态由 OFF 点击一下切换到 ON 就开始运行打印 Log了，我们向左滑就可以切换到 Setting 页面，配置好后开始抓包，我们来看一下 Charles 的效果。

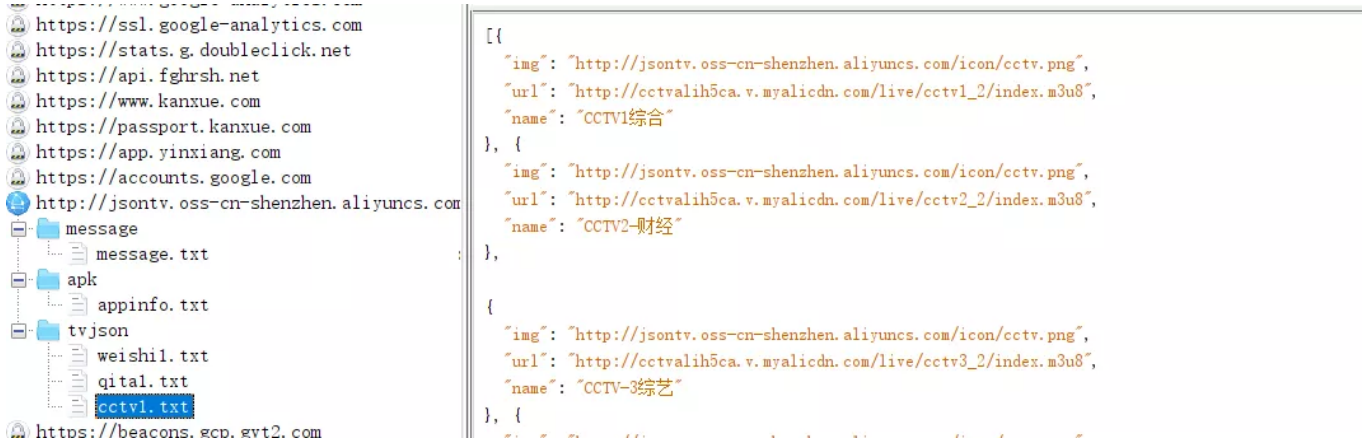
抓到的登录请求和返回数据的包：

Overview | Contents | Summary | Chart | Notes

Name	Value
name	tv_test1
pass	tv_test1
key	308202d5308201bda003020102020416 69d9bf300d06092a864886f70d(
rightkey	376035775
memi1	5c80b60fc1f73306
login	login

Headers | Text | Hex | Form | Raw


```
[{
  "memi2": "2020-04-11",
  "memi1": "5c80b60fc1f73307",
  "pass": "tv_test1",
  "jifen": "0",
  "today": "2020-04-11",
  "name": "tv_test1",
  "vipday": "0",
  "startviptime": "0",
  "endviptime": "0",
  "isvip": "false",
  "requestcode": "0",
  "yijian": "0"
}]
```



经过分析可以看到我们使用 frida 来模拟抓包的效果十分好。

>>>> 5.4、frida Hook 完整代码

```
import frida
import sys
import json

```

```
jscode = """
function log(){
    var Log = Java.use("android.util.Log");
    var Throwable = Java.use("java.lang.Throwable");
    console.log(Log.getStackTraceString(Throwable.$new()));
}


Java.perform(function(){

    /*
    * 模拟抓包
    */
    var client = Java.use("com.cz.babySister.c.a")
    client.a.overload("java.lang.String","java.lang.String").implementation = function(arg1,arg2){
        send("抓包**********************************************")
        send("request_url: "+arg1+arg2);
        var response_data1 = this.a(arg1,arg2);
        send("response_data: ");
        send(response_data1)
        return response_data1;
    }

    client.a.overload("java.lang.String").implementation = function(arg1){
        send("抓包**********************************************")
        send("request_url: "+arg1);
        var response_data2 = this.a(arg1);
        send("response_data: ");
        send(response_data2)
```

```
        return response_data2;
    }

     client.b.overload("java.lang.String").implementation = function(arg1){
        send("抓包**********************************************")
        send("request_url: "+arg1);
        var response_data3 = this.b(arg1);
        send("response_data: ");
        send(response_data3)
        return response_data3;
    }


    /*
    * hook UserInfo修改积分，积分修改之后消费一次积分会上传到服务器更新
    */
    var userinfo = Java.use("com.cz.babySister.javabean.UserInfo");
    userinfo.getJifen.implementation  = function(){
        return "100000";
    }


    /*
    * hook修改返回值，支付失败变成成功
    */
    var pay = Java.use("com.cz.babySister.alipay.q");
    pay.b.implementation = function(){
        return "9000"
    }
```

```
    /*
    * 修改vip会出现封号，服务器除了禁账号也会禁android_id，hook修改android_id
    */
    var sec = Java.use("android.provider.Settings$Secure")
    sec.getString.implementation = function(arg1,arg2){
        return "9774d56d682e549a"
    }

});
"""


def on_message(message, data):
    if message['type'] == 'send':
        try:
           print(json.dumps(json.loads(message['payload'].encode('utf8')), sort_keys=True, indent=4, separators=(', ', ':
'), ensure_ascii=False))
        except:
            print("[*] {0}".format(message['payload']))


    elif message['type'] == 'error':
        for i in message:
            if i == "type":
                print("[*] %s" % "error:")
                continue
            if type(message[i]) is str:
```

```
                print("[*] %s" %
                      i + ":\n    {0}".format(message[i].replace('\t', '    ')))
            else:
                print("[*] %s" %
                      i + ":\n    {0}".format(message[i]))
    else:
        print(message)


process = frida.get_usb_device().attach('com.cz.babySister')
script = process.create_script(jscode)
script.on('message', on_message)
script.load()
sys.stdin.read()
```

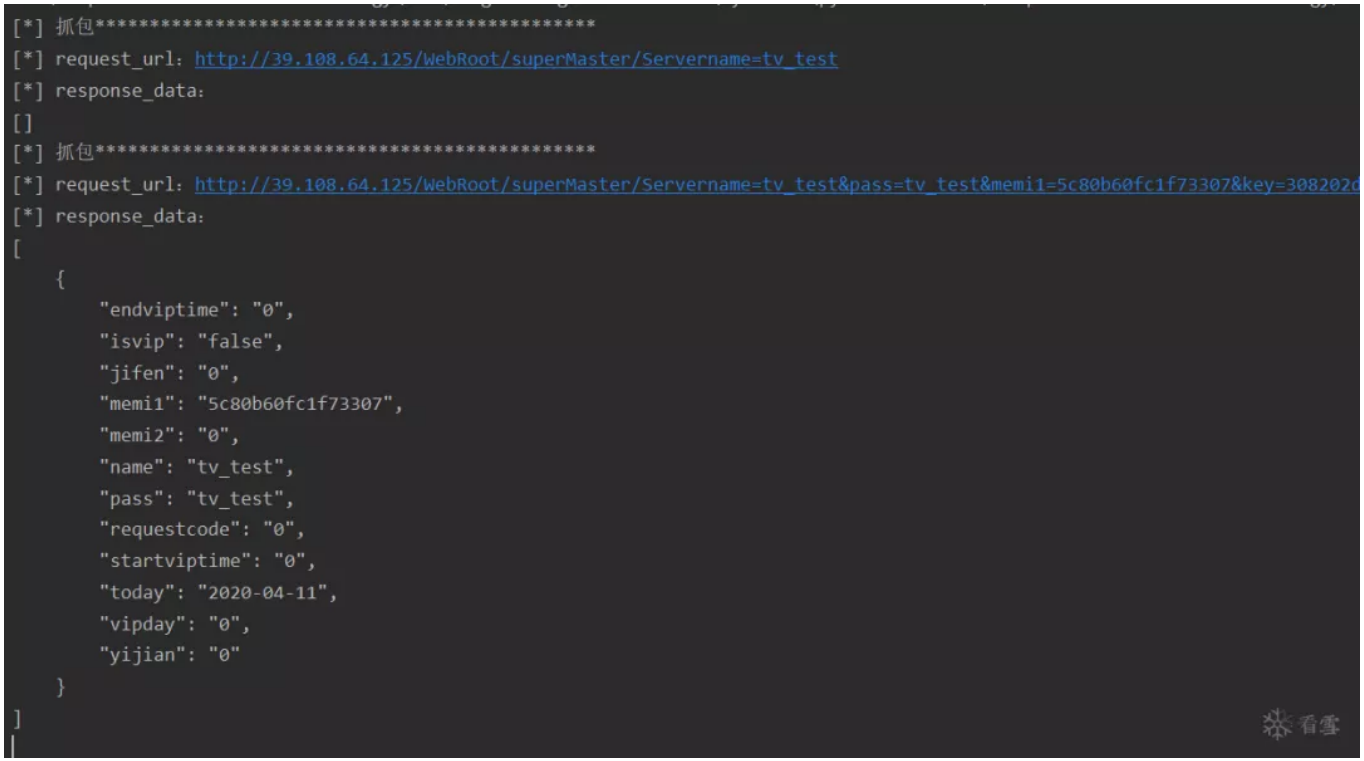
六、协议分析

当我们使用 frida 能模拟抓到清楚地看到网络请求后，协议分析也便不再话下了。一般的思路是抓包查看网络请求参数，然后在反编译的代码中搜索字符串定位相关代码，继而分析协议。

这里我们简单地分析下注册账号、登录账号、更新积分、注册 vip 的协议。

>>>> 1、注册帐号

我们注册一个账号并抓包，注册成功后查看。



我们可以看到两条网络请求，注册请求就是第二条，我们分析一下参数有 name、pass、memi1、key、rightkey，我们可以找到 apk 构造请求地方的代码。

```
public static String a(Context arg6, String arg7) {
    String v0 = null;
    try {
        String v1 = arg6.getSharedPreferences("userInfo", 0).getString("name", v0);
        String v2 = Settings$Secure.getString(arg6.getContentResolver(), "android_id");
        String v3 = a.a(arg6);
        String v6_1 = a.b(arg6);
        StringBuilder v4 = new StringBuilder();
        v4.append("file=readfile&filename=");
        v4.append(arg7);
        v4.append("&name=");
        v4.append(v1);
        v4.append("&memi1=");
        v4.append(v2);
        v4.append("&key=");
        v4.append(v3);
        v4.append("&rightkey=");
        v4.append(v6_1);
        return a.a("http://39.108.64.125/WebRoot/superMaster/Server", v4.toString());
    }
    catch(Exception v6) {
        v6.printStackTrace();
        return v0;
    }
}
```

前两个就是账号密码，memi1 我们前面在说过是 android_id，如果账号被封的话，这个也会一起被拉入黑名单，也就是这台设备不能再看不可描述的东西了。

key是什么呢，我们来看一下生成的代码，可以看出来是获取 APP 的签名，会在服务器进行验证，是对付修改 APP 二次打包的。

```
public static String a(Context arg4) {
    try {
        Signature[] v4_1 = arg4.getPackageManager().getPackageInfo(arg4.getPackageName(), 0x40).signatures;
        StringBuilder v0 = new StringBuilder();
        int v1 = v4_1.length;
        int v2;
        for(v2 = 0; v2 < v1; ++v2) {
            v0.append(v4_1[v2].toCharsString());
        }

        return v0.toString();
    }
    catch(PackageManager$NameNotFoundException v4) {
        v4.printStackTrace();
        return "";
    }
}
```

再看下 right_key 生成的代码，可以发现是获取公钥证书 X509Certificate 的序列号。

```
public static String b(Context arg5) {
    Iterator v0 = arg5.getPackageManager().getInstalledPackages(0x40).iterator();
    while(v0.hasNext()) {
        Object v1 = v0.next();
        if(!((PackageInfo)v1).packageName.equals(arg5.getPackageName())) {
            continue;
        }

        try {
            CertificateFactory v2 = CertificateFactory.getInstance("X.509");
            ByteArrayInputStream v3 = new ByteArrayInputStream(((PackageInfo)v1).signatures[0].toByteArray());
            Certificate v1_2 = v2.generateCertificate(((InputStream)v3));
            v3.close();
            String v5 = ((X509Certificate)v1_2).getSerialNumber().toString().trim();
            return v5;
        }
        catch(Exception v1_1) {
            v1_1.printStackTrace();
            continue;
        }
    }

    return "123";
}
```

可以发现注册账号的协议十分简单，前三个我们可以随便拟造，后两个是固定的。

>>>> 2、登录帐号

同理分析登录账号的请求，可以看到只是比注册请求多了一个 login 参数。

```
public void run() {
    SharedPreferences$Editor v9_1;
    String v0 = "date";
    String v1 = "name";
    String v2 = "http://39.108.64.125/WebRoot/superMaster/Server";
    String v3 = "name=";
    String v4 = "";
    long v5 = 0;
    try {
        String v7 = MainActivity.n(this.c);
        String v8 = this.c.b();
        String v9 = BaseActivity.rightkey(this.c);
        StringBuilder v10 = new StringBuilder();
        v10.append(v3);
        v10.append(this.a);
        v10.append("&pass=");
        v10.append(this.b);  // key
        v10.append("&key=");
        v10.append(v8);
        v10.append("&rightkey=");
        v10.append(v9);
        v10.append("&memi1=");
        v10.append(v7);
        v10.append("&login=login");
        v7 = a.a(v2, v10.toString());
        if(v7 != null && !v4.equals(v7)) {
            List v7_1 = ParseJson.parseRegisterName(v7);
            if(v7_1 != null && v7_1.size() > 0) {
                if(v7_1.get(0).getPass().equals(this.b)) {
                    v9_1 = MainActivity.l(this.c).edit();
                    v9_1.putLong("shengyuday", v5);
                    v9_1.putString(v1, v7_1.get(0).getName());
                    v9_1.putString(v1, v7_1.get(0).getName());
                    v1 = v7_1.get(0).getIsvip();
                    MyApplication.a = v7_1.get(0).getToday();
                    if("true".equals(v1)) {
                        MainActivity.l = true;
                        MyApplication.d = Integer.parseInt(v7_1.get(0).getVipday());
                    }
                    else {
                        a.a(v2, v3 + this.a + "&vip=vip");
                    }

                    v9_1.putString("pass", v7_1.get(0).getPass());
                    v1 = v7_1.get(0).getJifen();
                    if(v1 == null) {
                        goto label_100;
                    }

                    goto label_89;
                }
```

>>>> 3、更新积分

在进行积分消费的时候，观察到现有积分扣除后，会向服务器发送一次积分更新请求，抓包数据如下，发现有个 time 和 sign，猜测根据 time 生成 sign，然后会在服务器进行验证。

找到构造请求的代码处。

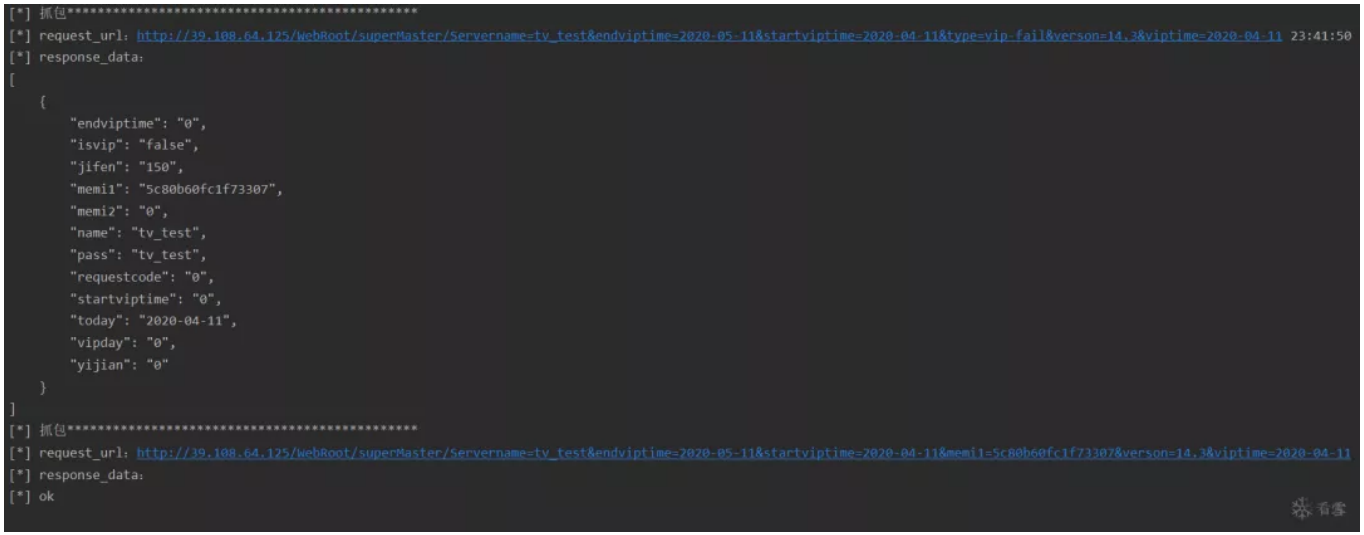
```
    public void run() {
        String v0 = "http://39.108.64.125/WebRoot/superMaster/Server";
        String v1 = "";
        String v2 = "name=";
        try {
            String v3 = BaseActivity.b(this.c).getString("name", null);
            String v4 = BaseActivity.b(this.c).getString("pass", null);
            StringBuilder v5 = new StringBuilder();
            v5.append(v2);
            v5.append(v3);
            v5.append("&pass=");
            v5.append(v4);
            int v4_1 = StringResource.queryJifen(v5.toString());
            if(v4_1 == -1) {
                return;
            }

            v4_1 -= this.a;
            long v5_1 = System.currentTimeMillis();
            StringBuilder v7 = new StringBuilder();
            v7.append(5 * v5_1);
            v7.append(v1);
            String v7_1 = BaseActivity.get_sign(v7.toString().getBytes());
            v2 = v2 + v3 + "&jifen=" + v4_1 + "&time=" + v5_1 + "&sign=" + v7_1;
            a.a(v0, v2);
            v0 = a.a(v0, v2);
            if(v0 == null) {
                return;
            }

            if(v1.equals(v0)) {
                return;
            }

            if(!v0.equals("ok")) {
                return;
            }

            BaseActivity.d(this.c).postDelayed(new b(this, v4_1), 0);
        }
        catch(Exception v0_1) {
            v0_1.printStackTrace();
        }
    }
}
```

可以看到 sign 是根据 time 生成的，我们通过交叉索引找到生成 sign 的算法，可以发现只是简单的 base64 加密。

```java
public static String a(byte[] 5time) {
    byte v0_1;
    int v15;
    if(5time == null) {
        return null;
    }

    int v1 = 8;
    int v0 = 5time.length * 8;
    if(v0 == 0) {
        return "";
    }

    int v2 = v0 % 24;
    v0 /= 24;
    int v3 = v2 != 0 ? v0 + 1 : v0;
    char[] v3_1 = new char[v3 * 4];
    int v4 = 0;
    int v5 = 0;
    int v6 = 0;
    while(v4 < v0) {
        int v7 = v5 + 1;
        v5 = 5time[v5];
        int v8 = v7 + 1;
        v7 = 5time[v7];
        int v9 = v8 + 1;
        v8 = 5time[v8];
        byte v10 = ((byte)(v7 & 15));
        byte v11 = ((byte)(v5 & 3));
        if((v5 & 0xFFFFFF80) == 0) {
            v5 >>= 2;
        }
        else {
            v5 = v5 >> 2 ^ 0xC0;
        }

        byte v5_1 = ((byte)v5);
        if((v7 & 0xFFFFFF80) == 0) {
            v7 >>= 4;
        }
        else {
            v7 = v7 >> 4 ^ 0xF0;
        }

        byte v7_1 = ((byte)v7);
        int v12 = (v8 & 0xFFFFFF80) == 0 ? v8 >> 6 : v8 >> 6 ^ 0xFC;
        int v13 = v6 + 1;
        char[] v14 = a.c;
        v3_1[v6] = v14[v5_1];
```

>>>> 4、注册 vip

上面我们提到怎么按破解内购的思路来 hook 改变支付逻辑，我们尝试一次并抓包。



可以看到第二个请求便是注册 vip，我们找到相应的代码处。

```java
public void run() {
    String v0 = "";
    int v1 = 6;
    try {
        String v2 = BuyVipActivity.e(this.d);
        String v3 = BuyVipActivity.c(MyApplication.a, this.a);
        String v4 = this.d.d();
        String v5 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.CHINA).format(new Date());
        String v6 = this.d.b();
        String v7 = BaseActivity.rightkey(this.d);
        StringBuilder v8 = new StringBuilder();
        v8.append("name=");
        v8.append(this.username);
        v8.append("&endviptime=");
        v8.append(v3);
        v8.append("&startviptime=");
        v8.append(MyApplication.a);
        v8.append("&memi1=");
        v8.append(v2);
        v8.append("&verson=");
        v8.append(v4);
        v8.append(this.c);
        v8.append("&viptime=");
        v8.append(v5);
        v8.append("&key=");
        v8.append(v6);
        v8.append("&rightkey=");
        v8.append(v7);
        v2 = a.a("http://39.108.64.125/WebRoot/superMaster/Server", v8.toString());
        if(v2 != null && !v0.equals(v2)) {
            if("ok".equals(v2)) {
                BuyVipActivity.g(this.d).postDelayed(new k(this), 0);
            }
            else {
                BuyVipActivity.i(this.d);
                if(BuyVipActivity.h(this.d) > v1) {
                    BuyVipActivity.a(this.d, this.a, v0);
                    BuyVipActivity.a(this.d, this.username, this.a);
                }
                else {
                    BuyVipActivity.a(this.d, BuyVipActivity.a(this.d));
                }
            }

            return;
        }
```

请求参数分别是 name、endviptime、startviptime、memi1、verson、viptime、key、rightkey，十分简单。

>>>> 5、写一份协议

然后很自然的，我们可以轻松地自己写一份简单地协议，尝试过程中发现 memi1 也就是 android_id 非常容易被封，需要不断更换。。以及尝试了很久写注册 vip 的请求都没有成功，可能在服务器还有别的验证。

```
import base64
import time

import requests
requests.packages.urllib3.disable_warnings()

class tv:
    def __init__(self):
        self.root = 'http://39.108.64.125/WebRoot/superMaster/Server'
        self.memi1 = "9774d56d682e549c"
        self.rightkey = "376035775"
        self.key =
"308202d5308201bda003020102020416 69d9bf300d06092a864886f70d01010b0500301b310b3009060355040613023836310c300a06035504031303776
569301e170d3136303731383038313935395a170d3431303731323038313935395a301b310b3009060355040613023836310c300a0603550403130377656
930820122300d06092a864886f70d01010105000382010f003082010a028201010095f85892400aae03ca4ed9dcd838d162290ae8dd51939aac6ecfde828
2f207c4cd9e507929a279e0a36f1e4847330cb53908c92915b2c6a93d7064be452d073a472093f7ca14f4ab68f827582fe0988e9e4bc8a6ea3b56001cbbb
b760f9eec571b0bbc97392e65aaf08c686f0e2ba353896d48a37c36716239977bd0e4dd878025cab497d8164537aec9f6599eefb98577dce972a1b794e21
1226520e23497beec3fd8548bb5b4d263120d40115cca28116bac32378df5033f536a0d7367fef78c587fefed28c5c9b35ba684ed6e46d9369c40950cf7a
d7236d10b7a51dfd2a8f218db72323bbd19f46947410b1191f263012ad4ba8f749223e37591254ee7f50203010001a321301f301d0603551d0e041604143
d43284bd5e4b0d322c9962a5b70aad4dcbc3634300d06092a864886f70d01010b05000382010100f04c51ff763311aa011777ba2842b441b15c316373d1
e1ed4116cf86e29d55c6ed3fa4c475251b1fb4fac57195dbca0166ebe565d9834552a3758b97c4528bab1f7ab82bb3a9faa932f5bc10943f3daf52e0fe58
89ffb58a6be67ea1c9a2fb37dc8aa6f3af476039a467336991a4e52dccd520195cd473eb5b984e702ed9ff638a14c3abb575a7a80ae4062084d1138a06a2
```

```
0e173be9df32df631311b07352898706198ddebaaa011f0da8e5f288f7cfb77505bc943f6476d6cc1feef56b68137aad91f23c4bb772169539d05653a6f0
d75f7192164e822b934322f3a975df677903b1667f5dc1e9ddb185da3281d31bfb8f67a84bd23bbcb398f8bb637dd72"

    def post(self, data=None):
        if data is None:
            data = {}
        return requests.post(url=self.root,data=data)

    def register(self, name, password):
        ret = self.post({'name': name, 'pass': password, 'memi1': self.memi1, 'key': self.key, 'rightkey': self.rightkey})
        print("Register response data: ")
        print(ret.content.decode('utf-8'))


    def login(self, name, password ):
        ret = self.post({'name': name, 'pass': password, 'memi1': self.memi1, 'key': self.key, 'rightkey': self.rightkey,
'login' : 'login'})
        print("Login response data: ")
        print(ret.content.decode('utf-8'))

    def updateSocre(self,name,password,jifen):
        t = int(round(time.time() * 1000))
        sign = base64.b64encode(str(5 * t).encode('utf-8')).decode('utf-8')
        ret = self.post({'name' : name, 'pass' : password, 'jifen' : jifen, 'time' : t, 'sign' : sign})
        print("UpdataScore response data: ")
        print(ret.content.decode('utf-8'))

if __name__ == "__main__":
    tv = tv()
```

```
# 注册账号
print(tv.register("mee4", "mee4"))

# 登录账号
print(tv.login("mee4","mee4"))

# 更新积分
print(tv.updateSocre("mee4","mee4","1000"))
```

七、结语

整个 frida 使用过程就是这样子，花时间整理不太容易，希望各位可以学到有用的东西能顺便点个赞就更好了（逃
更：评论区有问到 APP 要防止内购破解，可以做什么操作，简单写了些自己的看法，如果哪里有什么误解和偏见还请各位大佬指点下..

个人认为像这个 APP 虽然加了壳，也有根据 time 生成 sign 的算法然后把这两个值传到服务器去验证，也有获取 APP 的签名信息传到服务器防二次打包，还使用了防抓包的编程写法，但是都十分简单，一一解决起来十分容易，如果想防止内购破解，这些点都是必要的点，是都需要加强对抗的，根据木桶效应，一只水桶能装多少水取决于它最短的那块木板，APP 对抗同样也是。

首先是加的壳比较简单，dex 文件全部加载到内存中了轻松地可以 dump 出来，这样 APP 布置的第一道最重要的防线就没多少防护意义了，下面更是一败而溃了。

然后根据 time 生成 sign 的这么重要的算法不可放在 Java 层中去调用，不然反编译出来十分容易被分析出来利用构造协议，这个 APP 里面不仅放在 Java 层算法还只是简单的 base64 加密。逆过的一些防护优秀的 APP 是不加壳的，仅仅把根据 time 生成 sign 的算法放在 native 层进行对抗，分析起来难度十分大。

获取签名信息防二次打包这种重要的东西同样也是和生成 sign 算法一样不应该放在 Java 层的，应放在 native 层去对抗，以及获取签名的方式应该多样化，最好避免最常用的那种。

还有防抓包，感觉最好的解决方式还是对数据进行加密传输，加密算法放在 native 层。

还有反调试、防模拟器等等各种其他风险控制都要随着 APP 的防护需求去布置。

其实这样分析一下，很多时候不用对整个 APP 大动干戈地加壳，只需要对几个关键的点进行优雅地高强度对抗就能满足防护需求了。

- End -

看雪ID：0x指纹

https://bbs.pediy.com/user-802108.htm

*这里由看雪论坛 0x指纹 原创，转载请注明来自看雪社区。