華中科技大学

课 程 实 验 报 告

课程名称：移动应用安全

研究课题：安卓间谍应用研究

指导教师：王浩宇

报告日期：2022 年 11 月 18 日

网络空间安全学院

目录

1. 实验目的

远程管理工具（RAT）允许管理者远程访问安卓设备，当它被安装到安卓设备上后，使用者将能够通过计算机控制设备。通常，RAT 包括在管理机器上运行并监听特定 TCP/UDP 端口的服务端，以及充当服务器和客户端之间接口的客户端。

本实验将观察 RAT 应用各个家族的特定，并尝试使用静态分析、动态分析（主要是网络流分析）来自动化分析相关家族，旨在刻画其行为特征。

2. 实验环境和分工

2.1 实验环境

本实验主要在 Windows 系统中完成，并通过雷电模拟器 4 构建移动终端虚拟化环境。

- PC 操作系统：Windows10
- 手机型号：VIVO V1938T (x86)

在实验中还使用到了相关的网络流抓包工具和 API 监控工具，具体工具名称和版本如下：

- Fiddler Classic Latest
- Frida-server: 16.0.0-android-x86

2.2 小组成员信息

姓名	学号	班级
张博思	U201911157	网安 1903
宋静怡	U201912584	网安 1903
严子炜	U201812168	网安 1903
叶思源	U201916483	网安 1903

2.2 实验分工

本实验一共分为五大部分，分别是：资料收集、数据筛选与归纳、网络抓包环境搭建、网络抓包分析、API HOOK 分析。其中，由叶思源负责前期的资料收集，包括相关文献的收集以及工具的使用方法。数据筛选与归纳由宋静怡完成，她将从 1000 项数据中通过 avclass_family 归纳为 sandr、ahmyth 等几个类别。严子炜负责网络环境的搭建，配置雷电模拟器以及 Fiddler。而主要实验过程由张博

思完成，他负责网络抓包与 API HOOK 的大部分工作，最终的分析将由小组四人共同讨论完成。

3. 实验过程

3.1 背景调研

移动间谍应用使得管理员可以远程控制或监控安卓设备，这类应用通常会监听管理员机器上的 TCP/UDP 套接字以实现双方的通信。此外，该类应用会隐藏自己使用户无法检测到。

目前，主流的移动间谍应用包括：

1. FlexiSPY. 其官网简介是："世界上最具权威的电脑、手机和平板电脑的监控软件。不管你在哪，让计算机或智能手机上的一切仅在掌握"。该应用适用于所有设备，包括：Android 设备、iPhone 设备和计算机。其功能包括：提供家长控制软件、允许您跟踪员工的在线活动、无需麻烦的远程安装服务、跟踪用户登录/注销活动、远程卸载或停用软件、在隐藏模式下运行、阻止卸载软件、通过安全密钥组合访问、提供仪表板警报、从 Web 发送远程命令、自动远程更新。

2. Google Family Link. 其官网简介是："帮助家人养成良好的电子设备使用习惯。无论您的孩子尚且年幼还是已进入青少年阶段，您都可以使用 Family Link 应用设置数字设备基本使用规则，正确引导孩子在网络世界学习、玩乐和探索。"不过该应用只支持 Android 设备。

3. XNSPY. 其功能包括：在地图上检查孩子和员工的位置、记录并收听他们的电话录音、键盘记录器功能使您可以监视即时消息传递应用程序中的按键、允许您监视他们的所有电子邮件，并保留有关员工和孩子访问哪些网站的标签。其支持的设备包括：Android 设备和 iPhone 设备。

Android 操作系统 (OS) 是世界上使用最多的移动操作系统，占全球市场份额的 76.61% ，这使其成为网络犯罪分子的理想目标。目前，已经有恶意的移动间谍应用通过在 Google Play 商店上架来吸引用户下载，进而搜集用户的隐私数据。移动间谍应用已经成为犯罪分子收集用户隐私数据的有效方法之一。

研究人员 Abualola 等人开发了一种利用 Android 的 Notification Listener 服务功能的特洛伊木马间谍软件。该恶意应用被宣传为短信备份应用。然而，该应用程序带有一个后门，可以将通知内容从 WhatsApp、Facebook Messenger、BBM

和 SMS 转发到攻击者的电子邮件中。该应用程序能够通过使用两个权限来完成此操作："Notification Access"和"Internet"。除了研究人员制造的间谍应用外，还有通过 Google Play 商店进行的真实间谍软件攻击。2017 年，有恶意的开发者在 Google Play 商店成功上传了一个假的 WhatsApp 应用程序，名为"update WhatsApp Messenger"，这个假应用程序被下载了超过一百万次。安装后，该应用程序将下载另一个名为 whatsapp.apk，并且不会被 Google play protect 服务发现，使得用户的隐私信息暴露于风险之中。

目前，恶意软件的检测方法有多种，包括网络流量分析、基于应用行为的检测方法等等。例如：MADAM 是一种用于 Android 设备的新型基于主机的恶意软件检测系统，它同时分析和关联四个级别的特征：内核、应用程序、用户和程序包，以检测和阻止恶意行为。

因此，对移动间谍应用进行行为刻画非常重要，有利于研究人员根据其行为设计移动间谍应用的检测方法。

3.2 实验操作

3.2.1 前期准备

我们的数据集共有 1000 多项，并通过 avclass_family 归纳为 sandr、ahmyth 等几个类别，在实验中尝试手动通过模拟器随机安装具有代表性的不同类别的应用，但是有很大一部分应用无法正常安装，因此直接舍弃。

4	44381544c	a0f0959d8abdba324045d36b	ahmyth
5	1e8ca219b	bfb2fb30812bc8c39aafa2df2	ahmyth
6	23515d5cf	b210bd8b5fbe4d8f3771343cf	ahmyth
7	7335da33c	0c94d60a4260346b2716b7b5	sandr
8	f760f7480e	0b8be497f4606f9a62cd5bd12	androrat
9	0156f9590	676e7150f24693edf35a24c2d	sandr
10	7fc62d80d	b04e952eff71c698c03988aab	sandr
11	36f82d84b	19c8fcfffb9ae9059a3edd210b	sandr
12	0e12b6da7	f09677c83bfe499d9df59bc7cf	androrat
13	dd3d8081	e04873771d1291f792a3181f4	sandr
14	654f1e6ec	b6230da1a70b227d015c080f2	androrat
15	15076b749	99e139eb2a051cd1647570a72	sandr
16	7486712fa	c0da7b60f5d22a99a99449fbc	sandr
17	21542b43	ce89e4f53090aea77dd0d664b	sandr

图 3-1：RAT Apps 数据集

最终我们挑选出了几个有代表性的、并具有明显效果的应用进行抓包测试和后续的 API Hook 测试。

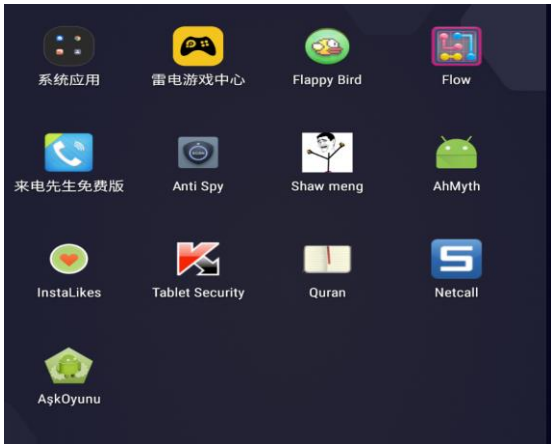

图 3-2：测试应用

我们针对的对象是间谍应用，其最明显的特征主要是通过 TCP/UDP 端口向远程建立连接发送消息，或者攻击者发送命令或消息以主动控制和获得用户手机的机密信息。

因此我们将重点放在对于网络流的抓包和 TCP/UDP 相关的网络 API 的 hook 检测上。

3.2.2 网络抓包分析

首先需要搭建网络抓包环境，如上所述，使用的是雷电模拟器和 Fiddler，需要配置相关代理，以使得模拟器手机的网络包经由 Fiddler。

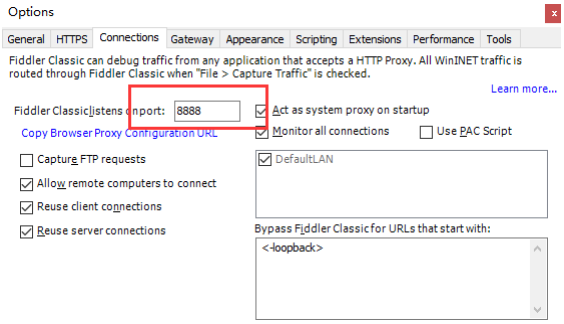

图 3-3： 代理设置

我们首先测试了几个游戏应用，发现并没有其他的异常网络请求，甚至基本就没有网络请求。因此无法判断是否包含间谍工具，或者需要特地的触发条件，但是由于时间有限，没有包含到。

图 3-4： 游戏应用-抓包情况 1

随后我们又测试了几个应用，发现了一些异常网络请求，并且方式各不相同。对于名称为“Shaw meng”的应用，见图 2，我们发现点击程序后，发送了 url 为 /userfiles/shaw-vaar2.json 的 Get 请求，但是 Response 的 error code 为 301 即页面

发送了永久移动。可以猜测该恶意程序可能是试图访问某个恶意用户配置文件。

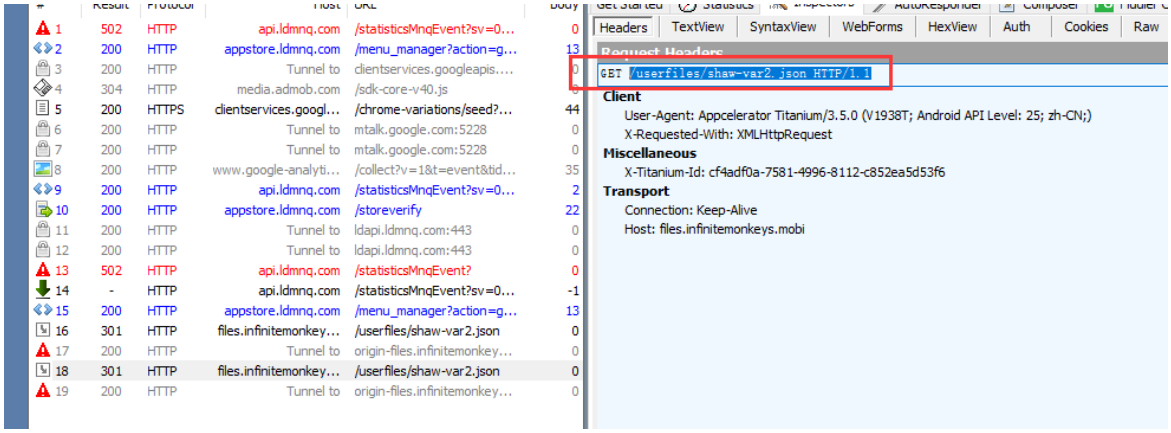

图 3-5： 抓包情况 2

对于名称为“AhMyth”的应用，抓包的情况又不相同，我们发现点击应用后，虽然程序直接退出了，但是能抓到一直尝试与 IP 地址为 192.168.1.62，端口号为 5050 建立连接，但是由于实验的 PC 处于校园网中，显然该 IP 也不会暴露端口，所以连接一直失败。我们猜测如果攻击者处于同一个局域网，他可以部署一个服务器使得其他机器能够建立连接，并持续监听受害者的信息，然后向其发送消息。

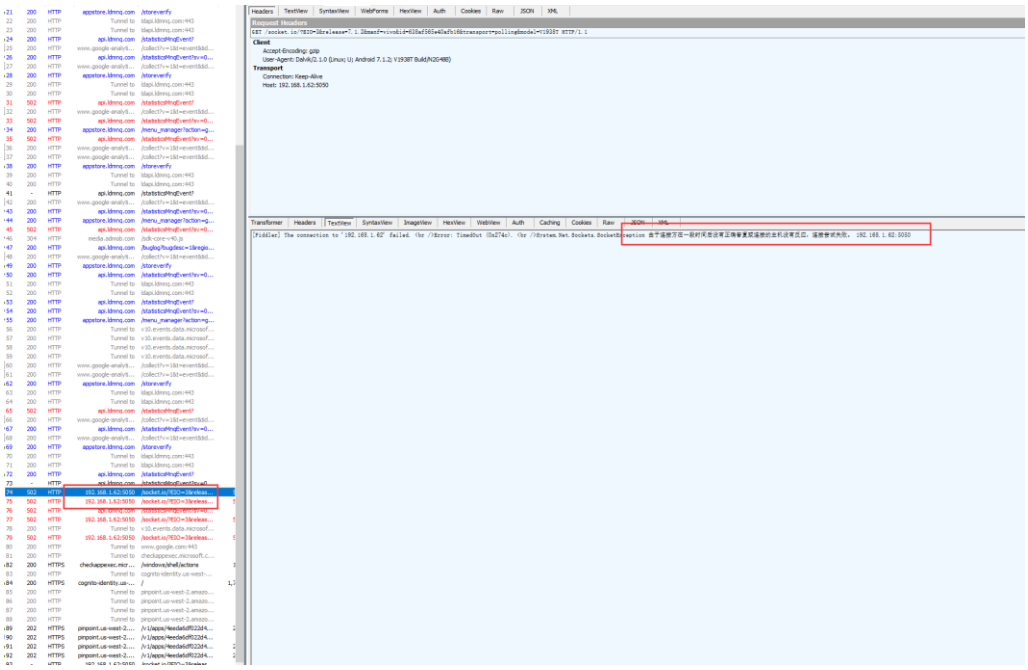

图 3-6：抓包情况 3

在实验中，我们还发现另一种情况，即点击程序后，会朝一个固定 url 发送请求 POST 请求，同时请求中包含 android ID 和用户的手机型号等重要信息。

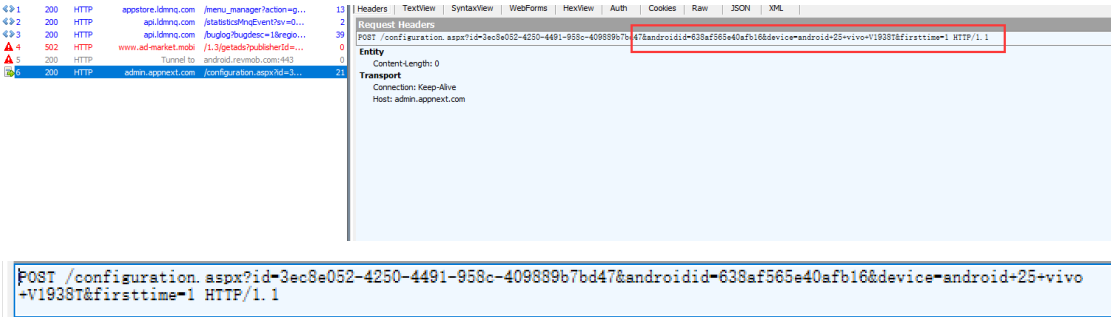

图 3-7：抓包情况 4

随后我们访问该网址，发现其能够正常访问(甚至包含一个页面 Logo)，但是没有任何的页面展示，猜测可能是攻击者的钓鱼门户网站。如果能进一步反过来攻击该网站，也许能获得网站服务器的数据，从而了解攻击者是否真的在收集不同手机用户的相关数据。

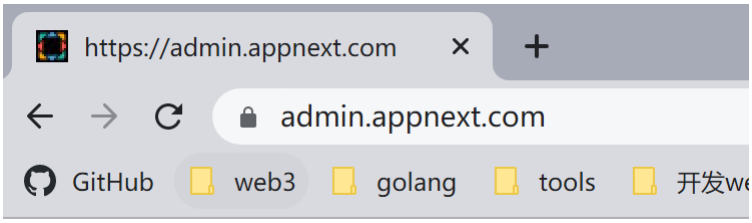

图 3-8：网站访问正常

3.2.2 API HOOK 分析

基于我们最初的判断，我们推测一个间谍应用要想远程控制或者传递机密信息，应该会隐秘地调用 TCP/UDP 以及 SSL 相关的库，比如 Android 常用的第三方库 OkHTTP。因此我们使用 API Hook 分析工具 Frida 来检测某一个应用是否使用了相关的库函数。

Frida 的配置较 Fiddler 稍微复杂，需要在本机上通过 pip 安装 frida 和 frida-tools，然后查看虚拟机的手机版本，并 push 进一个版本适配的 frida-server，然后暴露端口供本机连接。具体操作可查看参考文档的相关配置，在此不再赘述。最终的效果为，本机能够监听到虚拟机上跑的进程：

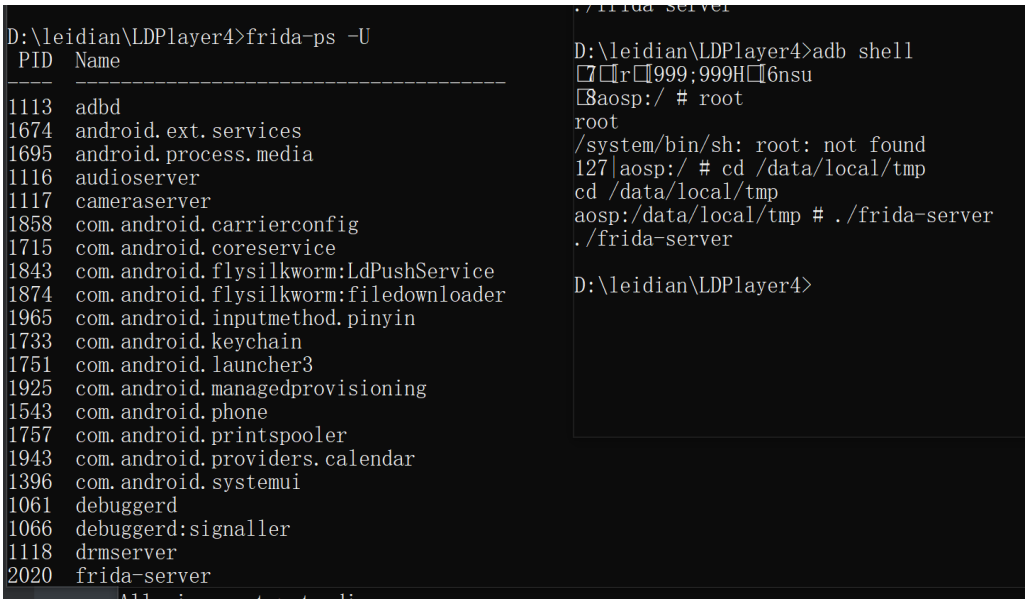

图 3-9：配置 frida 工具

由于 frida 的使用需要知道一个应用的具体包名，所以我们这里还需要用到 adb am monitor 来获取。

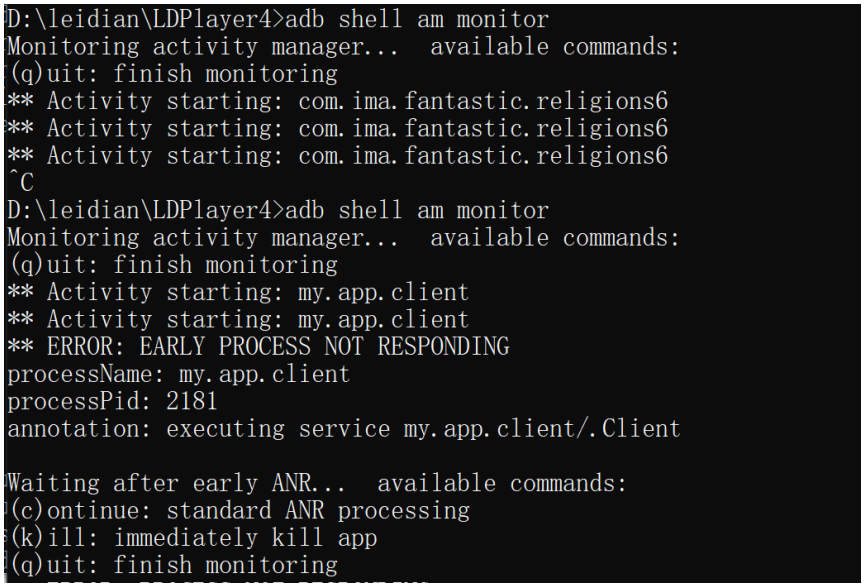

图 3-10：获取包名

并基于网上的代码框架编写了一个针对网络相关的 API 的 Hook JS 脚本，然后使用命令 frida -U- l ./frida-script.js -f package_name 来实现检测。

```
/// -- Specific targeted hooks: -- ///

// HttpsURLConnection
try {
    const HttpsURLConnection = Java.use("javax.net.ssl.HttpsURLConnection");
    HttpsURLConnection.setDefaultHostnameVerifier.implementation = function (hostnameVerifier) {
        console.log('  --> Bypassing HttpsURLConnection (setDefaultHostnameVerifier)');
        return; // Do nothing, i.e. don't change the hostname verifier
    };
    console.log('[+] HttpsURLConnection (setDefaultHostnameVerifier)');
} catch (err) {
    console.log('[ ] HttpsURLConnection (setDefaultHostnameVerifier)');
}
try {
    const HttpsURLConnection = Java.use("javax.net.ssl.HttpsURLConnection");
    HttpsURLConnection.setSSLSocketFactory.implementation = function (SSLSocketFactory) {
        console.log('  --> Bypassing HttpsURLConnection (setSSLSocketFactory)');
        return; // Do nothing, i.e. don't change the SSL socket factory
    };
    console.log('[+] HttpsURLConnection (setSSLSocketFactory)');
} catch (err) {
    console.log('[ ] HttpsURLConnection (setSSLSocketFactory)');
}
```

图 3-11：脚本代码

我们测试了两个效果比较好的应用。第一个是名称为“AhMyth”的应用，可与上面对应。我们发现他虽然执行的效果是点击后就退出，但是实际上却包含了建立 SSL 连接所需的所有库函数，包括第三方性能库，而通过我们上面的网络抓包分析，该程序正意图与某一 IP 地址建立 SSL 连接，也侧面证明了我们检测的正确性。

```
PS D:\作业\大四上\移动应用安全\MalRAT\noway> frida  -U -l ./frida-script.js -f ahmyth.mine.king.ahmyth
     /   |      Frida 16.0.2 - A world-class dynamic instrumentation toolkit
    | (_|  |
    >  _  |      Commands:
   /_/ |_|          help      -> Displays the help system
   . . . .          object?   -> Display information about 'object'
   . . . .          exit/quit -> Exit
   . . . .
   . . . .      More info at https://frida.re/docs/home/
   . . . .
   . . . .      Connected to Android Emulator 5554 (id=emulator-5554)
Spawned `ahmyth.mine.king.ahmyth`. Resuming main thread!
[Android Emulator 5554::ahmyth.mine.king.ahmyth ]-> ---
Unpinning Android app...
[+] SSLPeerUnverifiedException auto-patcher
[+] HttpsURLConnection (setDefaultHostnameVerifier)
[+] HttpsURLConnection (setSSLSocketFactory)
[+] HttpsURLConnection (setHostnameVerifier)
[+] SSLContext
[+] TrustManagerImpl
[+] OkHTTPv3 (list)
[ ] OkHTTPv3 (cert)
[+] OkHTTPv3 (cert array)
[ ] OkHTTPv3 ($okhttp)
[ ] Trustkit OkHostnameVerifier(SSLSession)
[ ] Trustkit OkHostnameVerifier(cert)
[ ] Trustkit PinningTrustManager
[ ] Appcelerator PinningTrustManager
[+] OpenSSLSocketImpl Conscrypt
[ ] OpenSSLEngineSocketImpl Conscrypt
[ ] OpenSSLSocketImpl Apache Harmony
[ ] PhoneGap sslCertificateChecker
[ ] IBM MobileFirst pinTrustedCertificatePublicKey (string)
[ ] IBM MobileFirst pinTrustedCertificatePublicKey (string array)
[ ] IBM WorkLight HostNameVerifierWithCertificatePinning (SSLSocket)
[ ] IBM WorkLight HostNameVerifierWithCertificatePinning (cert)
[ ] IBM WorkLight HostNameVerifierWithCertificatePinning (string string)
[ ] IBM WorkLight HostNameVerifierWithCertificatePinning (SSLSession)
[+] Conscrypt CertPinManager
[ ] CWAC-Netsecurity CertPinManager
[ ] Worklight Androidgap WLCertificatePinningPlugin
[ ] Netty FingerprintTrustManagerFactory
[ ] Squareup CertificatePinner (cert)
[ ] Squareup CertificatePinner (list)
[ ] Squareup OkHostnameVerifier (cert)
[ ] Squareup OkHostnameVerifier (SSLSession)
[+] Android WebViewClient (SslErrorHandler)
```

图 3-12： API HOOK 分析 1

随后我们测试了“Shaw meng”应用，也发现该简单应用也包含了建立 SSL 连

接的库函数，但是比上述应用的相关函数数目要少一些，可能只是单方面建立请求。

图 3-13 API HOOK 分析 2

3.2.2 实验总结

我们通过两种方式针对间谍应用需要基于网络通信的特点进行了实验，抓包得到了 4 种较为突出的通信情况：

1. 没有明显的网络通信
2. 通过 Get 请求非法网站获得恶意配置
3. 与指定 IP 地址建立 SSL 连接
4. POST 请求的同时，将用户信息作为携带参数

随后我们又通过 API HOOK 检测的技术，对发送网络请求的应用进行了验证，进一步坐实了其间谍应用的身份。也从另一个角度了解了间谍应用的行为特征。

除此之外我们还做了一个有意思的实验，我们找到了市面上比较流行的间谍应用检测软件 Anti Spy，用它来检测已经安装了的几个间谍应用程序。结果却令我们大跌眼镜，一个间谍应用都没有检测出来。我们推测可能是因为这几个应用不能造成实质上的危害。

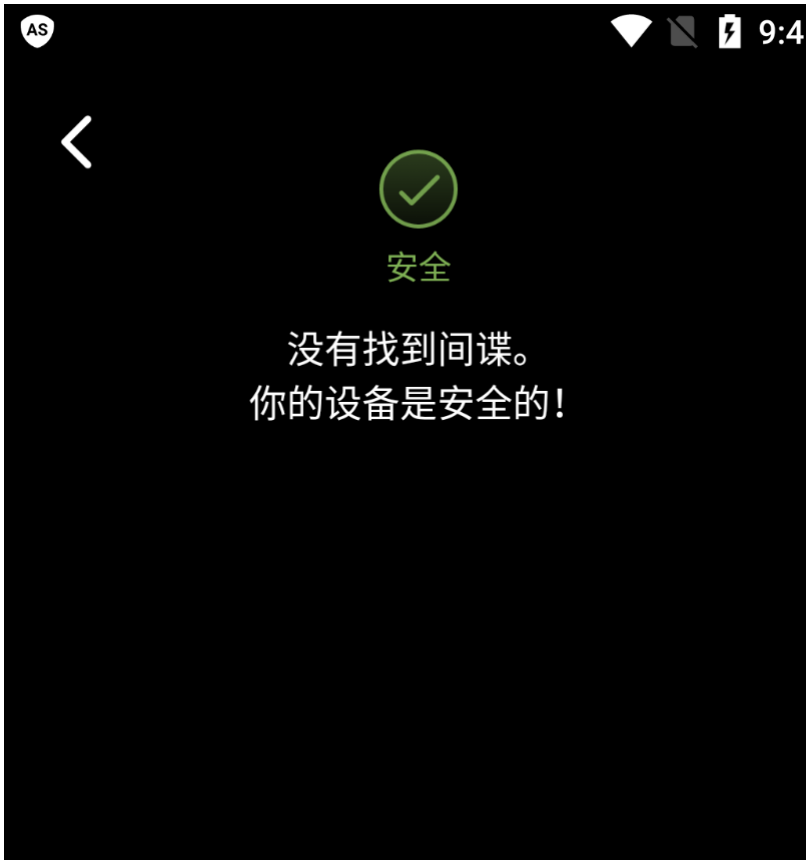

图 3-14 Anti Spy 检测结果

4. 心得体会

课程中提供给我们的数据集总共有 1000 多个应用，相对来说这个数据集还是较大的，如果对于数据集中的每一个软件都进行相应的测试，整体的工程量比较费时费力，所以我们讨论后决定在基于 avclass_family 的分类，选取数个具有代表性的软件进行测试。在实际应用中，市场上的安卓应用数以万计，对于我们个人来说使用的应用也会有数十甚至上百，可想而知，对于整个安卓市场来说，检测应用的安全性是一件极其复杂且工作量极其巨大的工作。

鉴于测试环境以及检测工具的限制，我们只是主要对测试应用的网络流进行抓包检测以及相关的 API Hook 检测。但是由于我们的测试是在模拟器上进行，并且对于每一个测试应用并未进行长时间的运行测试，所以我们认为我们所实施的测试方法并不能测试一些隐蔽性较强的间谍软件，比如软件只有在获取关键信息后才会触发间谍行为，或者软件只有在特殊的时间段才会进行间谍行为，这些我们认为也是间谍应用可能存在的一些行为特征，但是在我们目前的测试环境以及测试方法中，暂不能去识别鉴定这样的一些特征，有待进一步的工作去进行相应的验证。

此外，在整个实验过程之中，我们也遇到了一些困难。第一个困难就是限于实验环境模拟器的限制，很多应用根本没有办法正常运行，这给我们的测试工作带来了较大的阻碍。第二个困难就是，在实际执行 API Hook 时，所截获到的 API 数量较多，但是我们基本只需要与网络连接相关的 API，这就需要我们进行一个较为全面详细的整理，这样才能尽可能覆盖间谍应用可能使用到的 API。

然后，是对于这门课程的心得体会，就我们团队而言，我们认为这门课的知识覆盖面还是非常广泛的，首先可以让大家接触到移动应用前沿科学的一些相关知识，其次是可以让大家上手去做一些科研的实践，并且各个小组之间的任务分配不同，所能接触到的具体知识和工具也就不一样，大家可以根据自己的实际兴趣爱好去选择自己的研究任务，这样就可以在自己感兴趣的方向学习相关的知识和工具的使用方法，并且研究任务是开放式的，大家可以自由发挥，自主探索任务中的问题所在，这对于大家科研能力的培养是有一定的帮助的。

最后，是对于本门课程的建议。由于本门课程所涉及的知识点较为前沿，并且知识点的覆盖面比较广，所以我们认为增加本门课程的课时是有一定必要的，充足的课时可以使得同学们拥有充足的时间去消化理解课程所授知识，也可以让同学们有更多的时间去丰富自己的课题内容，更加深入探究自己的研究任务。此外，实验课程大家所分配的任务是独立的，小组之间也并没有交流，如果能够增加小组之间的成果交流，相信不仅可以更加激发同学们的研究兴趣，也可以让同学们接触到更多的有趣科研方向，丰富大家的知识体系。

5. 参考文档

[1] https://www.kancloud.cn/apachecn/guru99-zh/1954447

[2] https://www.cnblogs.com/testcodell/p/16803531.html

[3] https://blog.csdn.net/bi207186661/article/details/126085171

[4] https://zhuanlan.zhihu.com/p/102392715

[5] https://httptoolkit.com/blog/frida-certificate-pinning/

[6] https://frida.re/docs/examples/windows/

[7] https://www.52pojie.cn/thread-833964-1-1.html

[8] https://www.jianshu.com/p/51e6aef175a2

[9] https://juejin.cn/post/7039551824029810719