V-1.0

APP 安全检测指南 - Panda

【Android】

作者：Panda

Blog：http://blog.cnpanda.net

首发于： TOOLS

【0x01】前言

前一段时间业务应求，需要测试一个 APP，谷歌、百度、土司和九零搜索了一下，发现这方面的资料太少了，完整的测试流程、测试概要更是没有。所以有了这一篇文章。文章通过业务中的测试要点（如下图）进行了实例测试或者说明，同时参考了公司的 APP 测试白皮书进行了补充说明，算是一个手册吧。

APP渗透测试要点（安卓）	客户端程序安全	安装包签名	进程保护	内存访问和修改
		反编译保护		动态注入
		应用完整性校检	通信安全	通信加密
		组件安全		证书有效性
	敏感信息安全	数据文件		关键数据加密和校检
		Logcat日志		访问控制
	密码安全	键盘劫持		客户端更新安全性
		随机布局软键盘		短信重放攻击
		屏幕录像	业务安全	越权操作
	密码策略设置	密码复杂度检测		交易篡改
		账号登陆限制		重放攻击
		账户锁定策略		用户枚举
		问题验证		暴力破解
		会话安全		注入/XSS/CSRF
		界面切换保护	其他	其他
		UI信息泄露		
		验证码安全		
		安全退出		
		密码修改验证		
		Activity界面劫持		

声明：本文章仅作为业务中的 APP 渗透测试指南使用，请勿用于非法行为。本文首发于土司，禁止任何人转载！谢谢！

【0x02】测试环境

SDK

Java JDK，Android SDK

工具

7zip，dex2jar，jd-gui，apktool，activity 劫持测试工具等（部分工具见附录下载）

【0x03】客户端程序安全

1. 安装包签名

描述

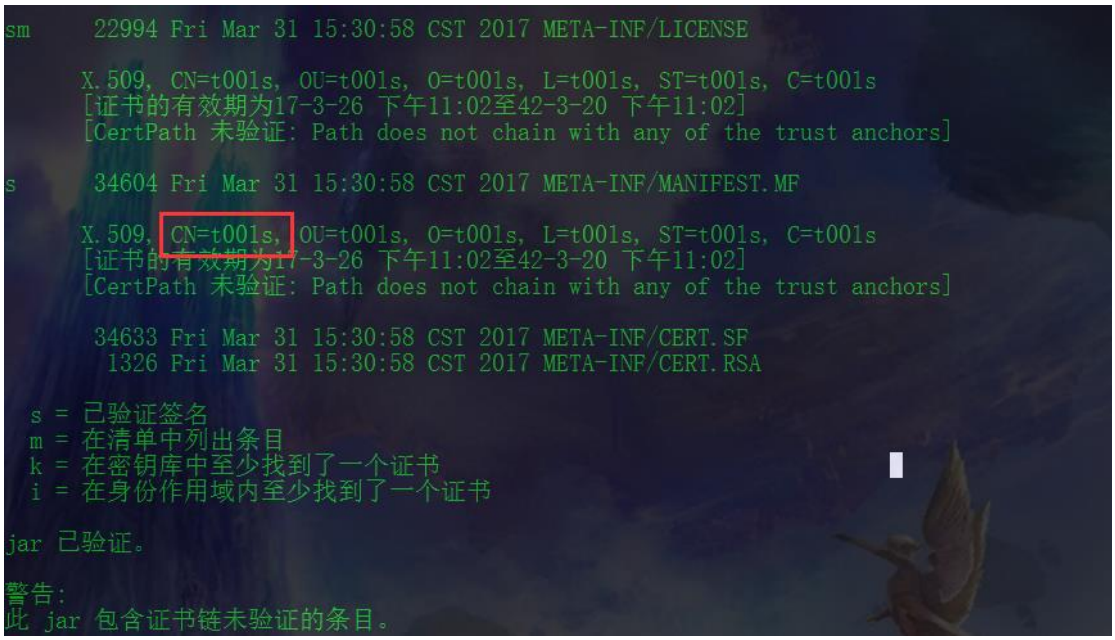
在 Android 中，包名相同的两个 APK 会被认为是同一个应用。当新版本覆盖旧版本时，签名证书必须一致，否则会被拒绝。(即使开启了“允许未知来源的应用”)。如果 APK 没有使用自己的证书进行签名，将会失去对版本管理的主动权。本项检测是检测客户端是否经过恰当签名（正常情况下应用都应该是签名的，否则无法安装），签名是否符合规范。

测试步骤

使用 JDK 中的 jarsigner.exe 检查安装包的签名，命令如下：

jarsigner.exe -verify APK 文件路径 -verbose -certs

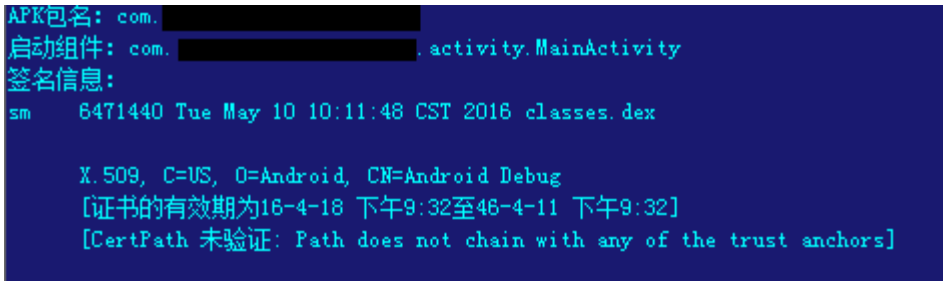
以土司 APP 为例，测试结果如下：



如上图，说明测试结果为安全。

要说明的是，只有在使用直接客户的证书签名时，才认为安全。Debug 证书、第三方（如开发方）证书等等均认为风险。

如下图就是认为存在风险：



威胁等级

安装包签名的威胁等级判断一般如下：

若客户端安装包签名有异常（例如签名证书为第三方开发商而不是客户端发布方），此时高风险；若无异常则无风险。

安全建议

将安装包进行签名并检测安装包签名的异常。

2. 反编译保护

描述

测试客户端安装程序，判断是否能反编译为源代码，java 代码和 so 文件是否存在代码混淆等保护措施。未作保护的 java 代码，可以轻易分析其运行逻辑，并针对代码中的缺陷对客户端或服务器端进行攻击。

成功的反编译将使得攻击者能够完整地分析 APP 的运行逻辑，尤其是相关业务接口协议、和通信加密的实现。

科普

smali 语言是一种 Android 系统特有的中间代码语言。Android 系统的 JVM 与其它常见操作系统有所区别，使用 Dalvik 指令（可执行文件为*.dex）代替了通常的 JVM 中间代码（可执行文件为*.class、*.jar）。对应 Dalvik 指令的“汇编语言”便是 smali。因此，从*.dex 中恢复 smali 代码比恢复 JAVA 代码要容易，成功率更高，但可读性差。尽管如

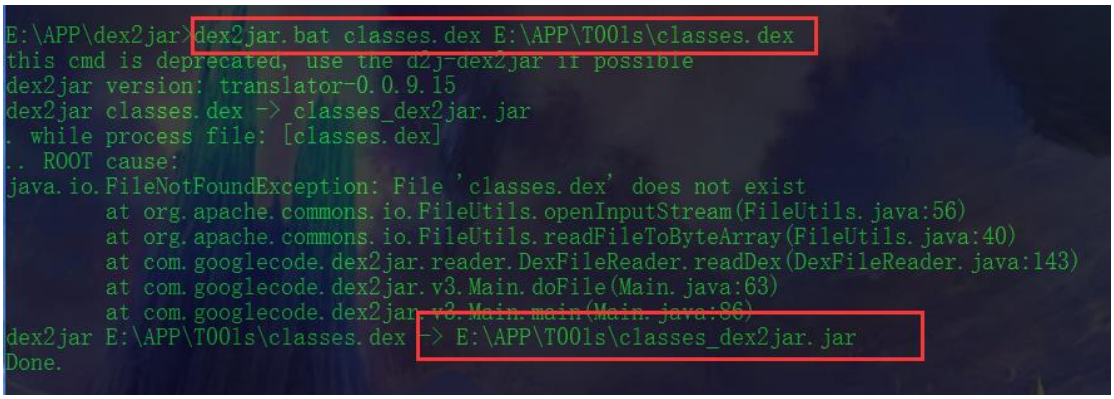
此，如果 APK 经过花指令处理，也会导致无法恢复 smali 代码（表现为 apktool 解包失败）。

测试步骤

把 apk 当成 zip 并解压，得到 classes.dex 文件（有时可能不止一个 dex 文件，但文件名大多类似），如下图：

使用 dex2jar 执行如下命令：

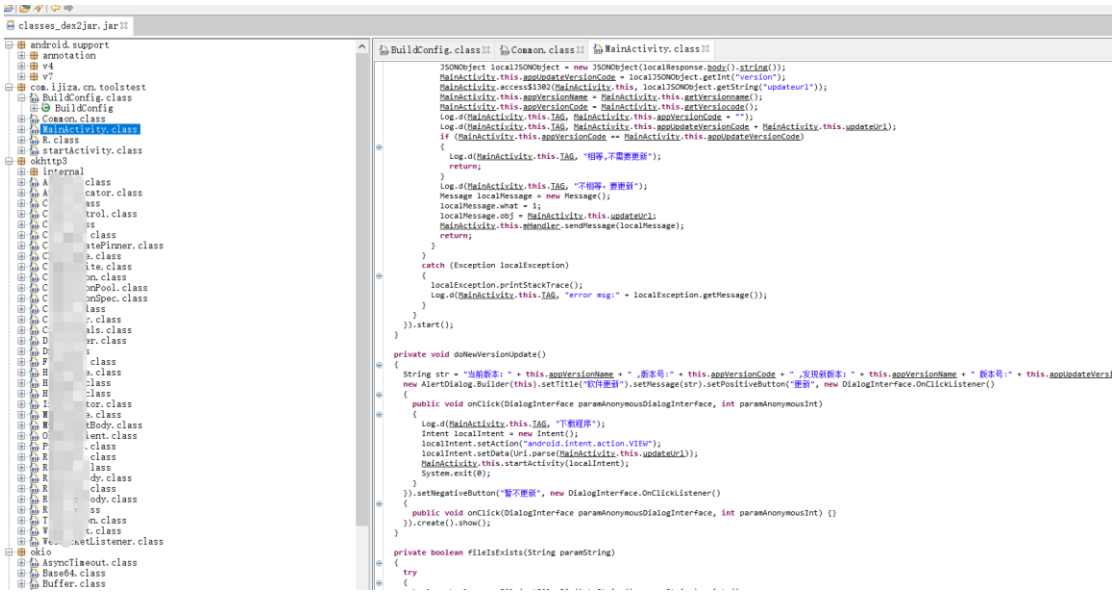
dex2jar.bat classes.dex 文件路径



得到 classes.dex.jar

然后使用 jd-gui 打开 jar 文件，即可得到 JAVA 代码。

【注：直接使用 smali2java 打开 apk 文件，也可反编译回 Java 代码。】

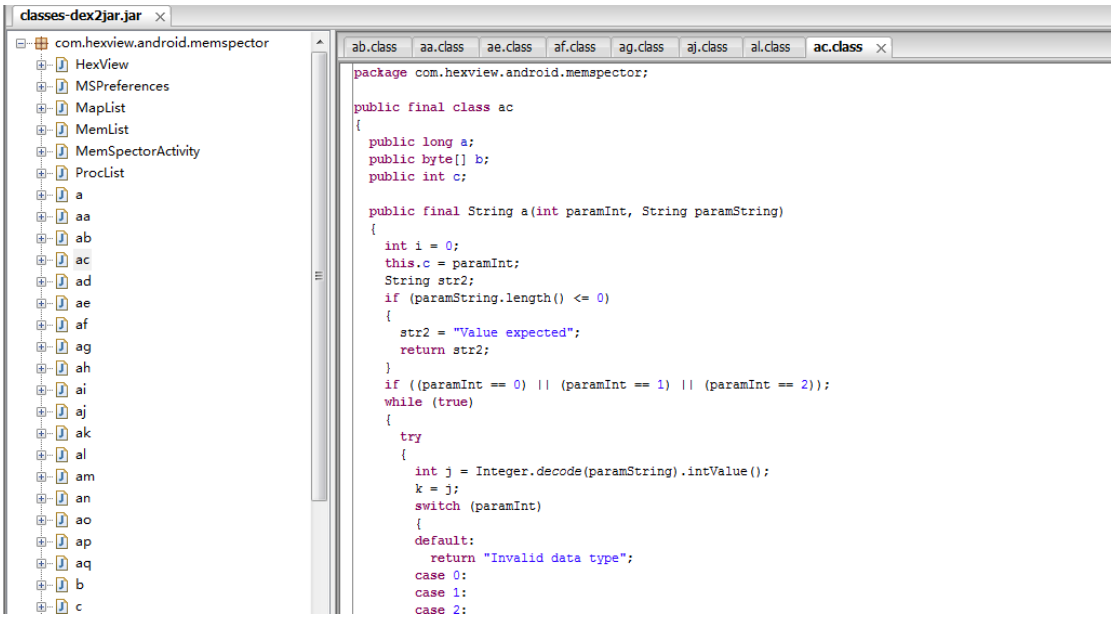


【注：有时用 apktool 能够解包并查看 smali，但 dex2jar 却不行。如果 dex2jar 反编译失败，可以试试看能不能恢复 smali 代码。】

如上图，逆向后发现是没混淆的情况，是不安全的。

如果代码经过混淆，或者有加壳措施，不能完整恢复源代码的，都可以认为此项安全。

下图为混淆后的代码样例，除了覆写和接口以外的字段都是无意义的名称：



威胁等级

若客户端进行加壳保护，此时认为无风险。
若大部分代码（包括核心代码）经过混淆，此时低风险。
若部分代码混淆，关键代码（加密或通信等）可以获知其关键代码，此时中风险。

安全建议

建议客户端程序可以把关键代码以 JNI 方式放在 so 库里。so 库中是经过编译的 arm 汇编代码，可以对其进行加壳保护，以防止逆向分析。

3. 应用完整性校检

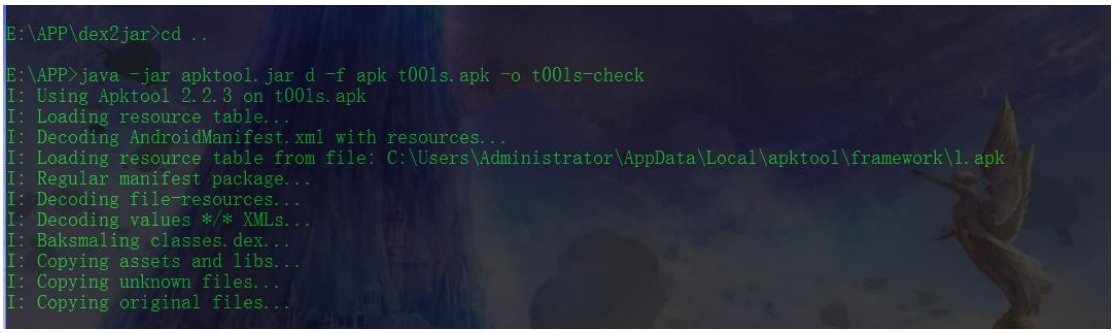
描述

测试客户端程序是否对自身完整性进行校验。攻击者能够通过反编译的方法在客户端程序中植入自己的木马，客户端程序如果没有自校验机制的话，攻击者可能会通过篡改客户端程序窃取手机用户的隐私信息。

测试步骤

用 ApkTool 将目标 APK 文件解包，命令如下：

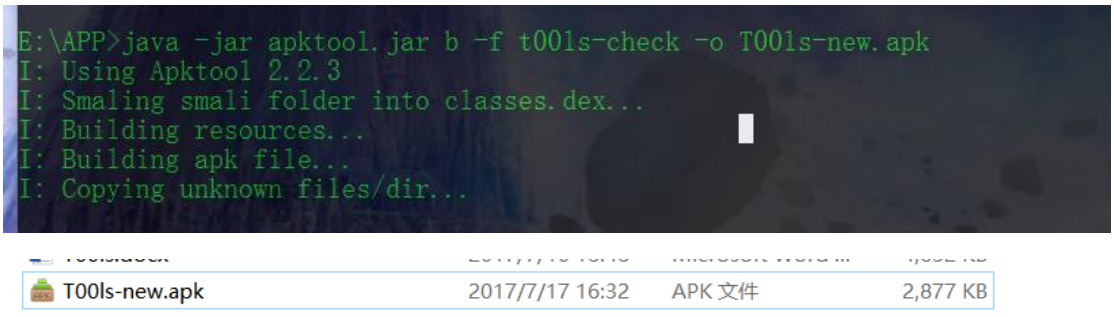
java -jar apktool.jar d -f apk 文件路径 -o 解包目标文件夹



随便找一个解包目录里的资源文件，修改之，推荐找到 logo 之类的图进行修改（因为容易确认结果）；

用 ApkTool，将解包目录重新打包成未签名的 APK 文件，命令如下：

java -jar apktool.jar b -f 待打包的文件夹 -o 输出 apk 路径



用 SignApk，对未签名的 APK 文件进行签名，命令如下：

java -jar signapk.jar testkey.x509.pem testkey.pk8 待签名 apk 文件路径 签名后输出 apk 路径

```
E:\APP\signapk>java -jar signapk.jar testkey.x509.pem testkey.pk8 E:\APP\t00ls-new.apk tools-finish.apk

E:\APP\signapk>
```

tools-finish.apk　　2017/7/17 16:37　　APK 文件　　2,896 KB

将签了名的 APK 安装、运行、确认是否存在自校验；

需要注意的是，如果之前安装的 APK 和修改后的 APK 签名不同，就不能直接覆盖安装，一般来说，先卸载之前安装的 APP 即可。

【注：APK 必须进行签名后，方可安装和运行。如果开启了“允许未知来源的应用”，那么 Debug 证书、自签名证书、过期证书的签名都是可以的，但是不可以不签名。】

将客户端程序文件反编译，修改源码或资源文件后重新打包安装运行，结果如下图：


下午3:51
35%
Panda
代码审计|渗透测试|CTF
代码审计 - Python - 技术杂谈 - CTF
- 大学项目 - 个人日志 - 关于 - Friends
搜索
关于
08年迈入安全圈。曾混迹于土司
（www.t00ls.net）、九零
（forum.90sec.org）等技术论坛。现
在归于平静，潜水党一员。
首页
资讯
文章
工具
我的


下午4:42
18%

网络错误,检查你的网络


首页
资讯
文章
工具
我的

如上图，我将土司的页面进行了修改，修改成我的博客、百度，其中一些图片也进行了修改。

经测试，APP 可以被重新打包运行。

上图为没有进行自校验的情况，下图为经过自校检的情况，，修改后无法正常启动。

威胁等级

若应用完整性校验不使用 MANIFEST.MF 中的数据，且核心代码通过 JNI 技术写入.so 库，同时于服务端进行相关校验，此时无风险。

若应用完整性于本地进行验证而不存在其他问题或使用 MANIFEST.MF 中的数据作为验证凭证（有新文件时提示应用完整性验证失败），此时低风险；

若在本地进行验证的基础上只通过 MANIFEST.MF 对客户端原有文件进行校验而忽略新增文件的检验，此时中风险；若未进行应用完整性校验此时高风险。

安全建议

建议客户端在每次开机启动时进行客户端自身的应用完整性校验，在验证逻辑中不使用 MANIFEST.MF 中的数据作为验证凭证，同时需验证是否有不属于该客户端版本的新文件添加，验证过程于服务器端完成。

4. 组件安全

描述

本项主要测试客户端是否包含后台服务、Content Provider、第三方调用和广播等组件，Intent 权限的设置是否安全。应用不同组成部分之间的机密数据传递是否安全。检查客户端是否存在组件劫持风险，查看客户端程序具有导出哪些应用信息的权限。反编译 APK 文件后，检查 AndroidManifest 文件中是否有多余的 android:export 声明，客户端是否存在导出其他应用信息的权限等。

科普

什么是组件？

安卓 APP 以组件为单位进行权限声明和生命周期管理；

组件有什么用？

安卓系统的组件共有四种，其主要用途分别为：
Activity：呈现可供用户交互的界面，是最常见的组件；
Service：长时间执行后台作业，常见于监控类应用；
Content Provider：在多个 APP 间共享数据，比如通讯录数据；
Broadcast Receiver：注册特定事件，并在其发生时被激活；

什么是权限声明？

安卓系统定义了许多权限声明项，分别对应一些操作系统功能；

权限声明有什么用？

如果一个 APP 或组件在没有声明权限的情况下就调用相关 API，会被拒绝访问；
但如果声明了相关权限，安装的时候就会有提示；

什么是组件导出？

简而言之，就是别的 APP 也可以访问这个组件。

组件导出有什么用？

有些 APP 的功能需要提供一些接口给其它 APP 访问，就需要把相关的接口功能放在一个导出的组件上。

组件导出有什么危害？

因为权限声明是以组件为单位的，A 组件调用 B 组件的功能来访问操作系统 API 时，适用于 B 组件的权限声明。

如果 B 作为导出组件，没有进行严格的访问控制，那么 A 就可以通过调用 B 来访问原本没有声明权限的功能，构成本地权限提升。

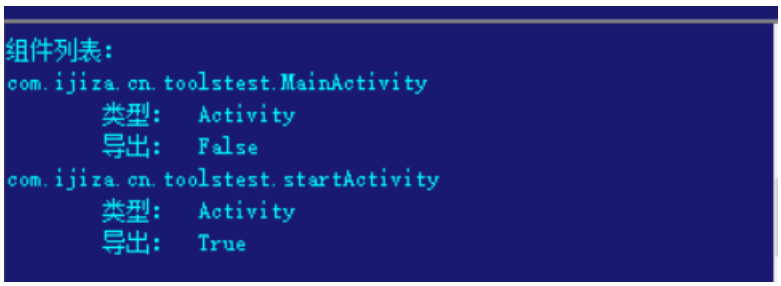
测试步骤

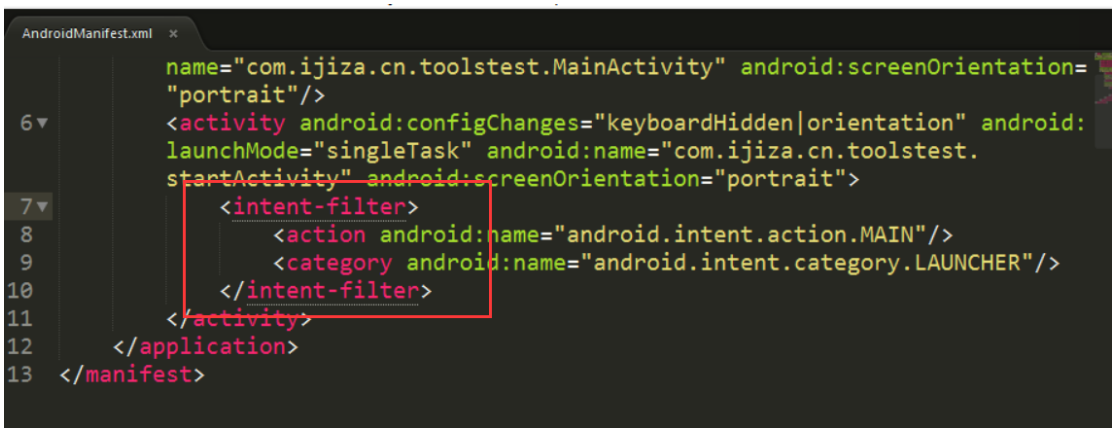
方案一：

使用 ApkTool 解包，打开解包目录中 AndroidManifest.xml，对其中声明的各个组件，根据以下规则判断是否可导出：

1. 显式声明了 android:exported="true"，则可导出；
2. 显示声明了 android:exported="false"，则不可导出；
3. 未显示声明 android:exported：
   a) 若组件不是 Content Provider：
      i. 若组件包含<intent-filter>则可导出，反之不可；
   b) 若组件是 Content Provider：
      i. 若 SDK 版本<17 则可导出，反之不可。

从测试的角度上，只能判断组件是否导出，但能否构成危害需要详细分析源代码后才能得出结论。一般来说，在测试时尽管写清所有的导出组件，由客户开发侧确认相关组件是否确实需要导出即可。

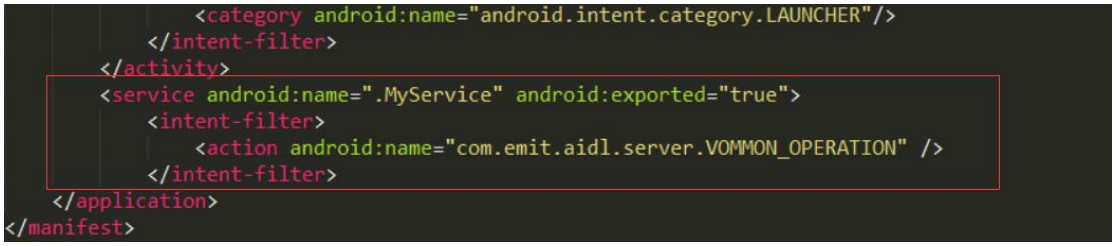




由于功能需要，启动 Activity 和 Content Provider 大多是导出组件，一般无须理会。如上图，土司的组件导出就是安全的：

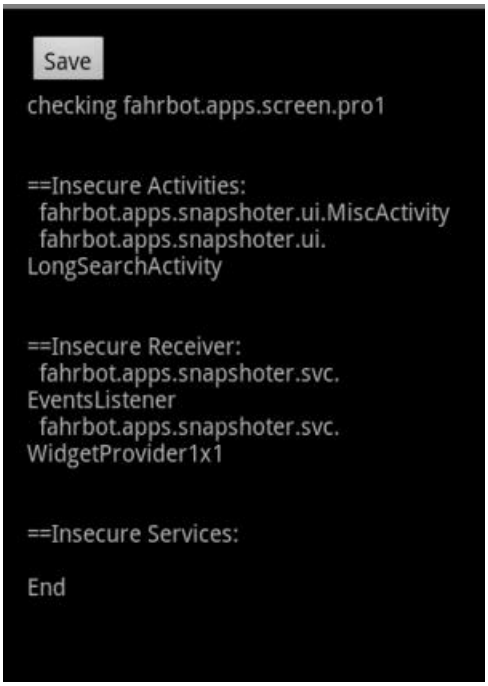
方案二：

检查 AndroidManifest.xml 文件中各组件定义标签的安全属性是否设置恰当。如果组件无须跨进程交互，则不应设置 exported 属性为 true。例如，如下图所示，当 MyService 的 exported 属性为 true 时，将可以被其他应用调用。（当有设置权限(permissions)时，需要再考察权限属性。如 android:protectionLevel 为 signature 或 signatureOrSystem 时，只有相同签名的 apk 才能获取权限。详情见附录参考资料 API Guides 系统权限简介）



可以使用“组件安全测试工具”来检测组件的 exported 属性（有些应用在代码中动态注册组件，这种组件无法使用“组件安全测试工具”测试，需要通过阅读代码确定是否安全。

如下图所示。凡是列出来的组件都是 exported 属性为 true 的。



当发现有可利用的组件导出时，（当然，并不是说所有导出的组件都是不安全的，如果要确定，必须看代码，对代码逻辑进行分析）可利用 drozer 测试工具进行测试，工具以及使用方法请看附录。

威胁等级

若不存在组件暴露的情况，此时无风险。

如存在组件暴露的情况，但暴露的组件无关客户端逻辑核心或不会泄露用户敏感信息，此时低风险；

若暴露的组件会泄露用户敏感信息（例如邮件客户端存在消息组件的暴露，攻击者可以通过编写 APK，通过组件利用的方式读取用户邮件信息）

安全建议

避免不必要的组件导出。

【0x04】敏感信息安全

1. 数据文件

描述

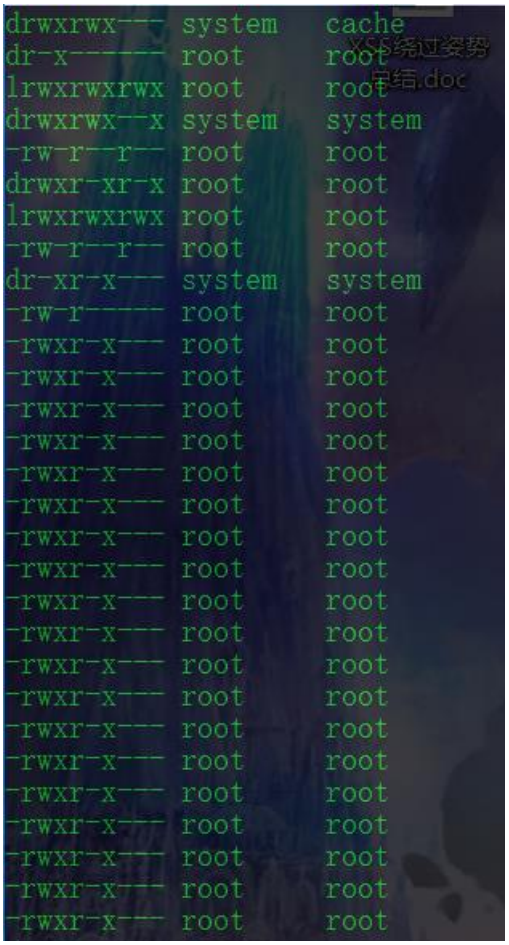
检测客户端是否保存明文敏感信息，能否防止用户敏感信息的非授权访问。

文件敏感信息泄露以明文存储“记住密码”居多。一般来说，先将安卓设备中的文件复制到有鼠标的主机上，方便查看。

测试步骤

首先查看相关文件的权限配置。

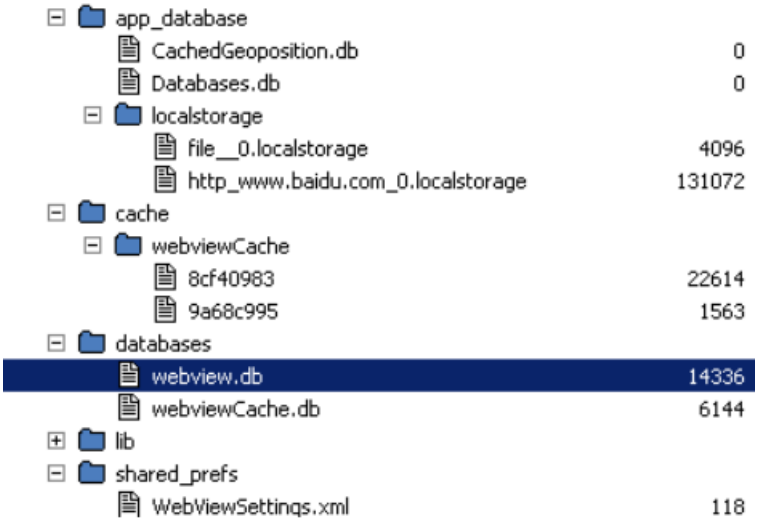
正常的文件权限最后三位应为空（类似“rw-rw----”），即除应用自己以外任何人无法读写；目录则允许多一个执行位（类似“rwxrwx—x”）。如下图所示，（lib 子目录是应用安装时由 android 系统自动生成，可以略过）：



当客户端使用 MODE_WORLD_READABLE 或 MODE_WORLD_WRITEABLE 模式创建文件时，shared_prefs 目录下文件的权限也会多出一些，这不一定是安全问题，如下图（Google 已不推荐使用这些模式）：

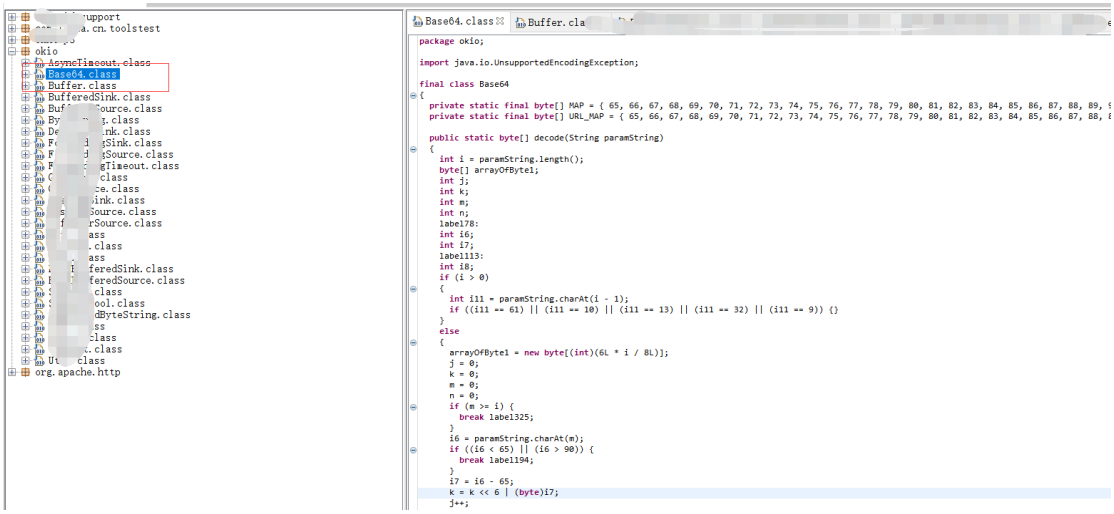
权限检测完整后，再检查客户端程序存储在手机中的 SharedPreferences 配置文件，通常是对本目录下的文件内容（一般是 xml）进行检查，看是否包含敏感信息。

最后在检测 SQLite 数据库文件，在私有目录及其子目录下查找以.db 结尾的数据库文件。对于使用了 webView 缓存的应用，会在 databases 子目录中保存 webview.db 和 webviewCache.db，如图所示。其中有可能会记录 cookies 和提交表单等信息。如下图：



使用数据库查看工具即可查看这些文件中是否有敏感信息。

还有些时候，客户端程序 apk 包中也是是保存有敏感信息的，比如检查 apk 包中各类文件是否包含硬编码的的敏感信息等。如下图土司 APP 的相关编码信息：



威胁等级

根据敏感信息泄露的程度进行威胁等级评分。若私有目录中存在存储了用户登陆密码（明文或只进行过一次单项哈希散列），手势密码（明文或只进行过一次单项哈希散列）或曾经访问过网址的 Cookie 等敏感信息的文件，此时为高风险，若不存在则无风险。

安全建议

尽量避免在文件、数据库、日志等位置写入敏感信息。如果确实需要存储，应当进行加密。对于内存中的信息泄露，可以通过反注入、反调试来解决。

此外，正常的文件权限最后三位应为空（类似“rw-rw----”），目录则允许多一个执行位（类似“rwxrwx—x”）。

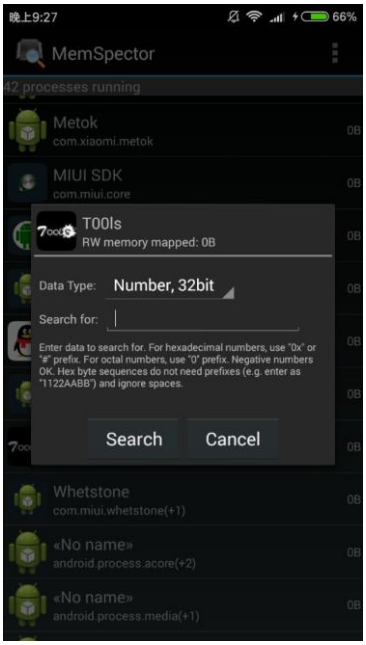
2. Logcat 日志

描述

本项主要是检查客户端程序存储在手机中的日志是否含有敏感信息。

测试步骤

MemSpcetor 中提供了搜索功能，可以将内存 DUMP 到 SD 卡（注意，虚拟机得先配置 SD 卡），然后用 adb 或 monitor 复制到主机上查看。



在这里使用 ADB 进行查看。

使用 adb 工具连接设备：

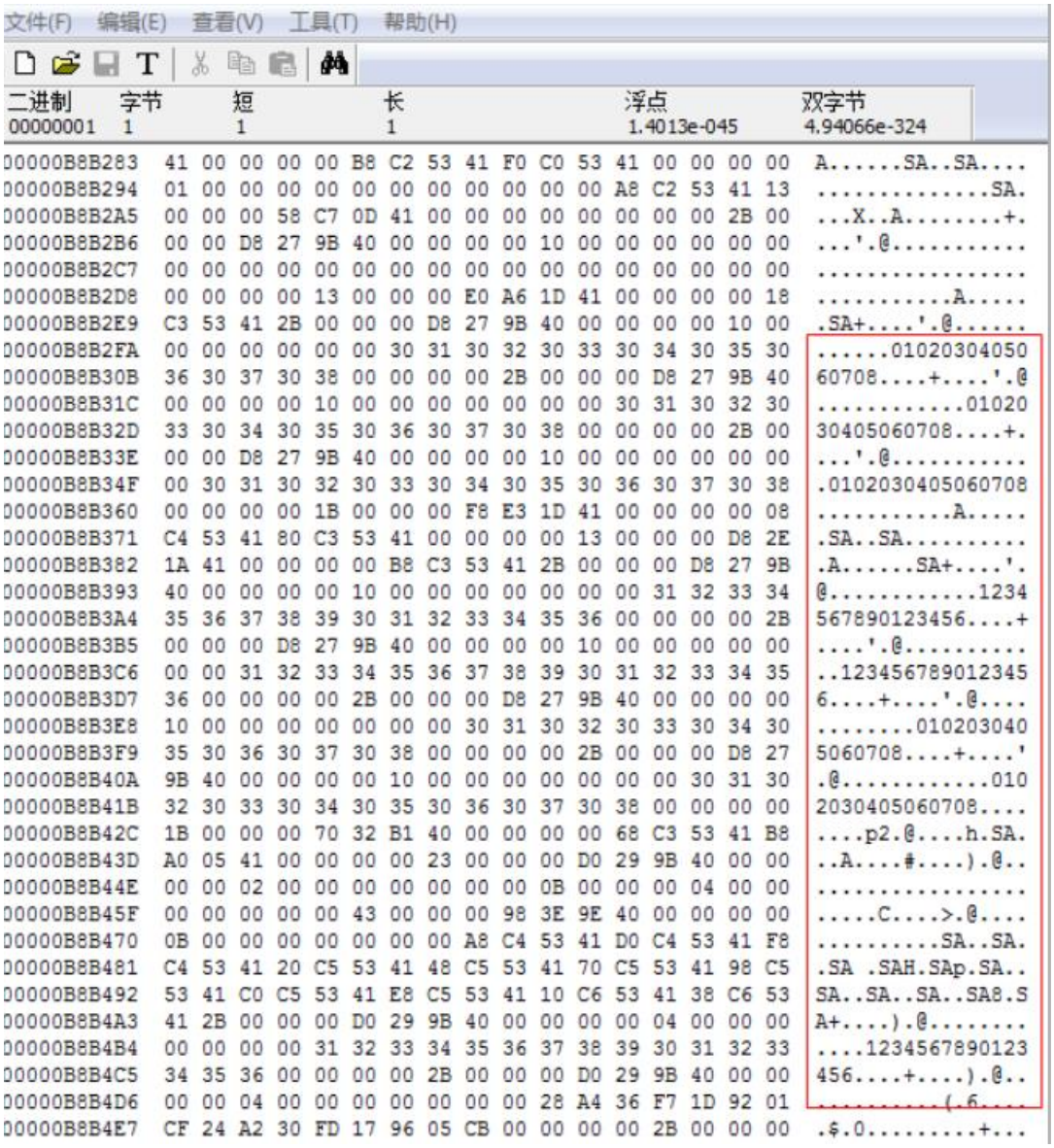
```
adb devices                          //查看安卓设备列表
adb -s 设备名称 其它命令              //当连接了多个设备时，选择操作的目标设备，否则会出错
```

如下图：

```
C:\Users\Administrator>adb devices
List of devices attached
6158ef0a        device
```

adb pull 手机目录名 PC 目录名 //从安卓设备中复制文件到电脑中

然后使用 WinHex 打开



这是查看内存遗留的信息，还可以直接使用 adb 查询 logcat 日志：

```
C:\Users\Administrator>adb shell logcat -d > E:\1.txt

C:\Users\Administrator>
```

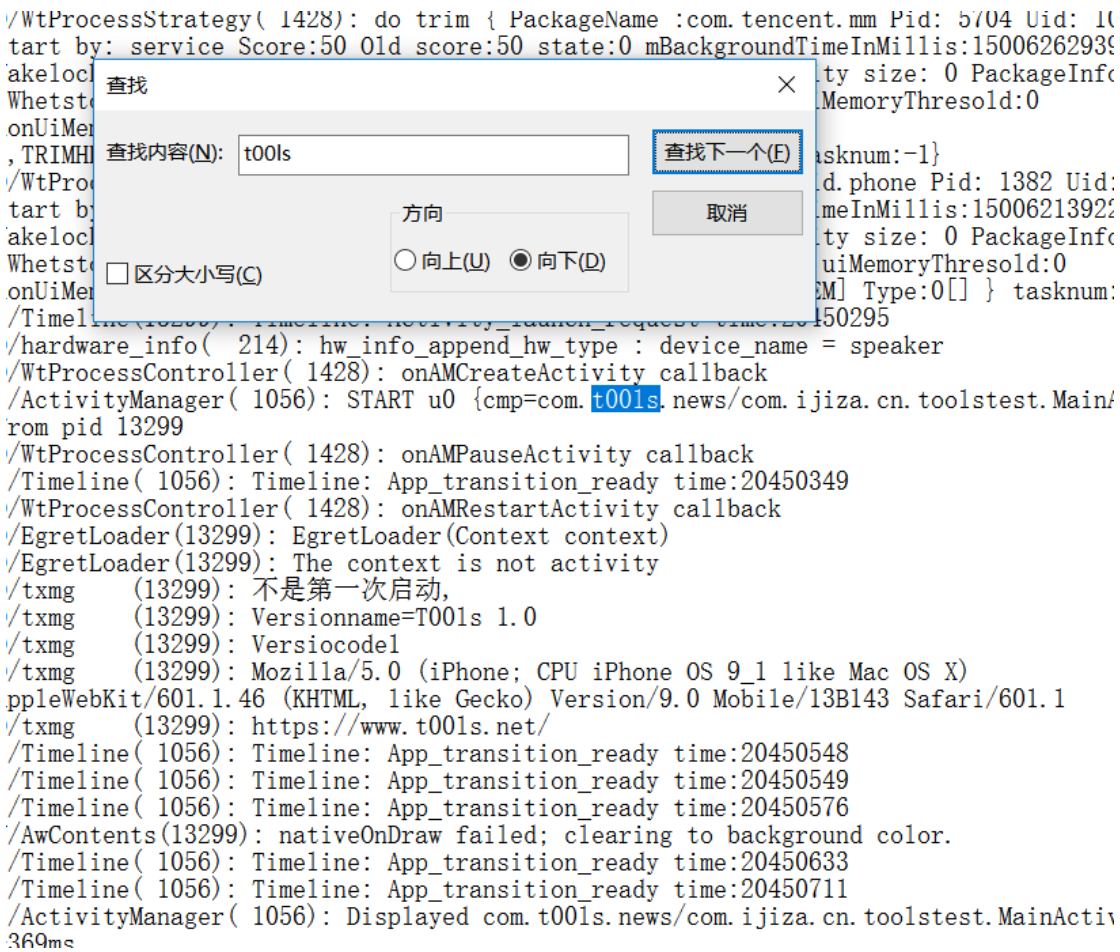
在 adb shell 中，有下列命令可用：

logcat　　//持续输出日志，直到 Ctrl+C

logcat -d　　//一次性输出日志缓存，不会阻塞

```
logcat -c       //清空日志缓存
```

查看 1.txt：




```
/WtProcessStrategy( 1428): do trim { PackageName :com.tencent.mm Pid: 5704 Uid: 1(
tart by: service Score:50 Old score:50 state:0 mBackgroundTimeInMillis:1500626293
akeloc                                                              ty size: 0 PackageInfo
Whetst                                                              MemoryThresold:0
onUiMe
,TRIMH                                                              asknum:-1}
/WtPro                                                              d.phone Pid: 1382 Uid:
tart b                                                              meInMillis:1500621392
akeloc                                                              ty size: 0 PackageInfo
Whetst                                                              uiMemoryThresold:0
onUiMe                                                              M] Type:0[] } tasknum:
/Timel                                                              450295
/hardware_info(  214): hw_info_append_hw_type : device_name = speaker
/WtProcessController( 1428): onAMCreateActivity_callback
/ActivityManager( 1056): START u0 {cmp=com.t00ls.news/com.ijiza.cn.toolstest.MainA
rom pid 13299
/WtProcessController( 1428): onAMPauseActivity_callback
/Timeline( 1056): Timeline: App_transition_ready time:20450349
/WtProcessController( 1428): onAMRestartActivity_callback
/EgretLoader(13299): EgretLoader(Context context)
/EgretLoader(13299): The context is not activity
/txmg     (13299): 不是第一次启动,
/txmg     (13299): Versionname=T001s 1.0
/txmg     (13299): Versiocode1
/txmg     (13299): Mozilla/5.0 (iPhone; CPU iPhone OS 9_1 like Mac OS X)
ppleWebKit/601.1.46 (KHTML, like Gecko) Version/9.0 Mobile/13B143 Safari/601.1
/txmg     (13299): https://www.t00ls.net/
/Timeline( 1056): Timeline: App_transition_ready time:20450548
/Timeline( 1056): Timeline: App_transition_ready time:20450549
/Timeline( 1056): Timeline: App_transition_ready time:20450576
/AwContents(13299): nativeOnDraw failed; clearing to background color.
/Timeline( 1056): Timeline: App_transition_ready time:20450633
/Timeline( 1056): Timeline: App_transition_ready time:20450711
/ActivityManager( 1056): Displayed com.t00ls.news/com.ijiza.cn.toolstest.MainActiv
369ms
```

并未发现有敏感的信息。

威胁等级

根据敏感信息泄露的程度进行威胁等级评分。若相关信息中存在存储了用户登陆密码（明文或只进行过一次单项哈希散列），手势密码（明文或只进行过一次单项哈希散列）或曾经访问过网址的 Cookie 等敏感信息，此时为高风险，若不存在则无风险。

安全建议

数据传输应做到加密处理，敏感信息不要输出在 logcat 日志中。

【0x05】密码安全

1. 键盘劫持

描述

测试客户端程序在密码等输入框是否使用自定义软键盘。安卓应用中的输入框默认使用系统软键盘，手机安装木马后，木马可以通过替换系统软键盘，记录手机键盘输过的密码。

测试步骤

通常来说，只有使用系统输入法的编辑框才能够进行键盘码记录。如果是自制的软键盘，则可以尝试进行触摸屏记录。像下图这样，不使用系统输入法，且按键随机分布的软键盘是安全的：

威胁等级

能够劫持键盘风险为高，不能则为无风险。

安全建议

尽量使用系统自定义的随机软键盘（而非系统输入法）来输入敏感信息。或者对 Native 层输入记录功能进行 Hook（需要 root 权限）。

2. 随机布局软键盘

描述

测试客户端实现的软键盘，是否满足键位随机布放要求。

测试步骤

人工观测……

威胁等级

当客户端软键盘未进行随机化处理时为低风险；当客户端软键盘只在某一个页面载入时初始化一次而不是在点击输入框时重新进行随机化也为低风险。

安全建议

键位随机布放。

3. 屏幕录像

描述

客户端使用的随机布局软键盘是否会对用户点击产生视觉响应。当随机布局软键盘对用户点击产生视觉响应时，安卓木马可以通过连续截屏的方式，对用户击键进行记录，从而获

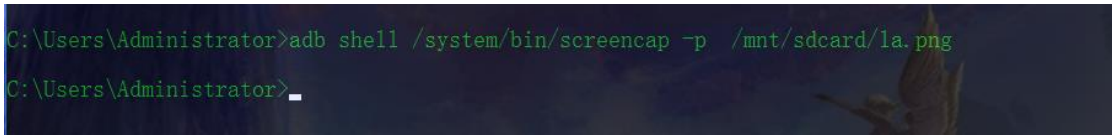
得用户输入。

测试步骤

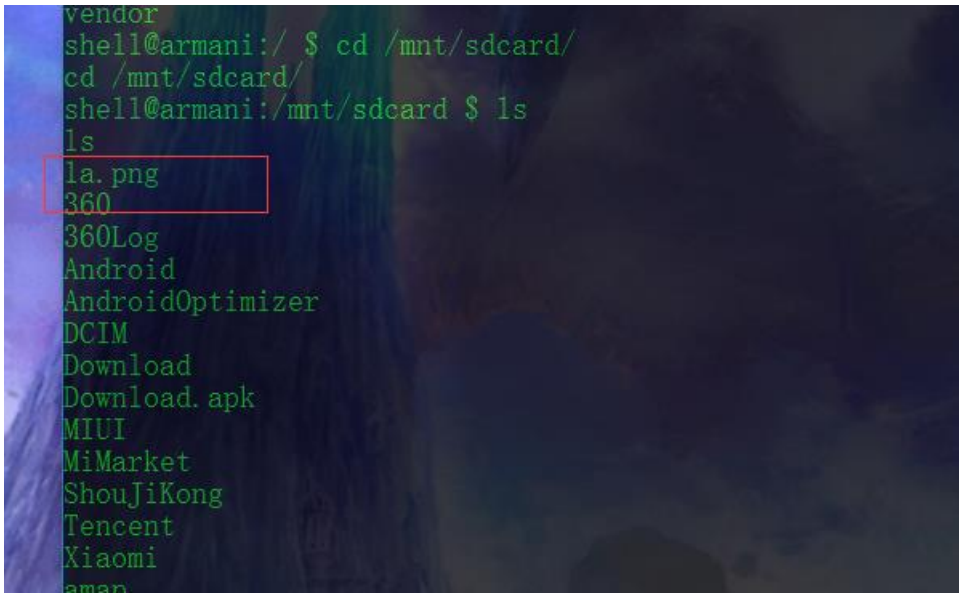
使用 ADB 进行测试：

adb shell /system/bin/screencap -p 输出 png 路径（安卓设备中）

如图：



在/mnt/sdcard/路径下，可以看到 1a.png：



打开：

成功截图。

攻击者可以在用户进入登录页面，在输入密码的同时，进行连续截图，即可记录用户输入的密码。如果没有防截屏，那么即使是随机分布的、没有视觉反馈的软键盘也会被记录：

还有一种验证方式是从代码方面进行验证：

首先检测需较高安全性的窗口（如密码输入框），看代码中在窗口加载时是否有类似下图的代码。按照 android SDK 的要求，开启 FLAG_SECURE 选项的窗口不能被截屏。

```
public class FlagSecureTestActivity extends Activity{
    @override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        getEindow().setFlags(LayouParams.FLAG_SECURE,LayouParms.FLAG_SECURE);
        setContenView(R.layout.main);
    }
}
```

目前 FLAG_SECURE 测试结果：

N一PASS，可截图，

ZTE 880E，可截图

ASUS TF300T，可阻止工具及 ddms 截图。

威胁等级

当使用第三方程序（或系统截屏）可以对客户端内容进行截屏时，为中风险；当客户端会对截屏操作进行有效抵抗时（无法截屏或截屏结果为黑屏等无意义图片）无风险。

安全建议

在敏感信息的输入过程尽量避免视觉反馈，或者在操作系统层面对截屏相关功能进行 Hook 以阻止敏感信息输入期间其它程序的截屏操作（需要 root 权限）。

4. 手势密码

描述

主要从手势密码的复杂度、修改和取消、本地信息保存、锁定策略、抗攻击测试等方面进行测试。

测试步骤

手势密码的复杂度：

1. 进入客户端设置手势密码的页面进行手势密码设置。
2. 进行手势密码设置，观察客户端手势密码设置逻辑是否存在最少点位的判断。
3. 反编译 APK 为 jar 包，通过 jd-gui 观察对应代码逻辑是否有相应的判断和限制条件。（一般设置手势密码若输入点数过少时会有相应的文字提示，通过此文字提示可以快速定位到代码位置）

手势密码的修改和取消：

1. 进入客户端设置手势密码的位置，一般在个人设置或安全中心等地方。
2. 进行手势密码修改或取消操作，观察进行此类操作时是否需要输入之前的手势密码或普通密码。
3. 观察在忘记手势密码等其他客户端业务逻辑中是否存在无需原始手势或普通密码即可修改或取消手势密码的情况。
4. 多次尝试客户端各类业务，观察是否存在客户端逻辑缺陷使得客户端可以跳转回之前业务流程所对应页面。若存在此类逻辑（例如手势密码设置），观察能否修改或取消手势密码。
5. 反编译 APK 为 jar 包，通过 jd-gui 观察对应代码逻辑，寻找客户端对于手势密码的修改和删除是否存在相应的安全策略。

手势密码的本地信息保存：

1. 首先通过正常的操作流程设置一个手势密码并完整一次完整的登陆过程。
2. 寻找/data/data 的私有目录下是否存在手势密码对应敏感文件，若进行了相关的信息保存，基本在此目录下。（关键词为 gesture，key 等）
3. 若找到对应的文件，观察其存储方式，为明文还是二进制形式存储，若为二进制形式，观察其具体位数是否对应进行 MD5（二进制 128 位，十六进制 32 位或 16 位）、SHA-1（二进制 160 位，十六进制 40 位）等散列后的位数。如果位数对应，即可在反编译的

jar 包中搜索对应的关键字以迅速对应代码。

4. 通过代码定位确认其是否进行了除单项哈希散列之外的加密算法，若客户端未将手势密码进行加密或变形直接进行散列处理可认为其不安全，一是因为现阶段 MD5、SHA-1 等常用的哈希算法已被发现碰撞漏洞，二是网络中存在 www.somd5.com 等散列值查询网站可以通过大数据查询的方式获取散列前的明文手势密码。

手势密码的锁定策略：

1. 首先通过正常的操作流程设置一个手势密码。

2. 输入不同于步骤 1 中的手势密码，观察客户端的登陆状态及相应提示。若连续输入多次手势密码错误，观察当用户处于登陆状态时是否退出当前的登陆状态并关闭客户端；当
客户未处于登录状态时是否关闭客户端并进行一定时间的输入锁定。

3. 反编译 APK 为 jar 包，通过 jd-gui 观察对应代码逻辑，寻找客户端是否针对输入次数及锁定时间有相应的逻辑处理。

手势密码的抗攻击测试：

1. 下载并安装 Xposed 框架及 SwipeBack 插件。

2. 启动客户端并进入手势密码输入页。

3. 启动 SwipeBack 插件，观察是否可以通过滑动关闭手势密码输入页的方式进入登陆后的页面。

威胁等级

手势密码的复杂度：

当用户设置或修改手势密码时服务器会对手势密码安全性（使用点数）进行判断时无风险，否则低风险。

修改和取消：

当取消或修改手势密码时，如果不会验证之前的手势密码则为中风险；若存在验证则无风险。

本地信息保存：

当本地保存了明文存储（数组形式）的手势密码时为高风险；当本地保存了只进行单项哈希散列的手势密码时为中风险。

锁定策略：

当服务器不会验证手势密码输入错误次数时为中风险，会进行验证时无风险。

抗攻击测试：

若客户端采用附着的方式将手势密码放置于登陆后的界面上时，如果无法抵抗 SwipeBack 插件的滑动攻击则高风险，如果可以抵抗则无风险。

安全建议

上述手势密码哪方面出现问题就根据哪方面的风险提出针对性的建议。

【0x06】安全策略

安全策略在实际测试中受限较多，因此建议的风险等级：安全策略类全部为低危。

1. 密码复杂度检测

描述

测试客户端程序是否检查用户输入的密码强度，禁止用户设置弱口令。

测试步骤

人工测试，尝试将密码修改为弱口令，如：123456，654321，121212，888888 等，查看客户端是否拒绝弱口令。也可以阅读逆向后的客户端 java 代码，寻找对用户输入口令的检查方法。

威胁等级

低危

安全建议

当系统允许用户设置弱密钥时为低风险，如果存在系统存在一定的安全策略（密码使用数字和字母组成，至少为 8 位）时无风险。

2. 账号登陆限制

描述

测试一个帐号是否可以同时在多个设备上成功登录客户端，进行操作。

测试步骤

人工测试。

威胁等级

低危

安全建议

禁止同一个账号可以同时在多台移动终端设备上登陆。

3. 账户锁定策略略

描述

测试客户端是否限制登录尝试次数。防止木马使用穷举法暴力破解用户密码。

测试步骤

人工测试。

威胁等级

低危

安全建议

设定账户锁定策略。

4. 问题验证

描述

测试对账号某些信息（如单次支付限额）的修改是否有私密问题验证。私密问题验证是否将问题和答案一一对应。私密问题是否足够私密。

测试步骤

人工测试

威胁等级

当用户进行忘记密码操作时，在发送邮件给用户邮箱前是否进行私密问题的验证，若验证则无风险；若不验证则低风险。

安全建议

对于敏感功能操作时，要进行私密问题验证。

5. 会话安全

描述

测试客户端在超过 20 分钟无操作后，是否会使会话超时并要求重新登录。超时时间设置是否合理。

测试步骤

人工测试。

威胁等级

当系统不存在会话超时逻辑判断时为低风险，若存在则无风险

安全建议

设置会话超时。

6. 界面切换保护

描述

检查客户端程序在切换到其他应用时，已经填写的账号密码等敏感信息是否会清空，防止用户敏感信息泄露。如果切换前处于已登录状态，切换后一定时间内是否会自动退出当前会话。

测试步骤

人工检测。在登录界面（或者转账界面等涉及密码的功能）填写登录名和密码，然后切出，再进入客户端，看输入的登录名和密码是否清除。登录后切出，5 分钟内自动退出为安全。

威胁等级

当移动终端设备进行进程切换操作，显示界面不为客户端页面时，若客户端提示用户确认是否为本人操作，则无风险；若无相应提示则为低风险。

安全建议

对于画面切换进行提示或者验证。

7. UI 信息泄露

描述

检查客户端的各种功能，看是否存在敏感信息泄露问题。

测试步骤

人工测试。使用错误的登录名或密码登录，看客户端提示是否不同。在显示卡号等敏感信息时是否进行部分遮挡。

威胁等级

若在用户名输入错误和密码输入错误时提示信息不同则存在 UI 信息泄露问题，此时为低风险，否则无风险。

安全建议

注意 UI 信息的防护。

8. 验证码安全

描述

测试客户端在登录和交易时是否使用图形验证码。验证码是否符合如下要求：由数字和字母等字符混合组成；采取图片底纹干扰、颜色变换、设置非连续性及旋转图片字体、变异字体显示样式等有效方式，防范恶意代码自动识别图片上的信息；具有使用时间限制并仅能使用一次；验证码由服务器生成，客户端文件中不包含图形验证码文本内容。

测试步骤

观察验证码组成，若简单，可以尝试使用 PKAVHttpFuzzer 的验证码识别工具进行识别：

也可以使用接地气表哥的验证码识别工具（见附录）：

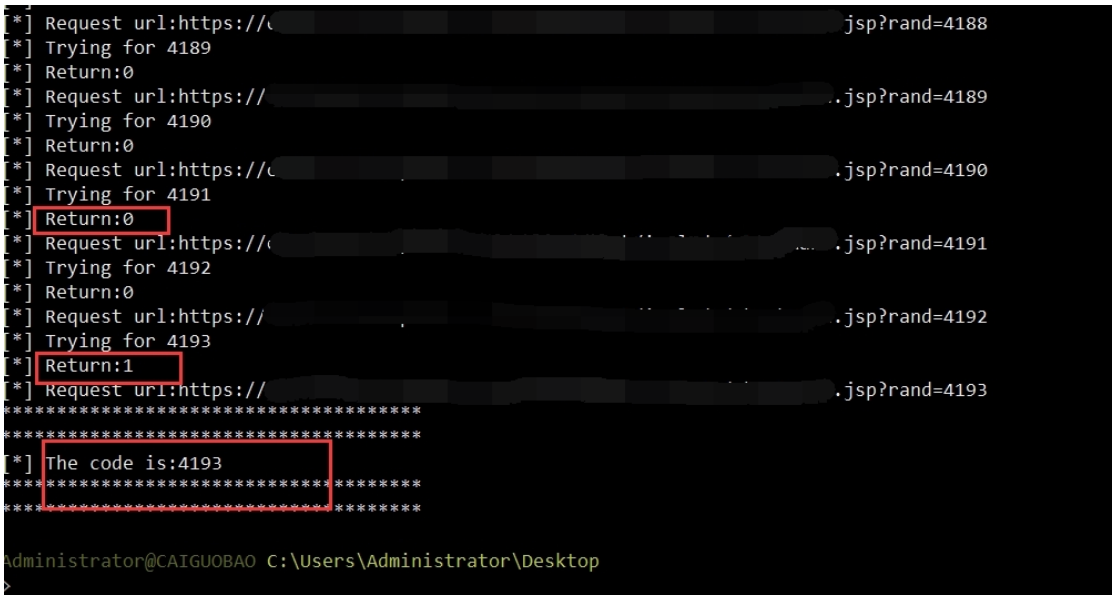
对于有的验证码验证形式为：

https://xxx.com/web/inc/CRrand.jsp?rand=xxxx

请求验证的方式，可以自己构建脚本进行验证（如下为例）：

```
# coding=utf-8
import requests
import re
s = requests.Session()
url_1 = 'https:// ‘xxxx.com/web/login.jsp'
re = s.get(url_1,verify=False)
for number in range(10000):
  url_2 = 'https://xxx.com/web/inc/CRrand.jsp?rand='+str(number).zfill(4)
  s.cookies.set('JSESSIONID', '9E426F03658582FE9CD5BF3725CFB444', path='/web/',
domain='xxxx.com')
  req = s.get(url_2,verify=False)
  result = req.text
  print '[*] Trying for ' + str(number).zfill(4)
  print '[*] Return:' + result.strip('\n')
  print '[*] Request url:' + url_2
  if '1' in req.text:
    print '**********************************'
    print '**********************************'
    print '[*] The code is:' + str(number)
    print '**********************************'
    print '**********************************'
    break
```

也可以成功的验证爆破：



（具体可参考附录中的【业务中的一次 python 验证码爆破】）

威胁等级

当图形验证码由本地生成而不是从服务器获取时为中风险；当验证码安全性低或不存在验证码时为中风险；不存在以上两个问题时无风险。

安全建议

加强验证码的识别难度，对于验证码的验证做到“一次一验”。

9. 安全退出

描述

测试客户端退出时是否正常终止会话。

测试步骤

检查客户端在退出时，是否向服务端发送终止会话请求。客户端退出后，还能否使用退出前的会话 id 访问登录后才能访问的页面。

威胁等级

若客户端退出登录时不会和服务器进行 Logout 的相关通信则为中风险，否则无风险。

安全建议

客户端退出时要做到和服务器进行 Logout 的相关通信。

10. 密码修改验证

描述

测试客户端在修改密码时是否验证旧密码正确性。

测试步骤

人工测试。

威胁等级

当进行密码修改时是否要求输入原密码已验证其正确性，若需要输入则无风险；如不需输入原密码则中风险。

安全建议

修改密码需要验证原密码的正确性。

11. Activity 界面劫持

描述

检查是否存在 activity 劫持风险，确认客户端是否能够发现并提示用户存在劫持。

测试步骤

安装 HijackActivity.apk，使用 activity 界面劫持工具，在工具中指定要劫持的应用进程名称。如图所示，从列表中选择被测试的应用，点击 OK。打开应用，测试工具会尝试用自己的窗口覆盖被测的应用。

如果劫持成功，会出现如下界面：

威胁等级

若客户端无法抵抗 Activity 界面劫持攻击时为中风险；若可以抵抗攻击则无风险。

安全建议

Activity 劫持通常依靠注册 Receiver 响应 android.intent.action.BOOT_COMPLETED 事件。客户端程序可以在启动前检查 Receiver 并报警；

由于 Activity 界面劫持攻击通常是将自己的页面附着在客户端之上，因此需要进行界面切换操作，因此在界面切换到后台时弹出警告信息也可以达到一定的效果。

除此之外，因为 Android 进程栈的工作原理，建议开发客户端时针对进程栈进行相应的保护，可禁止其他进程放置于客户端之上。

【0x07】进程保护

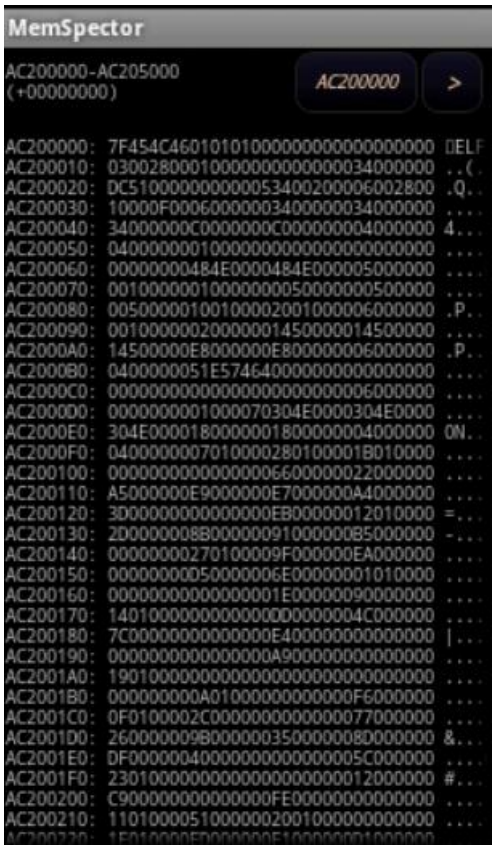
1. 内存访问和修改

描述

通过对客户端内存的访问，木马将有可能会得到保存在内存中的敏感信息(如登录密码，帐号等)。测试客户端内存中是否存在的敏感信息（账号、明文密码等等)。

测试步骤

需要 root 权限，可以使用 MemSpector 查看、搜索和修改客户端内存数据，如图所示。用户名、密码等数据通常会在/dev/ashmem/dalvik-heap 内存段。(目前大多数工具都是通过 ptrace 接口修改客户端内存，可以使用 ptrace 机制本身防护。)



威胁等级

当进行敏感操作后在内存中可以搜索到用户输入的敏感信息时为高风险，否则无风险。

安全建议

对于内存中的信息泄露，可以通过反注入、反调试来解决。

2. 动态注入

描述

通过注入动态链接库，hook 客户端某些关键函数，从而获取敏感信息或者改变程序执行。

测试步骤

检测 LD_PRELOAD 环境变量。使用 LD_PRELOAD 环境变量，可以让进程预先加载任

意 so，劫持函数。如图是劫持 ls 命令__libc_init()函数的效果。（可参考附录 Let’s Hook a Library Function）

```
# LD_PRELOAD=/data/data/debug/libhookm.so ls
LD_PRELOAD=/data/data/debug/libhookm.so ls
hooked __libc_init!
libhookm.so
```

或者使用工具动态注入应用进程内存。参考附录 ddi - Dynamic Dalvik Instrumentation Toolkit 或者使用 hook 框架来进行测试。

威胁等级

当客户端存在动态注入隐患时高风险，否则无风险。

安全建议

防止应用程序被 Hook

【0x08】通信安全

1. 通信加密

描述

如果客户端与服务器之间的通信加密协议实现不当，攻击者将有机会对当前网络环境中其他合法用户的通信内容进行窃听甚至篡改。

测试步骤

为了对服务器端进行测试，至少要先弄清楚客户端与服务器的通信协议。
一般来说，有以下三种情况：

1. 如果使用 HTTP(S)协议，大多可以通过设置系统 HTTP 代理来进行操作客户端的网络访问。这种情况占绝大多数：
   a) 在电脑上开启 Burp/Fiddler 等代理工具，并设置允许远程连接；
   b) 如果是 Emulator 虚拟机：
      i. Emulator 默认使用 3G 网络 NAT；
      ii. 在设置->无线和网络(->更多)->移动网络->接入点名称/APN->Telerik，即可设置代理地址和端口；
      iii. 一般来说，设置成物理机的出口网卡 IP 即可；

c) 如果是 Genymotion 虚拟机：
   i. Genymotion 默认使用 WIFI 网络 NAT；
   ii. 在设置->无线和网络->WLAN->长按连接的 WIFI 名称->修改网络->代理:手动，即可设置代理地址和端口；
   iii. 同上，设置成物理机的出口网卡 IP 即可；
d) 如果是真实手机：
   i. 使用 netsh wlan 开启承载网络（具体方法请自行百度）；
   ii. 用手机连接电脑的 WIFI，其余部分同 Genymotion。

2. 如果是 HTTP(S)协议，但是不接受操作系统代理：
   a) 如果设备已经 root，可以安装一个叫做“ProxyDroid”的 APP；
      i. Emulator 的 root 方法参考三、注释；
   b) 如果设备不能 root，但电脑性能充足，可以把安卓虚拟机装在 Windows 虚拟机里，在 Windows 虚拟机上安装 Proxifier，挂到物理机的代理工具上；
   c) 如果都不行，可参考 3.；
3. 如果不是 HTTP 协议：
   a) 那么一般就是由 TCP 或 UDP 实现的私有协议，大多数是 TCP；
   b) 虽然也可以用一些办法来操作 TCP 网络访问（比如用 WPE 附加到 Emulator 的进程上），但由于 TCP 是全双工流式协议，传输层上没有明确的报文边界。如果私有协议是请求-响应式的还勉强可以编辑，如果是委托-回调式，或者其它复杂形式的协议，使用通用工具进行操作是非常困难的。
   c) 这种情况可以通过 Wireshark 抓包分析，如果协议不复杂，可以自行实现代码进行仿制；
   d) 如果协议复杂，就需要对 APP 的代码进行分析，找到通信的部分，将其摘出并调用，或者自行实现代码进行仿制；

此外，通信过程如果有加密、签名等措施，通常需要从客户端代码入手，进行传统逆向分析以破解其加密。如果其实现非常复杂，此项可以认为安全。

逆向分析的常见入手位置主要有数据（字符串内容等）和特定 API（如界面、网络、文件、Native 操作等）两种。

有时会遇到客户端检查了 HTTPS 证书的情况（表现为，代理工具如果不替换 SSL 证书则正常代理，替换 SSL 证书则客户端网络异常），会有以下两种情况：

1. 客户端使用操作系统证书链验证服务端证书：
   a) 此种情况下，可以从代理工具中导出证书，然后安装到安卓设备中。
2. 客户端预存了服务端证书的公钥或 Hash，甚至整个证书：
   a) 此种情况下，如果客户端本身缺少完整性校验，可以尝试分析其代码，并修改其存储的证书信息为代理工具的证书信息；

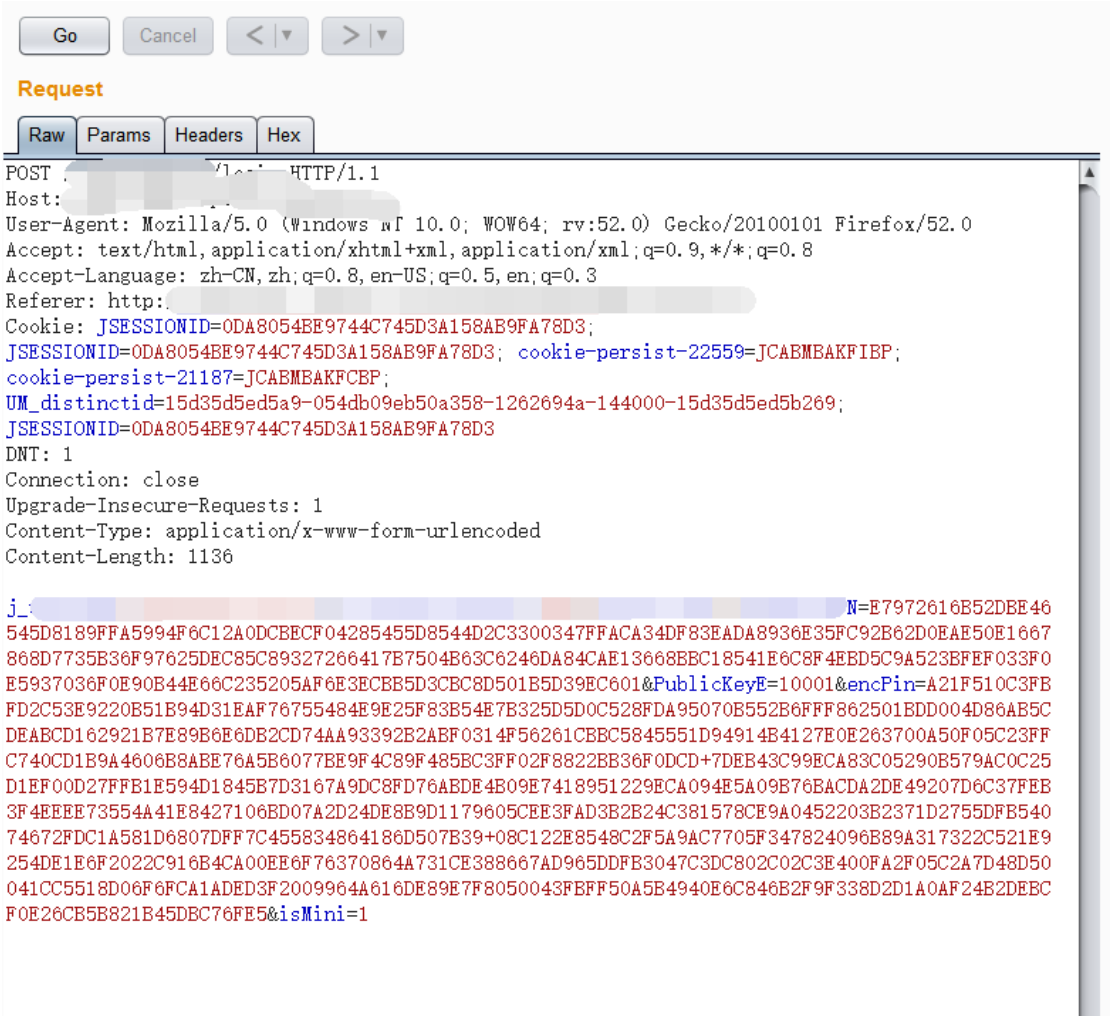
所有需要对客户端程序/设备进行操作后，才能解除的保密性或完整性措施，不算作风险。

加密签名措施的破解，最终还是要根据客户端的具体实现方式进行分析。

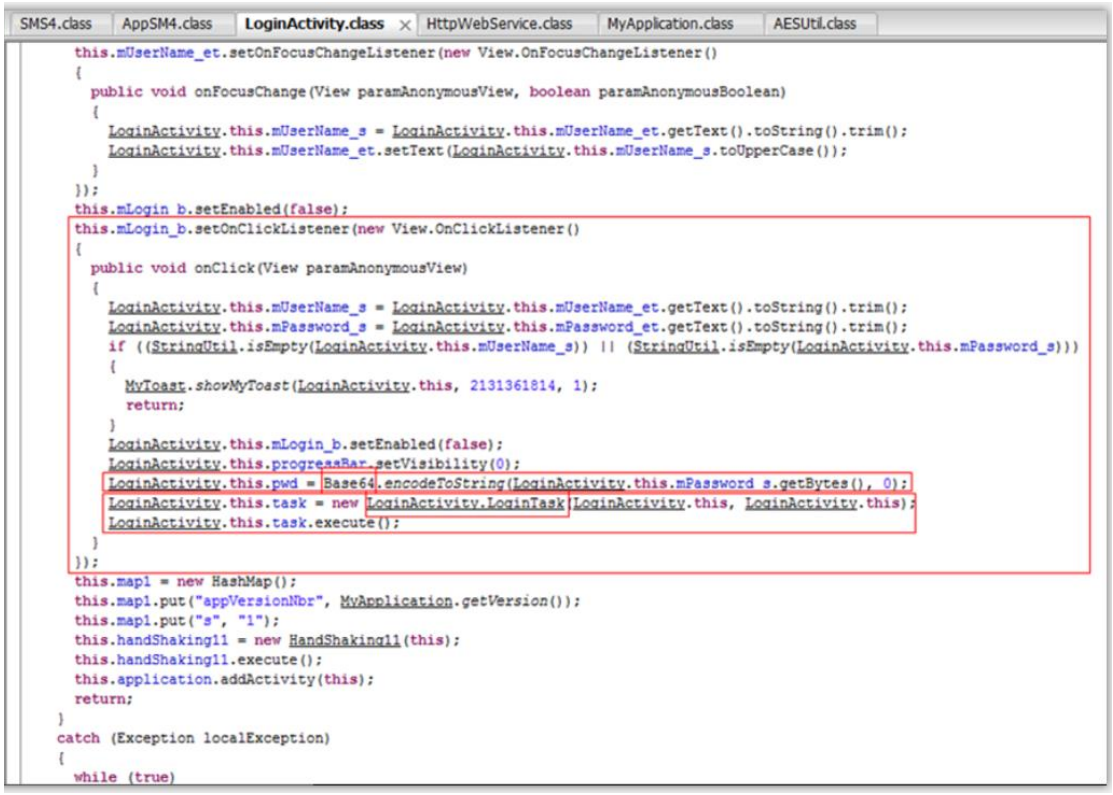
下面的例子（从界面按钮的响应入手，找到 JAVA 层常量中存储对称密钥）可供参考。

首先在手机终端设置 HTTP 代理，指向电脑上的代理工具（BurpSuite、fiddler 等）。

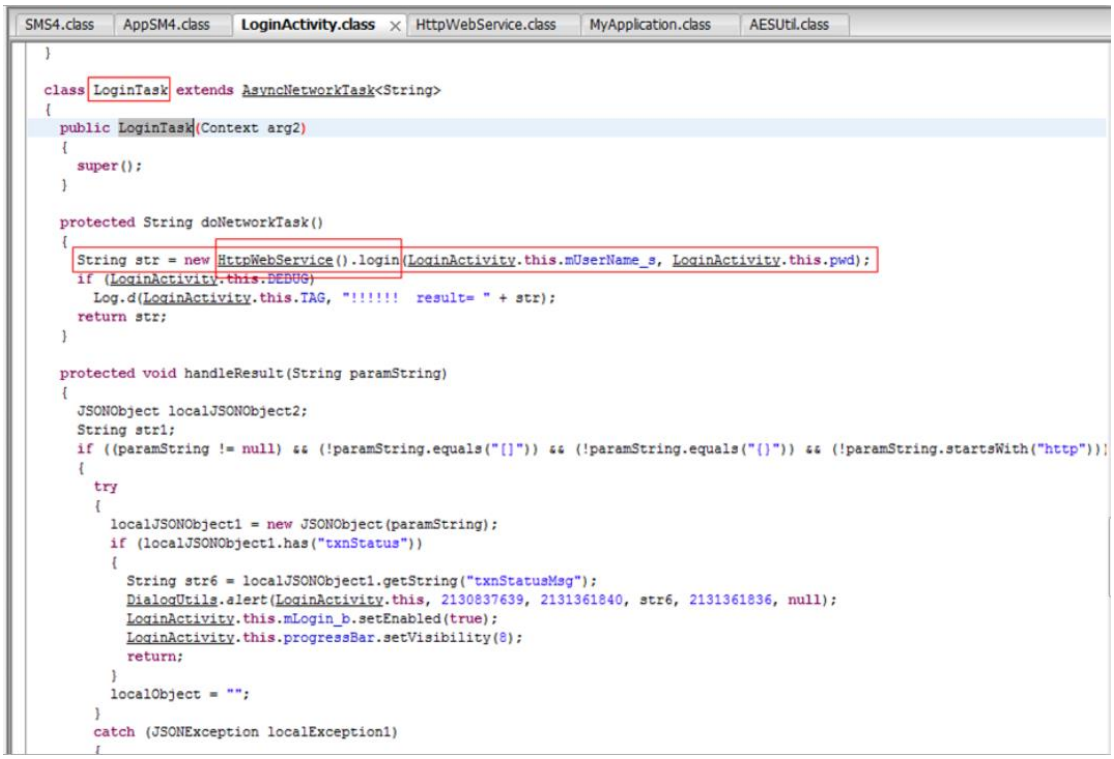
准备完成后打开手机客户端，在代理工具上替换服务器发送给客户端的证书，对客户端和服务器的通信进行 SSL 中间人攻击，成功捕获到客户端和服务器间的通信，但通信字段明显经过加密算法处理：



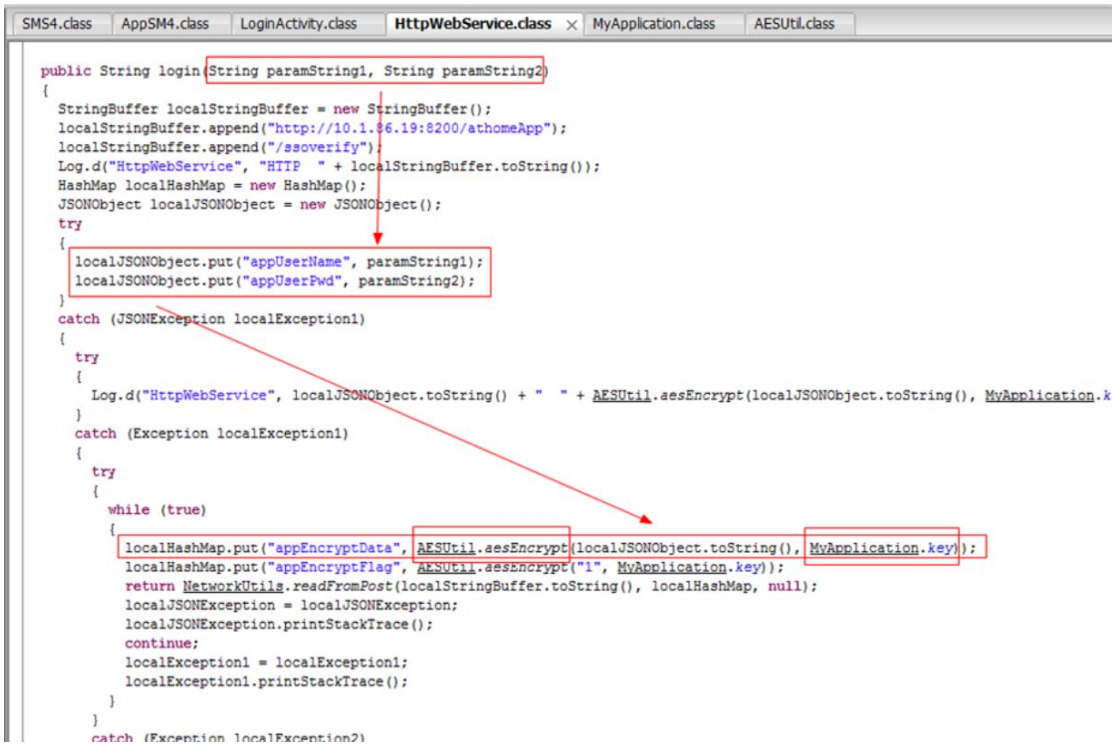
对客户端程序代码进行逆向分析，从登录按钮的 Listener 开始进行逻辑追踪，可见 Listener 对密码进行 Base64 编码后进行了异步过程调用：



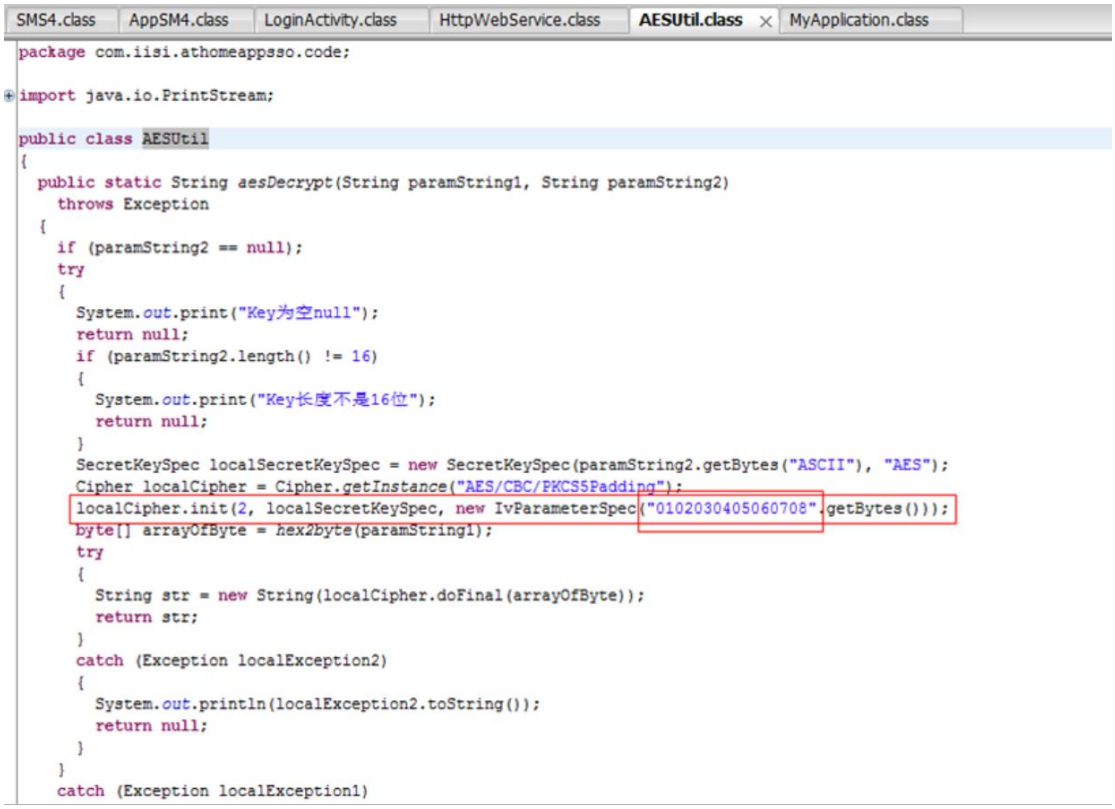
找到异步过程子程，如下：

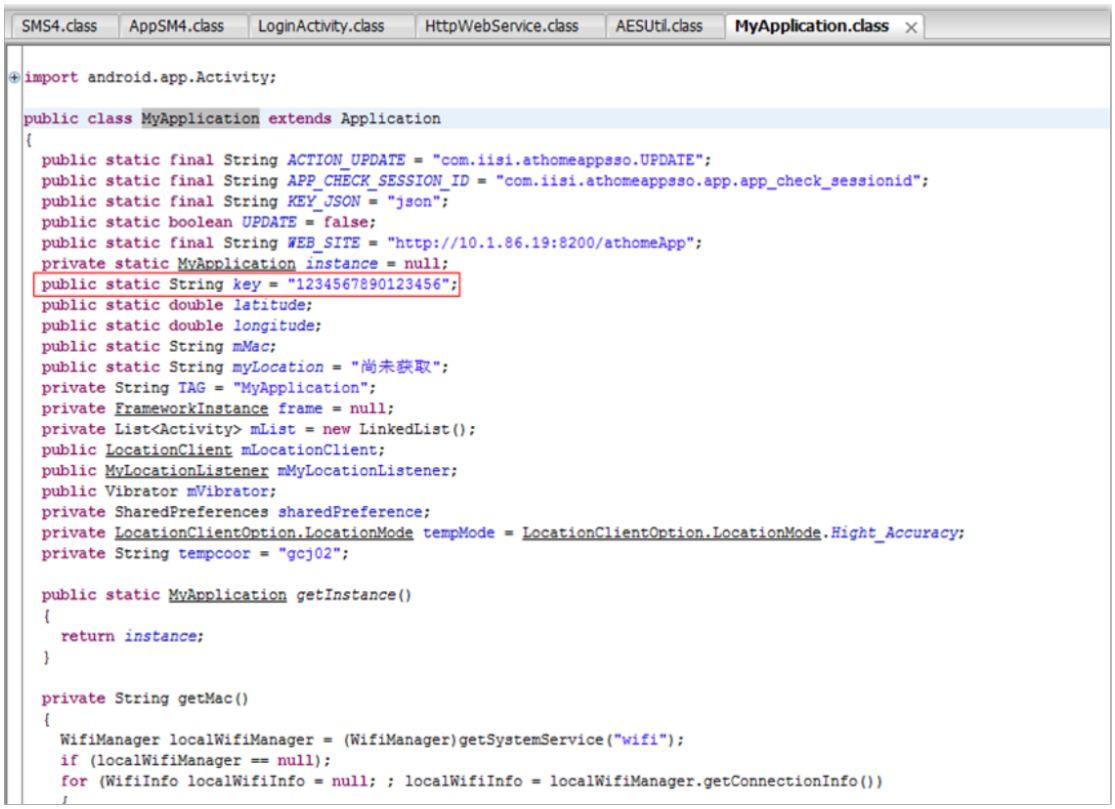


继续进行逻辑追踪，发现 HttpWebService.login(string,string)中调用了加密封装功能：



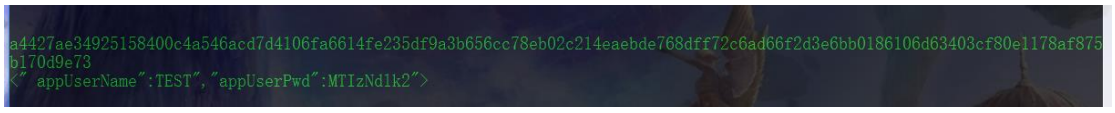
在相关功能中发现了以常量字符串形式硬编码存储的加密算法、初始化向量、和密钥：





编写代码仿制相关加密协议，成功对最初登录报文中的内容进行了解密：

```
static void Main(string[] args){
        AesCryptoServiceProvider test = new AesCryptoServiceProvider();
        test.IV = Encoding.ASCII.GetBytes("0102030405060408");
        test.Key = Encoding.ASCII.GetBytes("1234567890123456");
        test.Padding = PaddingMode.PKCS7;
        test.Mode = CipherMode.CBC;
        ICryptoTransform tdec = taes.CreateDecryptor();
        string tinput;
        while ( !sstring.IsNullOrEmpty(tinput = Console..ReadLine()))
            byte[] ttinput = EncodingEx.Hex(tinput);
            byte[] toutput =
tdec.ICryptoTransformFinalBlock(ttinput,0,ttinput.Length);
            Console.WriteLine(Encoding.UTF8.GetString(toutput));
    }
```



威胁等级

当客户端和服务器的通信不经过 SSL 加密（或没有参考 TLS 协议，RFC4346 等实现加密信道）时为高风险；当自实现通信算法存在漏洞可被解析或绕过时为高风险；使用低版本 SSL 协议（SSLV2，SSLV3 均存在漏洞，至少使用 TLSV1.1 以上算法）时为高风险；以上问题均不存在时无风险。

安全建议

建议使用 SSL 协议，并在客户端对服务端的证书进行验证。如果自行实现加密协议，建议在客户端预先存储服务端公钥，在建立会话时随机生成对称加密密钥，用服务端公钥加密并发送给服务端，随后服务端用私钥解密后，正式开始进行通信。加密过程尽量避免使用 CBC 模式。

2. 证书有效性

描述

主要测试 SSL 协议安全性、SSL 证书验证等。

测试步骤

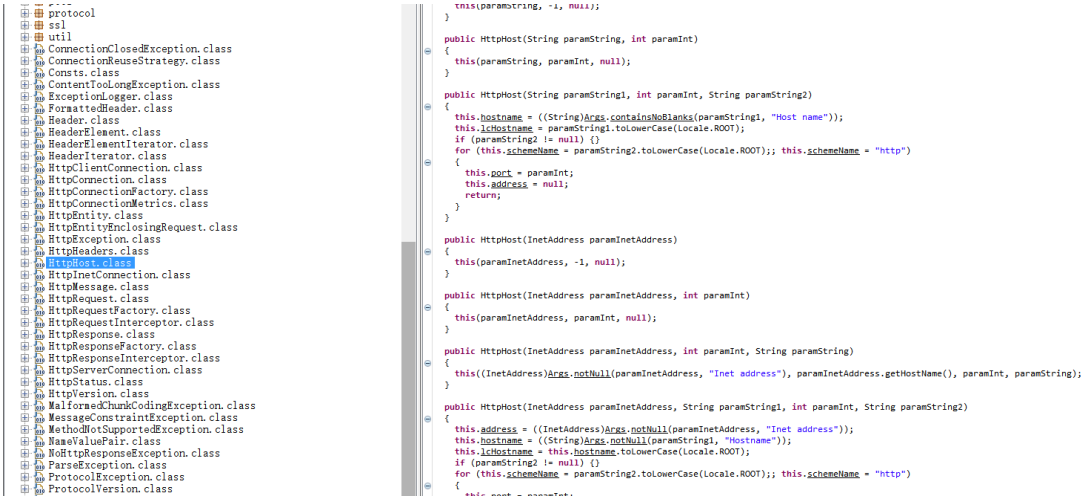
首先测试客户端程序是否严格检查服务器端证书信息。

通过修改 DNS，将客户端链接到的主页地址改为 https://mail.qq.com/，然后使用客户端访问服务端，查看客户端是否会提示连接错误。此项测试主要针对客户端是否对 SSL 证书中的域名进行确认。

也可以查阅代码中是否有 SSL 验证。下图是 Java 中进行服务端 SSL 证书验证的一种方式。关键函数为：java.net.ssl.HttpsURLConnection.setDefaultHostnameVerifier()，通过此函数查找 HostnameVerifier 的 verify 函数。如 verify()函数总返回 true，则客户端对服务端 SSL 证书无验证。（可能还有其他 SSL 实现，需要验证）。

```
{
  public boolean verify(String paramString, SSLSession paramSSLSession)
  {
    PrintStream localPrintStream = System.out;
    String str = "hostname: " + paramString;
    localPrintStream.println(str);
    return true;
  }
};
this.hnv = local2;
try
{
  SSLContext localSSLContext = SSLContext.getInstance("TLS");
  X509TrustManager[] arrayOfX509TrustManager = new X509TrustManager[1];
  X509TrustManager localX509TrustManager = this.xtm;
  arrayOfX509TrustManager[0] = localX509TrustManager;
  SecureRandom localSecureRandom = new SecureRandom();
  localSSLContext.init(null, arrayOfX509TrustManager, localSecureRandom);
  label74: if (localSSLContext != null)
    HttpsURLConnection.setDefaultSSLSocketFactory(localSSLContext.getSocketFactory());
  HttpsURLConnection.setDefaultHostnameVerifier(this.hnv);
  return;
}
catch (GeneralSecurityException localGeneralSecurityException)
```

如土司的相关验证：



具体可参考附录：HttpsURLConnection。

还需要检测 SSL 协议安全性。主要是检测客户端使用的 SSL 版本号是否不小于 3.0（或 TLS v1），加密算法是否安全。（安全规范要求）。

测试如下：

Openssl s_client -host www.t00ls.net -port 443

```
root@kali:~/Desktop# openssl s_client -host www.t00ls.net -port 443
CONNECTED(00000003)
depth=2 C = US, ST = Arizona, L = Scottsdale, O = "GoDaddy.com, Inc.", CN = Go Daddy Root Certificate Authori
ty - G2
verify return:1
depth=1 C = US, ST = Arizona, L = Scottsdale, O = "GoDaddy.com, Inc.", OU = http://certs.godaddy.com/reposito
ry/, CN = Go Daddy Secure Certificate Authority - G2
verify return:1
depth=0 OU = Domain Control Validated, CN = www.t00ls.net
verify return:1
---
Certificate chain
 0 s:/OU=Domain Control Validated/CN=www.t00ls.net
   i:/C=US/ST=Arizona/L=Scottsdale/O=GoDaddy.com, Inc./OU=http://certs.godaddy.com/repository//CN=Go Daddy Se
cure Certificate Authority - G2
 1 s:/C=US/ST=Arizona/L=Scottsdale/O=GoDaddy.com, Inc./OU=http://certs.godaddy.com/repository//CN=Go Daddy Se
cure Certificate Authority - G2
   i:/C=US/ST=Arizona/L=Scottsdale/O=GoDaddy.com, Inc./CN=Go Daddy Root Certificate Authority - G2
 2 s:/C=US/ST=Arizona/L=Scottsdale/O=GoDaddy.com, Inc./CN=Go Daddy Root Certificate Authority - G2
   i:/C=US/O=The Go Daddy Group, Inc./OU=Go Daddy Class 2 Certification Authority
 3 s:/C=US/O=The Go Daddy Group, Inc./OU=Go Daddy Class 2 Certification Authority
   i:/C=US/O=The Go Daddy Group, Inc./OU=Go Daddy Class 2 Certification Authority
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIFIDCCBAigAwIBAgIJAKOfJaVA+JOeMA0GCSqGSIb3DQEBCwUAMIG0MQswCQYD
VQQGEwJVUzEQMA4GA1UECBMHQXJpem9uYTETMBEGA1UEBxMKU2NvdHRzZGFsZTEa
MBgGA1UEChMRR29EYWRkeS5jb20sIEluYy4xLTArBgNVBAsTJGh0dHA6Ly9jZXJ0
```

```
---
New, TLSv1.2, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES256-GCM-SHA384
    Session-ID: AD57FA3DE3BA648AFAC9000CF70F3E1D546A3985DF2E1CFDE96309AF67FB6C81
    Session-ID-ctx:
    Master-Key: F3C6BEBD88AA78A4783ED5E93304A62482FE381F6D7EAAEA829CBFB6CD9577095C5A6045B0B70DE36C21B15D3587A
BE3
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 300 (seconds)
    TLS session ticket:
    0000 - 1b 33 48 c6 08 fd d1 09-8b 9f a1 34 2d e7 c2 0f   .3H........4-...
    0010 - 46 50 1a f8 a8 c0 31 f9-74 cc 06 98 5b 07 f3 ca   FP....1.t...[...
    0020 - 5d 22 8a 70 41 4c 86 61-48 2c dc b6 98 65 21 77   ]".pAL.aH,...e!w
    0030 - 37 97 dc 12 dc 06 d1 c5-63 34 21 ff f1 33 c2 ed   7.......c4!..3..
    0040 - b9 98 20 54 c4 f7 3a e2-ee e4 89 15 e5 b3 d8 05   .. T..:.........
    0050 - f7 dd ea df f3 82 1f ea-63 1b d5 22 e4 6d f6 68   ........c.."m.h
    0060 - 93 aa c1 82 0e b2 70 4f-5c b0 c8 dc f7 a6 6e 60   ......pO\.....n`
    0070 - 4b 5d d6 dd aa e7 c4 ca-92 9c af 55 34 39 e4 5d   K].........U49.]
    0080 - 3d 8f 22 68 9e 24 d3 89-f7 af 1b 0c 15 d2 61 58   =."h.$........aX
    0090 - fd 84 76 65 f8 b2 74 c2-bb e8 66 48 a7 68 76 1f   ..ve..t...fH.hv.
    00a0 - 34 75 73 0a 51 2a c0 d3-9b bb e8 38 a9 bd fa 2c   4us.Q*.....8...,

    Start Time: 1500793684
    Timeout   : 7200 (sec)
    Verify return code: 0 (ok)
    Extended master secret: yes
```

威胁等级

当客户端和服务器互相不验证证书时高风险，当只有客户端验证服务器证书时为中风险；当服务器不通过白名单的方式验证客户端时为中风险；当客户端和服务器进行双向认证，并且服务器通过白名单方式验证客户端证书时无风险。

安全建议

使用 Https 方式进行加密，且服务器通过白名单方式验证客户端证书。

3. 关键数据加密和校检

描述

测试客户端程序提交数据给服务端时，密码、收款人信息等关键字段是否进行了加密，防止恶意用户嗅探到用户数据包中的密码等敏感信息。

测试步骤

在手机上配置好代理，观察客户端和服务端的交互数据。检查关键字端是否加密。

如果客户端对根证书进行了严格检测，导致代理无法使用。则可以将代理的根证书安装到设备上，使根证书可信。或是替换客户端 apk 中的根证书文件。如果上述方法均失效，则反编译为 java 代码，将客户端逆向后，通过阅读 java 代码的方式寻找客户端程序向服务端提交数据的代码，检查是否存在加密的代码。

威胁等级

当账号，密码，卡号等数据明文传输，未进行二次加密时为高风险；当密码只进行了单项散列而未经过加密时为高风险；当返回数据中包含更新的 URL 且数据不加密时为高风险；当校验字段删除后服务器仍会处理所发送的数据包时为高风险；当校验字段的散列中不包含随机因子时为高风险。以上问题均不存在时无风险。

安全建议

对于重要的敏感信息传输时要进行严格的加密和校检。

4. 访问控制

描述

测试客户端访问的 URL 是否仅能由手机客户端访问。是否可以绕过登录限制直接访问登录后才能访问的页面，对需要二次验证的页面（如私密问题验证），能否绕过验证。

测试步骤

人工测试。在 PC 机的浏览器里输入 URL，尝试访问手机页面。

威胁等级

当 PC 端也可访问手机页面时低风险，当可以绕过登陆限制访问登陆后才能访问的页面时中风险。

安全建议

对也页面访问权限进行严格控制。

5. 客户端更新安全性

描述

测试客户端自动更新机制是否安全。如果客户端更新没有使用官方应用商店的更新方式，就可能导致用户下载并安装恶意应用，从而引入安全风险。

测试步骤

使用代理抓取检测更新的数据包，尝试将服务器返回的更新 url 替换为恶意链接。看客户端是否会直接打开此链接并下载应用。在应用下载完毕后，测试能否替换下载的 apk 文件，测试客户端是否会安装替换后的应用。

威胁等级

当客户端返回明文 URL 地址并可以通过篡改的方式控制用户下载恶意 APK 包进行安装，则高风险；若返回数据包经过二次加密则无风险。

安全建议

对于返回的数据包要进行二次加密处理。

6. 短信重放攻击

描述

检测应用中是否存在数据包重放攻击的安全问题。是否会对客户端用户造成短信轰炸的困扰。

测试步骤

尝试重放短信验证码数据包是否可以进行短信轰炸攻击

威胁等级

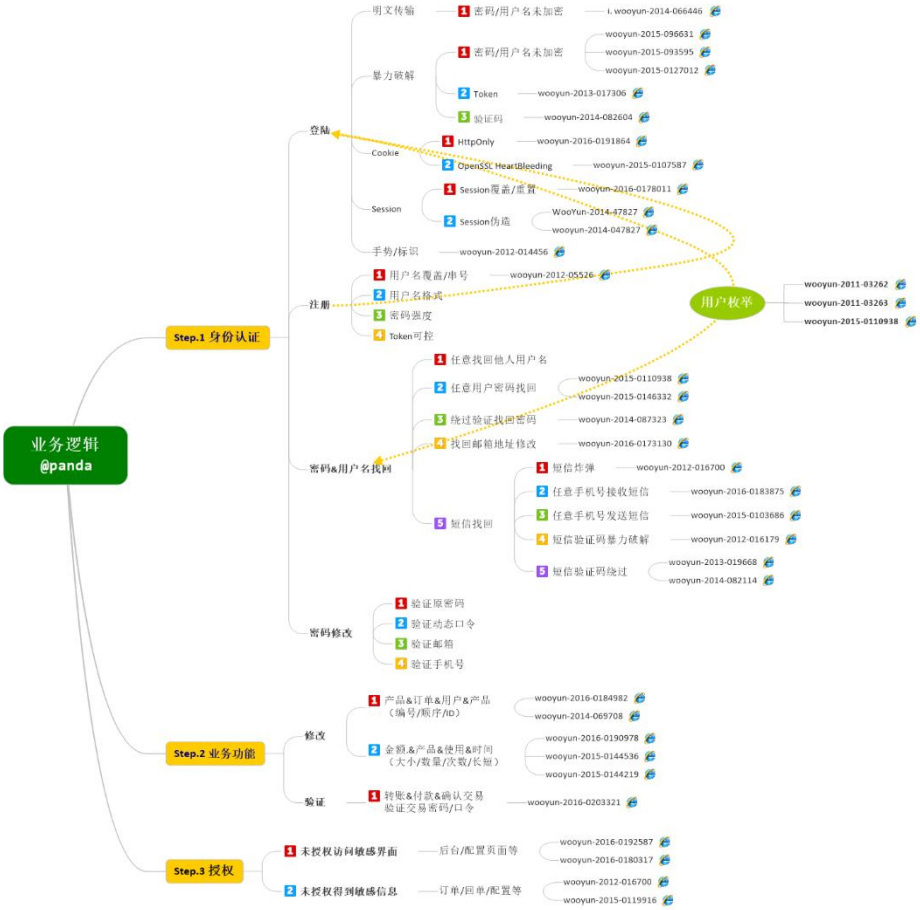
当存在短信轰炸的情况时为中风险，若短信网关会检测短时间内发送给某一手机号的短信数量则无风险。

安全建议

对于验证码的发送要进行相关身份验证，如：验证码。

【0x09】业务安全

业务安全基本上检测的业务逻辑安全。对于业务逻辑安全在进行测试的时候可以参考我的业务逻辑渗透测试思维导图：



由于手机 APP 的业务安全和 web 测试并无太大差别，因此这里就只是简单的说明。

1. 越权操作

描述

服务器端对客户提出的数据操作请求过分信任，忽略了对该用户操作权限的判定，导致攻击账号拥有了其他账户的增删改查功能。

测试步骤

属于业务逻辑相关的测试，因此需要手工测试。可参考附录：我的越权之道。

威胁等级

根据越权操作的危害性进行定级。

安全建议

1、基础安全架构，完善用户权限体系。要知道哪些数据对于哪些用户，哪些数据不应该由哪些用户操作；

2、鉴权，服务端对请求的数据和当前用户身份做校验；

2. 交易篡改

描述

本项测试主要是修改金额信息（如：转帐金额为负值），订单信息（如：订单的数量）等

测试步骤

手工抓包测试。

威胁等级

高（此项业务一般为核心业务，只要能修改相关信息，基本上都是高危漏洞）

安全建议

服务端要做到严格的验证。（根据业务情形不同，安全建议也不相同，因此这里只能简单的一句话描述）

3. 重放攻击

描述

主要就是进行抓包重放（如：重放产品购买、订单创造等）测试。

测试步骤

人工测试。

威胁等级

根据重放业务进行判定危险等级

安全建议

根据业务具体情境提供不同的安全建议。

4. 用户枚举

描述

尝试爆破枚举用户。

测试步骤

手工测试。
此类漏洞情境一般是：登录界面无验证码、有明显的返回信息（如：该账号不存在、密码错误等）

威胁等级

要看公司的规模、用户名的形式（如：身份证号码）等来判定危险等级

安全建议

一般的安全建议是设置严格的登录信息验证，如验证码、手机验证码等。

5. 暴力破解

描述

主要是测试业务中查询、登录等功能，尝试使用暴力枚举的方式进行破解。

测试步骤

手工测试。

威胁等级

根据暴力破解的业务和结果进行判定（如：若成功爆破出用户账号和密码，风险等级为高）。

安全建议

一般的安全建议是设置严格的信息验证，如验证码、手机验证码等。

6. 注入/XSS/CSRF

描述

和 WEB 测试类似，主要测试站点存在的常见的 web 漏洞。

测试步骤

可手工测试也可以使用工具扫描、测试。

威胁等级

此类漏洞一般为高，具体的可以看存在漏洞的业务进行判定。

安全建议

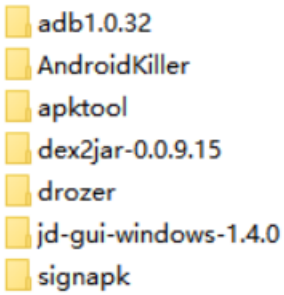
根据 WEB 漏洞的分类进行建议，可参考附录：渗透测试报告中的一些安全建议总结。

【0x10】附录

1. 测试工具

[*] signapk
[*] jd-gui-windows
[*] dex2jar
[*] apktool
[*] adb
[*] AndroidKiller
[*] drozer
[*] ddi

其中部分工具



下载链接：链接：http://pan.baidu.com/s/1qYv3bzQ 密码：xlkf

2. 参考资料

[*] API Guides 系统权限简介
[*] drozer 官方说明
[*] 【安全测试】Drozer 安装及使用
[*] Drozer - Android APP 安全评估工具小测
[*] 国产工具：渗透测试助手 PKAV HTTP Fuzzer 发布
[*] 业务中的一次 python 验证码爆破
[*] 【庆新春】--->>验证码绕过及识别专题
[*] Let’s Hook a Library Function
[*] ddi - Dynamic Dalvik Instrumentation Toolkit
[*] 动态注入技术
[*] HttpsURLConnection
[*] 我的越权之道
[*] 渗透测试报告中的一些安全建议总结