## 移动APP恶意行为识别与逆向分析技术研究

技术方案V1.0

### 1主要目标

本项目的主要目标是:

1. 建立移动APP恶意行为识别模型;
2. 开发恶意行为识别工具软件;
3. 研究逆向分析技术方法。

### 2技术内容

移动APP恶意行为识别与逆向分析技术研究主要包括2个部分：移动APP恶意行为识别与逆向分析技术研究。

#### 2.1 移动APP的恶意行为伪装与识别技术研究

移动APP的恶意行为伪装与识别技术研究，包括：

1. APP恶意行为归类（支付类，隐私类，远控类等等）；
2. APP恶意行为已知伪装技术研究和未知伪装技术预测；
3. APP恶意行为已知识别技术研究和未知识别技术创新；
4. APP恶意行为防范方法。

#### 2.2基于智能终端APP的逆向分析技术研究

基于智能终端APP的逆向分析技术研究，包括：

1. 基于Android的APP逆向工具研究；
2. 基于Android的APP逆向研究方法；
3. 基于IOS的APP逆向工具研究；
4. 基于IOS的APP逆向研究方法。

### 3总体需求

移动APP恶意行为识别与逆向分析技术研究项目的两个部分同时进行。两个部分均具有理论探索研究性质，因此项目实施过程中采取科学研究与软件工程相结合的模式。需要按照下列进度完成研究开发工作：

（1）移动APP的恶意行为伪装与识别技术研究：

* 第一阶段：Android和IOS相关技术资料收集分析；
* 第二阶段：关键技术突破;
* 第三阶段：研究成果实践及论文发布；

（2）基于智能终端APP的逆向分析技术研究：

* 第一阶段：技术资料收集分析;
* 第二阶段：关键技术突破、工具研究;
* 第三阶段：研究成果实践及论文发布。

基于以上总体研究路线，需要完成以下研究成果：

1. 移动APP的恶意行为伪装与识别技术方法手册及软件；
2. 移动APP的逆向分析技术方法手册及软件；
3. 提供开展移动APP研究实验所涉及的过程文档和录制的视频资料；
4. 协助甲方申请软件著作权1~2项，与甲方联合申请国家或省内资金支持课题一项。

## 4研究开发计划

本项目分为两部分，包括移动APP恶意行为识别与逆向分析技术研究

### 4.1移动APP的恶意行为伪装与识别技术研究

结合中国目前移动应用市场的实际情况，SecApp Lab联合OWASP中国、百度、互联网安全研究中心、通付盾等应用安全联盟成员，在对上百万应用安全分析，并参考行业各类资源，发布十大移动应用恶意行为。

1. 山寨应用 —— 二次打包，伪装成流行应用吸引用户下载实现攻击；
2. 隐私窃取 —— 窃取用户联系人信息、短信信息、照片信息、身份信息、银行卡信息等；
3. 资费消耗 —— 消耗用户网络套餐资源；
4. 恶意扣费 —— 通过恶意发送短信、订阅收费服务等扣除用户账户资金；
5. 远程控制 —— 远程控制用户手机设备，形成移动设备僵尸网络；
6. 窃取资金 —— 窃取用户移动数字钱包或网银资金；
7. 恶意传播 —— 利用传播路径漏洞进行恶意传播，影响市场规划；
8. 静默下载 —— 静默下载安装恶意应用，实现多方式攻击；
9. 跨平台感染 —— 通过移动设备与PC的USB连接感染PC设备；
10. 系统破坏 —— 破坏移动设备系统正常运作。

各类恶意行为所占的比分比如下表格：

|  |  |
| --- | --- |
| 恶意行为 | 比例 |
| 恶意扣费 | 30.0% |
| 山寨应用 | 20.2% |
| 静默下载 | 17.4% |
| 窃取隐私 | 14.5% |
| 恶意传播 | 5.1% |
| 话费消耗 | 5.0% |
| 窃取资金 | 4.9% |
| 远程控制 | 1.4% |
| 系统破坏 | 1.0% |
| 跨平台感染 | 0.5% |

#### 4.1.1 Android平台恶意行为伪装与识别技术研究

由于Android系统为开源系统，编程语言使用跨平台语言java，而且应用发布市场极不规范，所以APP市场一度呈现鱼龙混杂状态，充斥着各类恶意软件。基于此现状，市场上也出现了各类安全应用，通常采用的方法是基于已有APP特征库进行算法建模，从而产生恶意行为特征库，用于匹配新的APP是否属于恶意软件。下文将对实现技术及其流程进行讨论。

##### 4.1.1.1 获取Android APP资源库

在构建Android恶意行为特征库之前需要先获取市场已有APP作为分析库，需要获知这些应用是否具有恶意行为。

据统计数据，从Google Play下载的Android APP具有恶意行为的概率仅有0.5%。所以，一般认为从Google Play下载的APP是安全的。鉴于此，可以使用脚本程序从Google Play抓 取应用作为非恶意软件。

除此以外，还可以从众多的开源社区直接获取已知恶意属性的APP作为特征分析源。

##### 4.1.1.2 特征提取

通过特征提取，可以获取到Android应用的基本特征数据，比如权限信息、系统API调用、lib库数据等。可以通过静态分析和动态分析方法提取Android应用的特征数据。

静态分析方法主要通过对静态文件的分析来判断系统是否具有恶意行为的可能。常用的方法有如下几种：

（1）从AndroidManifest.xml获取权限等信息

Android程序需要在AndroidManifest.xml文件中注册权限、服务、Activity等信息。因此，只需要通过反编译技术获取AndroidManifest文件，即可获知该程序所有的基本信息，从而分析出该程序是否具有恶意行为的可能。

（2）获取lib库、字符串常量信息

Android调用的第三方库都位于libs文件夹下，通过分析该文件夹，可以获知应用是否调用了具有恶意行为的第三方库。通过分析strings.xml常量文件，可以获知应用是否具有敏感下载链接或者提示用户下载的行为等。

（3）从dex字节码文件获取系统API调用信息

阅读反汇编生成的Dalvik字节码，可以使用IDA Pro分析dex文件，或者使用文本编辑器阅读baksmali反编译生成smali文件。通过对Dalvik字节码的分析，可以获取到应用对系统API，例如短信API，通讯录API的调用情况，从而判断出应用是否具有恶意行为的可能。

（4）从java源代码获取系统API调用信息

通过反编译出来的java源码可以很容易的获取到应用对系统API的调用情况。如果应用调用了发送短信接口、网路连接接口等敏感借口，则该应用有较高的可能性具有恶意行为。

动态分析方法可以弥补了静态方法不能检测运行中释放的恶意代码的缺点，也绕过了静态方法遇到的代码加密和混淆问题。

动态分析主要采用沙盒技术，自动完成启动模拟器、安装应用程序、模拟用户行为和提取系统调用序列等工作。这里可以使用adb shell指令来控制软件的安装和卸载，使用monkey或monkeyrunner等来模拟用户输入。

##### 4.1.1.3 建立特征库

通过提取到的特种数据和学习算法可以建立出特征库。可以选用的算法有K means、朴素贝叶斯、K最近邻分类算法（KNN）、EM最大期望算法、Boyer-Moore模式匹配算法、决策树、支持向量机等。通过这些分类算法和聚类算法，可以生成效果不一的特征库，具体效果取决去APP资源库的选择、算法参数的设置等因素。

##### 4.1.1.4 行为检测

在拥有了恶意行为特征库后，可以通过使用匹配算法判断出新的APP是否具有恶意行为。

#### 4.1.2 IOS平台恶意行为伪装与识别技术研究

相比Android平台，IOS平台具有恶意行为的软件数量极少，这主要归功于IOS的闭源系统和规范的应用发布平台，尤其对于未越狱的IOS设备，基本没有恶意软件的威胁。但即便如此，IOS平台仍存在一定的安全隐患，以下将对安全隐患及相应的安全防范措施进行维系。

##### 4.1.2.1 iOS APP安全隐患

2015年，针对部分App使用ROP（return-oriented programming）来隐藏和混淆恶意函数，来达到突破沙盒获取用户隐私的目的。罗马尼亚布加勒斯特理工大学的Mihai Bucicoiu等人设计了一个加固沙盒，防止容易函数访问的Framework。

南澳大利亚大学的Christian D’Orazio等人在今年的IEEE computer society上发表文章，针对部分App获取用户敏感信息，结合静态和动态两种检测方式，提出了一种分析恶意行为，鉴定App设计缺陷的方法——iOSAAP（iOS app analysis process）。

上海交通大学的贺宇轩等人，在2014月6月的通信技术上发表文章，分析了iOS系统越狱后的安全隐患，并据此提出了恶意程序检测手段，及相关的安全加固措施。

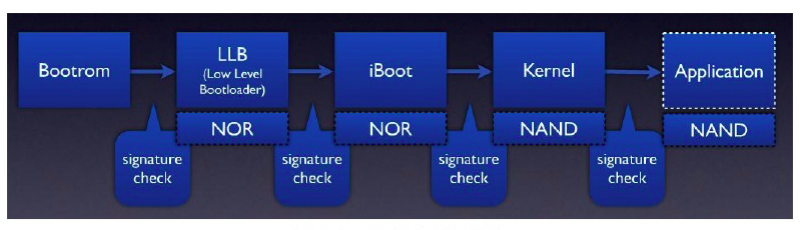
在2011年巴塞罗那举办的Black Hat Briefings安全系列会议上,Nitesh Dhijam 发表了新时代针对iOS平台的应用程序攻击报告,报告中指出了针对应用程序对 URLSchemes协议的处理缺陷的攻击以及针对Tple的APNS服务的欺骗性攻击,同时 还介绍了中间人攻击、隐私泄露攻击、身份显现攻击等攻击方法以及应对这些攻击可 以采取的对策等。

遗憾的是，直至今日URLSchemes的设计缺陷苹果官方并没有完美解决。2015年3月，港大的同学进一步发现相关漏洞，恶意应用可利用该漏洞进行URL Scheme 劫持，在未越狱的 iPhone 6上达到盗取支付宝和微信支付的帐号密码的目的。

##### 4.1.2.2 iOS安全防护措施研究

1. 系统安全（System Security）

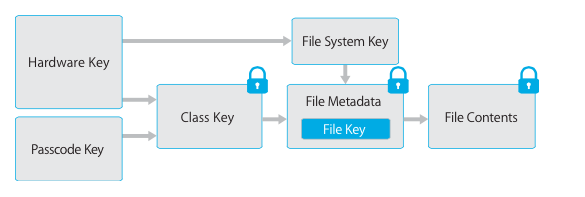
iOS系统启动过程的每一个步骤都包含了由Apple签名加密的组件用以保证该步骤的正确以及完整性,而且仅当验证了信任链过后步骤才得以进行,这些加密的部件包括bootloader、kernel、kernel extensions、还有baseband firmwareo每当一台iOS设备开机,它的应用处理器会立刻执行Boot ROM的只读内存上执行代码,这些无法被更改的代码在硬件芯片制造之初就被植入,因此显然是值得信任 的。Boot ROM的代码包含了Apple的根证书公匙,该公匙是用来在Low-Level Bootloader(LLB)加载之前验证其是否具有正确的Apple签名。当LLB执行完它的任务后,它就会验证和运行下一阶段的bootloader,iBoot,最后由iBoot验证并且启动iOS 内核。整个iOS的启动过程如下图所示:



这条安全启动链确保系统的最底层的软件不会被非法篡改也同时确保了 iOS启动只会在经过验证的iOS设备上运行。如果这条启动过程中的任何一个步骤验证出了问题,启动过程就会终止并强制系统进入恢复模式(Recovery Mode),如果甚至连Boot Rom都无法成功的启动LLB,那么设备会直接进入DFU工厂模式(Device Firmware Upgrade)。

1. 加密及数据保护（Encryption and Data Protection）

相较于PC等传统桌面设备,移动设备有着丢失、被盗、可能一直处于开机并且联网状态,因此Apple在iOS中引入了数据保护(Data Protection)技术来保护存储在设备flash内存上的用户数据。从iOS4开始Apple就提供了数据保护应用程序接口给iOS应用程序开发者用来更有效的保护好保存在文件与keychain中的数据。开发者只需要在声明哪些文件中的数据或者keychain里的项目是敏感数据并且在什么情况下才是可以获取的。在程序源代码中,开发者使用代表者不同保护类的常量来标记这些受保护的数据和keychain项目, 不同的保护类则以是保护的文件数据还是keychain项目、还有什么条件下(手机是否解锁等)才启用保护来区分成不同种的类。



不同的保护类是通过一个层次结构的密匙体系实现的,在这个体系里,每一个密匙都通过其他的一些密匙和数据继承而来。部分跟数据加密有关的体系如上图所示。 在这个结构的根部是UID密匙与用户密码。UID是一个设备唯一的、存在于板载加密运算器芯片里的数据,不可以直接被读取,但是可以用它来进行加密解密数据。每当设备被解锁过后,设备密码就会经过一个改进后PBKDF2算法进行多次加密运算以生成设备密码密匙(passcode key),设备密码密匙会一直存在在内存中直到用户再次锁定设备后才会被销毁。UID密匙还被用来加密一个静态字符串以生成设备密匙(device key),设备密匙用来加密各种代表着文件相关的保护类的类密匙(class key)。有一些类密匙也会同时经过设备密码密匙加密,这样能保证该类密匙只有在设备被解锁时才有效。数据保护应用程序接口(Data Protection API)是用来让应用程序申明文件或者keychain项目在什么时候被加密以及通过向现有的应用程序接口里面添加新定义的保护类标记使得这些加密后的文件或者keychain项目随时能重新被解密。要对某个文件进行保护,应用程序需要使用NSFileManager这个类NSFileProtectKey设置一个值,所有支持的值以及他们代表的含义如下表所示:

|  |  |
| --- | --- |
| 保护类 | 描述 |
| NSFileProtection Complete | 文件被保护，只有设备被解锁才能访问 |
| NSFileProtection CompleteUnlessOpen | 文件被保护，只有设备解锁后才能打开，但是打开后即使设备重新被锁定也可以继续使用和写入 |
| NSFileProtection CompleteUntilFirst UserAuthentication | 文件被保护直到设备被成功启动并且用户首次输入登陆密码过后才解除保护 |
| NSFileProtectionNone | 文件不被保护，任何时候都能访问 |

1. 应用安全（App Security）

一旦iOS内核启动起来,它就管控者那些进程或者程序可以送进内核得到运行,为了确保应用程序不被非法篡改,iOS要求所有执行代码必须要经过由Apple颁发的证书签名。代码签名机制受控于强制性访问控制框架(Mandatory Access Control Framework),该系统框架由FreeBSD的Trusted BSD MAC Framework继承而来。MACF允许有追加的访问控制策略,并且新的策略在框架的启动时刻被载入。 在iOS的MACF中,总共注册了两种新策略,AMFI与Sandbox。

$ otool -1 CommCenter | grep -A 5 SIGN

cmdLC CODE SIGNATURE

cmdsize 16

dataoff 128083

datasize 7424

通过对XNU的内核源码bsd/kern/mach loader.c进行分析,我们可以知道当执行文件被送去内核执行时,内核会调用parse—machiileO对该mach-o文件进行解析,在parse\_machfile()函数体内部调用load\_code\_signatureO函数载入程序签名。在载入过程中ubc\_cs\_blob\_add函数会检查签名是否合法,最终通过Mach的远程方法调用服务 向AMFID询问该签名是不是有效签名。因此我们知道当一个应用程序被成功加载后, 通过受信证书签名后的摘要信息(Hashs)也被载入,这些摘要信息存在与cs\_blob里面,并且跟实际的执行内存地址一一对应。最后在程序运行期间由AMFI对保存在cs—blob中间的摘要信息与出现在虚拟内存中间的实际代码摘要值进行比对,如果匹配 结果不一致,则产生分页错误并且终止跟该内存地址有关的进程。

### 4.2基于智能终端APP的逆向分析技术研究

#### 4.2.1 Android系统

##### 4.2.1.1 Android平台逆向工程工具研究

Google Android平台选择了Java Dalvik VM的方式使其程序很容易破解和被修改，首先APK文件其实就是一个MIME为ZIP的压缩包，通过修改ZIP后缀名方式可以看到内部的文件结构，类似Sun JavaMe的Jar压缩格式一样，不过区别是Android上的二进制代码被编译成为Dex的字节码，所有的Java文件最终会编译进该文件中去，作为代码既然虚拟机可以识别，那么就可以很轻松的反编译。所有的类调用、涉及到的方法都在里面体现到，至于逻辑的执行可以通过实时调试的方法来查看。

Android平台的逆向分析工具主要分为两类，一类工具将Dex字节码反编译为java编译后的class文件，通过其他的第三方工作可以将class文件再转化为java源码。另一类工具用户查看xml文件、资源文件等。这类文件在Android源码中一般为xml文件，包括布局文件、字符串常量等。

##### 4.2.1.2 Android平台逆向工程方法设计

Android平台逆向工程一般分为两类，一类将dex文件转换为class文件并最终转化为java源码。一类从APK中提取布局文件、字符串常量等资源文件。

（1）反编译Apk得到Java源代码

反编译Apk得到Java源代码需要两个工具dex2jar和JD-GUI。Dex2jar的作用是将apk中的classes.dex转化成Jar文件。而JD-GUI是一个反编译工具，可以直接查看Jar包的源代码。

具体步骤为，首先将Apk文件的后缀改为zip，解压，得到其中的classes.dex文件，它是java文件编译再通过dx工具打包而成的。解压dex2jar文件，将classes.dex复制到dex2jar.bat所在目录。在命令行下定位到dex2jar.bat所在目录，运行：

dex2jar.bat classes.dex

执行该命令后会生成classes.dex.dex2jar.jar文件。运行JD-GUI（jd-gui.exe），打开上面生成的jar包，即可看到源代码了。

（2）反编译Apk得到资源文件

反编译Apk得到资源文件需要使用工具apktool，在命令行下定位到apktool.bat文件夹，输入以下命令：

apktool d C:\\*.apk C:\\*文件夹

这样就可以得到包括AndroidManifest.xml在内的所有资源文件。

#### 4.2.2 IOS系统

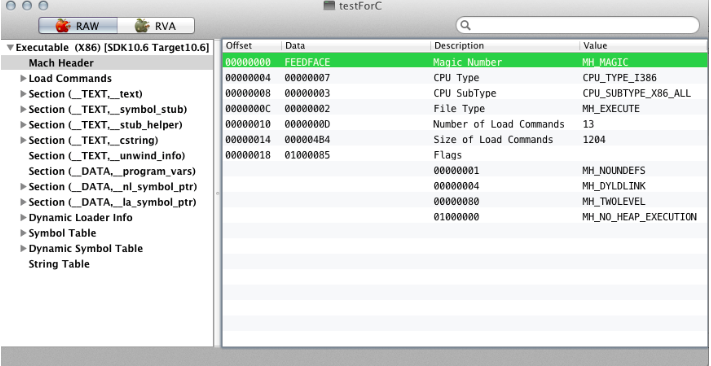
##### 4.2.2.1 IOS平台逆向工程工具研究

1. App Store应用加密破解工具

由于Appstore上的应用都采用了DRM（digital rights management）数字版权加密保护技术，一般直接用逆向工程工具进行破解得不到想要的结果，故需要工具破解DRM。常用的破解工具有：Crackulous，Clutch，PoedCrackMod，CrackTM及AppCrackr。

1. 可执行文件解析工具

MachOView：查看Mach-O文件基本信息。



1. 反编译工具

class-dump；将破解后的Mach-O可执行文件逆向工程反编译为头文件格式。

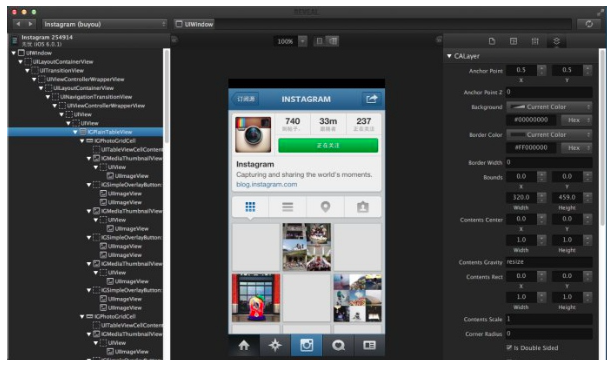


IDA：将破解后的Mach-O可执行文件逆向工程反编译为ARM64汇编指令集或伪代码。



1. UI解析工具

Reveal：可监控指定App的UI布局。



1. 断点调试工具

GDB：配合IDA使用，App运行过程中，设置断点，动态调试，确定函数作用。

1. 远程登录工具

OpenSSH：在PC或者Mac上远程输入命令操作iOS设备，免去使用MobileTerminal的不方便，同时，也是一种在多系统下管理iPhone、iPad系统文件的方法。

1. 越狱环境下文件管理工具

iTools：一款简洁有着“无需[越狱](http://baike.baidu.com/subview/36434/5381069.htm),即可同步”功能的苹果设备同步管理软件，也是继[iTunes](http://baike.baidu.com/view/365842.htm)后全球第二款支持对[iOS](http://baike.baidu.com/subview/158983/8747673.htm)管理的软件。它可以让你非常方便的完成对[iOS](http://baike.baidu.com/subview/158983/8747673.htm)设备的管理，包括信息查看、同步媒体文件、安装软件、备份[SHSH](http://baike.baidu.com/view/3992379.htm)等功能。

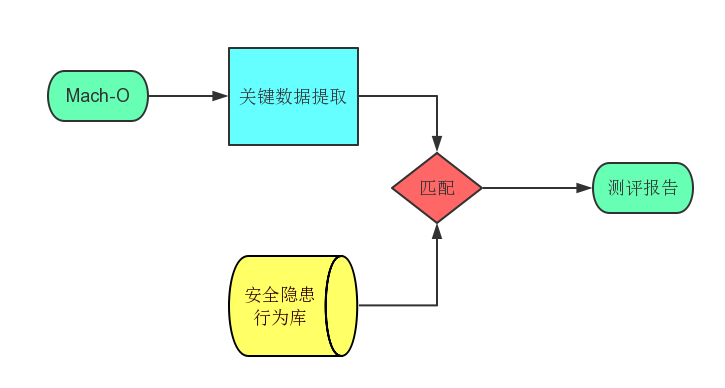
1. 越狱环境下tweak开发工具

Theos：越狱开发工具包，基于Cydiasubstrate的MobileHooker实现。跟其他越狱开发工具相比，最大的特点是：下载安装简单、Logos语法简单、编译发布简单，可以让开发者把精力都放在开发工作上去。

##### 4.2.2.2 iOS平台逆向工程方法设计

1. 静态研究方法

根据App中使用公有API获取用户隐私这一现状，利用逆向工程工具，提出设计并实现静态研究方法，对App安全隐患行为进行测评。



Mach-O文件：Mach-O为[Mach](http://zh.wikipedia.org/wiki/Mach) [Object](http://zh.wikipedia.org/wiki/%E7%9B%AE%E6%A0%87%E6%96%87%E4%BB%B6)文件格式的[缩写](http://zh.wikipedia.org/wiki/%E7%B8%AE%E5%AF%AB)，它是一种用于[可执行文件](http://zh.wikipedia.org/wiki/%E5%8F%AF%E6%89%A7%E8%A1%8C%E6%96%87%E4%BB%B6)，[目标代码](http://zh.wikipedia.org/wiki/%E7%9B%AE%E6%A0%87%E4%BB%A3%E7%A0%81)，[动态库](http://zh.wikipedia.org/wiki/%E5%87%BD%E5%BC%8F%E5%BA%AB)，[内核转储](http://zh.wikipedia.org/wiki/%E6%A0%B8%E5%BF%83%E6%96%87%E4%BB%B6)的文件格式。作为[a.out](http://zh.wikipedia.org/wiki/A.out)格式的替代，Mach-O提供了更强的扩展性，并提升了[符号表](http://zh.wikipedia.org/wiki/%E7%AC%A6%E5%8F%B7%E8%A1%A8)中信息的访问速度。

关键数据提取：利用逆向工程工具class-dump，将Mach-O反编译为头文件、函数、属性集。

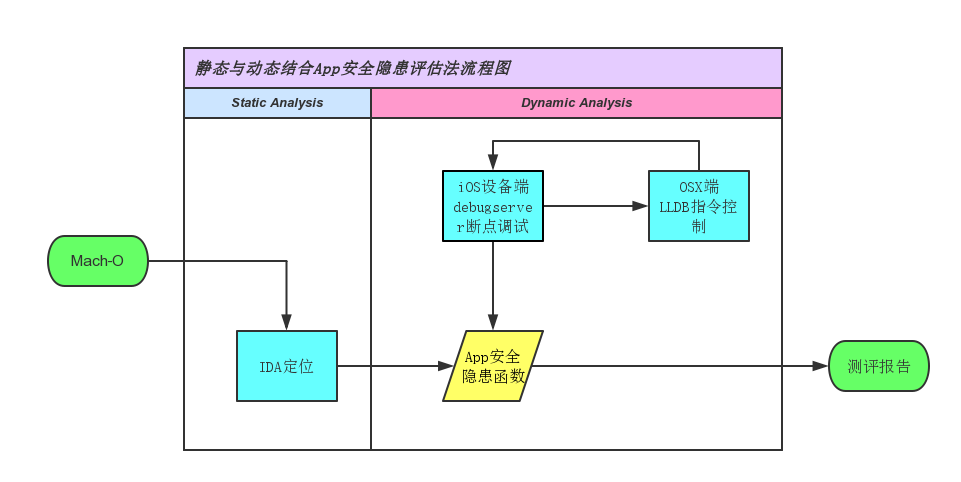
安全隐患行为库：拟将获取用户数据的公有API定位具有安全隐患行为的函数。

匹配：设计并实现自动化工具，可根据获取的关键数据与安全隐患行为库进行自动匹配。

评测报告：匹配后，生成评测报告，得出结论。

1. 静态与动态结合研究方法

根据App中通过非正常途径使用私有API获取用户隐私这一现状，利用逆向工程工具，提出设计并实现一种静态与动态结合研究方法，对App安全隐患行为进行测评。



IDA定位：根据逆向工程工具IDA，将Mach-O文件转换为ARM指令集及伪代码形式。据此，定位安全隐患函数。

LLDB指令控制：在OSX端安装LLDB，对iOS设备端的debugserver进行指令控制。

Debugserver断点测试：根据LLDB指令，对App设置断点、动态调试，并将结果反馈回OSX端。

评测报告：根据以上流程，生成评测报告，得出结论。

## 5人员安排

## 6时间表

## 7风险点