**第一部分 ADB抓取日志**

**整体目标：**

1）抓取android日志，并存储到本地（PC 还是 手机）

2）尤其是针对APP的所有请求和用户行为，以及各种数据

3）从中分析例如用户名/口令/载荷等字段，并还原用户行为，进行行为重现/行为分析

4）后续有两种手段：获取用户名+口令 或者 接管APP，修改载荷

**思路或工具：**

1）libdvm.so或libart.so探针，即 修改后重新 编译生成的Dalvik虚拟机（部分编译）

源码目录： Dalvik / vm / dalvik

输出目录： out / host / linux-x86 / lib / libdvm.so

2）得到的身份认证信息(数据)可能经过加密或者码制变换，利用des等开源解析工具

或者手动测试码制

3）虚拟机可能做不通，建议真机。但真机(非Nexus)无法下系统，虚拟机拿数据是否？

**基本指令：**

1）adb start-server + adb kill-server 开启/关闭服务器；adb devices 查看当前连接的设备；

2）adb shell可以进入安卓系统(类Linux)命令行；其中cd /data/log 该路径(没找到)

3）抓取日志

adb logcat -v time 1>.\\logcat.log 输出时间+日志，重定向到当前目录下的logcat.log 文件（.\\logcat.log）中

adb logcat -b main -v time>app.log 打印应用程序的log

adb logcat -b radio -v time>radio.log 打印射频相关的log,SIM STK也会在里面

adb logcat -b events -v time 打印系统事件的log,如触屏事件

adb shell cat /proc/kmsg 内核日志的抓取

4）Log.v 的调试颜色为黑色的，任何消息都会输出

Log.d的输出颜色是蓝色的，仅输出debug调试

Log.i的输出为绿色，一般提示性的消息information

Log.w的意思为橙色，可以看作为warning警告

Log.e为红色，显示错误信息

**当前进展（Windows及Linux）：**

1）ADB配置：利用ADB（android develop bridge）可以输出日志信息到本地，原理在于 以电脑为客户端，手机为服务器端，调用手机命令行

[利用ADB抓取实时日志（基本） logcat实时日志](http://jingyan.baidu.com/article/eb9f7b6d6fae86869364e88c.html)

经测试（仅充电 或者 存储设备 等模式类别 并不影响。。）

---------------------------------------------------------------------------------------------------------------------------------

当前问题：

在Windows环境中，C:/adb安装了本工具，但是adb logcat ... 后 显示“找不到设 备”，网上分析是找不到驱动，其实都是瞎扯

解决方案：

最后一种方法：[添加真机设备硬件ID](http://www.cnblogs.com/yanzisucheng/p/4369266.html)

路径为C:\Users\fujia\.android

---------------------------------------------------------------------------------------------------------------------------------

当前问题：

若是驱动问题，在设备管理器（我的电脑右击，第一个就是）中会有黄色感叹号

解决方案：

1.尝试在设备管理器中修改，更新驱动（各种选项）

2.如果不行的话，下载豌豆荚等手机助手，会自动安装驱动，后面卸载软件即可

---------------------------------------------------------------------------------------------------------------------------------

当前问题：

在Windows环境中， adb devices中有显示，但是为offline,而且 adb logcat 还是

没有效果

解决方案：

[android 版本与adb版本不兼容 更新adb版本至1.0.32](http://jingyan.baidu.com/article/e8cdb32b435ce637052bad14.html)

---------------------------------------------------------------------------------------------------------------------------------

当前问题：

在Linux环境中，adb shell 进入某路径时，显示[opendir failed, Permission denied](http://blog.csdn.net/ztguang/article/details/51015571)

解决方案：

似乎在Hisense山有效，Meizu上无效

[su 提高权限](http://www.imooc.com/article/1815)

[adb remout重置](http://blog.csdn.net/ztguang/article/details/51015571)

---------------------------------------------------------------------------------------------------------------------------------

当前问题：

在Linux环境中，adb reboot bootloader 时找不到设备

解决方法：

[Linux下ADB连不上Android设备的解决办法](http://blog.csdn.net/fireness/article/details/50060477)

--------------------------------------------------------------------------------------------------------------------------------- 当前问题：

在Linux环境中，adb reboot bootloader 时显示no permission

解决方法：

[Android ADB显示no permission](https://zhidao.baidu.com/question/589239941.html)

---------------------------------------------------------------------------------------------------------------------------------

当前问题：

在Linux环境中，adb fastboot时，显示waiting for device

解决办法：

在指令前增加 sudo，给予root权限（否则权限不够）

---------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------

1. 日志抓取，并定位到APP

通过 adb logcat -v time 1>.\\logcat.log 终于有内容写入了，格式如下：

01-18 14:22:53.500 D/PowerManager ( 481): AlarmManager [acquire]

Time 信息种类(VDIWE)/TAG PID MESSAGE

A. 基本的日志获取

a. [基本的实时日志 格式 + 种类 + 显示模式 + 过滤方法](http://www.cnblogs.com/JianXu/p/5468839.html)

主要： -v <format> 设置输出格式的日志消息

-d 将缓存区的log转存到屏幕中，之后退出

-s 设置过滤器（可以跟多个条件）

-b <buffer> 查看指定的日志类别，比如event\radio\main等

adb logcat -v time -d > .\\logcatt1.log

adb logcat -v time -d -s TAG > .\\logcatt2.log

adb logcat -v time -d -s TAG:W > .\\logcatt3.log

adb logcat -c 清空缓存区

adb logcat -v time -b main -d -s TAG:W > .\\logcatt4.log

adb logcat -v time -b main -d -s TAG:W \*:S > .\\logcatt5.log

b. Main\_log : app及framework相关活动的log

Events\_log: AvtivityManager及PowerManager等事件相关log

Kernel\_log:驱动相关的log

Net\_log:网络相关的，网络链接情况，抓取网络包

Modern\_log:通话相关的，信号强弱，连接的哪个网络

源码中应该有对应路径

**进一步工作（Windows及Linux）：**

目标一：

看起来日志是杂乱无章的，如何解析日志（定位到单个APP，所有行为）

相关资源：

3. [分析实例一](http://blog.chinaunix.net/uid-20459533-id-3109996.html)，针对的是一种更完整的日志(dumpstate)，不是logcat的实时日志

1) 定位到关键事件信息，

2) 如果信息不够明确的，再去搜索应用程序包的虚拟机信息(DALVIK THREADS)

3）查看具体的进程和线程跟踪的日志，来定位到代码

[分析实例二](http://blog.csdn.net/ameyume/article/details/7667574)，区分了实时日志和状态日志 抓取工具，分门别类/不同路径

目标二：

在安卓开发过程中实时获取logcat日志信息（自己写的日志+系统日志）

相关思路：

1. APP源码：监听安卓程序的一个按钮或者其他，一旦点击按钮，事件触发，将各 种日志都打印出来

[JAVA 监听一个按钮事件，时时获取logcat日志信息 + demo源码](http://www.jb51.net/article/79511.htm)

[JAVA 管理Android应用中的log日志/发布时全部去掉的示例](http://www.jb51.net/article/43952.htm)

2.系统源码：在编译so文件的c/c++文件之前加入几行简单代码，并简单修改配置 文件 -> 编译

C++ ：[C++ 在Android源码中打印出日志的示例](https://zhidao.baidu.com/question/1384240857848391780)

JAVA : [监视产品的logcat，捕获本机的日志 + demo源码](http://blog.sina.com.cn/s/blog_4a4f9fb50100x9x3.html) 很适合我们，前后两个桩

[Android程序中，如何将logcat日志输出到文件](http://www.iteye.com/problems/85431)

[真机获得日志，不用一直连接电脑，每天生成一个文件 -- JAVA源码](http://easion-zms.iteye.com/blog/981568)

[Android系统的日志系统及权限系统，为基本组件监听日志/增加权限 -- JAVA](http://www.cnblogs.com/androidNot/p/5613230.html)

相关工具及资源：

[多种安卓日志分析工具（ppt）+ tcpdump抓包](https://wenku.baidu.com/view/5f572bd32f60ddccda38a0f2.html)

**参考资源（基本原理）：**

1. [Android源码中java和c++的关系](http://blog.csdn.net/seniorwizard/article/details/7693573)

Java Native Interface (JNI) ：JAVA代码与其他代码相互调用的机制

Native层：都是C++ 应用层：虚拟机，都是JAVA类库

2. [dalvik虚拟机简要介绍与学习计划（系列4-5篇）](http://blog.csdn.net/luoshengyang/article/details/8852432)

即时编译（JIT）+ 垃圾回收（GC）+ Java本地方法调用（JNI）+ Java远程调试协议（JDWP）

[深入理解dalvik](http://www.infoq.com/cn/articles/android-in-depth-dalvik)

so是动态库，对应于Windows里面的dll（一个APP对应一个虚拟机 + 一个.so）

3. [dalvik虚拟机在android架构中的位置](http://www.oschina.net/p/dalvik/)

[ART运行无缝替换Dalvik虚拟机](http://blog.csdn.net/luoshengyang/article/details/18006645)

|  |  |
| --- | --- |
| Dalvik | ART |
| Android4.4(Kitkat)同时包含，Android5.0完全淘汰Dalvik，改变了APP的根本运行机制 | |
| JIT(Just In Time) | AOT(Ahead Of Time) |
| libdvm.so | libart.so |

---------------------------------------------------------分界线--------------------------------------------------------------

上述完成工作就是利用adb logcat来抓取日志，当前实现了基本的日志抓取功能，不足在于：1）手动的话 要连电脑，自动的话怎么实现更高级？更科学

1. 对于抓取到的日志怎样分析，一是定位到特定目标，二是怎么重现该行为，三是完整性
2. 是否可以自己写脚本过滤日志，定位APP，例如 时间段过滤
3. Logcat在模拟器emulator环境是否可行

......

---------------------------------------------------------分界线--------------------------------------------------------------

**第二部分 android源代码编译**

**整体目标：**

1）安卓系统代码层面 修改/抓取日志，为logcat做准备

2）APP源码层面 修改/抓取日志，为logcat做准备

3）构建emulator实验环境 ubuntu14.04 + android 4.4.4

**思路或工具：**

1）需要先配环境（基本就是linux系统安装，安卓系统代码和内核代码下载，系统编 译环境配置，编译系统+内核代码）

2）下真机 -- 编译安卓源码下真机，突然明白就是“刷机”

大概思路：编译后一堆img,在同一路径下，直接fastboot即可（机型对应编译版本）

具体说明：对照官方rom包，验证了本思路，因为./flash-all.sh完成的即为本工作 [Android源码编译后，发布到真机](https://www.zhihu.com/question/19824640)

**当前进展（Linux）：**

1）Android源代码编译配环境 -- Linux (Ubuntu 最好双系统 + 空间足够大 100G左右)

[较为完整的 配环境 + 下载android源代码 + 下载驱动 + 编译 + 下真机](http://blog.csdn.net/liu1075538266/article/details/51272398)

1.对照AOSP（android opensource project）[安卓版本 与Ubuntu版本+软硬件对应](http://source.android.com/source/requirements.html)

得知：要编译android4.4.x (Kitkat) 版本的源码

Linux操作系统为64位Ubuntu 14.04，磁盘空间预留100G，

JDK版本为1.6，make 版本3.81，python 版本2.7.6，

Git1.3及以上（这里获取源码不用Git+repo的方法，故暂不需要）

2. 安装JDK [Ubuntu下安装JDK图文详解](http://blog.csdn.net/jiangwei0910410003/article/details/37996723)

3. 对照官网 安装其他依赖包

[官网 配置编译环境](http://source.android.com/source/initializing.html)

[安装JDK之后，其他依赖包的安装 以及错误集](http://blog.csdn.net/jiangwei0910410003/article/details/37988637)

4. 对源码和驱动版本的一些准备 [官网 源码版本与驱动版本的对应](http://source.android.com/source/build-numbers.html)

第一列是build代码，我们根据build码查找对应设备的驱动程序，并下载； 第二列是分支（branch）代码，不同的分支会对应不同的设备；

第三列是Android版本代号，这里我们关注的是4.4系列的版本（KitKat）； 第四列是对应的设备名称，值得注意的是，同一设备即便是不同运营商发布 的也可能使用不同分支的源码，例如android-4.4.4\_r2就只针对上述运营 商的定制版本的分支，而r1就支持其余的Nexus 5设备

5. 源码的下载：

1）官网上推荐Git+repo的方式，但是[官网 Android源代码（被墙）](https://android.googlesource.com/)

在正常情况下无法访问，因此我们不采取官方提供的方式获取源码

2）到 百度云网盘 上面找对应版本源代码

注意：不要在Windows中解压，图形化工具解压会缺少内容，Linux解压

[Ubuntu中7z格式解压](http://www.educity.cn/linux/1241489.html) [Ubuntu中的解压缩文件方式](http://blog.csdn.net/jiangwei0910410003/article/details/37997899)

6. Ccache编译的加速，结合两个文档参考下

[提升Android源码编译速度的方法 -- Ccache](http://blog.chinaunix.net/uid-29140689-id-4033900.html)

7. 驱动的下载

[Linux中执行shell脚本的4种方法总结](http://www.jb51.net/article/53924.htm)

8. 开始编译 make -j4(电脑核数的二倍)

---------------------------------------------------------------------------------------------------------------------------------

问题一：unstable问题（原因是系统的源 版本过低或以失效）

解决方案：打开文件，原来的源删掉，新的源粘贴过来。其中extra..注释掉

[Ubuntu“下列软件包有不能满足的依赖关系”](http://blog.csdn.net/lvchaoshun/article/details/52245936)

[Ubuntu 14.04 完整软件源](http://wiki.ubuntu.org.cn/%E6%A8%A1%E6%9D%BF:14.04source)

---------------------------------------------------------------------------------------------------------------------------------问题二：collect2：Id terminated with signal 9错误 --> swap/RAM分配不足

解决方案： [collect2：Id terminated with signal 9错误 解决办法](http://blog.csdn.net/q35126156/article/details/53514187)

最后一步提示权限不够，直接打开文本修改就好

---------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------

9.经过超过10小时编译（这里未用到ccache..）,编译成功（Nexus Ham版本的）

10.下机有风险，先编译一个模拟器版本（lunch -> 1，之前真机lunch时选6对应)

1）在Linux中启动模拟器：

1.1 在源代码目录下直接输入emulator，默认会启动自带模拟器

重启或关闭时候，先要.（即source）build/envsetup.sh + lunch之后才能找 到emulator指令

1.2 在源代码目录下输入完整指令

emulator

-kernel  ./prebuilt/android-arm/kernel/ kernel-qemu-armv7

-sysdir ./out/target/product/generic -system system.img

-data userdata.img -ramdisk ramdisk.img

额外输入指令

emulator -partition-size 500 还可以设置启动的内存大小

2）在Windows中启动模拟器（前提：安装Android SDK）

...

11.Linux系统下尝试下真机（同样用ADB工具）

1）Cd到多个img的目录之上 ==> ham或者genetic文件夹下

---------------------------------------------------------------------------------------------------------------------------------问题一：adb reboot bootloader 时找不到设备

解决方法：[Linux下ADB连不上Android设备的解决办法](http://blog.csdn.net/fireness/article/details/50060477)

---------------------------------------------------------------------------------------------------------------------------------问题二：adb reboot bootloader 时显示no permission

解决方法：[Android ADB显示no permission](https://zhidao.baidu.com/question/589239941.html)

---------------------------------------------------------------------------------------------------------------------------------

问题三：fastboot时，显示waiting for device

解决办法：在指令前增加 sudo，给予root权限（否则权限不够）

---------------------------------------------------------------------------------------------------------------------------------问题四：fastboot时， [error: neither -p product specified nor ANDROID\_PRODUCT\_OUT set](http://blog.csdn.net/hejianhua1/article/details/50311715)

解决办法：[未设置rom的位置](http://blog.csdn.net/hejianhua1/article/details/50311715)  Note: -p 在 -w后面

---------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------

最终结果：发现确实 非Nexus设备不能下载原生系统

**进一步工作：**

2）Android原生环境上，安装 测试APP和 真实APP

[Linux下android模拟器安装apk](http://jingyan.baidu.com/article/1612d5007d5fe9e20e1eeee1.html)

3）并利用 前者测试 技术路线的可行性，在后者上进行实验分析

**参考资源（基本原理）：**

1.Build 结果的目录结构

所有的编译产物都将位于 /out目录下，该目录下主要有以下几个子目录：

/out/host/：该目录下包含了针对主机的 Android开发工具的产物。即 SDK中的各种工 具，例如：emulator，adb，aapt等。

/out/target/common/：该目录下包含了针对设备的共通的编译产物，主要是 Java应用 代码和 Java 库。

/out/target/product/<product\_name>/：包含了特定设备的编译结果以及平台相 关 的 C/C++库和二进制文件。其中，<product\_name>是具体目标设备的名称。

/out/dist/：包含了为多种分发而准备的包，通过“make disttarget”将文件拷贝到该目录， 默认的编译目标不会产生该目录。

2..Build 生成的镜像文件

Build 的产物中最重要的是三个镜像文件，它们都位 于 /out/target/product/<product\_name>/目录下。

system.img：包含了 Android OS的系统文件，库，可执行文件以及预置的应用 程序，将被挂载为根分区。

ramdisk.img：在启动时将被 Linux内核挂载为只读分区，它包含了 /init文件和 一些配 置文件。它用来挂载其他系统镜像并启动 init进程。

userdata.img：将被挂载为 /data，包含了应用程序相关的数据以及和用户相关的数据。

3. [Android系统 源代码阅读指南](http://blog.csdn.net/watermusicyes/article/details/38531537)

系统的讲解了阅读android源代码的步骤 并罗列了众多链接资源

1. Android系统分区及加载

Android的设备分区包括系统分区、数据分区、Cache分区及SD卡分区，概括如下：

1）系统分区通常加载为只读分区，包含操作系统内核、系统函数库、实时运行框架、 应用框架与系统应用程序等，由OEM厂家出厂时植入，外界不能修改：

/system/app：存放系统自带APP的APK

/system/lib：存放系统库文件

/system/bin和/system/xbin：存放的是系统管理命令

/system/framework：存放Android系统应用框架的jar文件

1. 数据分区通常存储各类用户数据和应用程序，一般挂载在/data：

/data/data：存放所有APK程序的数据。每个APK对应自己的Data目录，即在本目 录下有一个和package同名的目录

/data/app：存放用户安装的APK

/data/system：包括packages.xml、packages.list和appwidgets.xml等文件，记录安 装的软件和Widget信息等

/data/misc：存放WIFI账号和VPN设置等

1. SD分区较为特殊：SD卡是外置设备，可以从其他的计算机系统上进行操作，完全 不受Android系统的控制；而且，通常SD卡是FAT系统，无法单独对某文件设置权限
2. Android系统启动流程

Android系统启动 = 1）Android操作系统启动 + 2）Android应用系统启动

1）：ROM上电启动 → Bootloader引导程序启动 → Linux内核启动 → init进程启动

2）：Android本地服务启动 → Zygote进程启动 → Android系统服务启动 → Android

系统主界面启动

1. 专业分析工具与技术

|  |  |  |
| --- | --- | --- |
| 有助于 系统安全信息取证 | | |
| 分类 | 逻辑分析技术 | 物理分析技术 |
| 对比 | 提取保留数据：   1. 访问文件系统来获取数据 2. 保留数据指未被删除的、可访问的数据，特例：Sqlite保留数据可能仍然含有已删除数据 3. 逻辑分析技术主要包括分析系统文件数据、备份系统文件数据和商业用途的专业工具等 | 访问已删除数据，恢复恶意删除数据   1. 不依赖文件系统本身，而是以物理介质为研究目标 2. 文件系统通常只是把数据标记为废弃或已删除，而不会擦除存储媒介 3. 物理分析技术包括硬件分析工具和软件分析工具 |
| 特点 | 1. 分析系统文件数据：系统权限配置文件、系统属性文件、系统日志文件等 2. 备份系统文件：保存到SD卡、服务器端、云端等 3. 商业用途专业工具：政府部门、执法部门 | 1. 硬件分析工具：连接硬件设备或去除设备物理组件的方法实现；要求具有专业技术和比较专业的设备 2. 软件分析工具：设备上以root权限运行软件，获得数据分区的完整映像；具有易操作、直接访问文件系统、不损坏设备、可修改Android系统优点；适用无法正常开机/攻击崩溃情况 |

**其他版本：**

1.编译安卓源码 -- 详细步骤

1) [Ubuntu 12.04 + Android 4.4.4\_r1/4.4.4\_r2/4.0.3\_r6/2.3\_x/5.1.1\_r6/5.1.1\_r12](http://blog.csdn.net/u011422742/article/details/52068327)

虚拟机编译源码 + 模拟器 运行（模）+ 部分编译 + 内置app

2) Ubuntu 16.04 + Android 6.0.1

<http://blog.csdn.net/fuchaosz/article/details/51487585> 编译Android源码（模）

<http://blog.csdn.net/fuchaosz/article/details/52473660> 下真机各种详细分析

上一篇的基础上，进行补充，包含更为详细的 配环境+下载源代码+基础知识

3) Ubuntu 11.10 + Android 4.0.1

<http://blog.csdn.net/hens007/article/details/7406452> android环境建立

<http://blog.csdn.net/hens007/article/details/7465574> 编译源代码

4) Ubuntu 11.10 + Android 4.0

<http://blog.csdn.net/flydream0/article/details/7036156>

<http://blog.csdn.net/flydream0/article/details/7046612>

2. 安卓源码与系统源码 -- 区别

1) [安卓源码的版本与分支详解](http://www.tuicool.com/articles/RjeEZb)

2) [安卓系统和Linux内核的关系详解](http://www.cnblogs.com/andrew-wang/archive/2012/11/05/2755222.html)

3) [编译Android源码和内核源码的区别](http://blog.sina.com.cn/s/blog_b47e52490102uztf.html)

[如何下载并编译Android4.0内核源码goldfish](http://blog.csdn.net/flydream0/article/details/7070392)

内附 -- 下载android4.0源码 + 编译android4.0源码，即为上面的4)

4) <http://blog.csdn.net/c15522627353/article/details/51151156>

[编译源码和内核代码 归纳总结 问题汇总](http://www.linuxidc.com/Linux/2016-01/127291.htm )

3. 下真机（与第一个相同。。属于简略版->缺少源代码下载等，重点在下真机）

[源码版本 - - > build驱动下载 - - > 驱动生成/安装脚本 - - > 编译 - - >刷机]( http:/blog.csdn.net/u011422742/article/details/52253637)

---------------------------------------------------------分界线--------------------------------------------------------------

新思路 -- 直接修改系统源码，重新编译后下到真机

整体思路：

1. 配环境 + 下载系统源码/内核源码 + 编译成功 + 下真机
2. 上一阶段可能会花费很长时间，各种不同版本（Linux + JDK + android + 驱动 + 内核）

最好可以综合调试一些

1. 修改源码代码 + 重新编译 + 下真机

在实验室机器上发现：

如果选中 完全模式安装Linux,会修改系统启动设置，无法安装Windows

---------------------------------------------------------分界线--------------------------------------------------------------

**第三部分 抓取数据**

**整体目标：**

1. 抓取APP相关的各类敏感数据，构建本地存储
2. 区分数据：验证信息（用户名 口令 验证码 认证链等）+ 业务信息（内容）
3. 配合抓取到的日志，实现 行为重现/分析/APP缺陷/本地因素最小集等

**思路或工具：**

1. 本地数据（文件中+数据库中）：

---------------------------------------------------------------------------------------------------------------------------------

快快查汉语词典 [词典demo获取本地数据](http://www.jianshu.com/p/bc02c809772a)

1）三个路径 /data/data/包名 或 /sdcard/Android/data/包名（类似com.kk.dict）

或 sdcard/kkdict（与包名相近的）

Data/data 需要root权限，（su 即可进入），里面有好多待分析的文件

android系统中 [替代Linux中cp指令方法1](http://blog.csdn.net/fengye810130/article/details/9225783) [替代 cp 指令方法2](http://www.cnblogs.com/bravesnail/articles/5850335.html)

Sdcard/Android/data/com.kk.dict,发现有cache 和 files，files虽有内容，但无关

Sdcard/kkdict/dict,发现有两个.db文件

2）解析 zidian\_full.db 中 表zi 中的 zhujie字段：

乱码 -- read\_sqlite.py [python 二进制 十六进制 ascii码互转](http://blog.csdn.net/u010668907/article/details/50590096)

加解密工具 -- 在线有很多，不同格式的

---------------------------------------------------------------------------------------------------------------------------------

[安卓获取程序/DB路径](http://www.cnblogs.com/bluestorm/p/3688937.html)

[安卓本地文件存储](http://febsky.me/2015/07/14/2015-07-14-Android%20%E6%96%87%E4%BB%B6%E5%AD%98%E5%82%A8/)

[安卓本地数据存储](http://www.jianshu.com/p/3d73366093c3)

2.网络数据：

1）APP抓包工具Fiddler （将手机和电脑放在同一片局域网中，设置电脑为代理）

Fiddler数据包的原理是Fiddler将自己设置为一个代理服务器，默认监听127.0.0.1： 8888端口同时将浏览器的HTTP、HTTPS协议设置为使用代理服务器也就是使用自 己作为浏览器的代理服务器

Fiddler原理及配置 <https://my.oschina.net/u/587105/blog/322952>

Fiddler实例 <http://www.cnblogs.com/softlin/p/4966709.html>

<https://my.oschina.net/jhao104/blog/605963>

<http://www.cnblogs.com/wanghaixing/p/6262367.html>

Fiddler高级用法：可以设置过滤条件，定位到单个APP

2）Python爬虫（在抓包得知用户名+口令后），可以获取更多应用层面数据

<https://my.oschina.net/jhao104/blog/606922>

**当前工作：**

1. 本地数据（文件中+数据库中）：

对照着DEMO，利用python脚本取得了基本数据，但解密失败

2.网络数据 -- 爬虫 或 抓包，暂时放弃

**进一步工作：**

**第四部分 参考资源**

**参考文献：**

1. 基于沙盒的Android应用风险行为分析与评估

引用：李舟军, 吴春明,王啸.基于沙盒的Android应用风险行为分析与评估[J]. 清华大

学学报（自然科学版）, 2016, 56(5): 453-460.

摘要：（基于行为）在Native Code和应用程序框架层之间设置沙盒，通过注入系统

进程 Z y g o t e来实现对系统本身应用程序编程接口 AP I的重定向, 拦截其调

用地址, 将其地址指向一段自定义代码的地址, 然后执行自定义的代码, 最后 再回调系统的 A P I , 完成整个调用过程

补充：北航教授；测试了1200多个APP，利用python 脚本实现app的自动安装， 启动，测试，卸载

备注：其中【4】【5】【7】【8】【9】【10】【11】有价值

2.TaintDroid:针对实时隐私监控的信息流跟踪系统

引用：Enck W, Gilbert P, Han S, et al. TaintDroid[J]. Acm Transactions on Computer Systems, 2014, 32(2):1-29.

摘要：（动态分析）在消息+变量+类+文件等多个层面注入污点（隐私数据），跟随污点传播， 一直到网络层，从而判断APP是否侵犯隐私。分析的点包括LOW/HIGH-bandwidth

Sensors + Information Database + Device Identifiers + Network Taint Sink

补充：北加州大学；2010年开始，持续更新，这个看起来是标杆之一

3.DroidSafe:信息流分析工具

引用：Gordon M I, Kim D, Perkins J, et al. Information-Flow Analysis of Android Applications in DroidSafe[C]// Network and Distributed System Security Symposium. 2015.

摘要：（静态分析）从关键API入手，Java class中给它们打桩；除此之外，在安卓进程+应 用对象+文件系统中也会记录重要信息。

补充：开源的

备注：这里有一些参考文献值得看下

4.安卓系统非法控制流检测：保护智能手机的隐私数据

引用：EI Compendex：Graa M, Cuppensboulahia N, Cuppens F, et al. Detection of Illegal Control Flow in Android System: Protecting Private Data Used by Smartphone Apps[J]. 2014, 8930:337-346.

摘要：

1）TaintDroid的局限，只能侦测显式的数据流，未考虑控制流

2）静态方法：分析代码，得到CFG(Control Flow Graph)

动态方法：首先，针对CFG的节点（依赖条件+变量值）

其次，控制Dalvik的翻译器

5.杨博. Android系统下应用程序的安全性研究[D]. 上海交通大学, 2013.

摘要：

1）权限系统 和 通信系统 两方面

2）控制流：

基本调用图：源代码分析 -> 起始函数 -> 广度遍历

调用图补全：Java补全 -- 反射 / 创建新进程 / 类的继承

JNI补全 -- 调用Native C++代码

3）数据流：

数据流初始化 + 变量保存 + 静态数据模拟

**实验室内部材料：**

1. 贺恺学长 -- 大论文：基于Android平台的手机隐私保护技术的研究与实现

小论文：一种基于Android应用识别的隐私保护方法

目标：实现细粒度的保护Android隐私数据免受窃取

手段：对Android源码二次开发，为Android Framework层增加隐私保护

功能定义：

应用识别：能识别来自APP的隐私请求属于哪个APP（基于内核UID PID机制）

隐私保护：对APP的隐私请求进行额外处理，如果策略配置中此APP无权限，

返回虚假数据；反之，返回真实数据（基于预先的隐私配置-- DB）

策略配置：允许用户为APP自定义配置隐私策略，如APP0能访问手机号码，

但不能访问位置信息（手机号码+IMEI号+联系人列表+位置信息）

功能模块：

|  |  |  |
| --- | --- | --- |
| Application层 | Privacy Protector | 隐私配置 |
| Privacy Provider | 隐私策略数据 |
| Android Framework层 | Privacy Manager | 应用识别 |
| Privacy Service | 隐私处理 |

安卓进程通信机制：

对APP来说，主要是Activity,Service,Content Provider,Broadcast Receiver构成。这 些组件既可以工作在同一进程中，也可以工作在不同进程中，对于这些组件的数据 传输通信就需要用到进程间通信的方式。

|  |  |  |
| --- | --- | --- |
| Linux的进程通信机制 | | 本地Socket 管道或消息队列 |
| 缺点：Socket传输效率低，速度慢，开销大；管道和消息队列采用存储+转发的方式，两次拷贝的效率损失；同时，传统进程通信机制无法获取发送方的UID，缺乏足够安全机制 |
| Android的进程通信机制 | 特色 | Binder , Services ,Intents ,Content Provider |
| Binder | 是轻量级的远程进程调用方式，本质是通过代理技术进行远程过程调用，高效完成进程内部及跨进程方法调用。Binder工作在Linux层面，通过操作一小段内存，为用户层提供各种操作接口，从而实现相应的底层支持 |
| Service | 是Android的基本组件之一，用来实现后台运行耗时较长的应用程序。不同应用程序可启动同一个Service，而且其他基本组件也可通过Bind的方式绑定该Service |
| Intent | 是一个简单的消息体，代表完成一个动作的意愿。例如带有网址的Intent，Android分析后，会查找可处理它的应用程序 |
| Content  Provider | 是Android的基本组件之一，主要作用是作为数据仓库，为应用程序提供数据存储管理功能，典型示例是存储联系人信息的Content Provider,第三方应用通过它的机制读取/修改联系人数据，实现进程间通信行为 |

1. 黄洁学姐 -- 大论文：Android平台下 APP行为审计与管理方法 及原型系统

小论文：用户友好的Android隐私监管机制

|  |  |
| --- | --- |
| 痛点 | 主要工作 |
| 1）Android平台除安装时的授权提示外，对用户屏蔽权限使用细节 | APP行为审计办法：以APP对敏感数据的访问行为为研究对象，为用户提供实时的APP隐私泄露威胁量化值，并提供隐私资源的访问行为日志（资源使用细节可视化） |
| 2）Android平台只能在安装时授权且权限捆绑的问题 | APP访问控制办法：保留原有权限验证机制，并在此基础上增加细粒度的权限管理机制，为不同APP和用户提供个性化的细粒度敏感资源访问约束 |
| 3）Android平台细粒度权限管理机制下，权限定制低效的问题 | 预配置权限的用户辅助机制：设计并实现权限预配置生成工具，通过分析已有应用构建应用的基准权限轮廓，帮助用户便捷的完成细粒度权限决策 |
| 本系统将 行为审计和访问控制（行为监管/管理） 动态部署于Android进程间通信机制 控制流中 ，通过与用户接口交互实现APP行为审计与管理 | |

手段：进程注入（利用Ptrace机制，依赖root权限，但是不需要编译系统源码）

目标：

面向问题：授权应用访问功能非必需资源。

1）审计应用对敏感资源的访问行为，为用户提供动态的应用隐私泄露威胁量化 值及全局敏感数据访问行为日志；

2）同时，在Android原有权限机制的基础上，增加细粒度访问控制模块，为用 户提供 个性化的二次授权管理。

安卓 基本安全机制：对本文的面向问题 无能为力

|  |  |
| --- | --- |
| 基于应用隔离的  沙箱机制 | 为每个应用分配唯一的UID,不同UID的应用运行在不同的进程中，从而使进程得到隔离保护 |
| 面向资源访问控制的  权限机制 | 底层继承Linux的权限表达方式，高层以权限声明的方式表达；Android开发者在编写APP时必须将所用权限在AndroidManifest.xml文件中显式声明，应用安装时，系统会检测所用到的dangerous权限，打包以列表方式告知用户 |
| 面向开发者的  应用签名机制 | Android规定开发者需要为APP使用数字签名。在安装/升级/共享资源时涉及APP数字签名验证，此机制可用来识别应用开发者，可明确应用程序的来源，还可保证程序完整性和正确性 |

功能定义：

1. 细粒度权限配置：UI界面实现对应用的细粒度权限配置
2. 防崩溃数据配置：UI界面实现对应用防崩溃数据的配置
3. APP行为审计功能：作用于API调用控制流，通过API过滤敏感数据访问行为；

为用户实时量化敏感数据泄露威胁，并可视化敏感资源访问

1. APP行为管理功能：监控各应用对敏感资源访问API的调用情况，并依据细粒 度权限配置策略都API调用进行行为干涉：拒绝 模糊 允许
2. 预置配置生成功能：以已有应用的配置为基础构建不同类别应用的基准权限轮 廓，可作为预置权限辅助用户实现便捷高效的细粒度决策
3. 洪瑶彦学姐 -- 大论文：安卓平台下手机用户隐私管理与审计

研究目标：

1. 研究用户对于隐私信息的定义
2. 研究分析隐私信息流向的方式：采集 运输 发送
3. 研究分析应用或系统 操作 隐私信息的审计方式
4. 研究分析隐私信息管理和审计相结合的方式

方式：

动态信息流分析和实时审计相结合