TURING 图灵程序设计丛书

Android恶意代码分析与渗透测试

[韩] 赵涎元　等　著

金圣武　译

OWASP子明　审

人民邮电出版社

北　京

图书在版编目（CIP）数据

Android恶意代码分析与渗透测试 /（韩）赵涎元等著 ; 金圣武译. -- 北京 : 人民邮电出版社, 2015.7
（图灵程序设计丛书）
ISBN 978-7-115-39593-1

Ⅰ. ①A… Ⅱ. ①赵… ②金… Ⅲ. ①移动终端－应用程序－程序设计－安全技术 Ⅳ. ①TN929.53

中国版本图书馆CIP数据核字(2015)第131406号

内 容 提 要

本书详细讲解了 Android 恶意代码的散播渠道，并针对开发者和用户介绍如何应对此类威胁。还从分析人员的角度出发，通过渗透测试方法查看如何对应用程序进行迂回攻击以获得敏感信息。这些都是安全咨询人员或安全负责人可以直接应用的诊断方法。

◆ 著　[韩] 赵涎元 等
译　金圣武
审　OWASP子明
责任编辑　傅志红
执行编辑　陈　曦
责任印制　杨林杰

◆ 人民邮电出版社出版发行　北京市丰台区成寿寺路11号
邮编　100164　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京　印刷

◆ 开本：800×1000　1/16
印张：23.5
字数：555千字　2015年 7 月第 1 版
印数：1－4 000册　2015年 7 月北京第 1 次印刷
著作权合同登记号　图字：01-2014-5451号

定价：69.00元

读者服务热线：(010)51095186转600　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第 0021 号

版 权 声 明

推　荐　语

从业多年，我读过很多关于恶意代码分析的书籍。但是有关Android环境下的恶意代码分析，国内目前只有一本韩国的引进图书，也就是这本《Android恶意代码分析与渗透测试》。

本书完全从专业的技术手段进行分析，开始先讲原理——包括环境的搭建——接着通过具体的示例进行了详实的分析处理。最难能可贵的是，无论是安全方面的爱好者还是专业人员，都能参考本书。书中内容由浅入深，即使是刚刚从事分析工作的开发人员都可以轻松上手。诸多案例和第5章、第6章的多种专业工具都能手把手教会你如何分析，希望更多专业人士通过这本书进行系统化的学习。本书同时也可作为国际信息安全比赛移动分析部分的参考书，因为书中对每道试题都有十分具体的讲解。

应图灵公司之邀负责本书的审读工作，我感到非常荣幸。书中多次提及OWASP在移动安全开源方面的巨大贡献，作为“OWASP中国”的负责人之一，我向那些对Android环境下的恶意代码分析和研究感兴趣的人员强烈推荐本书。

OWASP子明

“OWASP中国”负责人、“知道创宇”信息安全培训负责人

作　者　序

赵涎元

不知不觉间，我们已经进入“移动设备就是必需品”的时代。除了工作或是编辑文档之外，我都通过移动设备搜索信息、管理网站、利用SNS与人联系。如今，移动服务的使用时间已经超过了PC台式机的使用时间。我们上下班途中或等人时，甚至和别人在一起的时候，手里也会拿着移动设备。可以说，人们现在的上网时间比以前翻了一倍。

针对台式机，特别是针对Windows用户的恶意代码攻击一直有增无减。经常访问的网站会突然变成恶意代码的宿主，在用户未能察觉的时候就感染到电脑，我们只能信任自己安装的杀毒软件。移动设备也暴露在同样的危险之中。众所周知，移动设备里保存的个人信息（公共认证书、照片、自动保存的服务账户信息等）比台式机里的更多，所以那些攻击者乐此不疲地向移动设备的用户发送大量恶意代码。每当出现重大的社会事件时，他们也会不知廉耻地利用这些新闻发送恶意代码。

韩国国内90%以上的移动设备用户都是Android系统用户，为了获取这些用户的信息，黑客们正在不分昼夜地进行研究。我们必须比这些人做更多研究以找出应对方案。

本书将详细讲解此类恶意代码会通过何种渠道散播，并针对开发者和用户介绍如何应对此类威胁。还会从分析人员的角度出发，通过渗透测试检验方法查看如何对应用程序进行迂回攻击以获得敏感信息。这些都是安全咨询人员或安全负责人可以直接应用的诊断方法。

我为写作本书付出了很多时间与努力。如果没有共同编写的同伴，我根本不可能完成这本书，非常感谢为了目标一直勇往直前的所有安全防范项目组成员。也非常感谢编写此书时一直在旁边为我加油的妻子金慧珍和儿子昊永，我爱你们。

朴炳旭

随着越来越多的用户开始使用Android系统的智能手机，一些不怀好意的人趁机对普通用户大肆实施短信诈骗，我也从这时起对Android恶意代码分析产生了兴趣。我现在也经常收到一些钓鱼短信，收到后并不立即删除，而是开始分析研究Android恶意代码，以了解恶意代码是怎样运行的。虽然刚开始有些辛苦，但随着分析的深入，逐步了解恶意代码的运行过程和损害用户利益的方法时，就好像拼图一样有趣。

目前也有很多人开始对Android系统的恶意代码分析和诊断感兴趣，想开始学习。为了让各

位也能像我一样从Android恶意代码分析中得到乐趣，本书包含了很多相关内容。

虽然书中没有收录所有Android恶意代码分析诊断相关内容，但已经包含了大量极为重要而核心的话题，希望能够指导各位的学习。

抚今追昔，这已经是我第三次写书了。虽然写作时感到艰难困苦，但是拿到成果时的兴奋真的无法言表。

感谢身边的家人和朋友，正因为有你们的不断支持，我才能体会这种难以言说的喜悦。

南大铉

这是我第一次写作者序，还生疏得很。虽然为技术杂志投过几次稿，但身为技术员，我感到写文章给别人看还是有些困难。

写这篇文章的时候我在想："读者能轻松理解我写的内容吗？"不记得从哪里听过（读过）："如果不能用一句话表达你想说的内容，那就说明你不了解这件事情。"我尽量用通俗易懂的表达方式对需要具体说明的地方展开了详细论述，但"这种程度应该都能理解吧"的想法也让我跳过了某些地方。当然，并不是所有人都能看懂全部内容。

移动环境已经占据了我们大部分的生活，也变成了一个重要的部分。无论怎样，我都希望自己写的内容能对学习移动恶意代码和移动安全的读者提供一点帮助。

感谢像亲人一样关心安全防范项目组成员的赵涎元先生，是他成就了今天的我。也感谢经常帮助我的江俊茂大哥和我深爱的妻子安正珠。

金衡范

2009年底，韩国国内引进了iPhone，之后的移动市场开始急速发展。无论男女老少，无论地铁、公共汽车、卫生间、办公室等何种环境，随时随地都可以利用无线接入网络。网民可以用移动设备使用银行服务、购买商品等，也有了越来越多使用移动设备处理公司业务的BYOD（Bring Your Own Device，自带移动设备）族。

高效与便捷使移动设备相关市场在短时间内急速增长，也因此暴露了很多安全漏洞。特别是Android系统，因为是开放性操作系统，所以变成了黑客的"游戏"平台。因此，国内外也提出了很多应对方案。OWASP（国际Web标准机构）也发表了"OWASP Mobile Top 10"安全威胁。

移动安全渐渐成为关注热点，我也对移动安全诊断产生了兴趣，并开始搜集资料学习相关技术。我怀着当时的热情和成员们一起编写了这本书，希望它能使更多人轻松进入移动分析和诊断领域。

本书的重点内容——恶意代码分析不仅能满足对该领域感兴趣的读者的要求，而且能提高对操作系统的了解，也可以通过渗透测试掌握Android漏洞诊断的概念。

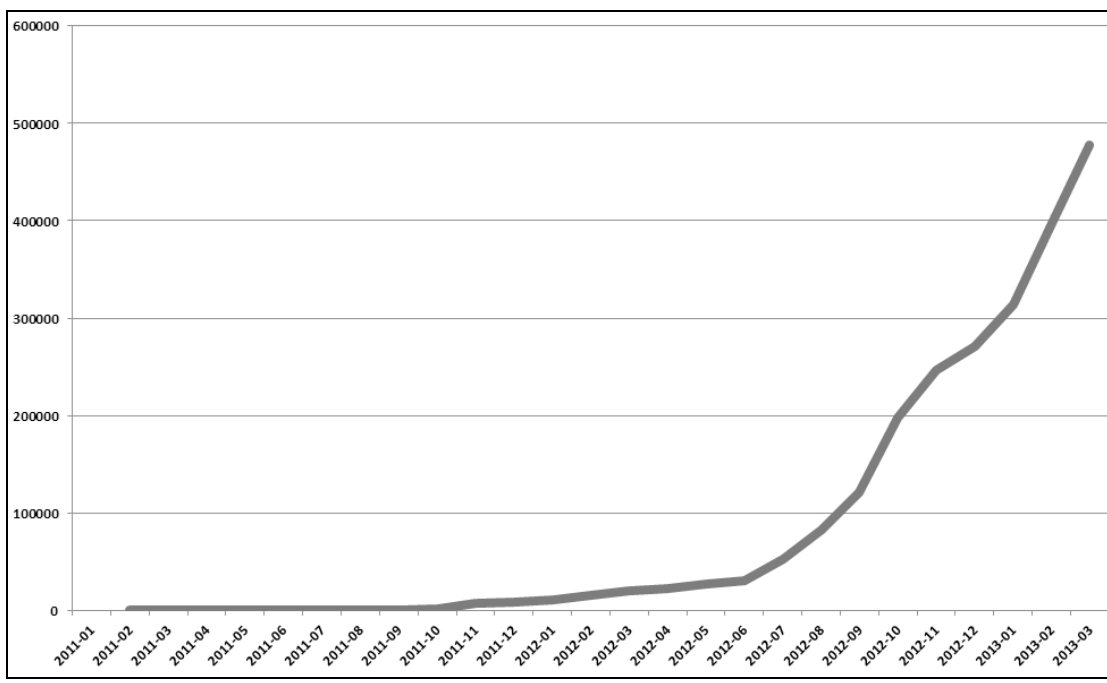
本书能使恶意软件分析的初学者体会到学习的乐趣，也会激发各位的热情。最后，感谢赵涎元先生和安全防范项目组允许我参与本书的编写工作。

前　言

韩国国内移动设备的普及时间并不长，但韩国网速极快，没几年时间，所有人都在使用智能手机。技术的发展需要以所有领域的高度发展为基础，而且要经过足够长的过程，但是过快的移动终端普及速度引发了移动领域的很多安全隐患。

移动终端的网络使用率已经超过了PC台式机的使用率，越来越多的个人信息安全问题得到关注。在PC端出现过的种种安全问题，也同样出现在了移动终端。

通过智能手机和平板电脑的普及，2013年的全球移动恶意代码比2012年增加了423%，特别在智能终端市场占有率非常高的Android系统中呈现增长趋势。每天新增的移动恶意代码数量也开始超过PC端。针对不熟悉手机操作人群的短信诈骗活动在日益增加，造成的损失也在不断扩大。



Android恶意代码增长示意图（2011年～2013年）（参考：安博士研究所（AhnLab）移动恶意代码统计资料）

与PC一样，移动设备也不能只依赖杀毒软件。新出现的恶意代码越来越智能，用户可以随时选择安装Android应用程序，所以只能自行防范恶意代码引发的威胁。

本书将讨论Android恶意代码的分析环境、分析方法、预防方法等所有领域，反映了截稿时[①]的所有最新信息，努力向有兴趣分析Android恶意代码的入门者和管理员提供更多信息。希望各位能通过学习本书有效应对恶意代码引发的威胁。

读者对象

本书面向移动安全分析领域的入门者和一线技术人员，同时适合以下读者。

- 想学习移动恶意代码分析技术的读者
- 想学习移动入侵/安全诊断工具的读者
- 想全面了解移动安全威胁的读者
- 想理解并实际运用移动服务诊断方法的读者

本书特色

近几年，移动应用发展迅猛，几乎覆盖了社会生活的方方面面，其安全性也越来越受到重视。本书从专业分析人员角度详细阐述了Android恶意代码的散播渠道及其引发的威胁，并从实战层面介绍了代码的分析环境、分析方法、预防方法等，读者不仅能通过渗透测试检验方法掌握Android漏洞诊断的基本概念，还可在多个项目中获得可直接运用于实际业务的技术和流程。

本书结构

本书面向关注Android移动安全威胁的读者，由“恶意代码分析”和“移动服务诊断”两大主题组成。各章节包含了分析步骤，作者们还亲自编写了黑客大赛应用程序试题，读者可以借此复习学过的内容。

各章节内容如下。

- **第1章　Android的基本概念**：介绍Android的基本概念。构建恶意代码分析环境前，把握Android的整体概念和结构。本书重点不在于Android开发，所以只讲解必要的部分。只有掌握了基本概念，才能理解后面要讲解的分析阶段。
- **第2章　Android应用程序诊断环境**：介绍Android分析环境的构建方法。分析恶意代码或移动应用程序服务时，需要构建Android开发环境。谷歌提供了Android SDK、NDK等工具包，以及完美支持Java应用程序开发工具的诊断环境。详细介绍了分析所需工具及其使用方法。
- **第3章　Android App分析方法**：介绍诊断Android恶意代码应用程序或漏洞时必须掌握的分析方法。第3章内容会用于第4章、第5章和第7章，必须熟练掌握。

① 原书出版于2014年5月29日。——编者注

- **第4章　恶意代码分析**：本书亮点之一，详细介绍分析恶意代码时使用的在线分析服务及其意义、手动分析步骤等。希望读者能在介绍的多种操作方法中选择适合自己的方法。
- **第5章　Android移动服务诊断**：介绍诊断Android移动应用程序服务的方法。通过测试应用程序详细讲解可应用于实际业务的方法，以OWASP提供的标准为基础，展示各种诊断方法和相应的应对方案。
- **第6章　使用Android诊断工具**：介绍分析Android应用程序时用到的其他重要工具。通过数据包分析、漏洞分析、开源框架诊断工具等内容全面了解诊断方法。
- **第7章　Android黑客大赛App试题**：介绍Android黑客大赛的试题解析，以复习之前通过Android恶意代码分析与诊断而掌握的技术和工具。黑客大赛出现了很多移动应用程序诊断试题，想参加黑客大赛的读者也会获得有用的信息。

注意事项

本书写作目的在于，为想了解移动安全威胁的读者和想从事分析工作的入门者提供帮助。书中详细描述了如何在本地电脑搭建测试环境。严禁利用这些工具非法入侵未获授权的服务，由此产生的所有法律责任均由实施者本人承担。

目　录

第1章
Android 的基本概念

开始分析前，先通过本章简单了解一下Android的基本概念。很多开发相关书籍都涉及了Android概念，所以本书只会提及一些必要的内容。本书重点不是内核区域，而会针对恶意代码分析和应用程序诊断，从应用程序和服务角度进行说明。

1.1 Android 的架构

Android是由Android Inc.开发的基于Linux平台的操作系统。谷歌公司收购后对其进行了修改，使其更适合在智能手机、平板电脑、相机、机顶盒等触屏设备上运行。使用Android系统的用户在不断增加，韩国国内80%以上的用户都在使用Android系统。越来越多的人关注Android系统的同时，相应的安全威胁也呈现增长趋势。

学习Android应用程序的基本结构前，先简单了解一下Android的系统架构。本书不会逐一讲解系统架构，只简单介绍不同区域。图1-1是很多文档中常见的Android系统架构。

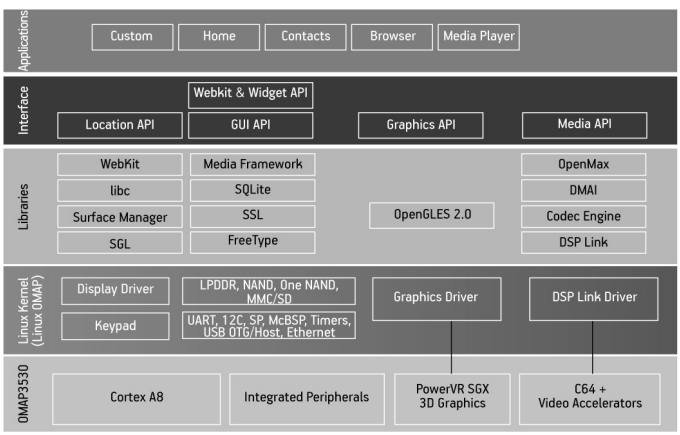


图1-1 Android系统架构

（出处：http://www.techdesignforums.com/practice/technique/android-for-the-rest-of-us）

1.1.1　Linux 内核

最底层由Linux内核构成。Android系统以部分结构微调后的Linux 2.6内核版本为基础，该层由相机、声卡、Wi-Fi、键盘等多种驱动程序构成。Android的安全、内存管理、进程管理、网络协议栈、驱动模型等主要系统服务都依赖于Linux。内核会在硬件和软件栈的剩余部分起到抽象层的作用。

用户在应用程序启动后运行的Windows或Linux等传统平台上工作。例如，用户安装并运行软件时，软件的执行权限与用户权限相同。假如该软件是恶意软件，其访问或窃取用户计算机中的敏感信息时，都会得到操作系统的许可。因为Windows和Linux都会在相同的用户权限下运行所有进程。

1.1.2　库

Linux内核的上层是Android的本地库，这些库是用C/C++语言编写的。库会使用Android系统的多个组件，通过Android应用程序框架（库的上层）展示给程序员。这些库也会在Linux内核中以进程方式运行。

库只是告知设备多种数据处理方式的命令集合。比如，媒体库支持录制或播放音频、视频格式。一部分重要的库如下表所示。

库	说　明
系统C库	嵌入式Linux设备专用，是继承了BSD而实现的标准C库
SQLite	可在所有应用程序中存储数据。虽然小，但是一个功能强大的关系型数据库引擎（渗透测试时可用于查看数据库中的所有敏感数据）
WebKit	提供网页检索工具的浏览器引擎（渗透测试时用于查看所有敏感页面缓存）
Surface Manager	负责设备屏幕上的图形图像
OpenGL	在屏幕上绘制2D或3D图形图像
媒体库	播放或录制音频、视频格式（mp3、mpeg4、jpg、png等）
3D库	基于OpenGL API，是硬件3D加速软件
SGL	支持2D图形图像

1.1.3　Android 运行时

Android运行时（Runtime）与库位于相同的层，以核心Java库和Dalvik构成。核心Java库用于开发Android应用程序。

众所周知，虚拟机是拥有操作系统的虚拟环境。Android使用Dalvik虚拟机概念，这样可以有效运行多个虚拟机。Android操作系统使用这些虚拟机将各应用程序运行为自己的进程。

Dalvik是由谷歌公司的Ban Bornstrein及其团队开发的，这个词源于Ban Bornstrein的祖先居住的一个冰岛村庄。Dalvik虚拟机的构建过程与JVM（Java Virtual Machine，Java虚拟机）类似，但前者使用Dex编译器进行转换，生成位元码后运行，如图1-2所示。这是考虑到当时的移动环境而设计的

性能，也是因为被Sun公司的Open Java（Open JDK和Apache Harmoney项目）替代了的缘故。

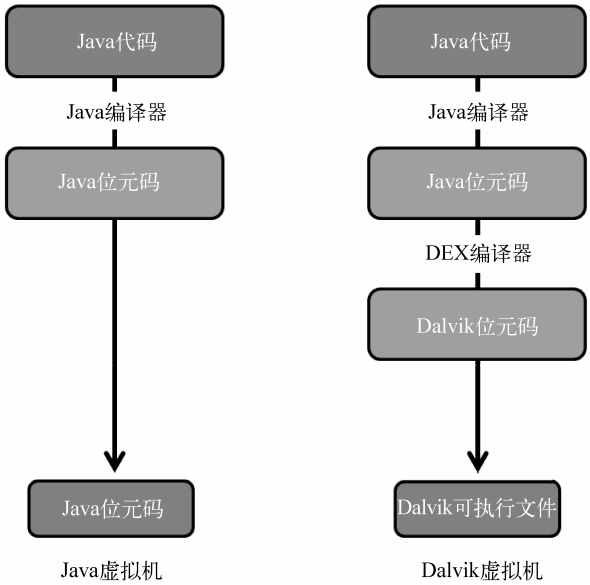


图1-2 Java VM与Dalvik VM的区别

Dalvik虚拟机的主要特点如下。

- 优化内存管理。
- 各应用程序未授权时不可干涉其他应用程序。
- 支持threading。

图1-3以图片形式表示Android环境。各Android应用程序在各自的虚拟实例中运行，每个应用程序会获得分配好的固定用户ID。

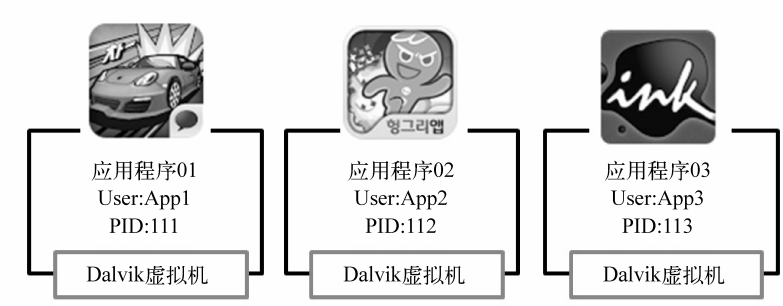


图1-3 赋予各应用程序固定的用户ID（出处：Google Market）

从图1-4可以看到设备中安装的工具包，并能看到以app_为前缀赋予了用户ID。简言之，就是每个进程都有一个Linux用户。

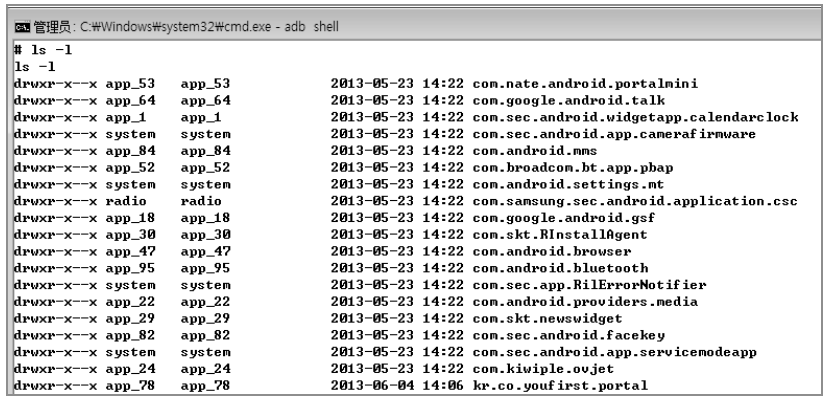


图1-4　赋予各应用程序固定的用户ID：在设备中查看

输入ps命令查看进程信息时，可以看到最左侧每个进程持有的用户权限。

```
# ps | more
... (略) ...
system    271   200    261244 70524 ffffffff afd0b7ec S system_server
system    354   200    152524 31240 ffffffff afd0c63c S com.android.systemui
system    362   200    147524 21920 ffffffff afd0c63c S
com.samsung.sec.android.inputmethod.axt9
radio     366   200    159304 25128 ffffffff afd0c63c S com.android.phone
app_22    367   200    155084 20816 ffffffff afd0c63c S android.process.media
bluetooth 375   200    135844 16056 ffffffff afd0c63c S com.broadcom.bt.app.system
app_1     382   200    210036 50772 ffffffff afd0c63c S com.sec.android.app.
twlauncher
app_18    427   200    207524 26392 ffffffff afd0c63c S com.google.process.gapps
app_1     439   200    142628 19428 ffffffff afd0c63c S android.process.acore
system    447   200    147540 25404 ffffffff afd0c63c S com.android.settings
app_40    477   200    137076 16700 ffffffff afd0c63c S com.sec.android.
provider.badge
app_1     486   200    135976 16076 ffffffff afd0c63c S
com.sec.android.provider.logsprovider
system    510   200    135940 15436 ffffffff afd0c63c S com.android.server.
vpn:remote
... (略) ...
```

1.1.4　应用程序与框架

库之上的层是应用程序框架，包含资源分配、语音通话等管理智能手机基本功能的程序。程序员使用此框架API开发更加复杂的应用程序。

该框架的一部分重要的“块”管理资源管理器，谷歌地图和GPS等定位服务、应用程序生命周期的Activity管理、语音通话管理、应用程序之间共享的数据管理内容提供商等均属此类。

栈的最上层是应用程序。与邮件地址、SMS账户、地图、浏览器等通过Android市场开发并发布的应用程序一样，也包含随Android系统一起提供的应用程序。

1.1.5 设备文件目录结构

本节讲解设备的主要文件目录。阅读本节之前，必须先搭建ADB（Android Debug Bridge）环境，所以请先阅读第2章。输入adb shell mount命令后可以看到如图1-5所示的多个分区。

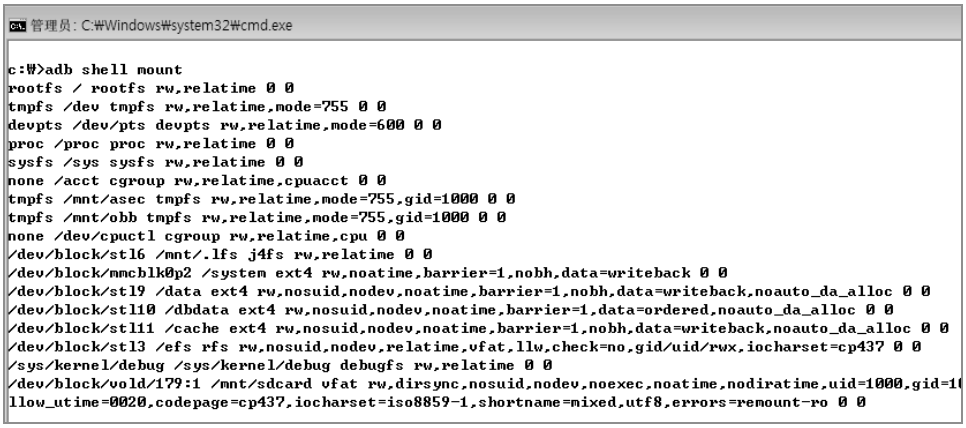


图1-5 各设备文件分区结构

主要文件目录的功能请参考表1-1。诊断应用程序时会经常从如下相应目录获得重要信息。

表1-1 文件系统目录

文件目录	说 明
/	只有读取权限的根（root）文件系统目录。浏览启动相关设置文件，包含初始进程信息
/system	只有Android系统读取权限的主目录，包括具备HAL和框架的库文件、守护进程相关可执行文件、字体、媒体、系统应用程序
/data	有读写权限，该文件系统目录包括可设置的用户应用程序及状态信息
/cache	有读写权限，包括浏览器缓存和用户临时状态信息

安装的应用程序会因为各种目的保存到系统目录。

表1-2 文件系统目录

文件目录	说 明
/system/app/应用名.apk	保存系统应用程序，优化后的dex代码保存到/system/app/应用名.odex。以安全模式启动时，运行可操作的系统应用（代码P28 第一段） # ls -l | more ls -l | more -rw-r--r-- root root 9000 2012-06-08 06:58 SkafLauncher.odex -rw-r--r-- root root 25072 2012-06-08 06:58 VisualizationWallpapers.odex -rw-r--r-- root root 2240033 2012-06-08 06:58 GoogleServicesFramework.apk -rw-r--r-- root root 36032 2012-06-08 06:58 TtsService.odex -rw-r--r-- root root 482608 2012-06-08 06:58 SelfTestMode.apk -rw-r--r-- root root 3490788 2012-06-08 06:58 ClockPackage.apk -rw-r--r-- root root 137632 2012-06-08 06:58 CSC.odex -rw-r--r-- root root 19118 2012-06-08 06:58 PackageInstaller.apk -rw-r--r-- root root 29575 2012-06-08 06:58 SKTNetworkTool.apk -rw-r--r-- root root 54624 2012-06-08 06:58 Stk.odex -rw-r--r-- root root 185223 2012-06-08 06:58 DualClock.apk -rw-r--r-- root root 8910 2012-06-08 06:58 Preconfig.apk -rw-r--r-- root root 10424 2012-06-08 06:58 SamsungWidget_ProgramMonitor.odex -rw-r--r-- root root 10294 2012-06-08 06:58 PhoneErrService.apk

（续）

文件目录	说　明
/data/app/应用名.apk	保存已注册的用户应用程序，优化后的dex代码保存到/data/dalvik-cache/应用名.odex
/data/app/＜app-package-name＞-1.apk	保存用户下载的应用程序，优化后的dex代码保存到/data/dalvik-cache/ data@app@＜app-package-name＞-1.apk@classes.dex（代码P28 第二段） # cd /data/dalvik-cache/ cd /data/dalvik-cache/ # ls ls data@app@com.tegrak.lagfix-1.apk@classes.dex system@app@HDVideoCall.apk@classes.dex data@app@stericson.busybox-2.apk@classes.dex system@app@DummySrn.apk@classes.dex system@app@GoogleFeedback.apk@classes.dex system@app@RInstallAgent.apk@classes.dex system@app@U3MobileInstaller.apk@classes.dex
/mnt/secure/asec/＜app-package-name＞-1.asec	保存移动到SD卡的应用程序，优化后的dex代码保存到/data/dalvik-cacge.nbt@asec@＜app-package-name＞-1@pkg.apk@classes.dex

这些目录中，能获取最多应用程序信息的是“/data/data/应用名”。访问相应应用程序时使用如下命令。

```
# cd /data/data/com.google.android.talk
```

图1-6　/data/data目录下各应用路径

虽然每个应用程序都不同，但是保存数据和设置文件比较多的应用程序会有如下目录结构。诊断应用程序漏洞或对移动设备取证时，需仔细查看如下目录，因为影响应用程序服务的重要信息全部保存于此。比如服务认证密钥、内容数据、应用程序设置文件等。

```
ls -l
drwxrwx--x app_24   app_24            2013-06-11 14:01 files
drwxr-xr-x system   system            2013-06-11 13:24 lib
drwx------ app_24   app_24            2014-01-27 14:42 databases
drwxrwxrwx app_24   app_24            2013-06-11 13:27 usrdata
-rw-rw-rw- app_24   app_24          6 2013-06-11 14:12 widget1.set
drwxrwx--x app_24   app_24            2013-06-17 12:46 cache
```

```
-rw------- app_24   app_24         183 2013-06-11 13:27 device_token.txt
drwxrwx--x app_24   app_24             2014-01-27 14:35 shared_prefs
```

表1-3 应用程序文件目录结构

目 录	说 明
files	保存管理员内部使用的文件（包括so文件、data文件、ini文件等）
lib	保存应用程序请求的库文件（存在so文件）
databases	包含设置文件、内容文件等的查询信息的SQLite数据库文件（存在db文件）
cache	有读写权限，包括浏览器缓存和用户临时状态信息
shared_prefs	保存为XML文件，是应用程序共享的设置文件

其中，shared_prefs目录内的preferences.xml文件包含应用程序的设置文件。因为包括升级、版本信息等内容，所以恶意访问时（或诊断漏洞时），此处会包含API密钥的盗用、认证密钥值等信息。

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<int name="whitelistrevision" value="1" />
<string name="productversion">2.0.1.1(Build 12)</string>
<string name="patchahnuibuilddate">2.0.4.61</string>
<int name="profilerevision" value="111" />
<int name="versioncode" value="61" />
<boolean name="initialzingstate" value="true" />
<string name="patchurl">http://www.test.co.kr/updates</string>
<string name="updateahnuibuilddate">2013.02.03.00</string>
<string name="productbuildnumberdate">2.0.4.11</string>
<string name="engineversion">2013.02.03.00</string>
</map>
```

下一个重要文件是保存于databases的db文件，它是SQLite数据库形式，可以使用SQLite数据库浏览器查看结构、浏览数据。

SQLite数据库浏览器工具下载URL：http://sourceforge.net/projects/sqlitebrowser/.

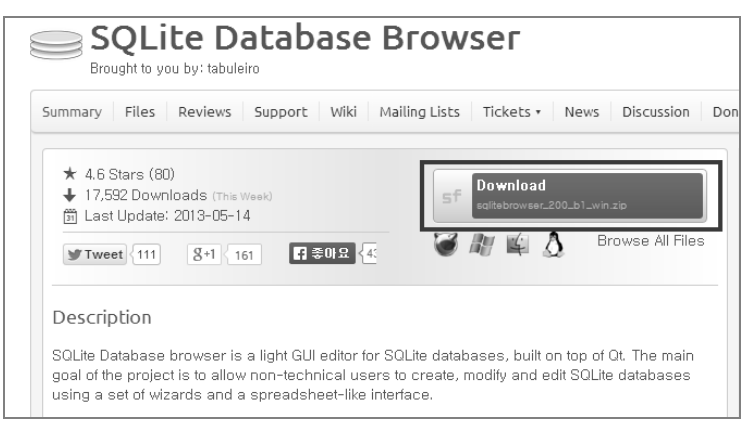


图1-7 下载SQLite数据库浏览器

如图1-8所示，访问某应用程序中的db文件。该db文件包括应用程序设置信息和结果导出语句。使用adb pull命令将需要分析的文件下载到个人PC。（adb命令参见第2章。）

```
master.db
# ls -l
ls -l
-rwxr-xr-x app_24   app_24       3072 2013-06-11 13:27 memo.db
-rw-rw--w- app_24   app_24       5120 2013-06-11 13:25 indicatoruser.db
-rwxr-xr-x app_24   app_24      14336 2013-06-11 14:05 webview.db
-rwxr-xr-x app_24   app_24       6144 2013-06-11 14:05 webviewCache.db
-rwxr-xr-x app_24   app_24       3072 2014-01-27 14:42 config.db
-rwxr-xr-x app_24   app_24       4096 2013-06-11 13:27 history.db
-rw-rw-r-- app_24   app_24     458752 2014-01-27 14:41 indicatordef.db
-rw-rw---- app_24   app_24      10240 2014-01-27 14:42 virus.db
-rwxr-xr-x app_24   app_24    2898331 2014-01-27 14:42 master.db
#
```

图1-8　查看应用程序内部数据库文件

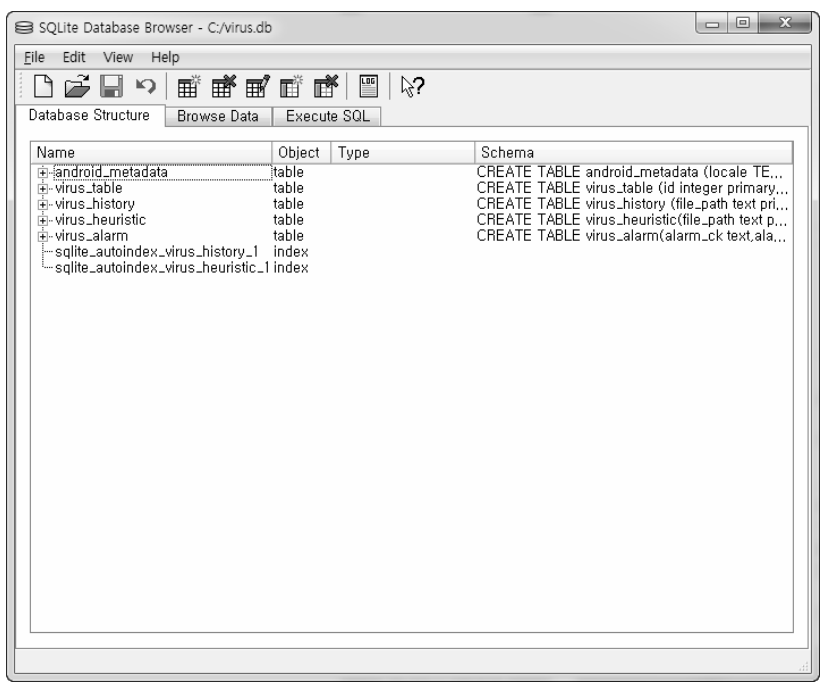


图1-9　查看数据库（db）文件结构和内容

其他常用工具还有SQLite Expert[①]。SQLite Expert有个人版（Personal）和专业版（Professional）。

各位可以在两个工具之间随意挑选。我们只对SQLite数据库信息进行简单查询，所以二者在功能上没有太大区别（诊断漏洞时也会修改一些数据库信息，以修改价格或绕过认证）。

① SQLite Expert主页：http://www.sqliteexpert.com

图1-10 下载SQLite Expert个人版

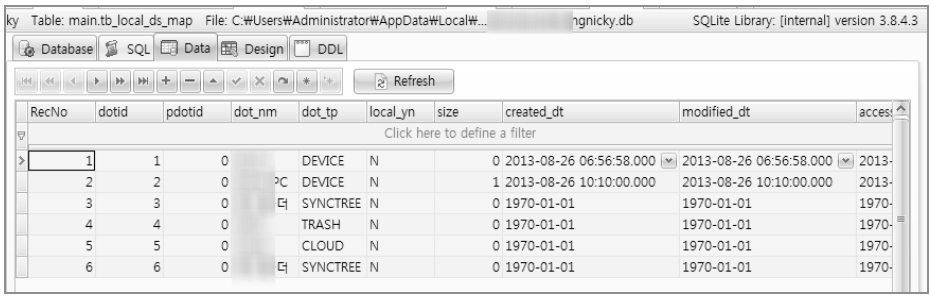


图1-11 用SQLite Expert查看到的数据库信息

使用本地PC检索工具Everthing查找所有已经保存的数据库文件（.db）时，会找到很多有趣的信息。数字取证时经常会用到这些信息，希望各位不仅要检查移动设备数据库，也要检查浏览器、应用程序数据库，看看有什么隐藏信息。

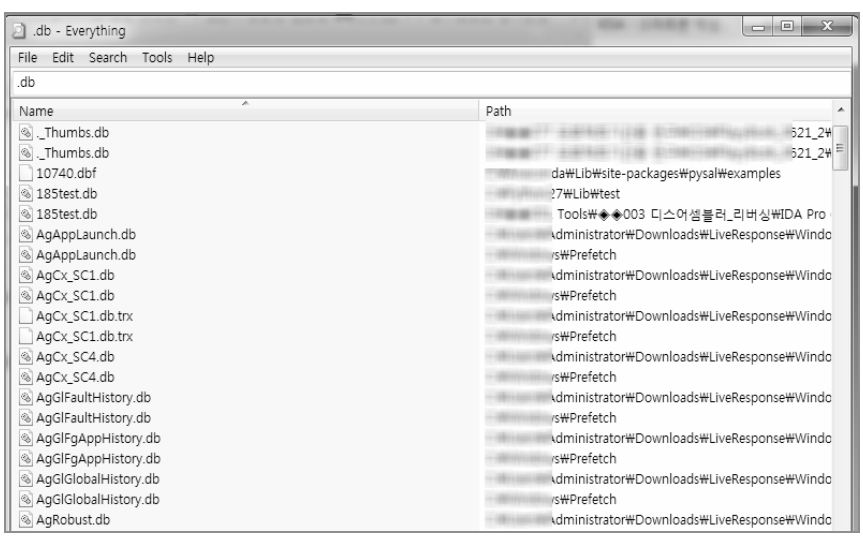


图1-12 本地PC内保存的数据库信息

1.2　Android 重要组件

Android由Activity、服务（Service）、内容提供商（Content Provider）、广播接收器（BroadCast Receiver）等4种主要功能组成。本书并不面向程序员，所以仅作简单介绍。

表1-4　Android重要组件

组　件	说　明
Activity	向用户显示的设备界面。通过点击菜单或按钮等特定动作转换的画面都可以称为Activity
服务	不显示到屏幕，在后台运行。如网络传输、读取文件等操作。Activity显示画面时，服务功能通常一同运行
内容提供商	应用程序共享的空间。即使数据保存到文件系统或其他地方，应用程序也能通过内容提供商访问数据
广播接收器	实时查看系统状态（电池状态、邮件提醒等），发生事件时响应。利用设备中发生的Notification等向用户发出警报

1.2.1　Activity

Android环境中，Activity大都在同一个画面显示。例如，在首页（Main Activity）点击一个按钮，就会跳转访问下一个页面（Second Activity），之后又会以其他动作触发另外的Activity。

此类Activity的执行就像图1-13的Java表示的生命周期。即使第二个Activity显示到屏幕，第一个Activity也会保存到其他空间，并变为静止状态（Stopped）。用户回到之前的Activity时立即显示，无需等待。

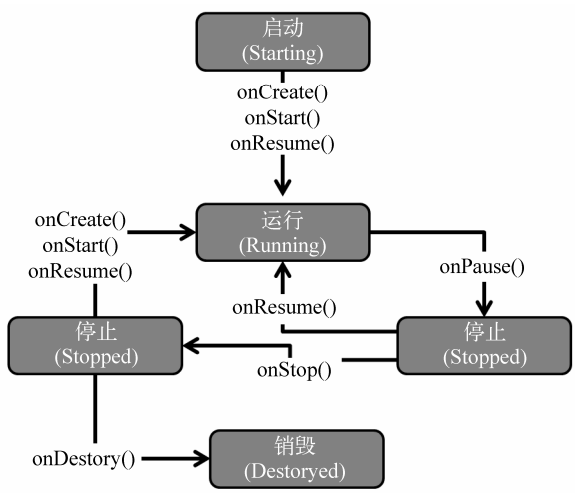


图1-13　Activity的生命周期

表1-5 Activity生命周期详解

状 态	说 明
启动（Starting）	Activity启动时，如果内存中没有相关信息，就以启动状态运行。回调函数执行操作，并转换为运行状态
运行（Running）	显示到用户屏幕，并实际在Activity上执行各项动作。输入文字或触摸屏幕时的状态
待机（Paused）	虽然屏幕上依然显示，但焦点（Focus）并不位于此处。例如，特定信息使对话框弹出到Activity之前时
停止（Stopped）	虽然不显示到屏幕，但仍存在于内存。用户可以随时从下一个Activity的执行过程迅速回到当前Activity
销毁（Destroyed）	从内存中销毁

1.2.2 Service

Service不会显示给用户，如图1-14所示，其运行过程与Activity运行过程相同。听音乐或录音都在后台运行，与其他软件的运行是同时的，这些都属于Service的功能。 Activity和Service都以名为UI线程的相同应用程序线程执行。

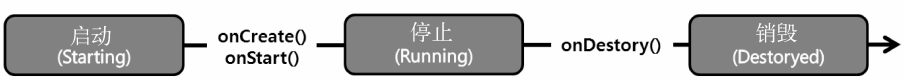


图1-14 Service的生命周期

1.2.3 Content Provider

Content Provider是应用程序之间共享数据的界面。Android的每个应用程序都默认在Sandbox中运行，所以与系统中的其他应用程序相互分隔，不能直接访问数据。Content Provider遵守CURD（Create、Update、Read、Delete，创建、更新、读取、删除）原则。应用程序通过Intent共享小数据。Content Provider适合共享音乐文件、图片文件等大容量文件。

1.3 Android 应用程序的基本结构

分析Android系统之前，首先要了解Android应用程序的编写顺序，因为理解了编译过程才能了解反编译过程。Android应用程序的开发顺序如图1-15所示。源代码编译后生成apk文件，该文件和压缩文件一致。apk文件大致包含.dex文件、resources文件、uncompiled resources文件、AndroidMainifest.xml文件。之后须经过signing过程才能在模拟器或移动设备上正常运行。

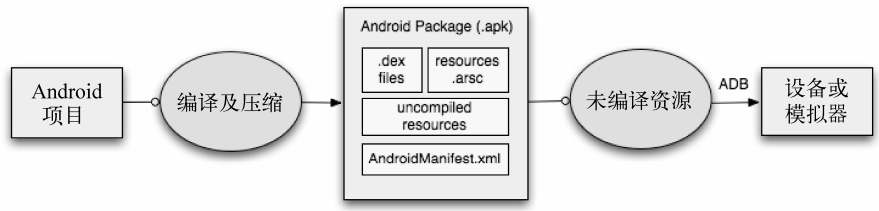


图1-15　Android应用程序开发流程

（出处：http://developer.android.com/tools/building/index.html#detailed-build）

本章将简单介绍生成apk文件的结果和权限。新建Android项目后，生成如图1-16所示的目录和文件。（第2章将详细介绍测试应用程序生成方法）

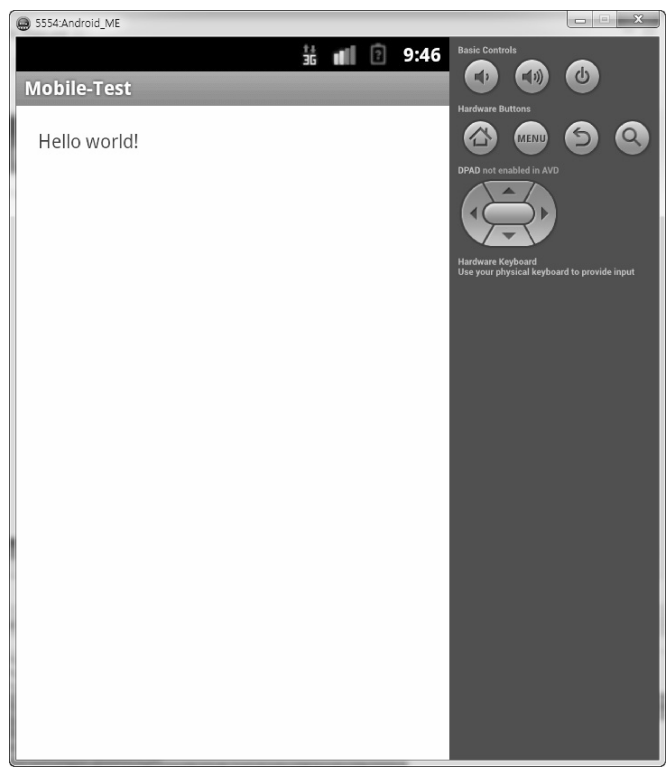


图1-16　在模拟器中运行测试应用程序

```
[2014-01-27 18:46:11 - AccelerometerPlay] Starting activity
com.example.android.accelerometerplay.AccelerometerPlayActivity on device
emulator-5554
[2014-01-27 18:46:12 - AccelerometerPlay] ActivityManager: Starting: Intent
{ act=android.intent.action.MAIN cat= [android.intent.category.LAUNCHER]
```

```
cmp=com.example.android.accelerometerplay/.AccelerometerPlayActivity }
[2014-01-27 18:47:24 - Mobile-Test] ------------------------------
[2014-01-27 18:47:24 - Mobile-Test] Android Launch!
[2014-01-27 18:47:24 - Mobile-Test] adb is running normally.
[2014-01-27 18:47:24 - Mobile-Test] Performing
com.example.mobile_test.MainActivity activity launch
[2014-01-27 18:47:24 - Mobile-Test] Automatic Target Mode: Unable to detect device
compatibility. Please select a target device.
[2014-01-27 18:47:31 - Mobile-Test] Uploading Mobile-Test.apk onto device
'emulator-5554'
[2014-01-27 18:47:35 - Mobile-Test] Installing Mobile-Test.apk...
[2014-01-27 18:47:37 - Mobile-Test] Success!
[2014-01-27 18:47:37 - Mobile-Test] Starting activity
com.example.mobile_test.MainActivity on device emulator-5554
[2014-01-27 18:47:38 - Mobile-Test] ActivityManager: Starting: Intent
{ act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER]
cmp=com.example.mobile_test/.MainActivity }
```

下面解压并查看用于测试的apk文件。如图1-17所示，把下载的apk文件或安装于移动设备的apk文件的扩展名修改为.zip后，即可使用常用的免费压缩软件ALZip或7-zip解压。

图1-17 将apk文件改为zip文件

解压后可以看到如图1-18所示的文件夹和文件，此处须仔细查看AndroidManifest.xml文件和classes.dex文件。分析恶意代码时可以从其他源文件内包含的多种图片和信息中获得提示，诊断应用程序时需考虑收费软件中可能包含内容文件。

图1-18 解压apk文件后的内部目录和文件

AndroidManifest.xml文件位于项目根目录，该文件包含了对组件的定义和应用程序的使用权限。Android中有很多可能遭恶意使用的API，所以要查看是否存在使用权限外的多余权限。

查看解压后的AndroidManifest.xml文件时显示乱码，因为当前是二进制格式，需要将其转换为xml文件格式才能正常显示。

如果只想转换一个XML文件，可以使用AXML Printer[①]，但我使用另一个反编译软件apktool.bat[②]。

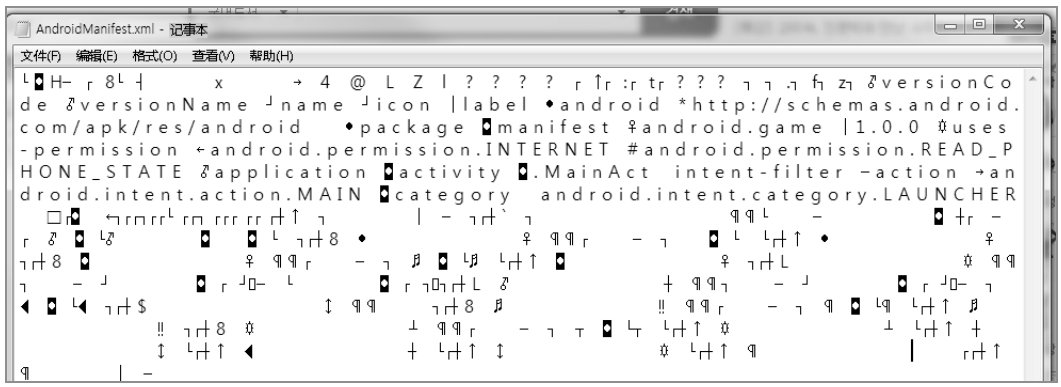


图1-19　二进制格式的AndroidManifest.xml文件

用调试模式解压程序。使用调试模式后，xml文件也同时转换为明文字符，可直接查看。图1-20中的两个文件是实际用作恶意代码的应用程序。

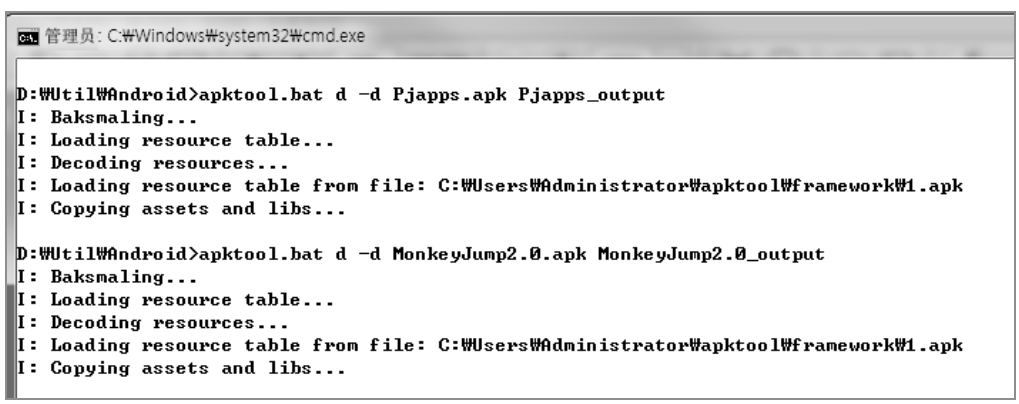


图1-20　转换AndroidManifest.xml文件

请看如下从正常程序提取的xml文件，显示了API层级版本信息和Activity信息。重要的是使用权限，此处只包含了INTERNET和READ_PHONE_STATE的API信息。

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest android:versionCode="1" android:versionName="1.0.0"
package="android.game"
  xmlns:android="http://schemas.android.com/apk/res/android">
```

① AXMLPrinter下载：http://code.google.com/p/android4me

② apktools下载：https://code.google.com/p/android-apktool/downloads/list

```
   <uses-permission android:name="android.permission.INTERNET" />
   <uses-permission android:name="android.permission.READ_PHONE_STATE" />
   <application android:label="@string/app_name" android:icon="@drawable/icon">
      <activity android:label="@string/app_name" android:name=".MainAct">
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
   </application>
</manifest>
```

下列示例分析了Android恶意代码应用程序。从Activity名推测，它应该是在中国开发的。通过下端的权限信息可知，存在SMS相关API。这部分比较危险，会利用SMS扣除话费或发送垃圾短信。

自动分析恶意程序代码的工具会浏览该xml文件，以判断是否存在发生危险行为的API。

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest android:versionCode="1" android:versionName="1.0"
package="com.mobile.app.writer.zhongguoyang"
 xmlns:android="http://schemas.android.com/apk/res/android">
   <application android:label="@string/app_name"
android:icon="@drawable/icon">
      <activity android:label="@string/app_name"
android:name=".ZhongGuoYangActivity">
          <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
      <activity
android:theme="@android:style/Theme.Black.NoTitleBar.Fullscreen"
android:name=".VideoPlayerActivity"
android:configChanges="keyboardHidden|orientation" />
       <activity android:theme="@style/Theme.CustomDialog"
android:label="?于" android:name=".AboutActivity" />
       <meta-data android:name="Wooboo_PID"
android:value="1a27f9a5e5f74dedafb56c3dd6f3475f" />
       <meta-data android:name="Market_ID" android:value="177" />
       <service android:name="com.android.main.MainService"
android:process=":main" />
       <receiver android:name="com.android.main.ActionReceiver">
          <intent-filter>
             <action android:name="android.intent.action.SIG_STR" />
          </intent-filter>
       </receiver>
       <receiver android:name="com.android.main.SmsReceiver">
          <intent-filter android:priority="100000">
             <action android:name="android.provider.Telephony.SMS_RECEIVED" />
          </intent-filter>
       </receiver>
       <activity android:theme="@android:style/Theme.Dialog"
```

```
android:name="com. android.main.TANCActivity" />
   </application>
   <uses-permission android:name="android.permission.INTERNET" />
   <uses-permission android:name="android.permission.RECEIVE_SMS" />
   <uses-permission android:name="android.permission.SEND_SMS" />
   <uses-permission
android:name="com.android.browser.permission.READ_HISTORY_BOOKMARKS" />
   <uses-permission
android:name="com.android.browser.permission.WRITE_HISTORY_BOOKMARKS" />
   <uses-permission android:name="android.permission.INSTALL_PACKAGES" />
   <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
   <uses-permission android:name="android.permission.READ_PHONE_STATE" />
</manifest>
```

如图1-21所示，开发应用程序时，使用开发工具提供的功能设置AndroidManifest.xml文件的Permissions选项，即可反映用户权限。

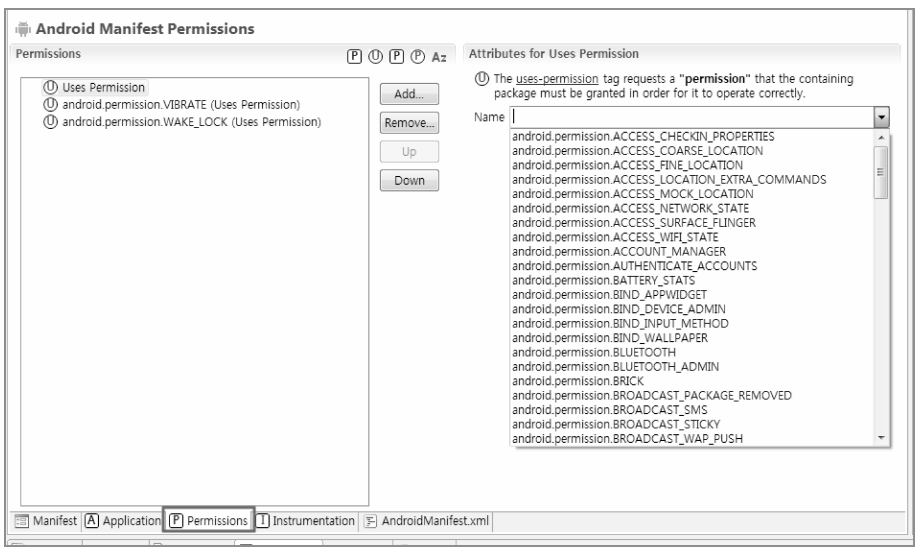


图1-21　向AndroidManifest.xml文件添加权限

编写恶意代码时也能简单适用。通常会调制正常的应用程序，先反编译为原应用程序后，使用XML编辑器直接修改源代码权限。

下面是调查了多个恶意应用程序后得到的、添加到AndroidManifest.xml文件的不必要的权限目录。虽然不能断定拥有这些权限的就是恶意应用程序，但对分析恶意代码应用程序功能有很大参考价值。通过后面的名称（Name）可以判断执行何种功能。

```
android.permission.SEND_SMS, android.permission.RECEIVE_SMS
android.permission.SYSTEM_ALERT_WINDOW
com.android.browser.permission.READ_HISTORY_BOOKMARKS,
com.android.browser.permission.WRITE_HISTORY_BOOKMARKS
```

```
android.permission.READ_CONTACTS, android.permission.WRITE_CONTACTS,
android.permission.READ_CALENDAR, android.permission.WRITE_CALENDAR
android.permission.CALL_PHONE
android.permission.READ_LOGS
android.permission.ACCESS_FINE_LOCATION
android.permission.GET_TASKS
android.permission.RECEIVE_BOOT_COMPLETED
android.permission.CHANGE_WIFI_STATE
```

●● 可以仅通过权限的数字判断是否为恶意代码吗？

F-Secure开发公布了一款很有趣的应用程序，它会检查用户手机上安装的所有应用程序的权限（Permission），并把权限的数字显示在程序图标旁。我刚开始认为这个程序很有意思，但后来发现，它只显示权限数字及其说明。用户会误以为权限大的软件就是恶意软件。下面是Android市场中注册的应用程序说明图片，人们常用的网络电话Viber位列第一，因为其使用权限最大。

我们必须赋予自己常用的SNS和视频通话、语音通话等服务很大权限，因为此类软件需要用户的很多信息。

https://play.google.com/store/apps/details?id=com.fsecure.app.permissions.privacy

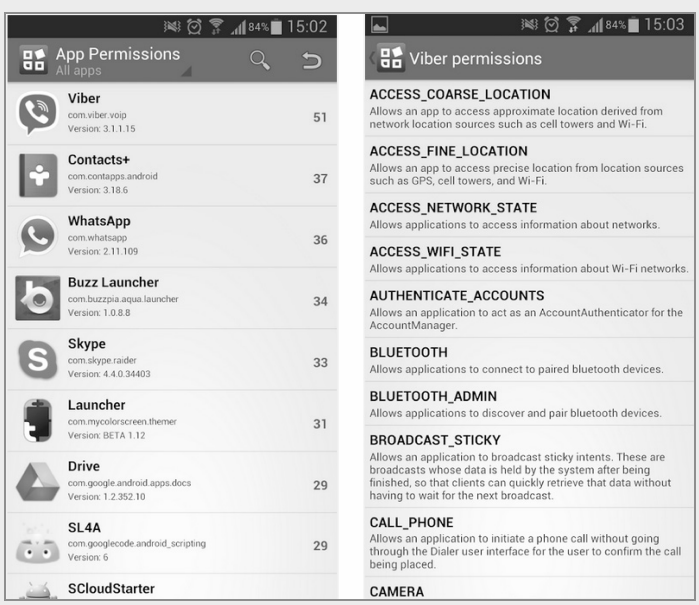


图1-22 F-Secure应用程序权限检测

图1-23摘自论文“Dissecting Android Malware: Char-acterization and Evolution”，统计分析了恶意软件权限（左）和从Android市场正常下载的免费软件权限。可以发现，虽然权限

类型相似，但恶意代码使用的权限数字（频率）很高，这说明恶意代码会要求大量权限以访问个人信息。

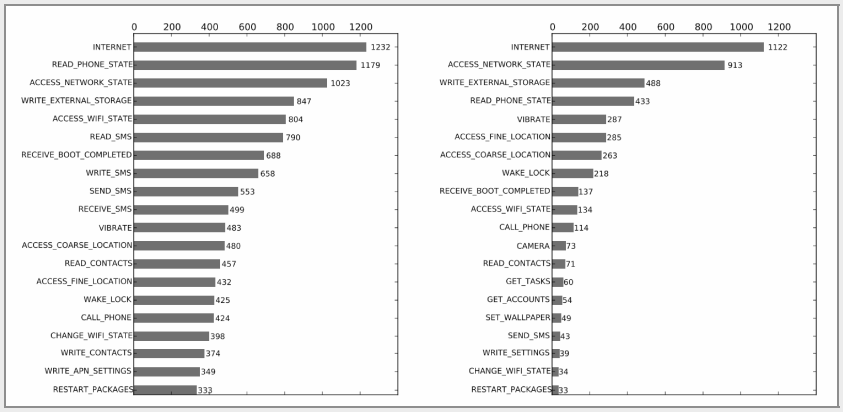


图1-23　恶意代码权限（左）和正常的免费工具应用程序权限（右）

以上述信息为参考，要想更准确地判断是否为恶意代码，可以为其权限按比例赋分，高于平均分的即可判断为疑似恶意应用程序。也就是说，恶意代码会经常依序使用INTERNET、READ_PHONE_STATE等权限，所以为这些权限赋予高分；对READ_PHONE_STAT等正常使用的权限，可以赋予低分。

可参考下列网址。

- http://ant.apache.org/bindownload.cgi
- http://developer.android.com/tools/projects/projects-cmdline.html#UpdatingAProject
- http://mrkn.co/s/post/1473/Android_Security_Underpinnings.htm
- http://www.csc.ncsu.edu/faculty/jiang/pubs/OAKLAND12.pdf

1.4　小结

本章讲解了Android系统的基本结构。如果以开发为目的，则需要深入学习基本概念和各组成部分。但分析或诊断恶意代码时，只查看应用程序功能也不会影响分析。第2章将讲解分析应用程序必备的环境构建方法。

第 2 章
Android 应用程序诊断环境

第2章将讲解如何构建分析Android应用程序时所需的移动设备环境和无线网络。分析Android系统前，需要安装开发软件时必备的SDK和相关工具。执行应用程序诊断时，不仅需要工具包中的模拟器，而且需要实际的设备。虽然使用现有设备也可以进行诊断，但是为了利用所有工具，必须切换（rooting）移动平台。这样并不安全，但这是诊断时的必经过程。

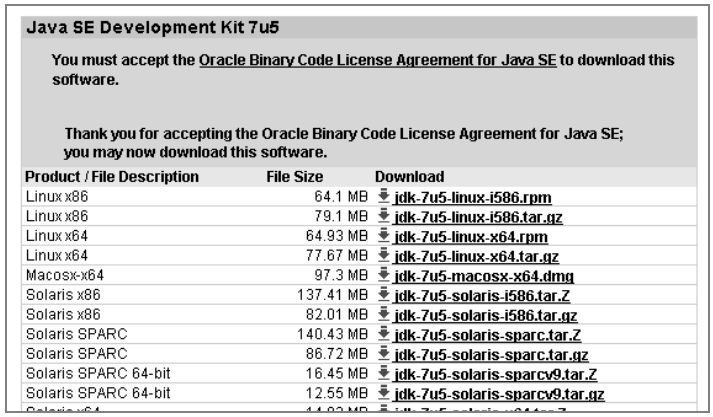
2.1 构建 Android 环境

本节是进入Android系统诊断、分析之前的环境构建阶段。首先查看Windows环境和Linux环境。分析恶意代码时，通常会使用虚拟机（模拟器）；但分析正常应用程序时，会使用真实的移动终端。因为应用程序的某些功能在模拟器上无法运行，并且性能也有限。无论是在模拟器上进行诊断还是在真实设备上诊断，都需要构建Android开发环境。

2.1.1 安装Android SDK

构建Android诊断环境之前，需要安装Java开发工具JDK（或JSE）。安卓以Java语言为基础，未安装JDK将无法诊断。

下载URL：http://www.oracle.com/technetwork/java/javase/downloads/index.html



Java SE Development Kit 7u5

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux x86	64.1 MB	jdk-7u5-linux-i586.rpm
Linux x86	79.1 MB	jdk-7u5-linux-i586.tar.gz
Linux x64	64.93 MB	jdk-7u5-linux-x64.rpm
Linux x64	77.67 MB	jdk-7u5-linux-x64.tar.gz
Macosx-x64	97.3 MB	jdk-7u5-macosx-x64.dmg
Solaris x86	137.41 MB	jdk-7u5-solaris-i586.tar.Z
Solaris x86	82.01 MB	jdk-7u5-solaris-i586.tar.gz
Solaris SPARC	140.43 MB	jdk-7u5-solaris-sparc.tar.Z
Solaris SPARC	86.72 MB	jdk-7u5-solaris-sparc.tar.gz
Solaris SPARC 64-bit	16.45 MB	jdk-7u5-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	12.55 MB	jdk-7u5-solaris-sparcv9.tar.gz

图2-1　必须安装JDK或JSE

●● 如果安装Windows 7 64位环境时出现错误

现在，大多数用户都在使用64位 Windows 7系统。64位环境下的JDK和Android SDK、Eclipse之间的搜索环境不同，经常会出现错误。因此，请先参考下列内容，再继续阅读本书。

下文参考了http://www.098.co.kr/?mid=blog&category=31862&document_srl=36613（短链接：http://goo.gl/T21ew9）。

- 从 http://www.oracle.com/technetwork/java/javase/downloads/index.html 下载 JDK 后安装。必须安装32位版本。即使操作系统是64位的，安装64位JDK后，Android SDK也无法识别。此外，还必须设置PATH变量。
- 从http://www.eclipse.org/downloads/下载32位Eclipse。解压后移动到适当的位置。（请参考后面的内容。）
- 从http://developer.android.com/sdk/index.html下载Android SDK后安装。选择相应的平台下载并安装。

为了测试Android平台的应用程序（与开发应用程序时一致），首先学习Android SDK的安装过程。

从http://developer.android.com/sdk/index.html可以下载到适用于Windows、Mac OS X、Linux（i386）等各种环境的SDK。本节学习构建Windows环境的方法。

表2-1　Android SDK安装文件目录

平　台	工具包	大　小	SHA-1校验和
Windows	installer_r24.2-windows.exe (Recommended)	107849819字节	e764ea93aa72766737f9be3b9fb3e42d879ab599
	android-sdk_r24.2-windows.zip (Recommended)	155944165字节	2611ed9a6080f4838f1d4e55172801714a8a169b