Computer Science and Application 计算机科学与应用, 2014, 4, 197-203
Published Online September 2014 in Hans. http://www.hanspub.org/journal/csa
http://dx.doi.org/10.12677/csa.2014.49028

Hans汉斯

Design and Control on a Self-Turn-On Program of Android System

Tianhong Zhou[1], Siqing Zhang[2], Mingliang Liu[3]

[1]Department of Information Engineering, Wuhan Business University, Wuhan
[2]Information Engineering College, Zhengzhou College of Science & Technology, Zhengzhou
[3]Department of Computer Science, Yunyang Teacher's College, Shiyan
Email: 109051308@qq.com

Received: Jul. 2nd, 2014; revised: Aug. 1st, 2014; accepted: Aug. 11th, 2014

Abstract

Android is the execution environment of applications on mobile devices. A self-turn-on program of Android system brings convenience to the user. Based on the analysis of the system architecture and the start-up process of Android, combined with the specific programming, a self-turn-on program in Android system has been achieved in simulation. At the same time three specific control methods on self-turn-on programs are put forward from the programmer's point of view, and the safety feasible measures from the user's point of view are proposed, to provide the basis for the analysis of Android system security.

Keywords

Android, Self-Turn-On, Program, Control

Android系统的自启动程序设计与控制

周天宏[1]，张思卿[2]，刘明良[3]

[1]武汉商学院，信息工程系，武汉
[2]郑州科技学院，信息工程学院，郑州
[3]郧阳师范高等专科学校，计算机科学系，十堰
Email: 109051308@qq.com

收稿日期：2014年7月2日；修回日期：2014年8月1日；录用日期：2014年8月11日

摘　要

Android是在移动设备上执行应用程序的环境，自启动应用程序会给用户带来便利。通过对Android的系统架构与启动流程分析，结合具体编程，模拟实现了Android系统下应用程序的自启动；同时从应用程序开发者的角度提出了三种具体控制自启动应用程序运行的方法，并从使用者的角度提出了可行的安全处理措施，为Android系统安全分析提供了依据。

关键词

Android，自启动，程序，控制

1. 引言

Android 本义是指“机器人”，同时也是 Google 于 2007 年 11 月 5 日宣布的基于 Linux 平台的开源手机操作系统的名称，它由操作系统、中间件、用户界面和应用软件组成。2012 年 7 月美国科技博客网站 Business Insider 评选出二十一世纪十款最重要电子产品，Android 操作系统和 iPhone 等榜上有名。2012 年 11 月数据显示，Android 占据全球智能手机操作系统市场 76%的份额，中国市场占有率为 90%。2013 年 09 月 24 日谷歌开发的操作系统 Android 在迎来了 5 岁生日，全世界采用这款系统的设备数量已经达到 10 亿台[1]。

特别，Android 操作系统是免费、开源的，Google 通过与运营商、开发商以及设备制造商等机构形成的战略联盟，希望通过共同制定标准使其成为一个开放式的生态系统。因而 Android 所有的应用程序都是可替换和扩展的，而程序之间可以无障碍沟通，开发人员能自主决定应用程序的权限。加上基于 WebView 内核的组件方便在应用程序中嵌入 HTML、JavaScript，以及主流开发环境 Eclipse + ADT + Android SDK 的使用。使得 Android 系统受到越来越多的手机厂商、软件厂商、运营商以及个人开发者的追捧，进而 Android 系统的安全性考虑就摆上桌面。现有的文献资料主要关注 Android 系统整体架构的安全[2] [3]，手机的病毒及预防[4] [5]，和应用程序自身的安全性[6]。但更需要我们关注 Android 下自启动应用程序带来的利与弊。

2. Android 的系统架构与启动流程

Android 的系统架构和其它操作系统一样，采取了分层的架构[7]。从 Google 提供的系统架构图看，Android 从高层到低层排列分为以下四层：

应用程序层：由运行在 Dalvik 虚拟机上的应用程序组成，Dalvik 是为 Android 专门设计的基于寄存器的虚拟机，运行一种 dex 文件，且被认为比 Java 类文件更加简洁和节省内存的文件。常见的应用如联系人管理、日历等

应用程序框架层：主要由 View、Notification Manager、Activity Manager 等组成，它们是开发人员可以直接调用的组件，都是由 Java 语言编写，更是开发者进行 Android 开发的基础。

系统运行库层：这是对应用程序框架层的支持层，主要包括 C 语言标准库、多媒体库、SQLite、OpenGL ES、Dalvik 虚拟机等。

Linux 内核层：主要由硬件驱动、内存管理、进程管理等组件组成。

因而，从某个方面来讲，Android 可以看成是 Linux 之上的一种 XWindow。而 Android 系统的启动过程主要分为四步：1) Init 启动；2) Zygote 启动；3) Systemserver 和应用服务启动；4) 应用程序启动[8] [9] (图 1)。

其中：

1) Init 是由内核启动的第一个程序。在 init 执行过程中，将会解析并执行脚本文件，执行命令，创建新服务。直至 Android 系统的底层框架搭建完成；

2) Zygote 是 Init 要创建的一个服务。由于 Init 创建服务的方式是采用创建子进程，多进程可同时进行。因而在 Zygote 执行过程中，与 Zygote 同层的服务也在启动和执行。当 Zygote 完成启动 Dalvik 虚拟机的工作后，Android 系统基本完成系统运行库层的搭建；

3) 虚拟机启动完成后，Zygote 将试图启动 Systemserver，在 Systemserver 中将逐个启动系统上层服务，这些服务就是 Android 系统应用程序框架层的内容。

4) 至此，系统拥有了启动应用程序的能力，系统通过活动管理器 Activity Manager 激活第一个应用程序，进入用户待机画面，完成启动。

3. 开机自启动程序的模拟

结合 Android 系统的启动流程，我们知道当 System Service 加载完所有的系统服务后，系统就会向所有的系统服务发送一个广播，通知系统已经准备完毕。当 Activity Manager Service 接收到系统已经准备完毕的广播后，会向 Zygote 进程发送创建 Dalvik 虚拟机实例的请求[10]。包括初始化一个 Dalvik 虚拟机实例，装载 Socket 请求所需的类和监听，创建虚拟机实例来管理应用程序的进程。然后 Activity Manager Service 在系统中查找并启动具有

<category android:name = "android.intent.category.HOME”/>

属性的 Activity。因此，Activity Manager Service 同样也会启动其它开机自启动的应用程序。而 Intent 对象不仅可以启动应用程序内部的或者其它应用程序的 Activity，而且可以发送广播动作 Broadcast Action。

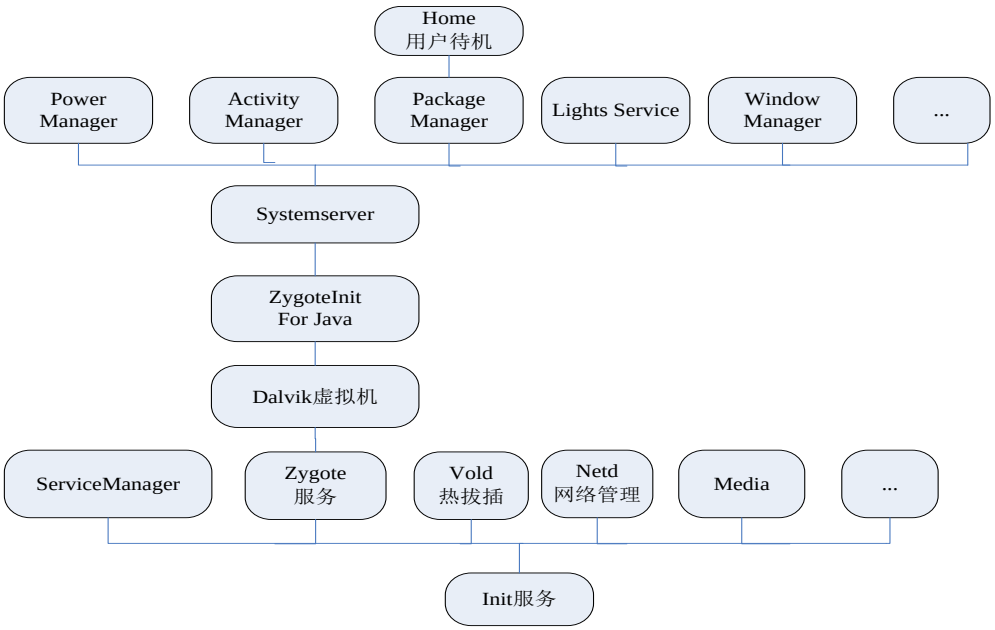

Figure 1. Start-up procedure of the file system in Android system
图 1. Android 系统中文件系统的启动过程图

Broadcast Action 和 Activity Action 一样，既可以由系统负责广播，也可以由自己的应用程序负责广播，以实现某些既定功能。实际上，在手机中发生类似的事件时，Android 都会向整个系统发送相应的 Broadcast Action。如果应用程序接收到这些 Broadcast Action，就可以来完成相应的功能。于是，实现开机自启动的功能就可以通过接收如下的系统广播：

android.intent.action.BOOT_COMPLETED

并编写接收该系统广播的类来实现。

从具体编程的角度来讲[11] [12]，设计自启动程序的大致步骤如下：

第一步：编写一个 android.content.BroadcastReceiver 的继承类 StartupReceiv，并实现 BroadcastReceiver 类中的 onReceiver 方法。以便应用程序接收到系统发送的广播时，去调用这个 onReceiver 方法。代码如下：

```
package net.blogjava.mobile.startup;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class StartupReceiver extends BroadcastReceiver
{
    @Override
    public void onReceive(Context context, Intent Intent)
    {
        Intent    mainIntent = new Intent(context, Main.class);
        mainIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(mainIntent);
    }
}
```

第二步：在 onReceive 方法中启动开发者希望的应用程序中的 Main，类似代码如下

```
package net.blogjava.mobile.startup;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class Main extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
```

```
        }
    }
```

以表示自启动应用程序已经启动。在应用程序中可以植入体现开发者的思想。

第三步：在 AndroidManifest.xml 文件中配置 StartupRecevier 类，使用<receiver>标签来描写可以接收哪一个 Broadcast Action。AndroidManifest.xml 文件是每个 Android 程序中必不可少的，它包含了组成应用程序的每一个组件的节点，并使用 Intent 过滤器和权限来确定这些组件之间以及这些组件和其他应用程序是如何交互的。Android 定义一系列 manifest 权限，以保护系统或其他应用程序的各个方面。请求权限可以在 manifest 文件中声明一个 <user-permission> 属性：

```
<uses-permission android:name="string" />
```

其中 android:name 指定权限的名称。类似代码如下：

```
<receiver android:name="StartupReceiver">
<intent-filter>
        <!—指定接收的 Broadcast Action -->
        <action android:name="android.intent.action.BOOT_COMPLETED" />
        <!—指定 Action 的种类，即 Android 系统启动后第一个运行的应用程序 -->
       <category android:name="android.intent.category.HOME" />
  </intent-filter>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
</receiver>
```

在程序安装在手机或者模拟器后，只要 Android 系统向外广播这个 Broadcast Action，并且应用程序没有被卸载的话，系统就会自动调用应用程序的 onReceiver 方法来处理它，从而实现程序的自启动。即重启模拟器或者手机重新开机启动后，就会发现模拟器或者手机启动后总是会先运行上述应用程序。

4. Android 自启动程序的控制

4.1. 从应用程序开发者的角度

开发 Android 应用程序时，必须处理很多与安全性相关的方面。虽然 Android 本身已提供应用沙盒、权限、应用签名等安全机制，但是这种以开发者为中心、依托用户完成的粗粒度授权机制存在诸多问题和漏洞，导致平台安全性大大降低，特别是容易造成应用层权限提升攻击，通过利用 Linux 内核的优势，Android 得到了大量操作系统服务，包括进程和内存管理、网络堆栈、驱动程序、硬件抽象层以及与安全性相关的服务[6]。

Android 系统通过应用程序自行在系统中登记注册事件(即 Intent)来响应系统产生的各类消息。正如自启动应用程序模拟中考虑的那样，配置表示应用程序会响应系统产生的 android.intent.action.BOOT_COMPLETED (系统启动完成)信号，以此来实现应用程序自启动的控制。具体控制自启动应用程序的方法有

1) 静态分析

静态分析技术是将可执行程序反汇编，通过分析反汇编代码来理解其代码功能，即在不执行应用程序的情况下对其行为进行分析的技术[4]。

基于上述原理，我们可以通过对系统中已安装的程序去除其 Android Manifest 的上述配置片段来控制应用程序的对系统的响应。在没有源码可修改编译的情况下只能实现屏蔽其对有些信号的响应，完成屏蔽该程序不再开机自启动。手工方法就是利用有关工具直接在解压其 APK 包后，删除其 Android Manifest

的上述配置行后再打包成 APK，最后安装到系统中就实现了屏蔽其自启动功能。

有两种主流方式可以实现对 Android 恶意程序的静态逆向分析，一种是将 APK 程序利用 Apktool 等工具进行反编译，生成程序的源代码、图片、XML 配置和语言资源等文件，然后对相关文件进行查看，查找具有恶意意图的代码，并对其进行验证；另外一种是先使用 Dex2jar 工具将程序反编译成 jar 格式，然后再用 JD-GUI 直接查看源代码进行分析。

2) 编程处理

静态分析需要借助 Apktool 或 Dex2jar 等工具，步骤比较繁琐，我们可以通过自己开发程序来实现该功能。主要利用的是

a) Package Manager，本类 API 是对所有基于加载信息的数据结构的封装，其中具有增加，删除 permission 功能

b) Activity Manager，本类 API 是对运行时管理功能和运行时数据结构的封装，其中具有注册／取消注册动态接受 intent 功能。

具体 API 可参考 SDK 文档。通过调用相应现成的 API 实现就可以实现自启动程序的控制。

目前，已有的控制开机自启动的成熟应用程序主要有 autostarts，它不只是控制开机启动信号，也可以控制程序对大部分信号的响应行为。

3) 系统自带工具

Android 自带的 activity 管理工具/system/bin/am 和程序包管理工具/system/bin/pm，都可以用于自启动程序的控制。

a) am 全称 activity manager，可以使用 am 去模拟各种系统的行为，例如去启动一个 activity，强制停止进程，发送广播进程，修改设备屏幕属性等等。

在 Android 系统 adb shell 命令下执行 am 命令：

am <command>

也可以在 adb shell 前执行 am 命令：

adb shell am start -a android.intent.action.VIEW：启动一个 activity

而 force-stop <PACKAGE>是强制停止指定的 package 包应用。

b) pm 全称 package manager，可以使用 pm 命令去模拟 android 行为或者查询设备上的应用等，也有上述二种执行方式，其中

enable <PACKAGE_OR_COMPONENT>：使 package 或 component 可用。

disable <PACKAGE_OR_COMPONENT>：使 package 或 component 不可用。

4.2. 从使用者的角度

作为手机使用者，自身积极的防范是最重要的。建议用户在购买手机时尽量选择从正规渠道购买；在选择应用下载网站时，尽量选择大型可信站点，如 Google Play 商店等；尽量不要下载安装功能不清的软件；安装应用软件时，注意观察软件权限；慎重查看那些来历不明的短信和彩信；及时安装有效的手机防护软件并定期升级，在用户端构建安全墙；及时备份手机数据并定期检测手机系统是否正常等。

5. 结语

自启动应用程序确实给用户带来方便；同时根据 Android 开机自启动过程的分析，一般用户只须写一个广播接收器，收到这个广播后启动自己喜欢的应用(甚至包括监视等功能)，从而留下漏洞。但我们可以从应用程序开发者和使用者的角度，对自启动应用程序来加以控制。

随着手机朝着智能化方向的发展，Android 系统必然为人们关注，并会在应用中不可避免地出现大量针对 Android 的攻击行为和恶意软件。这些程序的制造者更多的是受到经济利益的驱使，制造并传播手机病毒，发动基于应用软件的恶意攻击，并利用应用软件发布缺乏验证机制的安全漏洞，发布含有恶意代码的程序供用户下载，从中获取不当收益。但 Android 开源的途径会促使安全性不断地改进和提升。

基金项目

2014 武汉商学院科研项目 2014A009，2013 年武汉市教育局教学研究项目 2013030，2013 年武汉商学院优秀教学团队项目。

参考文献 (References)

[1] 百度百科 android [EB/OL]. http://baike.baidu.com/subview/1241829/9322617.htm?fromId=1241829&from=rdtself

[2] 蒋绍林, 王金双, 等 (2012) Android 安全研究综述. *计算机应用与软件*, **10**, 205-210.

[3] 廖明华, 郑力明 (2011) Android 安全机制分析与解决方案初探. *科学技术与工程*, **6**, 6350-6355.

[4] 舒心, 王永伦, 张鑫 (2012) 手机病毒分析与防范. *第*27 *次全国计算机安全学术交流会论文集*, 九寨沟, 2012 年 8 月 21 日, 54-56.

[5] 毕倩倩 (2013) Android 手机系统病毒及保护机制研究. *第*28 *次全国计算机安全学术交流会论文集*, 贵阳, 2013 年 10 月 24 日, 81-83.

[6] Zhou, T.H., Liu, M.L. and Shen, J. (2014) Design for a self-turn-on program of a Android system. *International Conference on Electrical, Control and Automation*, Shanghai, 22-23 February 2014, 809-813

[7] Payet, E. and Spoto, F. (2012) Static analysis of Android programs. *Information and Software Technology*, **54**, 1192-1201.

[8] 禹建磊 (2012) Android 系统启动程序的优化设计与实现. 硕士论文, 西北大学, 西安.

[9] 金智义, 张戟 (2011) 嵌入式 Android 系统的启动研究. *佳木斯大学学报*(*自然科学版*), **4**, 521-528.

[10] Nicola, C.U. (2009) Einblick in die Dalvik Virtual machine. *IMVS Fokus Report*, **3**, 5-12.

[11] 李宁 (2012) Android 开发完全讲义. 中国水利水电出版社, 北京.

[12] E2EColud 工作室 (2009) 深入浅出 Google Android. 人民邮电出版社, 北京.