计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(9):262−267 [doi: 10.15888/j.cnki.csa.006547]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

Android 应用中 Exported Activity 测试途径研究①

王国珍, 杨红丽

(北京工业大学 信息学部, 北京 100124)
通讯作者: 王国珍, E-mail: wauoen@163.com

摘　要: Android 系统提供了多种应用间交互机制, 其中开放活动 (Exported Activity, EA) 不需要复杂的跨进程交互就可以被其他应用在运行时调用. 现在很多研究主要关注 GUI 组件的功能性测试, 但是在 Android 应用本身往往不会启动内部的开放活动, 所以开放活动有时候很难被覆盖到. 本文提出了一种系统化测试开放活动的方法, 使用该方法可以生成一组代理应用作为测试驱动程序启动应用中的开放活动. 首先, 使用静态分析技术解析 APK 文件, 提取出开放活动列表和启动它们需要数据的键值和类型; 其次, 将相应的数据填充到预先设置好的模板中, 生成测试驱动应用. 本文基于提出的测试方法开发了一款原型工具——EASTER, 使用一些真实的应用进行了实验. 实验结果显示, 所有测试应用共有 65 个开放活动, 其中有 20 个开放活动在被外部应用启动过程中存在漏洞.

关键词: 测试驱动程序; 开放活动; Android 应用; 系统化测试; 程序分析

引用格式: 王国珍,杨红丽.Android 应用中 Exported Activity 测试途径研究.计算机系统应用,2018,27(9):262–267. http://www.c-s-a.org.cn/1003-3254/6547.html

Research on Test Methods of Exported Android Activity

WANG Guo-Zhen, YANG Hong-Li

(Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China)

Abstract: The Android system provides various mechanisms for interactions between apps, of which the exported activity is an activity that can be launched by other apps during runtime without complex inter-process communication. Most of the existing works on testing Android apps mainly focus on the functionalities bound to the GUI components in the app, while the app often does not include the GUI callbacks to activate its exported activities. This study proposes a method to systematically test the exported activities in the way of generating a set of agent apps as test drivers to launch these activities. It first statically analyzes the APK file to figure out the exported activities and extract the keys and types of their required data items, and then fills this corresponding data to a pre-set template to build the test drivers. All these proposed techniques are implemented into a prototype tool called EASTER. The preliminary experiments on several real-world apps show that without comprehensive testing, some exported activities are vulnerable to various external apps launches.

Key words: test drivers; exported activity; Android application; systematically test; program analysis

随着移动互联网的高速发展, 智能手机已经成为每个人生活中必不可少的工具, 手机操作系统也正在高速发展. 截止到 2017 年, Android 智能手机在全球市场中占有超过 85% 的市场份额[1], 占据着市场的主导地位. Android 应用市场为用户提供了功能各异的应用程序, 截止 2017 年 2 月, Google Play 已经发布了 2700 000

① 收稿时间: 2018-01-28; 修改时间: 2018-02-27; 采用时间: 2018-03-09; csa 在线出版时间: 2018-08-16

个应用[2]. Android 应用的种类变得更加多样, 提供的功能也更加专业, 应用之间可以分享信息并且相互协作完成一些复杂的任务. 例如, 一个电子支付应用可以被多个第三方的电子商务应用调用来完成支付功能. Android 系统提供了开放活动 (Exported Activity, EA) 机制, 可以将应用内特定的 Activity 分享给其他的应用.

EA 往往包含开发者想要推广的关键功能, 可以被其他的应用重复执行. 因此, 它们的质量是关键功能得到有效推广的关键因素. 另一方面, 因为 EA 可以被任意的应用执行, 因此包含 EA 但是开发不是很完备的应用很容易被恶意应用滥用[3]. 综上, 开放活动的质量验证是一个非常值得关注的课题.

在 Android 应用质量验证方面, 测试是一个很热门的课题. 大部分现有的工作主要集中在分析和测试单个应用的图形交互界面 (Graphical User Interface, GUI) 上, 被测应用与外部应用之间的交互往往被忽略. 我们的实验结果表明, 在现有工作中 EA 很难被覆盖到. 因此, 在对 Android 应用的测试过程中应该使用针对性的技术对这类 Activity 进行测试.

本文讨论了如何系统的测试 Android 应用中的 EA. 首先, 自动的生成一系列的应用作为测试驱动应用, 然后利用这些测试驱动应用启动目标应用中的 EA, 发现其中的漏洞. 本文使用数据流分析技术得到启动 EA 所需的信息, 结合预先定义的模板生成测试驱动应用. 选取了 10 个现实生活中使用非常广泛的安卓应用进行了测试并且有效的检测到了一些问题. 通过分析测试结果, 将检测到的问题分为两类: 应用崩溃和界面异常. 实验结果表明本文提出的方法可以有效的对 EA 进行系统化测试, 间接提高了存在 EA 应用的质量.

1 相关背景知识

这部分内容将介绍在一些相关的背景知识, 主要包括 EA 和 Android 系统提供的 Intent 机制.

1.1 开放活动

Activity 是使用率最高的 Android 组件, 它提供了与用户交互的 GUI 窗口. Activity 可以分成两类: 内部活动 (Internal Activity, IA) 和开放活动 (Exported Activity, EA). 前者只能被同一应用中的其他组件启动, 后者允许被外部应用调用. EA 是一种应用间有效的交互、协作方式. 默认情况下除了应用的 Main Activity 之外所有的 Activity 都是 IA 类型, 开发者可以在 Manifest 文件中将 Activity 的属性 (android:exported) 设置为 TRUE, 使得该 Activity 变为 EA 类型. 如果不设置 android:exported 属性, 也可以在 Manifest 配置文件中 EA 对应的<activity>标签下添加一个或者若干个<intent-filter>标签, 将 Activity 设置为 EA 类型.

1.2 Intent

外部应用通过 Intent 机制调用其他应用中的 EA. Intent 是消息传递的载体, 其中包含目标组件的标识符和目标组件初始化所需的数据.

发送方组件通过一系列重载的 API (Application Programming Interface) 以键值对 (key-value) 的形式将不同类型的数据附加到 Intent 中, 例如, putExtra (String, int) 可以添加整形数据, putExtra (String, String) 可以附加字符串型数据. 该方法的第一个参数是用于识别不同数据的键值 (key), 第二个参数是表明该数据的值 (value). Android 系统还提供了接收方组件获取特定键值所对应数据的 API. 例如, getIntExtra (String key) 可以根据指定的 key 值来获取其对应的整形数值.

第三方开发商开发的应用通常会通过该机制为 EA 传递数据来完成一些任务, 所以数据的有效性检验和 EA 的容错性是非常重要并且需要特别关注的方向.

1.3 示例说明

本小节将通过一个简单的例子介绍本文研究的动机. 图 1 是样例应用中一个 EA(Foo) 的代码片段, 该 EA 不会被应用中的其他组件调用. 在 onCreate 方法中包含一条字符串类型数据和一条整形数据, 它们的键值分别为: “key1”、“key2”. 由于该 Activity 不会被应用自身调用, 所以现有的技术和工具[4,5]不会覆盖到它.

```
public class Foo extends Activity {
  protected void onCreate(Bundle b) {
    Intent intent = this.getIntent();
    String key1 = "key1";
    String key2 = "key2";
    String value1 = intent.getStringExtra(key1);
    int value2 = intent.getIntExtra(key2, 0);
      ......
}}
```

图 1 EA 的代码片段

图 2 是一个调用 EA 的测试驱动应用的示例代码. 它创建了一个 Intent 实例, 设置它的目标 Activity 组件为 Foo, 并且设置了 Foo 初始化数据, 最后通过

startActivity (Intent intent) 方法执行该 Intent. 通过这个例子可以发现, 对 EA 系统化测试的关键问题是如何为 Intent 中各种各样的数据赋值.

```
public class FooTestDriver extends Activity {
  protected void onCreate(Bundle b) {
    Intent intent = new Intent(
      FooTestDriver.this, Foo.class);
    String key1 = "key1";
    String key2 = "key2";
    intent.putExtra(key1, "abcdefg");
    intent.putExtra(key2, 12345678);
    this.startActivity(intent);
}}
```

图 2 Foo 的测试驱动应用

2 测试方法

本文提出了一种系统化测试 Android 应用中 EA 的方法. 该小节中, 首先对方法进行了简单的概述, 然后详细介绍几个关键模块.

2.1 方法概述

为了测试待测样本中所有的 EA, 本文所提方法的主要思想是自动化的生成一批测试驱动应用, 每个测试驱动应用会执行携带不同数据 (包括无效数据) 的 Intent, 启动被测 EA 以检测它处理不同数据时是否会出现漏洞. 图 3 是方法的概述. 它包含四个模块: Android 安装包 (Android Package, APK) 解析、Intent 分析、测试驱动生成、测试执行. 第一个模块提取待测样本中所有的 EA 信息, 包括: EA 的包名和类名等; 第二个模块获取每个 EA 启动所需的数据信息; 测试驱动生成模块生成一批可调用 EA 的测试驱动应用; 最后一个模块在一台真实的设备上执行生成的测试驱动应用并记录执行结果.

2.2 APK 解析

Android 应用大多数都采用 Java 语言实现并被编译为 Dalvik 字节码. 应用中的字节码文件、资源文件和配置文件最后被打包成一个二进制形式的 APK 文件. 为了进行后面的分析, 本文首先利用 Apktool[6]对 APK 文件进行反编译, 反汇编二进制代码和配置文件. Apktool 是 Google 官方提供的一款对 Android 应用逆向工程的工具. 基于 Android 官方文档中对 Manifest 配置文件的介绍, 解析 Manifest 文件提取所有 Activity 信息, 根据 EA 的 android:exported 属性以及<intent-filter>标签过滤出所有的 EA. EA 的判断依据分为以下两方面: 其一, AndroidManifest.xml 文件中<activity>标签的 android:exported 为 true; 其二, 在<activity>标签没有设置 android:exported 属性的情况下, 若该<activity>标签中包含一个或者若干个<intent-filter>属性, 该 Activity 也是 EA.

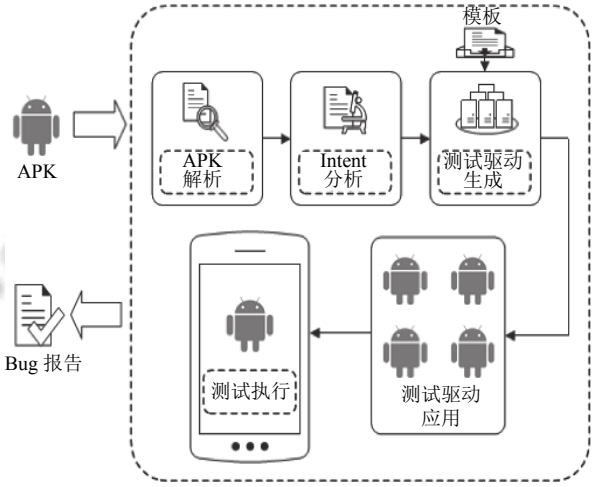

图 3 方法概述

2.3 Intent 分析

为了构建测试驱动应用, 需要获取每个 EA 所需的数据信息. 根据上文所述, Activity 主要利用若干特定的系统 API(参见 1.2) 将键值 (或者数据名称) 指定为 API 的参数来获取数据内容. 因此, 该模块的目标就是提取每个 EA 中数据的键值, 并通过判定 API 的类型确定数据的类型.

以图 1 中的源码为例, 需要分析该 EA 代码中所有的方法并且确定 getStringExtra 和 getIntExtra 的第一个参数的值, 即 Intent 中附加数据的键值. 实际上, 该问题可以被看作是一个到达定值问题, 它是数据流分析中最普遍和最有用的模型. 到达定值模型可以静态的确定那些预定义的变量在到达代码指定位置时的值. 本文中使用 Soot[7,8]提供的数据流分析模型 ForwardDataFlowAnalysis 实现了该算法.

2.4 测试驱动生成

该模块生成测试 EA 的测试驱动应用, 预先设计了一个测试驱动应用的模板, 只包含一个 Activity. 在入口函数 onCreate 中创建了一个 Intent 实例, 它的目标 Activity 是被测 EA, 这样就可以在测试驱动程序启动后可以自动的启动目标 EA. 之后我们随机生成之前

模块中提取到的键值所对应数据类型的值, 并且使用 putExtra APIs 将生成的数据添加到 Intent 实例中. 在 onCreate 的最后将 Intent 作为 startActivity API 的参数启动 EA.

除了这些代码, Manifest 文件也是应用执行的重要部分, 所以创建了一个 Manifest 文件并且在其中声明 Activity 列表. 最后, 将源码和 Manifest 文件一起打包成一个 APK 文件.

2.5 测试执行

测试驱动应用生成之后, 将它们部署到设备中并执行它们去启动 EA, 为了检测 EA 中存在的问题, 需要记录执行过程中设备中出现的信息. 使用 Adb 工具将上述过程自动化, Adb 是 Android 软件程序工具包 (Software Development Kit, SDK) 提供的一款用于连接 Android 设备的工具. Adb 可以批量的将测试驱动应用安装到设备中并执行这些应用, 测试驱动应用被 Adb 启动后会自动化的启动测试目标应用中的 EA, 并且保存设备的屏幕截图供之后的分析用.

3 试验结果与分析

为了评估方法的有效性, 本文根据上文提到技术实现了一个工具 EASTER (Exported Activity: Specific TestER), 使用若干真实应用在一台 Google Nexus 5 上进行了实验.

3.1 实验设计

从 Google Play 和其他的 Android 市场上下载了 10 个应用作为实验应用. 表 1 展示了这些应用的详细信息, 包括应用大小、Activity 的数量和 EA 数量. 从表中可以发现 EA 在现实应用中被广泛使用, 系统的测试它们是一个非常重要的研究课题. 本文使用了很流行的 Android 应用黑盒测试工具 Monkey[9]对处理过的应用进行测试, 用以评估 Monkey 对应用中 EA 的覆盖情况, 本次试验中对应用中 EA 的字节码进行了插桩处理. 实验中设置 Monkey 对每个应用进行测试时生成的事件数量为 10 000. 覆盖率如表 1 中最后一列所示, 实验表明应用中大约有 17% 的 EA 可以被 Monkey 生成的事件序列所覆盖.

3.2 EA 的测试覆盖率

实验中首先使用 EASTER 为应用中每个 EA 生成一个测试驱动应用, 然后让它们在设备中有序的运行. 试验中, 如果一个测试驱动应用可以成功的启动该 EA, 就认为该测试驱动应用可以覆盖该 EA. 表 2 展示详细的实验结果, 其中第三列给出了被覆盖的 EA 数量, 最后一列表明了覆盖率. 从表中可以发现, 生成的测试驱动应用可以覆盖实验应用中大多数的 EA(平均 73.7%), 覆盖率远远大于 Monkey 对 EA 的覆盖率. 有几个特殊的样本的覆盖率较低, 例如: Snapchat 和腾讯新闻. 这些应用中对一些 EA 设置了某些系统或者自定义的权限, 拒绝来自于未授权应用的调用请求. 这个问题将在之后的工作中解决.

表 1 实验应用

应用名称	大小 (MB)	Activity 数量	EA 数量	Monkey 覆盖
Artisto	24.8	25	1	1
百度地图	39.5	92	5	1
爱奇艺	21.4	286	13	2
Pinterest	17.2	33	5	0
Snapchat	78.8	13	3	1
SuperBright	5.7	33	1	1
腾讯视频	22.3	142	5	1
腾讯新闻	25.7	145	8	1
WiFi 万能钥匙	5.7	54	2	1
优酷	32.0	157	22	2

表 2 EASTER 的覆盖率

应用名称	EA 数量	覆盖数量	覆盖率
Artisto	1	1	100
百度地图	5	2	40
爱奇艺	13	12	92.3
Pinterest	5	4	80
Snapchat	3	1	33.3
SuperBright	1	1	100
腾讯视频	5	4	80
腾讯新闻	8	3	37.5
WiFi 万能钥匙	2	2	100
优酷	22	19	86.4

3.3 实验结果分析

实验中使用 EASTER 结合不同的 Intent 数据为应用中每个 EA 生成了 5 个测试驱动应用, 并深入研究了它们执行过程中出现的问题. 可以将实验过程中出现的问题分成两类: 应用奔溃和显示异常, 其中前者表示该 EA 在测试中会崩溃, 后者代表 EA 显示不正常, 例如显示没有友好用户提示的空白窗口. 表 3 展示了详细的实验结果, 其中第二和第三列是生成测试驱动应用的数量和报告出有漏洞 EA 的数量. 最后两列分别列出了在 EA 测试过程中出现应用崩溃和显示异常两类问题的测试驱动程序的数量.

表 3 EASTER 检测到漏洞数量

应用名称	测试驱动程序数量	有漏洞 EA 数量	应用奔溃	显示异常
Artiso	5	0	0	0
百度地图	25	2	0	10
爱奇艺	65	5	5	20
Pinterest	25	1	0	5
Snapchat	15	0	0	0
SuperBright	5	0	0	0
腾讯视频	25	2	0	10
腾讯新闻	40	2	5	5
WiFi 万能钥匙	10	1	0	5
优酷	110	7	15	20

3.4 案例分析

本小节中进一步的分析了存在漏洞应用的代码, 准确的定位漏洞的位置. 首先利用反编译工具 jd-gui[10]将有漏洞的应用的字节码转换成 Java 程序, 之后人工的检查检测报告中 EA 的代码. 下面分别对两个应用中的两种漏洞进行分析.

图 4 显示爱奇艺应用中一个类名为 FalconActivity 的 EA 的代码片段. 推测第三行中开发者忽视了对 “argument”键值所对应数据使用前的有效性校验. 在实验中发现, 随机生成一个字符串赋值给该变量, 应用总会出现崩溃.

```
if (localIntent.hasExtra("argument")) {
  loadUrl("javascript: var argument=\"" +
    localIntent.getStringExtra("argument") + "\"");
}
```

图 4 爱奇艺应用的代码片段

图 5 给出了优酷应用中类名为 GameDetailsActivity 的 EA 中 onCreate 方法的代码片段. 我们可以发现如果 Intent 不为空并且 appid 为空时 loaddata 方法不会被执行并且该 Activity 会显示了一个空白窗口, 这样会迷惑用户.

```
if (getIntent() == null) {
  layout_no_result.setVisibility(View.VISIBLE);
}
if (getIntent() !=null
  && getIntent().appid != null) {
  loaddata();
}
```

图 5 优酷的代码片段

4 相关工作

这部分将简要介绍了几种 Android 应用自动测试生成方法的相关工作.

Fuzzing 是一个被普遍使用, 对 Android 应用进行黑盒测试的方法. Machir 等人[4]开发了一款随机生成系统事件的工具——Dynodroid, 该工具生成的事件与用户事件相同.

基于模型的测试也是测试 GUI 程序典型的技术, 也可以用来测试 Android 应用. Wang 等人[5]将 GUI 行为构建成一个 FSM 模型, 他们利用静态分析技术提取与控件相关的事件, 使用动态探测技术捕获 Activity 的迁移. Azim 和 Neamtiu[11]也提出了一个测试生成工具——A3E, 它使用静态分析技术构建 GUI 模型, 基于深度优先搜索和目标搜索策略并结合 GUI 模型生成事件序列.

以上研究值关注为应用生成事件序列. Mirzaei 等人[4]将 Android 应用中的事件输入和数据输入区分开, 利用符号执行引擎 Symbolic PathFinder(SPF) 生成数据输入. Jensen 等人[12]使用符号执行技术生成事件和数据覆盖指定代码.

现有的工作主要是通过动态执行应用或者静态分析代码来提取和遍历程序行为. 这些方法有一个共同的缺点就是它们不能有效的覆盖应用的特殊部分——EA, 因为 EA 往往在应用内部不会被调用, 大部分 EA 是提供给外部应用进行访问的.

5 结语

本文介绍了一个自动化测试 Android 应用中 EA 的方法, 现有的工作中很少有相关的讨论. 实验结果表明 EA 被广泛的应用在现实的应用中, 但是很多时候开发者对它的开发不是非常完备. 由于 EA 通常包含开发者渴望推广的功能, 对 EA 进行系统化的测试在保证 Android 应用质量的研究中是非常值得关注的问题. 相信本文的工作是对现有工作的一个非常有用的补充, 之后的研究也是很有意义的.

本文只是对 Android 应用中 EA 进行了初步的研究, 并没有全面考虑 EA 和 Intent 机制的所有特性, 例如 Intent Filter 和 Permission 等. 之后的工作将解决这些问题以加强 EASTER 的性能. 除此之外, Intent 的数据也是通过自己开发的工具随机生成的, 之后的工作中也可以使用模糊测试或者混合测试技术加强 EASTER 的功能.

参考文献

1 IDC. Smartphone OS Market Share, 2017 Q1. IDC Research, Incorporated. http://www.idc.com/prodserv/smartphone-os-market-share.jsp. [2017-07-25].

2 The introduction of Google Play on wikipedia. Wikimedia Foundation, Incorporate. https://en.wikipedia.org/wiki/Google_Play. [2017-07-25].

3 Ren CG, Zhang YL, Xue H, *et al*. Towards discovering and understanding task hijacking in Android. USENIX Security 2015. 2015. 945–959.

4 Machiry A, Tahiliani R, Naik M. Dynodroid: An input generation system for Android apps. Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering. 2013. 224–234. [doi: 10.1145/2491411.2491450]

5 Yang W, Prasad MR, Xie T. A grey-box approach for automated GUI-model generation of mobile applications. In: Cortellessa V, Varró D, eds. Fundamental Approaches to Software Engineering. FASE 2013. Lecture Notes in Computer Science, vol 7793. Springer, Berlin, Heidelberg. 2013. 250-265. [doi: 10.1007/978-3-642-37057-1_19]

6 Apktool–a tool for reverse engineering Android apk files. Connor Tumbleson and Ryszard Wisniewski. http://ibotpeaches.github.io/Apktool/. [2016-04-01].

7 Soot–a framework for experimenting with comiler and software engineering technique for Java programs. Sable Research Group and Secure Software Engineering Group. https://sable.github.io/soot/. [2016-11-09].

8 Lam P, Bodden E, Lhoták O, *et al*. The Soot framework for Java program analysis: A retrospective. Cetus Users and Compiler Infrastructure Workshop. 2011.

9 Monkey–a popular black-box testing tool for Android apps. Google. https://developer.android.com/studio/test/monkey.html.

10 JD-GUI–a standalone graphical utility that displays Java sources from CLASS files. Free Software Foundation. http://jd.benow.ca/.

11 Azim T, Neamtiu I. Targeted and depth-first exploration for systematic testing of android apps. ACM Sigplan Notices, 2013, 48(10): 641–660. [doi: 10.1145/2544173]

12 Jensen S, Prasad MR, Møller A. Automated testing with targeted event sequence generation. Proceedings of the 2013 International Symposium on Software Testing and Analysis. 2013. 67–77. [doi: 10.1145/2483760.2483777]