Android 应用程序 GUI 遍历的自动化方法①

赵耀宗, 程绍银, 蒋 凡

(中国科学技术大学 信息安全测评中心, 合肥 230027)

摘 要: 近来针对 Android 应用程序的基于 GUI(图形用户界面)的分析和测试方法已经成为一个研究热点. 自动化技术和较高的 GUI 覆盖率可以提高大部分方法的效率和效果. 然而以前的工作并不能充分满足自动化和高 GUI 覆盖率的要求. 提出了一种在不需要程序源代码的情况下遍历 Android 应用程序 GUI 的自动化方法. 其主要思想是通过模拟用户的行为自动探测 Android 应用程序的 GUI. 我们的工作主要解决了 UI 元素提取和处理、用户行为模拟、GUI 遍历算法设计和模型构建三方面中的一些关键问题. 实验结果表明, 该方法能获得较高的 GUI 覆盖率可以有效遍历应用程序的 GUI. 此外, 该方法也将有助于程序安全分析、GUI 测试等其他研究.

关键词: Android; GUI 遍历; 自动化方法; GUI 覆盖率; 用户行为模拟

Automatic Method for GUI Traversal in Android Applications

ZHAO Yao-Zong, CHENG Shao-Yin, JIANG Fan

(Information Technology Security Evaluation Center, University of Science and Technology of China, Hefei 230027, China)

Abstract: GUI-based analysis and testing methods of Android applications have become a research focus in recent years. Automation and high GUI coverage could improve the efficiency and effectiveness of most methods. However, previous work is insufficient to meet the requirements of automation and high GUI coverage. In this paper, we propose an automatic method of traverseing the GUIs in Android applications whose source codes are not available. The main idea of the method is to simulate user actions to explore the GUIs of an app automatically. The work tackles some key issues mainly in three aspects: extraction and process of UI elements, simulation of user actions, algorithm design and model building of GUI traversal. The result of experiments shows that the method is effective to traverse the GUIs of an application with a high GUI coverage. Furthermore, this method could assist other researches, such as program security analysis, GUI testing.

Key words: Android; GUI traversal; automatic method; GUI coverage; simulate user actions

1 引言

近年来针对 Android 应用程序的分析和测试方法的研究在学术界和工业界发展迅速. 由于 Android 应用是事件驱动的程序, 且用户通过和应用程序的 GUI 交互来触发一系列的事件去运行它. 因此出现了一些基于 GUI 的方法用以解决 Android 应用的正确性、安全性、性能等问题, 而且其中大部分方法都建立在遍历应用程序 GUI 的基础之上, 例如通过探测应用的 GUI 发现程序缺陷的 GUI 测试方法[1-5]或者通过遍历应用的 GUI 检测程序隐私泄露、恶意功能以及敏感行为的动态安全分析方法[6,7]或者通过遍历应用程序的 GUI 以测试其设备和屏幕兼容性的云测试技术[8].

上述方法的结果很大程度上取决于遍历的 GUI 覆盖率以及自动化程度. 目前存在一些工具和方法可以用于遍历应用程序 GUI. Monkey 工具[9]可以通过向应用程序发送用户事件的伪随机流和一些系统级的事件

① 基金项目:高等学校博士学科点专项科研基金新教师类资助课题(20113402120026);安徽省自然科学基金(1208085QF112);安徽省高等学校优秀青年人才基金(2012SQRL001ZD);中央高校基本科研业务费专项资金(WK2101020004,WK0110000007)

收稿时间:2015-01-06;收到修改稿时间:2015-02-16

引起 GUI 的转换, 但是这些转换是随机的, 这使得 GUI 遍历极其低效. 用户可以使用 MonkeyRunner[10]、Robotium[11]、Uiautomator[12]等工具编写脚本去遍历应用程序的 GUI. 从某种意义上来说, 编写脚本的方式和用户直接操作应用遍历 GUI 是等价的, 而用户直接操作应用程序遍历 GUI 的覆盖率只有 30.8%[13]. 因此虽然这些手工的方式可以精确的遍历 GUI, 但是效率低且易出错. 其他一些已有的方法也存在一些限制: 文献[1,2,14]只能探测包含少量 Activity 组件和 GUI 的简单应用. 文献[5,15]则是先通过静态分析技术构建 Activity 转换图, 再基于该图动态遍历 GUI, 因此这些方法过程复杂且需要程序源码. 另外大部分方法都只考虑了一个 Activity 组件对应一个 GUI 这种情况, 以至于其遍历模型都基于 Activity 组件而不是真正的 GUI.

为了解决这些问题, 本文提出了一种遍历 Android 应用程序 GUI 的自动化方法, 首先提取当前 GUI 上的控件元素并根据这些控件元素生成相应的用户行为, 再通过模拟用户行为使 GUI 发生转换, 然后基于遍历模型和遍历算法重复上述过程从而实现应用程序 GUI 的遍历. 该方法可针对现实生活中流行的应用程序且不需要程序源码. 文中对该方法进行了详细的介绍, 实验表明, 该方法可以有效地自动遍历 Android 应用程序的 GUI, 具有较高的实用性, 能广泛应用于其他基于 GUI 的方法研究.

2　方法概述

在方法的描述和模块的设计中, 本文定义了如下新的概念:

① GUI 定义为显示在屏幕上的界面. 当屏幕界面变化超过一定阈值后则 GUI 发生变化.

② 动作 可以引起 GUI 发生变化的一个或多个相关用户事件. 动作分为两类: 只包含一个非输入事件如点击的简单动作; 包含一个或多个输入事件和一个简单动作的复合动作.

③ 状态 定义为二元组(Activity, GUI 截图). 因为一个 Activity 组件可以显示一个或多个 GUI, 所以用该二元组表示一个 GUI 的状态. 通过 Activity 类名比较和截图图片比对判断两个 GUI 的状态是否相同.

④ 任务 定义为二元组(动作, 状态). 表示一个在该状态下执行的动作. 一个任务被执行只有当应用到达该状态且该动作发生.

⑤ GUI 节点 代表 GUI 树上的一个节点, 存储一些关键信息如父节点、子节点、GUI 状态、从初始状态到当前状态所执行的任务列表.

⑥ 转换 定义为二元组(任务, GUI 节点). 表示执行一个任务后产生一个子 GUI 节点. 在每一个父 GUI 节点中都保存了一个代表其所有孩子节点的转换列表.

方法的总体架构如图 1 所示. 从图中可知, 自动遍历应用程序 GUI 的过程主要包括四个步骤:

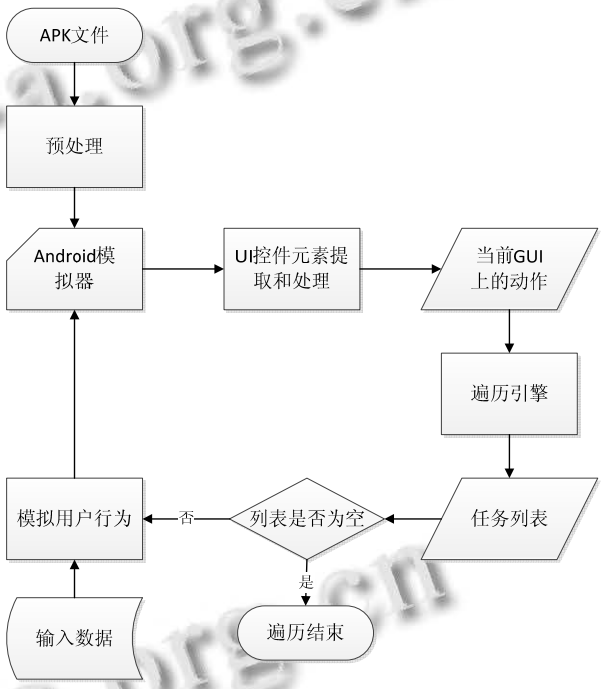

图 1　方法架构图

(1)预处理模块通过解析目标应用中包含的 AndroidManifest.xml 文件获得目标应用的 Activity 组件列表、进程名、主 Activity 名. 之后分别通过进程名和主 Activity 将目标应用安装到 Android 模拟器并启动目标应用.

(2)当目标应用运行到一个新的 GUI 时, UI 控件元素提取和处理模块提取当前 GUI 上的控件信息如控件的层次结构、控件属性, 并根据这些控件生成该 GUI 上的相应的动作.

(3)遍历引擎模块负责构造遍历模型、获取当前 GUI 状态、结合当前状态将步骤(2)中生成的动作转化成当前 GUI 上的任务并将其添加到全局任务列表.

(4)模拟用户行为模块执行具体的动作模拟包括简

单动作和复杂动作. 在模拟数据输入时根据启发式算法选择与上下文相关的数据进行输入. 重复步骤(2)(3)(4), 直到满足停止条件.

3 关键模块设计与实现

3.1 预处理模块

该模块首先使用开源工具 apktool 从目标应用中抽取 AndroidManifest.xml 文件, 之后解析该文件得到目标应用中定义的所有 Activity、应用的进程名、应用的主 Activity 名.

Android 系统规定应用中出现的每一个 Activity 必须在 AndroidManifest.xml 文件中进行定义, 格式如下:

```
<activity android:name=”string” …>
    <intent-filter>
        <action/>
    </intent-filter>
</activity>
```

解析 AndroidManifest.xml 文件时,识别<activity>标签并提取其中 android:name 属性的值. 该值是实现的一个 Activity 类名, 通常是一个全称限定名如 com.example.project.ExtracurricularActivity. 但有时是一个简写, 如果以‘.’开头如.ExtraccurricularActivity, 全限定名则是前接<manifest>标签中的 package 属性值. 该属性值也是应用的进程名. 主 Activity 定义包含一个子标签<intent-filter>,且该子标签包含<action android:name=”android.intent.action.MAIN”>子标签.

3.2 UI 控件元素提取和处理模块

在 Android 应用程序开发中视图是 UI 组件的基本构成要素. 一个视图占据屏幕中的一个矩形区域并且负责绘图和事件处理. 视图是控件的基类, 控件用来生成可交互的 UI 元素. 视图组是可以包含其他视图的特殊视图, 是用于 UI 布局和容器的基本类. 通过视图和视图组的组合可以形成 GUI 上的一棵控件树.

利用 Google 提供的开源工具 HierarchyViewer 可以获得 Activity 上的控件树. 但是由于复杂的界面设计和控件元素布局使得 Activity 上的有些控件并没有显示在当前屏幕范围内. 这增加了动作生成和模拟的难度. 而另一个开源工具 UiautomatorViewer 则能够只获得显示在当前屏幕范围内的控件树也即 GUI 上的控件树. 其中每一个节点对应屏幕上的一个控件, 而且在每个节点中保存着控件的一些关键属性, 例如文本信息、类名、是否可点击、控件的边界. 在本文中我们对开源工具 UiautomatorViewer 进行修改, 利用其提供的接口获得当前屏幕上控件的树形结构.

当探测到一个新的 GUI, 首先获得当前 GUI 上的控件树, 之后遍历该控件树生成相应的动作. 该过程包括两个步骤:

(1)生成复合动作: 通过分析一些应用程序发现一个 GUI 上的所有输入控件之间在逻辑上都是相关联的, 用户依次向所有输入控件输入数据后再点击一个按钮, 因此在一个 GUI 上最多存在一个复杂动作. 该模块遍历控件树根据节点中保存的控件 calss 属性提取出所有的输入控件. 每当遇到一个输入控件时, 获得与该输入控件相关的上下文信息即输入控件所接受的数据类型和含义. 例如“用户名”、“密码”、“手机号”等文本信息. 通常情况下这些信息存储在该输入控件附近的文本控件中, 通过搜索输入控件的子节点、兄弟节点、叔节点寻找与其相关文本控件提取该信息. 得到当前 GUI 上的所有输入控件后, 还应获得与这些输入控件相关的一个点击控件. 本文根据表示输入完毕的关键字信息如“提交”、“确定”、“注册”等在控件树中搜索相关的点击控件. 最后根据输入控件和点击控件生成对应的复合动作.

(2)生成简单动作: 目前本文中的方法只生成点击, 滑动等简单动作. 通过再次遍历控件树, 根据控件的 class 属性和是否可点击属性提取可点击的控件, 并根据这些控件生成对应的简单动作.

总之, 通过遍历控件树生成当前 GUI 上可模拟的动作, 包括至多一个复杂动作和一系列简单动作.

3.3 用户动作模拟模块

该模块负责模拟用户的动作, 主要包括触摸动作和按键动作两类. 触摸动作模拟屏幕上点击、拖动、输入等动作, 这些动作需要依据动作所针对的控件在屏幕上的坐标来进行模拟. 该坐标信息可以通过控件中的边界属性计算出, 该属性包含控件区域在屏幕上的右上角坐标、控件的长度和高度信息. 另外, 在本文方法中我们提出了针对输入动作的启发式输入策略, 即参考输入控件的上下文信息向输入控件输入有效数据而不是随机数据. 按键动作主要包括 BACK、HOME、音量键等手机按键动作.

3.4 遍历引擎模块

遍历引擎模块是本文方法的核心. 算法 1 详细描

述了 GUI 遍历算法. 算法的输入为一个启动的目标应用. 1-2 行初始化任务列表和 GUI 树. 3-5 行生成初始 GUI 上的任务并生成 GUI 树的根节点. 6-7 行从任务列表中取一个任务执行直到任务列表为空或满足其他终止条件. 当执行一个任务时 8-10 行负责先将应用恢复到任务所在的 GUI. 11-15 行模拟用户动作并判断 GUI 是否发生变化. 如果 GUI 发生改变, 16-19 行更新当前 GUI 状态并根据当前状态生成任务和子节点. 20-22 行增加子节点到父节点并更新当前节点.

Algorithm 1. The GUI Traversal Algorithm

Input: The launched target APK

```
1:  TaskList ← Φ;
2:  GUIRoot ← NULL;
3:  curGUIState, curGUIActions, curGUITasks ←
    obtain the state, actions, tasks from the initial GUI
    of the app;
4:  add the curGUITasks into TaskList;
5:  GUIRoot ← GUINode ← generate the GUI Node
    according to the current GUI;
6:  while TaskList is notΦ|| meet the stop condition do
7:      task ← get a task from the TaskList;
8:      if curGUIState is not task.state then
9:          find the state required by the task;
10:     end if
11:     simulate the task.action;
12:     newGUIState ← obtain the GUI state after
        simulating the action;
13:     if curGUIState is equal to newGUIState then
14:         break while;
15:     end if
16:     curGUIState ← newGUISate;
17:     curGUIActions, curGUITasks ← obtain the
        actions, tasks from the current GUI;
18:     add the curGUITasks into the TaskList;
19:     childGUINode ← generate the GUI Node by
        the current GUI;
20:     transition ← generate the transition by task
        and childGUINode;
21:     add transition and task into childrenList and
        executionPath of curGUINode respectively;
22:     curGUINode ← childGUINode;
23: end while
```

遍历引擎模块解决了遍历过程中遍历方式、GUI 状态表示及遍历模型构建、GUI 状态恢复等关键问题.

通常当一个应用启动后, 用户点击初始 GUI 上的按钮将会导致当前 GUI 切换到新的 GUI, 用户重复相同的动作使 GUI 不断切换到下一个新的 GUI 或者用户点击返回键使 GUI 切换到前一个 GUI. 用户访问应用程序 GUI 的过程类似于以深度优先方式遍历树, 而且该访问过程会生成一颗 GUI 树, 其中应用程序的 GUI 对应树的节点, 用户动作对应树的边. 本文方法采用深度优先的方式探测应用程序的 GUI 树, 相比其他探测方式如广度优先该方式有如下优点: 更加符合用户操作应用程序的行为; 能有效记录程序的执行路径, 便于程序的动态分析; 提高探测的效率, 尤其是广度优先方式遍历效率较低.

Android 中 Activity 组件用于显示应用程序的界面. 大部分情况下, 一个 Activity 组件显示一个界面. 因此先前的方法大多基于一个 Activity 组件代表一个 GUI 的认识来构建遍历 GUI 的模型, 如图 2 所示. 然而随着应用程序界面布局越来越复杂, 一个 Activity 组件往往会显示多个不同的 GUI. 若仍以 Activity 为基础建立 GUI 遍历模型则会增加动作生成和模拟的难度和降低遍历的效果. 因此本文的遍历模型建立在真正的 GUI 基础之上, 如图 3 所示.

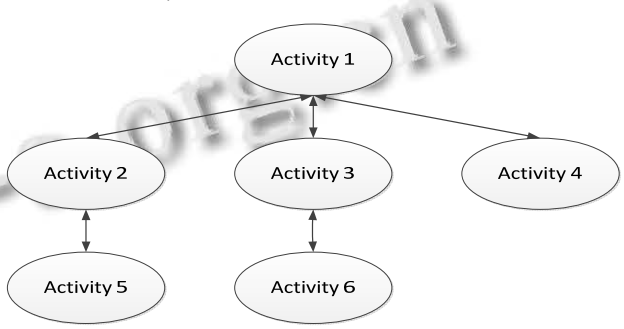

图 2 基于 Activity 的遍历树

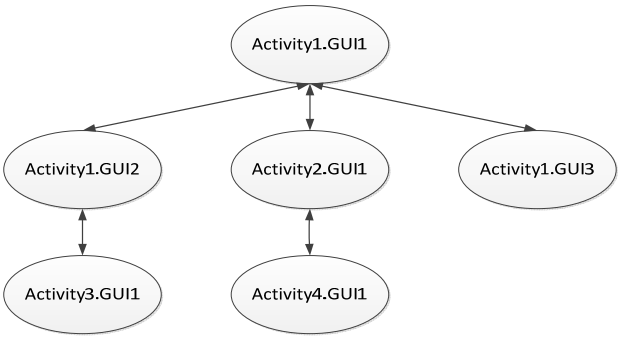

图 3 基于 GUI 的遍历树

另一个问题是如何表示 GUI 的状态. 先前的方法中, 研究人员使用 Activity 组件中包含的所有控件集及其属性来表示一个 GUI 的状态, 该方法效率较低且不准确. 本文中采用了一个巧妙的方法即使用 Activity 和 GUI 的屏幕截图来表示一个 GUI 的状态.

基于上文定义的概念实现模型的构建. 当应用程序运行到一个新的 GUI, 遍历引擎生成当前 GUI 的状态并构建一个 GUI 节点. 之后结合当前 GUI 状态将 UI 元素提取和处理模块生成的动作生成相应的任务并将这些任务加入全局任务列表. 然后再根据深度优先的方式选择一个任务执行, 任务执行后将会导致应用程序切换到一个新的 GUI, 并根据新的 GUI 生成一个子 GUI 节点. 最后构建一个转换将父节点和子节点联系在一起. 重复上述过程即可构建遍历模型.

在遍历过程中, 当执行一个任务时必须确保应用程序处于待模拟动作所需的 GUI 状态. 本文提出了回退和重放相结合的方法解决该问题. 定义执行任务所需的 GUI 状态为目标状态. 方法描述如下: 当执行一个任务时首先判断当前状态是否为目标状态, 若是则直接执行任务, 若不是则模拟 BACK 按键使应用回退到上一个 GUI 并判断该状态是否为目标状态. 重复上述过程, 直至回退到初始状态. 当回退到初始状态后, 则根据与目标状态相对应的 GUI 节点中保存的任务轨迹列表从初试状态依次模拟这些任务直到目标状态.

4 方法验证

4.1 工具实现

根据上述方法实现了一个原型工具用来验证本文方法的可行性和有效性. 该工具用 java 语言实现运行在 PC 平台, 目标 Android 应用程序运行在模拟器. UI 元素提取通过修改开源工具 UiautomatorViewer 使用其提供的 API 接口实现. 用户动作模拟模块主要借助 MonkeyRunner 工具和 Monkey 工具实现. Monkey 工具是运行在模拟器或者设备上的命令行工具, 可以通过执行指定的命令发送伪随机用户事件流或者系统事件到系统, 模拟用户或系统事件. 当 Monkey 通过“monkey --port”命令启动时, 其可从所监听的端口接受指定的用户事件. 通过 adb 技术实现模拟器和主机之间的 Socket 通信, 可以从主机发送命令到模拟器从而精确的控制 Monkey 模拟用户的行为. MonkeyRunner 工具提供了模拟各种动作的函数接口, 其底层则会向模拟器发送“monkey --port”命令启动 Monkey 并向指定的端口发送相应的命令来控制 Monkey 模拟具体的用户事件. 该原型工具可以自动安装并启动目标应用, 模拟用户动作自动地遍历应用程序的 GUI, 直至满足终止条件停止.

4.2 评估标准

直观感觉, 访问到的 GUI 越多意味着更好的遍历结果. 但是一个应用包含的所有 GUI 没办法提前统计, 因此不能通过 GUI 给出方法的定量评估标准. 上文提及, 一个 Activity 组件可以显示一个或多个 GUI, 因此尽管大部分情况下 Activity 的数量和 GUI 的数量不是完全相等的, 但是本文依然可以借助 Activity 组件的数目近似的对方法的有效性进行定量的评估.

本文定义 Activity 覆盖率对实验结果进行评估, 该覆盖率为执行遍历过程中探测到的 Activity 数与应用程序包含的所有 Activity 总数的比率. 显然 Activity 覆盖率越高探测到的 Activity 越多, 同时也意味着能探测到更多的 GUI. 借助 UiautomatorViewer 工具可以在遍历过程中记录探测到的 Activity. 应用程序的 Activity 总数在预处理模块从 AndroidManifest.xml 文件中统计得到. 在两部分中均不包括来自第三方库或其他应用的 Activity 如广告相关的库、浏览器, 因为这些 Activity 并不是目标应用基本代码的一部分. 尤其是当从 AndroidManifest.xml 文件中得到所有 Activity 后仍需手动的除去来自第三方库的 Activity.

4.3 实验

为了验证方法的可行性和有效性, 使用原型工具对三个流行的 Android 应用程序进行实验. 该工具运行在操作系统为 Ubuntu10.10、内存为 2G 的 PC 平台. 目标应用运行在搭载 Android4.3 操作系统、内存为 1G 的 Android 模拟器上. 实验结果如表 1. 第一列为应用程序名, 第二列为应用程序的大小, 第三列为应用程序中 Activity 总数(AT),第四、五列分别为遍历执行过程中探测到的 Activity 数目(AE)和 GUI 的数目(GE), 第六列为 Activity 覆盖率(AC).

表 1 实验结果

App	Size(KB)	AT	AE	GE	AC(%)
iReader	3,552	31	19	23	61.29
BaiduMusic	5,868	47	28	34	59.57
eLong	5,610	142	65	65	45.77

从实验结果可知, 本文方法具有可行性能自动遍历 Android 应用程序 GUI, 而且能获得较高的 Activity 覆盖率. 本方法还具有很好的适用性可以遍历流行且较为复杂的 Android 应用程序的 GUI.

4.4 结果分析

三个应用的 Activity 覆盖率分别为 61.29%、59.57%、45.77%, 均高于以人工方式遍历应用程序的 Activity 覆盖率 30.8%. 从表中分析可知覆盖率的高低与应用程序包含的 Activity 组件总数有很大关系. 应用程序包含的 Activity 组件越多意味着应用的功能越多、设计越复杂, 因此会增加遍历的难度从而导致相对较低的 Activity 覆盖率. 另外, 遍历过程中探测到的 GUI 数量略大于探测到的 Activity 组件数量, 这也符合本文方法所设计的遍历模型.

通过对应用程序和遍历过程进行分析, 限制覆盖率进一步提高的原因主要包括以下两个方面: 针对类似 Webview 控件、地图控件等控件的点击动作会使这些控件显示的内容发生变化, 从而导致屏幕显示的界面发生变化. 但是这从本质看并不是 GUI 发生了变化, 处理这样的控件会使基于 GUI 的遍历陷入死循环. 还有对列表控件、音频播放控件等也不易进行精确的模拟, 从而影响 GUI 的遍历结果; 应用程序的身份验证对遍历过程也会产生影响, 如在处理登录用户和游客身份两种情况时会导致不同的界面切换路径, 这必然会使得一部分界面无法访问到; 另外本文也没有对付费动作进行模拟, 目前很多应用都有类似这种付费后额外享受其他服务的功能, 因此不执行该付费操作也就不能访问到一些特殊界面.

5 结语

本文提出了一种遍历 Android 应用程序 GUI 的自动化方法, 该方法基于 Activity 组件和 GUI 建立遍历模型, 提取 GUI 上控件元素生成相应的用户动作, 并按照遍历算法自动的模拟用户动作遍历一个 Android 应用程序的 GUI. 根据该方法实现了一个原型工具, 并对三个流行的 Android 应用进行试验, 验证了本文方法的可行性和有效性. 目前, 基于本文方法已经完成了自动检测 Android 应用程序 XSS 漏洞的工作. 未来工作计划主要有两方面: 一方面继续优化本文方法, 针对一些特殊和复杂的控件制定相应的模拟行为而不仅仅是模拟点击、滑动等简单动作, 对不同类别的应用程序如视频类、浏览器类提供不同的模拟策略和遍历算法; 另一方面, 我们将继续基于本文方法在 GUI 测试、动态安全分析等方面展开深入的工作, 如基于本文方法自动运行一个应用程序, 对应用程序进行监控, 记录应用程序运行的信息分析应用程序的行为.

参考文献

1 Hu C, Neamtiu I. Automating GUI testing for Android applications. Proc. of the 6th International Workshop on Automation of Software Test. ACM. 2011. 77–83.

2 Amalfitano D, Fasolino AR, Tramontana P. A gui crawling -based technique for android mobile application testing. 4th IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). IEEE. 2011. 252–261.

3 Yeh CC, Huang SK, Chang SY. A black-box based android GUI testing system. Proc. of the 11th Annual International Conference on Mobile Systems, Applications, and Services. ACM. 2013. 529–530.

4 Yang S, Yan D, Rountev A. Testing for poor responsiveness in Android applications. 2013 1st International Workshop on the Engineering of Mobile-Enabled Systems (MOBS). IEEE, 2013: 1–6.

5 Jensen CS, Prasad MR, Moller A. Automated testing with targeted event sequence generation. Proc. of the 2013 International Symposium on Software Testing and Analysis. ACM. 2013. 67–77.

6 Zheng C, Zhu S, Dai S, Gu G, Gong X, Han X, Zou W. Smartdroid: an automatic system for revealing ui-based trigger conditions in android applications. Proc. of the 2nd ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. ACM. 2012. 93–104.

7 Rastogi V, Chen Y, Enck W. Appsplayground: automatic security analysis of smartphone applications. Proc. of the third ACM Conference on Data and Application Security and Privacy. ACM. 2013. 209–220.

8 Mobile Test Center. August 2012. http:// mtc.baidu.com/.

9 Android Developers. UI/Application Exerciser Monkey. August.http://developer.android.com/tools/help/monkey.html.

10 Android Developers. MonkeyRunner. June 2013. http://developer.android.com/tools/help/monkeyrunner_concepts.html.

11 Google Code. Robotium. June 2013. https://code.google.com/p/robotium.

12 Android Developers. Uiautomator. June 2012. http://developer.android.com/tools/help/uiautomator/.

13 Azim T, Neamtiu I. Targeted and depth-first exploration for systematic testing of android apps. Proc. of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications. ACM, 2013: 641–660.

14 Wang P, Liang B, You W, Li J, Shi W. Automatic Android GUI traversal with high coverage. 4th International Conference on Communication Systems and Network Technologies (CSNT). IEEE. 2014. 1161–1166.

15 Yang W, Prasad MR, Xie T. A grey-box approach for automated GUI-model generation of mobile applications. Fundamental Approaches to Software Engineering. Springer Berlin Heidelberg. 2013. 250–265.