一种基于组合事件行为触发的Android恶意行为检测方法

张国印[1] 曲家兴[1,2] 付小晶[1] 何志昌[1]
(哈尔滨工程大学计算机科学与技术学院 哈尔滨 150001)[1]
(黑龙江省国防科学技术研究院 哈尔滨 150001)[2]

摘 要 当前Android恶意应用程序在传播环节缺乏有效的识别手段,对此提出了一种基于自动化测试技术和动态分析技术的Android恶意行为检测方法。通过自动化测试技术触发Android应用程序的行为,同时构建虚拟的沙箱监控这些行为。设计了一种组合事件行为触发模型——DroidRunner,提高了Android应用程序的代码覆盖率、恶意行为的触发率以及Android恶意应用的检测率。经过实际部署测试,该方法对未知恶意应用具有较高的检测率,能帮助用户发现和分析未知恶意应用。

关键词 Android,恶意行为检测,动态分析,组合事件,自动触发
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.5.018

Android Malicious Behavior Detection Method Based on Composite-event Trigged Behaviors

ZHANG Guo-yin[1] QU Jia-xing[1,2] FU Xiao-jing[1] HE Zhi-chang[1]
(College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China)[1]
(Heilongjiang Province National Defense Science and Technology Institute, Harbin 150001, China)[2]

Abstract For lack of effective means to identify Android malware application in transmission link at current time, this paper proposed an Android malicious behavior detection method based on automated testing techniques and dynamic analysis techniques. Android applications behavior is triggered by automated testing technology and monitored by a virtual sandbox. This paper presented a model of triggering malicious behavior named DroidRunner using combined operations on malware, which improves the Android application code coverage and the trigger rate of the malicious behavior. It benefits to improving the detection rate of malicious Android applications. After the actual deployment and testing, this method has a high detection rate to the unknown malicious applications. It can help users to find and analyze the unknown malicious applications.

Keywords Android, Malicious behavior detection, Dynamic analysis, Composite event, Automatic trigger

1 引言

近年来,移动互联网技术及移动通信技术进行着飞速的发展与革新,人们生活中的日常消费、社交、娱乐、购物、获取信息的途径也逐步向智能手机迁移。其中,Android平台凭借开源特性和众多厂商的支持成为智能手机领域的第一大平台,Android平台的安全形势也愈加严峻。根据DCCI互联网数据中心公布的最新数据显示,2013年新增移动恶意软件69万余个,相比2012年同比增长了400%,这些恶意软件主要的传播途径是第三方电子市场及网页下载。2014年,Android恶意代码的主要行为依然是恶意扣费,大量广告应用被植入恶意代码,恶意代码利用各种技术手段躲避手机安全软件的查杀。

目前Android平台的恶意行为检测方法主要分为静态分析和动态分析两种[1,2]。静态分析则试图通过浏览程序代码来理解程序的行为,静态分析需要事先收集和更新恶意行为特征,不能检测未知恶意代码。动态分析是指在严格控制的环境中(虚拟机或真机)执行恶意软件,并使用检测工具记录并分析其所有行为,可以有效检出未知恶意代码。近年来,动态分析方面的研究受到研究者的极大关注。Blasing等[3]提出了一种名为AASandbox的Android应用恶意行为动态检测系统,该方法使用Monkey工具[4]模拟用户的操作行为,获取系统调用日志信息以进行分析,该行为触发的方式过于盲目,不能很好地触发Android应用恶意行为的发生。路程等[5]提出将自动化测试系统与Android系统的行为监控技术相结合,文中指出使用Monkeyrunner工具[6]模拟用户的操作来测试应用软件,终端监控程序主要负责监控网络连接和短信发送的情况。该方法使用迭代的算法遍历控件,对于有页面切换的复杂界面,不能很好地遍历所有的控件。杨欢等[7]提出了一种综合考虑Android多类行为特征的3层混合

到稿日期:2015-04-13 返修日期:2015-08-03 本文受黑龙江省国防科学技术研究院支撑项目:移动APP恶意行为识别与逆向分析技术研究(20150309)资助。

张国印(1962—),男,教授,博士生导师,主要研究方向为网络与信息安全、嵌入式系统,E-mail:zhangguoyin@hrbeu.edu.cn(通信作者);曲家兴(1979—),男,博士生,高级工程师,主要研究方向为网络与信息安全,E-mail:smilingqu@126.com;付小晶(1980—),女,博士,讲师,主要研究方向为网络与信息安全,E-mail:fuxiaojing@hrbeu.edu.cn;何志昌(1988—),男,硕士,主要研究方向为网络与信息安全,E-mail:358560623@qq.com。

算法用于检测 Android 未知恶意应用。在动态分析中的行为监控环节使用 strace 命令提取应用运行期间的行为数据，在行为触发环节使用 Monkey 工具模拟用户的操作。Spreitzenbarth 等[8]提出了一个沙箱系统，使用静态分析技术和动态分析技术相结合的方法，利用 Monkeyrunner 和 HierarchyViewer工具[9]，使用相应的行为触发机制来触发待检测的恶意样本。Karami 等[10]提出了一种 Adnroid 恶意代码检测机制，使用自动仪表系统与被测软件 UI 进行交互，检测系统调用并创建行为档案，但是没有给出该机制在代码检测覆盖率和准确性方面的实验和分析。

目前，在大多数 Android 恶意代码动态分析方法中，都引入了 Android 应用自动化测试技术以提升恶意行为检测方法的效率，使用 Monkey 或 Monkeyrunner 方式实现对 APP 的自动控制，触发其可能存在的恶意行为。现有方法至少存在以下缺点之一：(1)由于 Monkey 工具的实现方式是随机的，利用 Monkey 工具模拟人对应用的操作无法遍历执行整个应用中的所有路径，无法有效地触发应用的恶意行为性；(2)执行过程中对应用程序恶意行为的触发效果不明显，未充分考虑多种组合操作可提升触发恶意行为的概率，例如某些特定的权限组合可能会产生恶意行为，并且权限组合的复杂度越高，其安全隐患越大；(3)利用 Monkey 工具对应用的操作未充分考虑界面控件测试的均衡性，可能导致部分界面内控件平均执行次数高，其它界面操作较少，进而导致部分界面覆盖率不足。

基于上述存在的问题，本文建立了恶意代码检测系统，提出了组合操作触发行为模型，提高了 Android 应用的控件覆盖率和函数覆盖率，能尽可能地触发 Android 应用的恶意行为；同时采用多层行为监控的方法更好地分析当前应用是否存在恶意的行为。

2 检测系统架构

Android 恶意行为检测系统框架如图 1 所示。该框架由 4 个部分组成：预处理模块、行为触发模块、行为监控模块和行为分析模块。

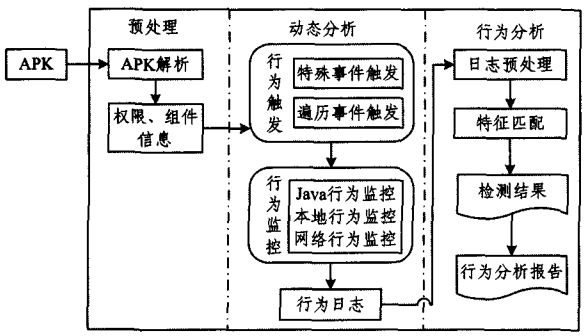

图 1 恶意行为检测系统框架

具体介绍如下：

(1)预处理模块负责对待检测 APK 进行反编译，提取 Main Activity、包名称、申请权限、组件等信息，为后续模块提供基础数据。

(2)行为触发模块负责模拟人对应用的操作，根据程序运行时环境特征，模拟人对应用内控件操作和模拟程序运行期间的外部干预，尽可能覆盖程序内部的路径，触发应用的恶意行为。

(3)行为监控模块从 Java 行为、本地行为、网络行为 3 个层面监控应用运行时的行为。Java 行为监控子模块负责提取应用程序运行期间调用 Java 核心库和应用框架层的 API 行为，并提取部分敏感函数调用的参数信息。本地行为监控子模块负责监控本地调用方法名、本地调用 Java 资源的行为及部分敏感本地调用行为。网络行为监控子模块负责提取应用与外部网络交互的信息，监控的信息主要有 IP 地址、连接的端口、消息的内容等。

(4)行为分析模块负责对行为监控日志进行预处理，提取行为特征，通过与恶意行为特征库和可疑行为特征库进行匹配分析，得出检测结果和行为分析报告。

恶意行为检测系统框架中，行为触发模块负责实现对应用的自动触发控制，其核心目标是快速高效地触发 Android 应用程序的恶意行为。Android 应用的恶意行为一般只在特定条件发生时才会被触发执行，以此隐藏其恶意功能、躲避杀毒软件的查杀和增强病毒分析人员对其分析的难度。通过对大量恶意样本的分析，恶意行为的触发方式大致可分为用户行为触发、广播接收器触发和检测运行环境触发 3 类。针对上述恶意行为触发方式，本文提出了一种组合行为触发模型——DroidRunner，通过组合操作控件触发恶意事件。该模型由多组合均衡遍历算法和特殊事件触发库两部分构成。

3 组合事件行为触发模型

3.1 模型描述

组合事件行为触发模型 DroidRunner 包括 5 个组件：主控、界面分析、界面控制、特殊事件触发和特殊事件触发库。其总体架构如图 2 所示。界面控制和特殊事件触发是模型的核心部分，用于在待测试应用运行期间模拟用户操作及运行期间可能发生的特殊事件，提升在应用运行期间触发恶意应用中恶意行为的概率。

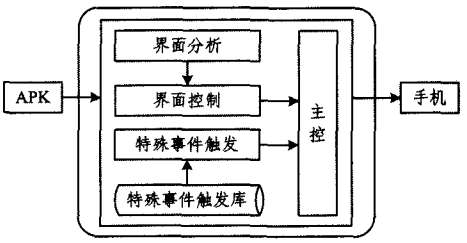

图 2 DroidRunner 模型结构

(1)主控组件

主控组件的主要功能是通过 ADB 与设备之间进行通讯与控制，并在界面分析组件、界面控制组件、特殊事件触发组件的支撑下，实现对应用的安装、启动、运行、卸载等自动化操作。

应用安装阶段：安装 APK，如果成功，则进入下一步；如果失败，说明安装失败的原因，将其获取失败原因，写入运行日志，结束当前检测流程。

应用启动阶段：以预处理阶段获取的 APK 的包名称和 Main Activity 名为参数启动 APP 主界面。如果成功，则进入下一步；如果失败，获取失败原因，将其写入运行日志，结束当前检测流程。

应用自动运行阶段：协调界面控制组件和特殊事件触发组件实现模拟用户对 APP 的操作。为方便分析不同操作对应用的影响，界面控制组件产生的用户基本操作与特殊事件触发机制产生的事件是交互发生的，并在每次操作被执行之

前截取当前界面图像。每个 APP 运行时间预定义为 10 分钟，如果在 10 分钟之内程序出现异常崩溃或卡死等异常事件，则结束运行当前 APP，并重新启动运行，同时保留当前测试的数据和运行的时间。如果在 10 分钟内出现第二次异常崩溃或卡死等异常事件，则自动结束应用自动运行阶段，进入应用卸载阶段。

应用卸载阶段：以预处理阶段获取的 APK 包名为参数卸载应用程序，如果成功，则进入下一步；如果失败，则说明卸载失败的原因，将其写入运行日志，结束检测流程。

(2)界面分析组件

界面分析组件的主要功能是获取当前 Activity 中显示的所有可操作控件的绝对坐标、控件类型、控件可操作指令等属性，生成控件的操作序列，为界面控制组件提供基础数据。

(3)界面控制组件

界面控制组件的主要功能是模拟用户对应用的操作，这些操作根据多组合均衡遍历算法调度生成。目前，界面控制组件支持 4 种指令，如表 1 所列。

表 1　界面控制组件指令

指令名称	参数个数	参数描述	示例
点击指令 (click)	2	点击控件的中心坐标位置	click 430 340
输入指令 (input)	1	输入的文本值	input weixin
按键指令 (keyevent)	1	该按键事件的编号	keyevent 82 (菜单键)
滑动指令 (drag)	4	滑动的最初位置和结束位置	drag(100,20) (200,200)

(4)特殊事件触发组件

特殊事件触发组件的主要功能是根据预处理组件获取的待检测应用的权限及组件信息从特殊事件触发库中抽取符合条件的特殊事件列表，之后在主控组件的控制下在 APP 运行期间随机触发这些特殊事件。

(5)特殊事件触发库

特殊事件触发库存储已知恶意应用中触发恶意行为的事件，在动态检测期间触发这些行为可提升触发检测应用恶意行为的概率。

3.2　多组合均衡遍历算法

多组合均衡遍历算法是界面控制模块的核心，为了在有限时间内提升对应用逻辑代码的覆盖率，更好地达到提升触发恶意行为的效果，需要对界面内的控件进行不同的组合操作，同时需要均衡地点击不同界面内的控件。为便于分析应用程序内部的界面间的调度关系，本文将 Android 应用界面间跳转关系转化为一个有向赋权图，其逻辑表示如式(1)所示：

$$\begin{cases} G=(L,W,E) \\ L=\{l_i \mid i=1,2,\cdots,n\} \\ W=\{w(l) \mid \forall l \in L\} \\ E=\{e_{ij}=\langle l_i,l_j\rangle \mid i,j=1,2,3,\cdots,n\} \end{cases} \tag{1}$$

其中，G 表示界面跳转关系的有向图；L 表示应用所有界面的集合，集合中每个节点表示一个界面；W 表示界面权值的集合，权值表示需要调度到该界面执行操作的需求程度，该值越大说明调度到该界面进行操作的需求越迫切；E 表示界面间跳转的边的集合，存储的是导致界面跳转的操作。

界面节点的权值确定了界面间调度优先级，如何确定界面节点的权值直接影响了多组合均衡遍历算法的效率。本文利用界面节点中操作控件的执行情况确定界面节点的权值。每一个界面节点均维持了 3 个列表：unKnowList、notJumpList 和 jumpList。unKnowList 列表用于存储不确定是否会导致界面跳转的控件操作，在进入一个新的界面时，所有的控件操作均存储在这个列表中。notJumpList 列表用于存储不会导致界面跳转的操作，初始值为空。jumpList 列表用于存储会发生界面跳转的操作，初始值为空。3 个列表中的所有的操作都具有 4 个属性：count、index、command 和 class。count 初始值为 0，表示该操作被执行的次数，index 表示操作标识，command 表示具体的操作命令，class 表示控件的类型。每个界面都有一个权重，在计算界面权值时不是以整个列表计算，而是以当前识别出的控件情况来计算，因为不同的操作可能导致界面内的控件发生变化。界面节点的权值计算公式如式(2)所示：

$$L_i.weight= \begin{cases} 0, & len(L_i.unKnowList)=0, \\ & len(L_i.notJumpList)=0 \\ n, & n=len(L_i.unKnowList)>0 \\ \dfrac{n}{\sum x_k}, & n=len(L_i.notJumpList)>0, \\ & len(L_i \cdot unKnowList)=0 \end{cases} \tag{2}$$

其中，x_k 表示 $L_i.notJumpList$ 列表中第 k 个操作被执行的次数。界面的权重大于 1 说明这个界面还有多个操作没有被调度执行，权值越大，调度到这个界面节点执行的优先级越高；当界面的权重小于 1，说明该界面中所有的操作都被执行过；界面的权值等于 1，说明界面存在一个操作没有被执行或者这个界面中所有的操作都被执行了 1 次。

多组合均衡遍历算法的具体描述如下：

(1)如果 $L_i \notin L$，即发现一个新的界面节点 L_i，将界面分析组件获取的当前界面的所有操作置于 $L_i.unKnowList$ 表中，则根据式(2)计算界面权值 $L_i.weight$，转到步骤(2)；如果 $L_i \in L$，则转到步骤(2)。

(2)如果 $len(L_i.unKnowList)\neq 0$，在 $L_i.unKnowList$ 列表中随机选择执行一个操作 x_k，$x_k.count$ 自加 1，根据式(2)计算界面权值 $L_i.weight$。如果界面没有发生变化，则将操作 x_k 从 $L_i.unKnowList$ 列表移动到 $L_i.notJumpList$ 列表中，跳到步骤(2)；如果界面发生跳转，新的界面为 L_j，则将操作 x_k 从 $L_i.unKnowList$ 列表移动到 $L_i.jumpList$ 列表中，并建立界面 L_i 过操作 x_k 跳转到界面 L_j 的指向关系，跳到步骤(1)；如果 $len(L_i.unKnowList)=0$，则跳到步骤(3)。

(3)如果 $len(L_i.notJumpList)\geqslant 3$，根据执行次数少优先和不同类型优先随机在 $L_i.notJumpList$ 中选择 3 个操作组合 $Commands$ 并执行；如果 $len(L_i.notJumpList)<3$，执行所有操作。被执行操作 $count$ 属性自加 1，根据式(2)计算界面权值 $L_i.wight$，完成后跳转到步骤(4)；如果 $len(L_i.notJumpList)=0$，则跳转到步骤(4)。

(4)$L_k=\max(W(L))$，如果 $i=k$，跳到步骤(3)；如果 $i\neq k$，跳到步骤(5)。

(5)根据广度优先算法在图中搜索当前界面 L_i 到界面 L_k 的路径 S，如果 S 存在，按照跳转操作执行，操作的 $count$ 属性自加 1，跳转到步骤(2)；如果 S 不存在，执行返回操作，跳到步骤(5)。

3.3 特殊事件触发库

界面控制模块模拟人对应用程序进行操作，但是，这种算法无法模拟接收到短信、锁屏等特殊事件。特殊事件触发技术通过在应用程序运行期间触发接收到短信、锁屏等这类特殊事件，进一步提升对应用程序内部处理逻辑的覆盖率，达到触发应用程序恶意行为的目的。对于由广播接收器和检测运行环境等触发应用程序恶意行为的事件，本文采用了命令行实现方式、脚本实现方式和预装实现方式实现了这些特殊事件的自动触发。Android 系统通过权限机制实现应用程序对系统资源的访问控制，需要通过申请相应的权限实现特定的操作或访问系统的资源。所以，根据应用程序申请的权限信息可分析出应用程序对哪些特殊事件敏感。因此，通过建立特殊事件与权限的对应关系，可以快速根据应用申请的权限提取对应特殊事件，提高检测的精度，节约动态检测的时间。

基于分层的设计思想和自动化测试中的数据驱动思想，在特殊事件-权限映射表中将脚本与数据分离。特殊事件-权限映射表的具体描述如表 2 所列。在特殊事件-权限映射表中，特殊事件与权限之间是 1 对 1 或 1 对多的关系，即某一个特殊事件发生可能需要多种权限的组合。

表 2 特殊事件-权限映射表

字段名	字段类型	字段描述
ID	Varchar(32)	表项编号
Event_ID	Varchar(32)	事件编号
Event_des	Varchar(64)	事件触发的行为描述
Permission_ID	Varchar(32)	权限编号
Permission_des	Varchar(64)	事件触发的行为描述
way	Bool	事件的实现方式，0 表示命令行实现，1 表示脚本实现
command	Varchar(256)	当 way 值为 0，存储的是命令行；当 way 值为 1，存储的是脚本路径
data	Varchar(256)	脚本数据文件存储路径

特殊事件的提取流程分为以下两步：

(1)提取应用申请的权限信息，根据提取的权限从特殊事件-权限映射表中抽取对应的特殊事件；

(2)根据提取的特殊事件在特殊事件-权限映射表中查找需要的权限，如果需要的权限均被应用申请，则表明这个特殊事件满足触发条件，需要在应用运行时执行这个特殊事件；若不满足，则舍弃这个特殊事件。

4 实验结果与分析

实验设计了 3 组测试。第 1 组测试中有 5 个自编辑的恶意应用程序，这 5 个自编辑的恶意样本的相关信息如表 3 所列，分别使用 Monkeyrunner 和 DroidRunner 对这些测试样本进行触发并统计应用程序执行期间的控件覆盖率、控件的点击次数差额和函数覆盖率，测试的统计结果如表 4 所列。目前 Android SDK 里自带的测试工具有 Monkey 和 Monkeyrunner 两个。总的来说，Monkey 主要应用在压力和可靠性测试上，运行该命令可以随机地向目标程序发送各种模拟键盘事件流，并且可以自己定义发送的次数，以此观察被测应用程序的稳定性和可靠性，应用起来也比较简单。而相比之下 Monkeyrunner 更强大一些，它主要应用于功能测试、回归测试，并且可以自定义测试扩展，灵活性较强，测试人员也可以完全控制。因此本文使用 Monkeyrunner 与 DroidRunner 作比较。

表 3 自编测试样本描述

测试样本	样本恶意行为描述	恶意行为触发方式
样本 A	后台发送短信	界面内操作触发
样本 B	读取用户位置、通讯录及短信内容，上传到指定服务器	由锁屏、网络状态连接等广播触发
样本 C	向以 106 开头的号码发短信，并拦截 106 开头的短信	监听短信接收器
样本 D	动态载入 DEX 文件的方式加载恶意代码，执行窃取用户通讯录、短消息	由锁屏、网络状态连接等广播触发
样本 E	获取当前运行进程列表，终断其中的杀毒软件进程，之后后台发送短信	检测运行环境中是否存在杀毒软件进程

表 4 恶意行为触发性能比较

统计值 \ 触发方式	Monkeyrunner[5-9]	DroidRunner（本文方法）
控件的点击次数最大差额	9	3
界面平均点击次数最大差额	5	1
控件覆盖率	72.3%	89.2%
平均函数覆盖率	69.1%	85.3%

测试结果显示 Droidrunner 相比 Monkeyrunner 在控件点击次数最大差额和界面平均点击次数最大差额两个数据上均有大幅度下降，反映出 DroidRunner 的多组合均衡遍历算法起到了很好的调节作用。在函数覆盖率、敏感 API 覆盖率和恶意行为触发率 3 个统计项中，DroidRunner 比 Monkeyrunner 均有一定的提升。

第 2 组测试有 20 个病毒样本，分别使用 Monkey、Monkeyrunner 和 DroidRunner 3 种方式实现对样本的自动控制，并统计函数覆盖率和恶意行为触发率。图 3 展示了 3 种触发方式在函数覆盖率、恶意行为触发率参数的统计值。

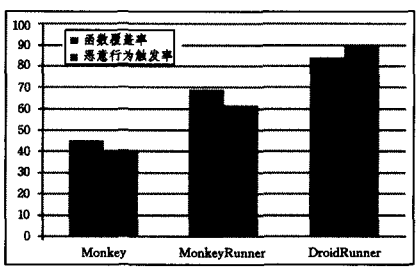

图 3 多种行为触发方式效果对比

由图 3 可知，DroidRunner 比其它两种触发方式在函数覆盖率和恶意行为触发率统计项上均有大幅提升。DroidRunner 与 Monkey 触发方式相比，在函数覆盖率上提升了42.7%，在恶意行为触发率上提升了近一倍。DroidRunner 与 Monkeyrunner 触发方式相比，在函数覆盖率上提升了21.7%，在恶意行为触发率上提升了 43.5%。这些数据显示 DroidRunner 能更加高效地触发应用的敏感行为，反映了它具有高效提升触发应用程序恶意行为的概率的能力。文献[3,5,7-9]主要从行为监控和分析的角度提升了恶意代码检测方法的性能，在行为触发模块中采用 Monkey 工具产生用户事件流，与本文提出的 DroidRunner 相比，恶意行为触发概率较低，从而降低了恶意代码检测效果。

第 3 组测试的目的是验证本文提出的方法对恶意行为的检测效果。通过网络论坛、QQ 群和博客等途径收集到了可用的恶意应用样本 35 个，同时自主开发了 5 个恶意样本，共计 40 个恶意应用样本。设置每个样本的测试时间为 15 分钟，检测结果如表 5 所列。自编辑样本的恶意行为检测率高

(下转第 139 页)

[17] Cai T T, Wang Lie. Orthogonal matching pursuit for sparse signal recovery with noise [J]. IEEE Trans. on Information Theory, 2011, 57(7): 4680-4688

[18] Liu En-tao, Temlyakov V N. The orthogonal super greedy algorithm and applications in compressed sensing [J]. IEEE Trans. on Information Theory, 2012, 58(4): 2040-2047

[19] Do T T, Gan Lu, Nguyen N, et al. Sparsity adaptive matching pursuit algorithm for practical compressed sensing[C]// Proc. of the Fortieth-Second Asilomar Conference on Signals System and Computers. 2008: 581-587

[20] Yang Yi, Liu Bo-wen, Li Yang. DWT-SVD Domain Video Watermarking Algorithm Based on Motion Analysis[J]. Journal of Chongqing University of Technology(Natural Science), 2015, 29(10): 132-138(in Chinese)
阳溢, 刘博文, 李扬. 基于运动分析的 DWT-SVD 域视频水印算法[J]. 重庆理工大学学报(自然科学), 2015, 29(10): 132-138

(上接第 99 页)

于网络样本的恶意行为检出率，由于自编辑恶意样本的恶意行为的触发方式比较单一，且对恶意行为的隐藏能力较差，因此本文方法能够较好地触发恶意行为，具有较好的检测效果。网络搜集的样本的检测率相对较低，这是因为网络搜集的样本的恶意行为触发方式更加多样，增加了恶意代码检测的难度。

表 5 DroidRunner 恶意行为检测率

检测效果 / 样本	识别数量	漏报数量	检测率(%)
网络样本	35	4	88.6
自编辑样本	5	0	100
总计	40	4	89

同时，对 40 个恶意样本使用 6 折交叉验证方法，通过使用 DroidRunner 和不使用 DroidRunner 来检测 Android 恶意程序，实验结果如下。

表 6 为使用 MonkeyRunner 和 DroidRunner 的 6 次交叉验证的实验结果对比，使用 MonkeyRunner 计算 6 次实验检测率的平均值为 84.1%，6 次实验误报率的平均值为16.2%，6 次实验漏报率的平均值为 15.6%。而使用Droid-Runner计算 6 次实验检测率的平均值为 89.6%，6 次实验误报率的平均值为 11.4%，6 次实验漏报率的平均值为9.5%。由实验结果可以看出，使用 DroidRunner 在检测率、误报率和漏报率方面都有较好的表现，平均检测率提高了5.5%，平均误报率下降了 4.8%，平均漏报率下降了 6.1%。

表 6 恶意行为检测结果比较

检测方式 / 检测结果	Monkeyrunner[5-9]	DroidRunner (本文方法)
平均检测率(%)	84.1	89.6
平均误报率(%)	16.2	11.4
平均漏报率(%)	15.6	9.5

通过对以上测试的统计数据分析，本文方法与传统的行为触发方法相比，能够更好地覆盖应用程序运行期间绝大部分的函数调用，对应用程序中的敏感行为触发效果显著，能够较好地检测 Android 应用的恶意行为。

结束语 本文深入研究了 Andorid 恶意代码动态检测过程中的行为触发方法，设计了 Android 恶意行为检测框架，提出了组合事件行为触发模型——DroidRunner，提出了多组合均衡遍历算法，利用组合操作控件等组合事件触发恶意行为，提高了触发恶意行为的概率。在行为监控环节提出了多层行为监控技术，实现了对 Java 代码和本地代码的函数层调用监控，进而实现了对应用程序恶意行为的高效触发。实验证明，本文提出的恶意行为检测方法具有较高的准确性和覆盖率。

参 考 文 献

[1] Hu Wen-jun, Zhao Shuang, Tao Jing, et al. A Detection Method and System Implementation for Android Malware[J]. Journal of Xi'an Jiaotong University, 2013, 47(10): 37-43(in Chinese)
胡文君, 赵双, 陶敬, 等. 一种针对 Android 平台恶意代码的检测方法及系统实现[J]. 西安交通大学学报, 2013, 47(10): 37-43

[2] Cai Zhi-biao, Peng Xin-guang. Detection of Android malware based on system calls[J]. Computer Engineering and Design, 2013, 34(11): 3757-3761(in Chinese)
蔡志标, 彭新光. 基于系统调用的 Android 恶意软件检测[J]. 计算机工程与设计, 2013, 34(11): 3757-3761

[3] Blasing T, Batyuk L. An android application sandbox system for suspicious software detection[C]// Proceedings of the 5th International Conference on Malicious and Unwanted Software. 2010: 55-62

[4] Hao Peng, Sarma C G B. Using probabilistic generative models for ranking risks of Android apps[C]// Proceedings of the 2012 ACM Conference on Computer and Communications Security. 2012: 241-252

[5] Lu Cheng, Yang Yi-xian. Design and Implementation of Malwares Detection System on Android[D]. Beijing: Beijing University of Posts and Telecommunications, 2012(in Chinese)
路程, 杨义先. Android 平台恶意软件检测系统的设计与实现[D]. 北京: 北京邮电大学, 2012

[6] Android developers. Monkeyrunner[OL]. http://developer.android.com/tools/help/ Monkeyrunner_concepts.html

[7] Yang Huan, Zhang Yu-qing, Hu Yu-pu, et al. A Malware Behavior Detection System of Android Applications Based on Multi-Class Features[J]. Chinese Journal of Computers, 2014, 37(1): 15-27(in Chinese)
杨欢, 张玉清, 胡予濮, 等. 基于多类特征的 Android 应用恶意行为检测系统[J]. 计算机学报, 2014, 37(1): 15-27

[8] Spreitzenbarth M, Freiling F, Echtler F, et al. Mobile-sandbox: having a deeper look into android applications[C]// Proceedings of the 28th Annual ACM Symposium on Applied Computing. ACM, 2013: 1808-1815

[9] HierarchyView[OL]. http://developer.android.com/tools/help/hierarchy-viewer.html

[10] Karami M, Elsabagh M, Najafiborazjani P, et al. Behavioral Analysis of Android Applications Using Automated Instrumentation[C]// Proceedings of the International Conference on Software Security and Reliability Companion. 2013: 182-187