一种基于传感器的 Android 应用行为分析技术

杨 频，冉 涛，张 磊，刘 易
(四川大学网络空间安全学院，成都 610064)

摘 要：大多数针对恶意软件识别的研究都是基于应用程序接口(Application Program Interface，API)调用来实现的，但是目前基于 API 的研究大都没有考虑到设备的状态，设备状态能够直接体现程序运行的外部环境，这对分析应用的行为有着重要作用. 本文提出一种基于传感器的应用行为识别技术，首先，通过传感器数据来判断设备实时状态；然后，结合 API 调用时序和图形用户界面(Graphic User Interface，GUI)首屏时序产生的多元时序数据，设计算法识别应用行为的恶意性；最后，设计实现包括静态打桩、动态行为监控和传感器实时状态采集的恶意行为分析原型系统，选取典型案例验证了本文提出方法的准确性，并通过黑盒测试验证了本文恶意应用识别方法的有效性.

关键词：传感器；应用行为；API 调用；行为分析
中图分类号：TP391.1 **文献标识码**：A DOI：10.19907/j.0490-6756.2021.013002

An analysis technology of Android application behavior based on sensors

YANG Pin, *RAN Tao*, *ZHANG Lei*, *LIU Yi*
(College of Cybersecurity, Sichuan University, Chengdu 610064, China)

Abstract: Most of the research on malware identification is based on the application program interface (API) call, but most of the current API based research does not consider the state of the device. However, the device state can directly reflect the running environment of the program, such as human operation or program automation, and it plays an important role in the analysis of application behavior. In this paper, a sensor based application behavior recognition technology is proposed. Firstly, the real-time status of the device is judged by the sensor data. Secondly, the algorithm is designed to identify the malicious application behavior using the multiple time series data generated by combining the API call time series and the first screen time series of graphical user interface (GUI). Finally, the malicious behavior analysis prototype system is designed and implemented, and it includes the functions of static piling, dynamic behavior monitoring and real-time status collection of sensors. Typical cases were selected to verify the accuracy of the proposed method, and the black box test was performed to verify the effectiveness of the malicious application identification method in this paper.

Keywords: Sensor; Application behavior; API call; Behavior analysis

收稿日期：2020-06-10
基金项目：国家重点研发计划(2017YFB0802900)
作者简介：冉涛(1989)，男，重庆万州人，硕士研究生，研究方向为 Android 恶意代码检测.
通讯作者：张磊. E-mail：zhanglei2018@scu.edu.cn

1　引　言

Android 平台凭借其开放性和生态圈的丰富多样性,吸引了大量的开发者,也包括恶意攻击者.由于现有的安卓系统没有统一的入口,很多恶意软件能够轻易地绕过官方市场,进入普通用户的设备.因此,针对恶意软件的检测一直是安全领域的一个重要研究.

大多数恶意软件主要通过调用系统的特定 API 来获取用户信息.因此,目前识别恶意软件的方法主要以 API 调用分析为主,刘凯等人[1]以 API 调用为基础,提出一种基于图卷积网络的恶意代码聚类检测方法.赵翠镕等人[2]通过提取真机的 API 序列进行恶意代码检测从而避免反虚拟机检测.Zhang 等人[3]则是通过抽象 API 调用产生的行为语义来实现恶意代码检测.Jung 等人[4]则是通过构建良性与恶性 API 列表来提高检测准确度.还有其他检测方法:例如,Utku 等人[5]实现了一个基于权限的多层感知系统用于检测恶意软件.刘晓建等[6]、葛文麒等人[7]通过对应用数据流进行分析,提取特征值来识别应用行为.AppIntent[8]提出一种基于数据转移来判断应用程序是否可疑的方法.目前已有研究表明应用行为模式分析[9-10]比数据流分析更能揭示应用存在的恶意性.目前的研究都没有考虑到设备状态这一技术点,但是设备的实时状态在反映应用行为恶意性这一点有更加明显迅速的优势.

为了弥补大多数基于 API 的行为分析研究与设备外部状态割裂造成的研究不准确,本文提出一种基于传感器的应用行为分析方法,该方法通过生成多元时序:设备外部状态时序、应用 API 调用时序和当前系统 GUI 首屏时序,基于多元时序数据提出一种恶意行为识别算法来识别应用的恶意性.本文主要贡献如下:(1) 设计并实现一种推算设备实时状态的算法,能利用传感器数据计算出当前设备的外部状态;(2) 设计了一种 Android 平台下高效准确的打桩方法,能通过该方法获取 API 调用的时序数据;(3) 提出一种分析应用行为的三元时序模型,以及基于该模型数据的恶意行为识别算法,并通过典型案例验证了模型的准确性.

2　基于传感器的应用行为识别技术

相对于常规行为分析的静态分析模块用来对需要分析的 APK 进行打桩外,本文还有采集器模块用来采集传感器数据用于计算设备外部状态、采集 GUI 首屏序列用于得出当前前台运行的程序,以及分析模块用于根据所得数据分析该应用是否恶意行为应用,得到如图 1 的技术流程图.

静态分析模块实现静态分析和打桩,通过逆向 APK 得出 smali 文件,并对敏感的 API 调用点标记输出应用调用该 API 的记录,再重新打包 APK.采集器模块通过传感器采集数据,并实时分析数据生成手机状态时序得到手机的实时状态.GUI 首屏采集通过实时采集系统首屏 APP 生成交互时序,用来查看当前的前台应用是否目标应用.恶意行为分析器,基于前面得到的数据生成多元时间序列模型,用于识别应用的恶意行为.

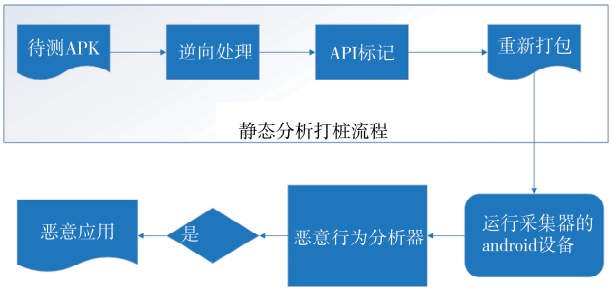

图 1　基于传感器的应用识别技术流程

Fig. 1　Application recognition technology flow based on sensor

2.1　传感器数据采集与分析

为了能够计算出设备运行时的平放、翻转、手持、静止等外部状态,本文设计了一种计算设备实时状态的算法,该算法通过对加速度传感器的数据进行分析,最后,计算出当前设备处于的状态.例如当俯仰角角度为 $-180°$ 左右,此时如果设备移动加速度小于 1,通过上述算法就能推断出此时设备处于反扣静止状态.所以需要对传感器中的加速度传感器数据进行采集.

Android 系统定义的传感器框架使用 3 轴坐标系来表示数据值.对于大多数传感器,当设备处于默认屏幕方向时,会相对于设备屏幕来定义坐标系(如图 2).当设备处于默认屏幕方向时,X 轴为水平方向右延伸,Y 轴为垂直方向向上延伸,Z 轴为垂直于屏幕向外延伸.

由于人体受脉搏跳动影响,在人处于静止不动状态时,捕获到的移动加速度传感器数值基本都是大于 1.通过移动加速度的大小可以判断此时设备是处于静止的平面,还是在用户身上.移动加速度公式如下式.

$$\text{speed}=\frac{\sqrt{dx^2+dy^2+dz^2}}{t}*10000 \quad (1)$$

其中，speed 是我们要求的移动加速度，dx、dy、dz 分别为加速度沿着图 2 坐标系的分量在采集时间 t（单位：ms）间隔内的增量.

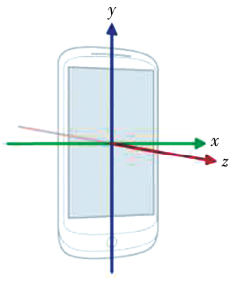

图 2　Android 传感器坐标系
Fig. 2　Android sensor coordinate system

设备状态计算算法思路：首先根据加速度在各方向的绝对值大小确定设备当前主要操作，然后，根据对应增量来判断具体情况. 例如，当 $|z|>|y|\&\&|z|>|x|$ 时，再根据 z 此时的值是否大于 0 来判断设备是正面向上还是正面向下.

2.2　应用静态分析及打桩

目前的静态打桩主要是针对需要分析的应用进行逆向分析，然后得到二进制代码插入可执行代码段输出自己需要的内容. 由于 Android 应用还包含了大量的资源文件以及一些配置文件，所以在打桩之后会有一个打包过程，打包的同时会对包进行签名. Android 对于 APK 的签名有着严格的验证，当发现当前签名包与官方签名值不一致的时候，就会进行异常签名处理. 文献[11]提出利用 Android 的“Master Key”漏洞将修改过后的 Dex 文件植入原 Apk 文件. 鉴于本文的目的仅仅是为了分析应用的行为，所以这里直接采用第三方签名，虽然不同于官方签名，但是能保证 APK 正常运行，也能达到分析应用行为的目的.

图 3 展示了针对 APK 静态分析以及打桩的具体流程，通过逆向工具对需要追踪的 APK 进行解包，反编译生成 Smali 文件，分析 Smali 文件在需要进行追踪的位置进行污点日志输出，然后重新打包并签名生成可执行 APK.

为了得到 APK 对应的二进制运行文件，需要用分析工具将 APK 解码，分析工具一般使用 Apktool，它可以将 APK 解码成资源文件形式，也能将解码后的资源文件重新打包成 APK. 打桩器主要用于向 Smali 文件中添加 Smali 语法的 Log 方法，用于输出对应 API 调用的日志记录.

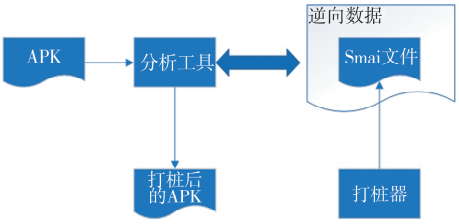

图 3　静态分析与打桩流程图
Fig. 3　Static analysis and plot flow chart

为更加高效地追踪 API，本文提出一种结合 AndroidManifest 配置文件的高效打桩方法. 该方法通过提取配置文件中的 Uses-Permission 权限列表，然后通过分析权限列表针对性的打桩. 相对于根据敏感 API 普遍撒网式的打桩，它根据实际使用权限对应的 API 进行打桩，更加高效明确.

打桩语法如下所示.

1) const-stringv0, "\u8c03\u7528\u6444"

2) invoke-static{v1,v0}, Landroid/util/Log;->i(Ljava/lang/String;Ljava/lang/String;)I

第一行定义了要输出的日志内容. 引号内的 Unicode 内容是日志里面输出的内容，可以根据要记录的内容自定义，放在寄存器 v_0 里面. 第二行执行 android 系统方法 Log 方法，用于打印日志信息. v_1 寄存器是 tag 用于查看日志时快速区分，一般是使用应用的包名. 当需要记录某个 API 调用的时候只需要在调用指令处，添加以上内容就能在调用该 API 的时候输出对应的日志.

2.3　GUI 首屏时序获取

应用是否处在与用户交互的状态，对于判断其行为是否具有恶意性是一个很重要的指标. GUI 首屏数据能很好地反映应用是否处于与用户交互状态. 为了方便后期分析应用行为，本文在采集 GUI 首屏数据的时候，加上了时间戳形成了时序序列. GUI 首屏也就是应用栈栈顶 APP，更细分就是栈顶 APP 的对应的 Activity. 对于移动应用来说，除了在跟用户直接交互之外，其他时候都判定为后台时刻. 本文把首屏时序作为识别恶意行为的一个非常重要的参考因子，因为绝大多数恶意行为都是在后台悄悄进行的.

Android5.0 以前可以通过 ActivityManager 获取位于栈顶的应用包名，但是 5.0 以后 Google 对此做了限制，只允许获取桌面包名和自身应用的包名. 因此本文通过 UsageStatsManager 来获取栈顶应用包名. UsageStatsManager 是使用情况记录

类,通过它可以获取应用的使用情况,应用在前台的时间、次数等.

2.4　恶意行为分析器

恶意行为分析器主要通过分析传入的首屏记录、API 调用记录和设备状态记录来确定应用行为的恶意性. 每条记录在输出的时候都会标记当前系统时间戳,这就使得不相关的数据能通过时间线合并成一个多元时间序列,作为最终分析的数据来源.

如图 4 所示,本来互不相关的设备外部状态、API 调用记录和 GUI 首屏通过时间相互之间联系在一起有了交集 E.

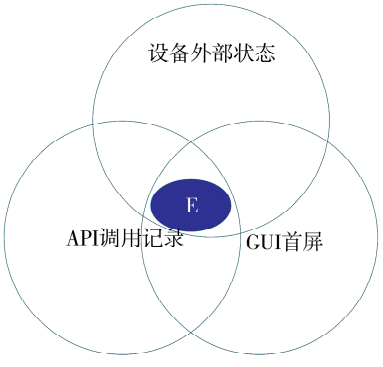

图 4　多元时序原理

Fig. 4　Principle of multivariate time series

2.4.1　应用行为恶意性分类　根据彭国军等人[12]的研究表明,android 恶意软件特征主要体现在其恶意负载上,主要有特权提升、远程控制、话费吸取、隐私窃取和自我保护这 5 类,具体如下.

1) 特权提升:通过利用 android 漏洞,突破 android 的权限机制和沙盒机制,允许恶意软件进行特权操作;

2) 远程控制:远程控制信息回传,更新本地恶意功能等;

3) 话费吸取:通过手机的短信功能,偷偷利用短信功能定制服务并从中牟利;

4) 隐私窃取:获取用户的各种隐私信息,例如通讯录,定位,通话录音,背景录音,拍照,通话记录等,这些信息多用于进行进一步攻击的前奏;

5) 自我保护:Android 恶意软件开始采用商业级代码保护技术,造成自动化静态分析失败,人工反汇编反编译困难,启发式检测、已知特征检测等难度加大.

2.4.2　恶意行为识别　为了能够更加准确地识别应用行为的恶意性,在推断之前需要进行时序数据的过滤. 根据时序数据我们可以分为 W_a:API 调用时序,W_g: GUI 时序,W_p:设备状态时序. 最终需要得到 W_e:应用事件时序. W_e应该是 3 个序列的交集:

$$W_e = W_a \cap W_g \cap W_p \tag{2}$$

首先,我们根据对应应用的包名在 W_a里面获取应用的 API 调用序列,过滤掉噪声数据;然后,根据每次 API 调用记录的时间,在 W_g和 W_p里面过滤掉其他的噪声数据;最后,就得到了 W_e用于识别应用行为的恶意性.

在进行应用行为识别的时候一般有以下两个着手点,一种是根据需要监控的 API 着手,然后通过 W_e得到的数据并结合设备的行为推断当前应用行为的恶意性. 一种是根据设备行为,先找出最能辨别出应用高恶意性的时段(这里是静止状态),然后根据这个时间段在 W_e里面查找应用是否做出了高敏感的 API 调用. 两种方法各有优点,可以根据实际需求在分析的过程中灵活选择.

3　实验设计与结果分析

3.1　实验设计

本文实验环境:红米 note5a,Android7.0,采集应用授予传感器权限,不限制后台用电,未 root. 不限制后台用电是防止采集应用被系统回收,不能正常采集数据.

```
2020-05-18 10:57:11.235 21331-21331/xyz.loadnl.guilog D/GUI: xyz.loadnl.guilog
2020-05-18 10:57:11.250 21331-21331/xyz.loadnl.guilog E/origen: screen up
2020-05-18 10:57:11.250 21331-21331/xyz.loadnl.guilog E/msg: speed= 0.25251055623469576
2020-05-18 10:57:11.565 21331-21331/xyz.loadnl.guilog E/origen: screen up
2020-05-18 10:57:11.566 21331-21331/xyz.loadnl.guilog E/msg: speed= 0.11415857618636555
2020-05-18 10:57:11.881 21331-21331/xyz.loadnl.guilog E/origen: screen up
2020-05-18 10:57:11.881 21331-21331/xyz.loadnl.guilog E/msg: speed= 0.2241301019209372
2020-05-18 10:57:12.117 21331-21331/xyz.loadnl.guilog E/origen: screen up
2020-05-18 10:57:12.118 21331-21331/xyz.loadnl.guilog E/msg: speed= 0.34032956907867373
2020-05-18 10:57:12.246 21331-21331/xyz.loadnl.guilog D/GUI: xyz.loadnl.guilog
```

(a)完全静止桌面

```
2020-05-18 10:58:11.827 21331-21331/xyz.loadnl.guilog D/GUI: xyz.loadnl.guilog
2020-05-18 10:58:11.944 21331-21331/xyz.loadnl.guilog E/origen: screen up
2020-05-18 10:58:11.944 21331-21331/xyz.loadnl.guilog E/msg: speed= 5.805367109528465
2020-05-18 10:58:12.180 21331-21331/xyz.loadnl.guilog E/origen: screen up
2020-05-18 10:58:12.181 21331-21331/xyz.loadnl.guilog E/msg: speed= 1.3359279806966402
2020-05-18 10:58:12.417 21331-21331/xyz.loadnl.guilog E/origen: screen up
2020-05-18 10:58:12.417 21331-21331/xyz.loadnl.guilog E/msg: speed= 2.6511545201257936
2020-05-18 10:58:12.654 21331-21331/xyz.loadnl.guilog E/origen: screen up
2020-05-18 10:58:12.654 21331-21331/xyz.loadnl.guilog E/msg: speed= 1.9959975828644532
2020-05-18 10:58:12.836 21331-21331/xyz.loadnl.guilog D/GUI: xyz.loadnl.guilog
```

(b)手持设备静止状态

图 5　手持静止与静止桌面加速度对比

Fig. 5　Comparison of acceleration between static and static desktop

为了评估更加的准确,本文首先验证了人手持设备靠桌静止下的设备加速度和设备放在静止的桌面上的设备加速度如图 5,由于脉搏跳动的影

响，人始终都在振动，所以手持设备如图 5(b)就算人完全保持静止状态，设备得到的加速度还是要大于设备处于完全静止的桌面图 5(a). 然后通过典型案例验证了模型对恶意应用识别的准确性，以及一个黑盒测试验证模型的识别能力.

3.2　实验结果

为验证本文模型识别的准确性，本文从 Drebin[13]恶意样本库提取出了 Установщик，在静态打桩阶段分析其主要权限请求里面包括读取联系人，发短信等敏感 API 操作. 从而判断其恶意行为可能是利用短信进行的扣费行为，所以在验证本文提出方法的可行性时，主要是针对短信调用 API 打桩，然后在经过一段时间的日志抓取之后，从中得到图 6 的行为记录，图中各顶点与交点代表右边对应的事件，其中设备明显处于闲置状态，手机屏幕向下翻转，首屏 APP 是小米自带的桌面管理器. 然后，能明显看到 Установщик 发送了一条短信. 以上证明本文系统能够准确地识别应用的恶意行为.

接着本文以聚会相片一款伪装成相片管理的恶意应用作为行为分析对象，通过本文提出的基于权限高效打桩方法，对聚会相片 APP 进行打桩并运行采集行为，发现这款应用安装之后桌面并不显示其图标，在其运行过程中采集到其行为如图 7，发现该应用在非人为操作情况下会向固定手机号码和邮箱发送信息对应图最高顶点和次高顶点，根据该异常行为，最后，调试发现该应用在手机接收短信之后拦截短信，并发送到其指定邮箱和手机. 以上实验证明本文提出的方法能有效地识别恶意行为，方便安全人员调试.

最后，为验证模型能识别未知恶意行为，本文采用黑盒测试的方法来验证识别未知行为的能力，在不知道样本具体恶意行为的前提下，对样本进行行为分析，然后得到图 8 所示行为图，根据行为图我们得到以下分析：传感器数据表明此时设备处于正面向下静止状态，17 点 27 分 4 秒应用调用了摄像头，如图 8 中次高顶点显示，然后几秒之后应用产生了一次网络请求，图 8 最高顶点显示，而且这些操作都是在后台完成，图中首屏显示为小米桌面管理. 根据该行分析为有理由认为该样本具有恶意性，因为在无人操作的情况下，应用在后台自动拍照，并进行了网络请求，其行为包含恶意行为中的第 2 条远程控制和第 4 条隐私窃取.

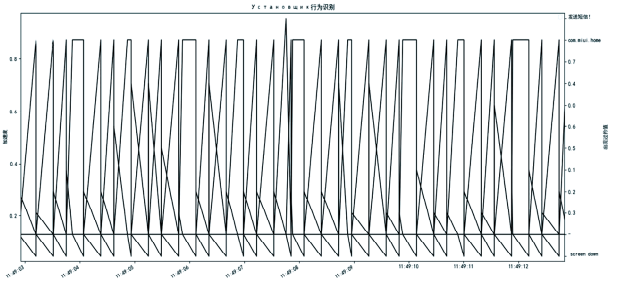

图 6　Установщик 行为分析图
Fig. 6　Behavior analysis of Установщик

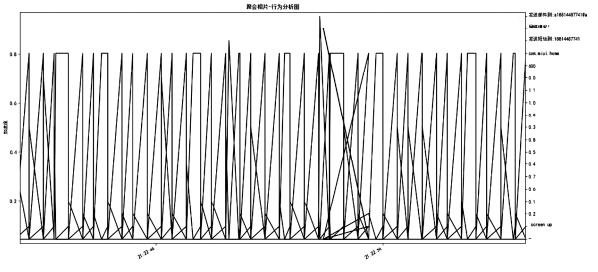

图 7　聚会相片行为分析图
Fig. 7　Juhuixiangpian behavior analysis chart

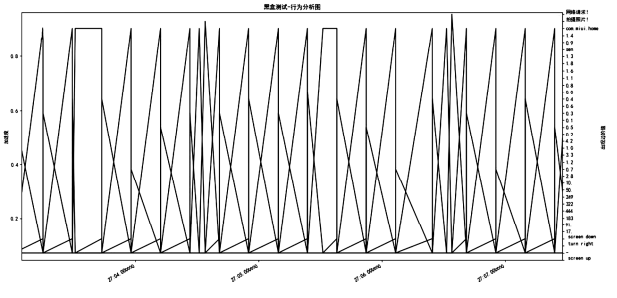

图 8　黑盒测试行为分析图
Fig. 8　Black box test behavior analysis chart

3.3　实验对比

为了更直观地对比本方法的功能，本文选用了 Mobile-Security-Framework(MobSF)[14]一个使用 python 实现的恶意样本分析平台，选择其主要原因是 MobSF 是基于经典开源分析平台 Androguard[15]实现，并在其基础上增加了新版本系统的支持，以及一些优化. 选择的测试样本是从 Drebin 恶意样本库中选取的一款短信类恶意软件 Установщик. 通过 MobSF 分析该样本得到的结果图 9 所示，以下主要展示了其基础信息、安全分析以及恶意性分析模块的截图. 通过分析结果我们能看到，MobSF 会对 APP 有一个比较全面的分析，结果会罗列出 APP 的基础信息和各种组件数量，以及用到的一些权限，并针对一些高风险的权限标记 dangerous，然后在安全分析以及恶意分析里面仅仅给出了可能存在高风险的操作，并不能捕获到具体的恶意行为. MobSF 并未像本文提出的基于

传感器的应用行为识别那样,能直接捕获到应用的恶意行为如图 6,并且本文提出的方法捕获到的行为能作为有力的取证分析切入点或者证据.两者主要对比如表 1 所示.

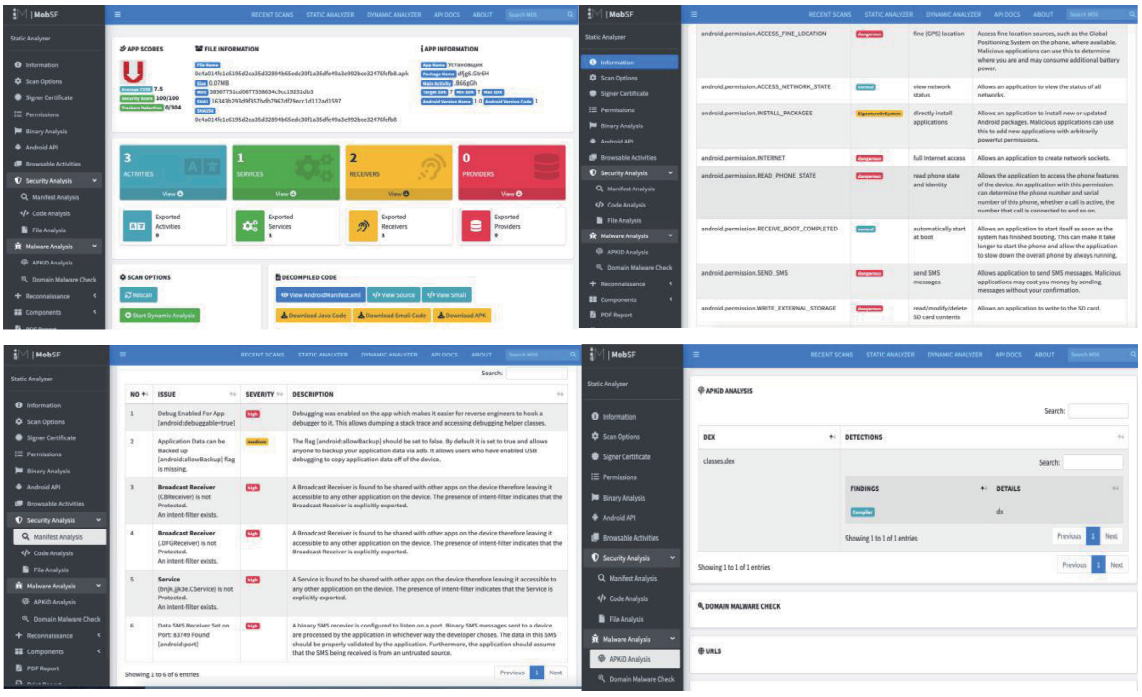

图 9　MobSF 下 Установщик 分析结果

Fig. 9　Analysis result of Установщик on MobSF

表 1　本文方案与 MobSF 对比

Tab. 1　ComparesbetweenMobSF and this Paper

比较项	MobSF	本文方案
自动化	高	低,需要手动打桩
检测项	多	仅涉及安全项
直观性	弱	强
准确度	弱/仅参考价值	高/能直接捕获异常行为

4　结　论

本文针对应用恶意行为识别提出一种基于传感器的应用行为识别方法.针对目前 API 识别都忽略的设备状态问题,设计了一个基于传感器的设备状态时序、API 使用时序和 GUI 首屏时序的三元时序模型,以及一个基于该模型的恶意行为识别器,用于识别应用行为的恶意性.通过典型案例验证了模型对恶意应用识别的准确性,利用黑盒测试验证模型的识别能力,从而证明该方法的有效性,最后通过一个对比实验,演示了本文方法相对于现有方法更加直观更加高效.

参考文献:

[1]　刘凯,方勇,左政,等.基于图卷积网络的恶意代码聚类[J].四川大学学报:自然科学版,2019,56:654.

[2]　赵翠镕,方勇,刘亮,等.基于语义 API 依赖图的恶意代码检测[J].四川大学学报:自然科学版,2020,57:488.

[3]　Zhang H, Luo S, Zhang Y, *et al*. An efficient Android malware detection system based on method-level behavioral semantic analysis [J]. IEEE Access, 2019, 7: 69246.

[4]　Jung J, Kim H, Shin D, *et al*. Android malware detection based on useful API calls and machine learning [C]//Proceedings of the 2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE). Laguna Hills: IEEE, 2018.

[5]　Utku A, DoGru I A, Akcayol M A. Permission based android malware detection with multilayer perceptron [C]//Proceedings of the 2018 26th Signal Processing and Communications Applications Conference (SIU). Lzmir: IEEE, 2018.

[6]　刘晓建,雷倩,杜茜,等.多上下文特征的 Android 恶意程序静态检测方法[J].华中科技大学学报:自然科学版,2020(2):1671.

[7]　葛文麒,杨清,廖俊国,等.基于特征加权的深度学习 Android 恶意检测系统研究[J].计算机工程,

2019(1)：1000.

[8] Zhe M Y，YangM，Zhang Y，*et al*. AppIntent：Analyzing sensitive data transmission in Android for privacy leakage detection [C]//AcmSigsac Conference on Computer & Communications Security. Berlin：ACM，2013.

[9] UlrichB，Paolo M C，Clemens H，*et al*. Scalable，behavior-based malware clustering [C]//NDSS Symposium. The Catamaran Resort Hotel and SpaSan Diego. CA：NDSS，2009.

[10] Iker B，Urko Z. Crowdroid：behavior-based malware detection system for android [C]//Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices (SPSM '11). New York：ACM，2011.

[11] Freeman J. Android bug superior to master key [EB/OL]. [2020-2-20]. http：//www. saurik. com/id/18，Isla Vista，CA：Saurik，

[12] 彭国军，李晶雯，孙润康，等. Android 恶意软件检测研究与进展[J]. 武汉大学学报：理学版，2015，61：1671.

[13] Daniel A，Michael S，Malte H，*et al*. Drebin：efficient and explainable detection of android malware in your pocket [C]//Proceedings of the NDSS. California：IEEE，2014.

[14] Ajin A，Ma G F，Matan D，*et al*. Mobile-Security-Framework [EB/OL]. (2018-03-08)[2020-08-01]. https：//github. com/MobSF/Mobile-Security-Framework-MobSF.

[15] Anthony D. Androguard [EB/OL]. (2018-03-13)[2020-08-01]. https：//github. com/androguard/androguard.

引用本文格式：

中　文：杨频，冉涛，张磊，等. 一种基于传感器的 Android 应用行为分析技术[J]. 四川大学学报：自然科学版，2021，58：013002.

英　文：Yang P，Ran T，Zhang L，*et al*. An analysis technology of Android application behavior based on sensors [J]. J Sichuan Univ：Nat Sci Ed，2021，58：013002.