基于关联分析的 Android 权限滥用攻击检测系统①

陈宏伟, 熊 焰, 黄文超, 黄建盟

(中国科学技术大学 计算机科学与技术学院, 合肥 230022)

摘 要: 为了限制应用软件的行为, Android 系统设计了权限机制. 然而对于用户授予的权限, Android 应用软件却可以不受权限机制的约束, 任意使用这些权限, 造成潜在的权限滥用攻击. 为检测应用是否存在权限滥用行为, 提出了一种基于关联分析的检测方法. 该方法动态检测应用的敏感行为与用户的操作, 并获得两者的关联程度. 通过比较待检测应用与良性应用的关联程度的差别, 得到检测结果. 基于上述方法, 设计并实现了一个原型系统 DroidDect. 实验结果表明, DroidDect 可以有效检测出 Android 应用的权限滥用行为, 并具有系统额外开销低等优点.

关键词: Android 安全; 权限滥用攻击; 关联分析; 检测系统

Association Analysis Based Detection System for Android Permission Abuse Attacks

CHEN Hong-Wei, XIONG Yan, HUANG Wen-Chao, HUANG Jian-Meng

(School of Computer Science and Technology, University of Science and Technology of China, Hefei 230022, China)

Abstract: In order to restrict the behaviors of applications, a permission system is designed in Android system. However, for the permissions granted by the users, applications will no longer be restricted and can use these permissions at will, which may cause the potential permission abuse attacks. To detect the permission abuse behaviors of applications, an association analysis based detection method was proposed. This method dynamically detects sensitive behaviors of applications and operations of users, then calculates the degree of association between them. Detection result will be obtained through comparing the differences between detected applications and benign applications. A prototype system named DroidDect was designed and implemented based on the above method. The experimental results show that permission abuse behaviors in Android applications can be effectively detected by DroidDect with advantages including low system overhead.

Key words: Android security; permission abuse attacks; association analysis; detection system

随着智能手机使用的普及, Android 系统已经成为市场上最流行的手机操作系统[1]. 和其他智能手机操作系统不同, 除了官方应用市场 Google Play, Android 系统还允许用户从第三方市场下载和安装应用程序. 部分第三方应用市场在发布应用之前, 并没有进行任何形式的安全检查[2], 这就为恶意应用的传播提供了相当程度的便利. 由于通过手机可以获取大量的用户个人信息, Android 系统成为了众多恶意软件的攻击目标. 有数据显示, 超过 95%的恶意应用是运行在 Android 平台上的[3]; 而且平均每 18 秒就有一个恶意 Android 应用诞生[4]. 面对如此严峻的安全挑战, Android 恶意应用的检测成为了人们的研究热点.

传统的恶意应用检测方法主要包括提取待检测应用的静态和动态特征, 将这些特征和已经预定义好的恶意软件特征相比较. 通过比较的结果, 判定待检测应用与恶意应用的相似程度[5-9]. 检测的特征包括应用

① 基金项目:国家自然科学基金(61572453,61202404,61520106007,61170233,61232018);安徽省自然科学基金(1508085SQF215);中央高校基本科研基金(WK0110000041)

收稿时间:2015-08-16;收到修改稿时间:2015-10-14

所需要获取的权限、方法调用、应用的信息流等. 随着检测方法的加强, 恶意应用也出现了新的攻击手段, 其中包括权限滥用攻击.

权限滥用攻击是指恶意应用利用 Android 权限机制的漏洞, 将自己伪装成良性应用, 在完成用户的需求之外, 滥用用户所赋予的权限, 窃取用户的隐私信息. 例如, 一个申请了录音权限的恶意应用程序, 当用户按下“录音”按钮后, 这个应用确实是录音了; 但是当用户停止录音后, 这个应用仍在后台继续录音, 并且将获得的音频数据发送至远程的服务器上, 对用户实施了窃听攻击.

文献[10]提出一种方法, 可以将用户说话时引发的手机陀螺仪传感器振动信号恢复成为音频信号, 再配合使用语音识别算法, 可以最多恢复出 65%的用户语音信息. 文献[11]发现了一个恶意应用, 可以监听用户的通话状态, 当用户进行通话时, 应用将在后台自动开启录音功能, 窃听用户的通话内容并将其存储在手机的 SD 卡上. 文献[12]发现了一个恶意应用, 可以根据远程指令, 让用户的手机偷偷地进行拍摄、录音等. 这些恶意应用实施攻击的根本手段, 都是滥用用户所赋予的权限. 有研究显示[13], Android 应用经常在没有征得用户许可的情况下, 将用户的隐私信息发送出去.

这一类恶意应用只使用了完成用户指定功能所需要的权限, 而且实施攻击时也仅仅只利用这些权限, 并没有涉及其他敏感权限, 具有较强的隐蔽性. 传统的检测手段由于没有将软件的静态和动态特征同用户的使用意愿相联系, 所以无法检测出这一种攻击行为.

为了检测恶意应用的权限滥用攻击, 现有的解决方案包括使用污点分析技术[14]、应用行为重构技术[15]、用户意图确认[16]等方法. 但是这些方案存在额外开销大, 对 Android 系统代码改动多的缺点.

滥用权限的恶意应用和良性应用的根本区别在于, 软件的行为是否由用户所触发. 根据这一基本思想, 本文设计并实现了一种新的轻量级的检测系统 DriodDect. DroidDect 基于软件动态检测技术, 获取软件运行时的行为和用户的操作. 再基于关联分析, 检测软件行为和用户操作的置信度, 通过置信度结果, 判定 Android 应用中是否存在权限滥用攻击.

经过实验, DroidDect 可以有效检测出 Android 应用中的权限滥用攻击. 相比于现有的检测方案, DroidDect 具有对 Android 系统代码的修改程度小, 系统额外开销低的优点, 具有较高的实用性.

权限滥用攻击是指一类基本思想相同的攻击方式, 区别仅在于滥用的具体的权限不同, 所以检测的基本思想是一样的. 下面将以检测权限滥用攻击中的窃听攻击为重点, 对比传感器和拍照权限滥用攻击的检测, 阐述本系统的分析、设计和实现.

1 系统设计

1.1 攻击模型与假设

本文假设 Android 的用户是诚实的, 但是应用可能是恶意的. 恶意应用会根据远程攻击者的指令, 悄悄在后台开启录音、传感器、拍照功能, 对用户进行窃听等攻击. 由于本系统的实现需要对 Android 框架层进行修改, 并且需要从修改后的 Android 框架层获取信息, 所以本文假设 Android 框架层和之下的 Linux 内核层是安全的, 不会拦截或篡改返回的信息. 而且恶意应用也不能获取系统的 root 权限.

基于上述攻击模型和假设, 本文设计并实现了 Android 权限滥用攻击检测系统 DroidDect. DroidDect 系统可以获取应用的行为和用户的操作, 并且检测两者之间的置信度. 当检测到置信度小于一个预定的阈值时, DroidDect 系统会自动向用户报警, 提醒用户应用存在权限滥用攻击.

1.2 各模块设计

DroidDect 系统的框架如图 1 所示. 该系统由六个部分组成: 应用行为检测模块、用户操作检测模块、数据处理模块、关联分析模块、用户配置数据库和用户配置管理器, 各部分具体介绍如下:

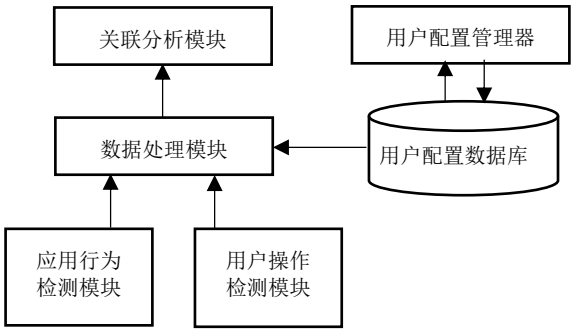

图 1 DroidDect 系统框架

(1) 应用行为检测模块. 此模块用于动态检测应用运行时的行为, 为之后的数据处理模块提供原始的应

用行为数据. 检测的数据取决于需要检测的攻击类型. 两者的对应关系和具有代表性的恶意应用如表 1 所示.

表 1 攻击类型与需检测的数据

攻击类型	典型恶意应用	需检测的方法调用
录音权限滥用攻击	文献[11]	应用是否访问麦克风音频数据
传感器权限滥用攻击	文献[10]	应用是否访问传感器数据
拍摄权限滥用攻击	文献[12]	应用是否访问摄像机数据

为了检测不同种类的权限滥用攻击, DroidDect 需要检测不同的数据. 当系统运行时, 用户可以通过用户配置管理器配置自己需要检测的攻击类型. 因为检测方法相同, 用户可以指定检测其中一种攻击, 也可以同时检测几种攻击.

(2) 用户操作检测模块. 此模块获取用户的操作数据, 与应用行为数据一起作为数据处理模块的输入. Android 是一个由事件触发的系统. 其中, 由用户触发的事件可以大致分为两类: 第一种是 GUI 事件触发, 比如用户按下某个按钮; 另一种是非 GUI 事件触发, 比如用户转动设备, 触发陀螺仪传感器检测到设备运动. 在检测录音滥用攻击中, 我们调查了 Android 市场上最受欢迎的 20 个录音应用. 我们发现这 20 个应用启动录音的方式都是通过用户按下按钮触发, 也就是由 GUI 事件触发. 所以在这里, 我们默认将用户的按钮操作作为录音应用中用户操作的检测对象.

在传感器和拍照应用中, 本文发现其中既存在 GUI 事件触发方式, 也存在非 GUI 事件触发方式. 对于如何检测由非 GUI 事件触发的用户功能, 本文将在第三部分中讨论.

(3) 数据处理模块. 前两个模块产生的检测结果是时间序列数据, 此模块的作用是对前两个模块产生的数据进行处理与合并. 处理包括统一时间间隔, 去除重复数据, 插入遗漏数据. 然后以两个文件的时间项为主键, 对两个文件做连接操作, 形成一个文件作为输出. 文件的格式为 CSV(Comma Separated Values)格式, 即逗号分隔值格式. 文件的每一行为每一秒测试的数据值.

(4) 关联分析模块. 在这个模块中, 本文以关联分析算法思想为基础, 针对本应用场景, 实现了一个简单的关联分析算法. 此模块将数据处理模块产生的数据作为输入, 计算出应用行为和用户操作的置信度. 当计算出的置信度低于预设的阈值时, DroidDect 会提醒用户该应用存在权限滥用攻击.

(5) 用户配置管理器和用户配置数据库. 用户配置管理器用于接收用户的配置输入, 并且将用户的数据存储在用户配置数据库中. 当 DroidDect 系统运行时, 数据处理模块会查询用户配置数据库, 获得数据处理需要的参数.

2 系统实现与算法

2.1 行为检测部分实现

基于 Android4.4.3_r1, DroidDect 系统修改了系统应用程序框架层, 实现了对软件行为特征和用户的操作特征的动态检测. 下面分别作详细介绍.

2.1.1 软件行为检测

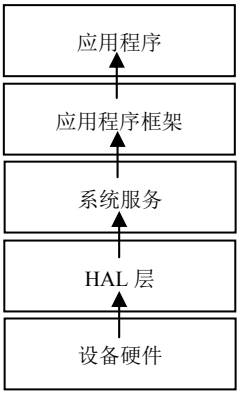

图 2 Android 应用数据流图

Android 系统架构是一个堆栈结构的软件系统. 如图 2 所示, 当一个应用需要访问底层硬件数据时, 需要首先向相应的管理器类注册, 当相应的事件发生时, 管理器类会调用系统服务向应用程序返回数据. 在应用程序框架层中, 和录音功能实现相关的类是 AudioRecord 和 MediaRecorder 类. 其中, 和录音相关的方法是 AudioRecord 类中的 read() 方法和 MediaRecorder 类中的 start()方法. 为了不遗漏地检测应用从麦克风获取到的数据大小, DroidDect 系统修改了 AudioRecord 类中的 read(byte[], int, int)、read(short[], int, int) 和 read(ByteBuffer, int) 方法; 以及 MediaRecorder 类中的 start()方法. DroidDect 系统在这些方法返回之前, 将方法返回的数据量大小输出. 因此任何调用录音方法的应用, DroidDect 都可以检测到这个应用获取的音频数据量大小及调用时间.

同理, 为了检测传感器权限滥用攻击和拍摄权限

滥用攻击. DroidDect 系统修改了和获取传感器数据相关的 SensorManager 类以及和拍摄相关的 Camera 类. 表 2 列出了改动的方法签名.

表 2 SensorManager 与 Camera 类修改的方法

检测传感器权限使用	SensorManager.registerListener (SensorEventListener, Sensor, int, int)
	SensorManager.registerListener (SensorEventListener, Sensor, int, Handler)
	SensorManager.unregisterListerner (SensorEventListener, Sensor)
	SensorManager.unregisterListerner (SensorEventListener, Sensor)
检测拍摄权限使用	Camera.open()
	Camera.takePicture()
	Camera.release()

值得注意的地方是: ①在一个应用对传感器注册监听器与取消注册之间的时间段里, 应用既可以选择获取传感器数据, 也可以选择不获取. 为了最大程度检测应用可能存在的恶意行为, DroidDect 系统选择最坏情况来检测, 即默认为注册监听器与取消监听器的时间段里, 应用一直是获取传感器数据的. ②根据 Android 开发官方文档[17], 应用获取拍摄功能有两种方法: 第一种是通过发起 Intent, 使用系统原生的拍摄应用; 第二种是使用自己编写的拍摄应用. 第一种方法使用时, 用户界面会自动切换到原生的拍摄应用, 界面会显示摄像机拍摄的实时画面. 本文认为, 使用这种方法会暴露应用访问了摄像机, 用户会发现应用的偷拍行为. 所以本文认为存在拍摄滥用攻击的恶意应用一定是采取第二种方法访问设备相机的. 因此修改了这种方法相应的系统调用(如表 2 所示).

2.1.2 用户操作行为检测

(1) GUI 事件检测

通过调研 Android 市场上最受欢迎的 20 个录音应用, 我们发现, GUI 事件触发录音有两种方式. 第一种是点击按钮开始录音, 再点击按钮结束录音; 第二种是长按按钮录音, 释放按钮停止录音.

Android 系统中的 Button 类继承于 TextView 类. 当用户点击按钮时, 会触发 TouchEvent 事件, 进而会调用 TextView 类中 onTouchEvent(MotionEvent)回调函数. 为了检测用户点击按钮事件, DroidDect 系统修改了这个回调函数. 当检测到 TouchEvent 事件为 ACTION_DOWN(手指接触屏幕)或 ACTION_UP(手指离开屏幕)时, DroidDect 系统会产生相应的输出, 内容为: 事件发生的时间 t, 事件类型 action_type, 按钮所在的程序包名 class_name, 以及按钮的文字 btn_text.

由于存在上文提及的两种不用的触发方式, 在获得了用户操作行为的原始数据后, 需要将原始数据由数据处理模块进行处理.

对于第一种触发方式, 用户两次点击按钮之间的时间段是作为用户的操作意愿, 所以需要将两组 ACTION_DOWN、ACTION_UP 作为起始和终止时间点. 处理如图 3 所示.

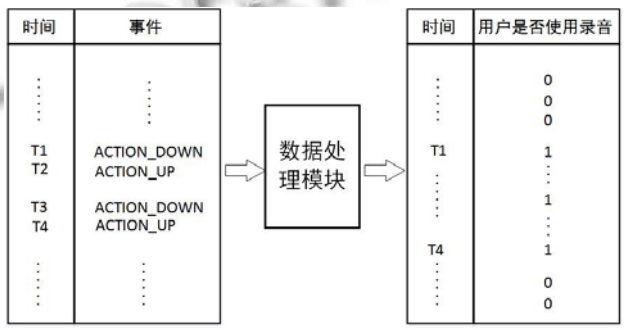

图 3 第一种触发方式数据处理

对于第二种触发方式, 用户长按的时间段是用户的操作意愿, 所以应该将一组 ACTION_DOWN、ACTION_UP 作为起始和终止的时间点. 处理如图 4 所示.

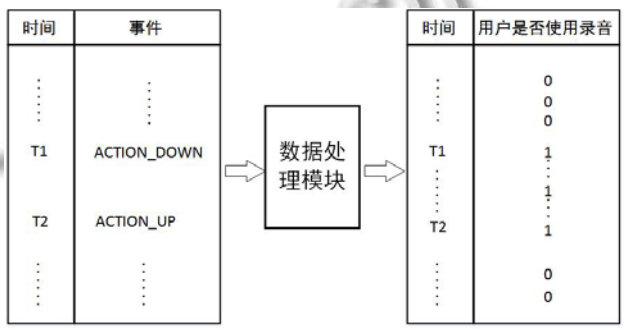

图 4 第二种触发方式数据处理

由于无法预先知晓应用的使用方式, 所以, 具体采用何种方式处理数据需要用户事先通过配置管理器输入.

(2) 非 GUI 事件检测

对于检测使用传感器的应用, 由于大多数应用会自动注册传感器监听器, 不会提供 GUI 界面启动(比如微信的摇一摇功能), 所以无法直接利用上文的检测方法. 本文设计了一种方法, 可以将对非 GUI 事件检测

转换为对 GUI 事件检测. 具体的方法是, DroidDect 系统提供了一个悬浮按钮框. 当用户开始使用传感器时, 需要点击这个悬浮按钮. 当用户结束使用传感器时, 再点击这个悬浮按钮. 这样, DroidDect 系统只需要检测这个按钮的触发事件, 就可以获得用户的操作意愿. 这样设计的目的, 是为了模拟 GUI 的触发方式, 获得同图 3 一样的原始数据, 从而可以复用 GUI 事件触发的检测方法.

2.2 关联分析部分实现

2.2.1 关联分析

关联分析是用来发现数据集中有意义的联系规则, 这个数据集通常是事务的集合, 可以表示为 T={t_1, t_2, …, t_N}. 每个事务由许多项组成, 所有的项集可以表示为 I={i_1, i_2, ..., i_d}. 每个事务 t_i 都是 I 的子集. 关联分析所发现的联系可以用关联规则的形式表示. 例如, 在数据中, 当项 T_1 出现时, 项 T_2 也经常出现, 则说明 T_1 和 T_2 有着很强的关联, 这个关联规则可以表示为: $\{T_1\} \rightarrow \{T_2\}$.

包含 0 个或多个项的集合为项集, 项集 X 出现的次数可以在数学上表示为: $\sigma(X)=|\{t_i|X \subseteq t_i, t_i \in T\}|$. 可以用支持度(support)和置信度(confidence)度量一个关联规则. 公式定义如下:

$$s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N} \quad (1)$$

$$C(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} \quad (2)$$

从公式中可以看出, 规则 X→Y 的支持度高意味着此规则在数据集中出现的频率高. 而置信度高意味着, 在一条事务中, 当 X 出现时 Y 也出现的可能性高. 关联分析的目标是找出支持度、置信度大于等于阈值的所有规则. 可以通过两个步骤解决这个问题: (1)发现超过支持度阈值的所有项集. (2)从上步发现的项集中提取所有高置信度的关联规则.

2.1.2 软件行为和用户操作的关联分析

DroidDect 系统检测权限滥用攻击的核心思想是判断软件行为和用户操作的关联程度. 本文认为, 良性应用的软件行为同用户操作之间有较强的关联, 而存在权限滥用攻击的应用则相反. 根据关联分析的基本思想, 对于一条关联规则, 可以用支持度和置信度去衡量这条规则是否足够有意义.

在 DroidDect 系统中, 关联分析模块获得的数据有 3 列, 分别是: 时间(time)、是否使用敏感权限(isSenPer)、是否有用户操作(isUsrOpr). 良性应用的检测数据中, 当 isSenPer 为 true 时, isUsrOpr 有很高的可能性也为 true; 存在权限滥用攻击的应用的可能性相比则低许多. 因此, 需要检测的是 c(*isSenPer= true → isUsrOpr=true*). DroidDect 计算置信度的算法根据公式 2-2 设计, 如下所示. 注意, 支持度在这里并没有实际意义, 因为在一段时间里, 用户的操作时间长度是不可预知的, s(*isSenPer= true→isUsrOpr=true*)的大小只能反映用户操作的时间长度, 而不能反映任何关于权限滥用的信息.

算法 1. 计算置信度

输入:

D_i: 待检测数据

D_{ij}: 每一条数据分为 3 个列:时间、敏感权限、用户操作, j=0,1,2

输出:

C: 置信度

```
Function CalConfidence(Di)
{
  C←0
  SenPerCount←0
  UsrOprCount←0
  FOR each Di in D DO:
    IF Di1 is TRUE:
      SenPerCount←SenPerCount+1
      IF Di2 is TRUE:
        UsrOprCount←UsrOprCount+1
  C←UsrOprCount / SenPerCount
  Output C
}
```

3 实验评估

3.1 阈值确定

为了测试 DroidDect 系统的有效性和性能, 需要首先确定良性应用中软件行为和用户操作置信度的阈值. 为了获得良性应用, 本文选择 Android 官方应用市场 Google Play 作为获取良性应用的渠道. 数据显示, 从 Google Play 下载的应用中, 只有不到 0.001%的应用是恶意的[5,18]. 所以, 可以认为 Google Play 中的应用基本上都是良性的. 为了保证实验结果的可靠性,

本文还对从 Google Play 上下载的应用进行了额外的检测, 如使用其他病毒检测工具、查看用户使用评论等, 确保应用不包含恶意行为.

为了使阈值的确定具有代表性, 本文依照 Google Play 的排名, 为每一类选取了 8 至 15 个最受欢迎的应用, 并测试了每一个应用的置信度, 部分样本如表 3 所示. 实验环境为基于 Android4.4.3_r1 的修改版 Android 操作系统, 硬件平台为 Nexus7.

表 3 部分良性应用样本与总体置信度

敏感权限	部分良性应用样本	总体平均置信度
录音	Voice Recorder, Smart Voice Recorder, Audio Recorder, Easy Voice Recorder	0.992
传感器	Steps Mania, Steps Counter, Compass, Cover Lock Screen	0.988
拍照	HD Camera, HD Camera Ultra, Open Camera, Camera360 Ultimate	0.991

DroidDect 系统将这些良性应用的总体平均置信度作为阈值, 当被检测的应用的置信度低于这个阈值时, DroidDect 系统则认为此应用存在权限滥用攻击.

3.2 有效性评估

为了测验 DroidDect 系统的有效性, 本文为每类权限滥用攻击设计了一个典型的恶意应用, 并将这个恶意应用同良性应用混合, 在不同的使用场景下重复测试多次.

为了测试 DroidDect 的检测效果, 本文分别实现了文献[10-12]中所描述的权限滥用攻击应用, 作为恶意应用测试, 并且同使用敏感权限的良性应用混合, 一同检测. 设计用于测试的恶意应用的思路是, 该应用不仅可以通过用户点击图形化界面触发应用功能, 还可以通过用户不知情的隐蔽方式获取用户的敏感数据. 例如, 为实现文献[10]中所描述的权限滥用攻击应用, 本文设计并实现了一个计步器应用, 当用户启动计步器的计步功能后, 该应用会获取手机的加速度传感器和陀螺仪传感器信号, 计算用户的步数. 但当该应用检测到手机的麦克风处于被其他应用占用状态时, 该应用会在后台自动监听上述传感器的信号, 并写入文件保存在手机的 SD 卡中, 待之后发向网络, 还原为音频信号, 实现窃听攻击.

为了准确测试恶意行为的存在, 测试恶意应用的权限滥用行为需要遍历恶意应用使用中所有的使用场景并随机确定每种场景的测试时间长度. 例如检测录音滥用攻击时, 上述恶意应用在用户通话时实施窃听, 因此测试的场景必须包含随机时间长度的通话场景.

检测方法是对于每个应用, 在各种使用场景中测试 50 次, 其中 20 次是用户主动发起使用敏感权限, 这属于合法使用; 剩余 30 次测试中, 用户并不主动发起使用敏感权限功能, 而是遍历所有可能的应用使用场景. DroidDect 系统检测这 50 次测试中, 应用所使用敏感权限的总次数, 并且同时计算这 50 次测试中应用软件的置信度. 下表列出的是恶意应用的检测结果.

表 4 权限滥用攻击检测数据

敏感权限	测试次数	测试正常功能次数	检测到敏感权限次数	置信度	良性置信度阈值
录音	50	20	47	0.37	0.992
拍照	50	20	39	0.38	0.991
加速度	50	20	31	0.53	0.988
陀螺仪	50	20	30	0.47	0.988
磁场	50	20	34	0.17	0.988
光线	50	20	28	0.51	0.988
GPS	50	20	25	0.37	0.981

当恶意应用和良性应用混合检测时, 由于恶意应用的置信度远低于良性应用的总体平均置信度, DroidDect 通过对比待检测应用和预设的置信度阈值, 100%检测出了存在权限滥用行为的应用.

3.3 性能评估

本文使用了一组测试基准以检测 DroidDect 系统的性能. 测试软件为 CaffeineMark 3.0. 测试基准包括运行寻找素数运算、运行循环、方法、浮点运算等, 以检测系统的性能. 测试结果如图 5 所示.

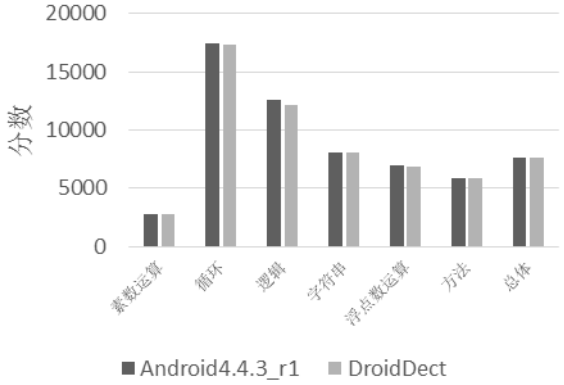

图 5 性能评估结果

通过比较检测结果, DroidDect 的系统性能和原生

的 Android 系统性能相比, 两者的差别很小, DroidDect 系统为检测权限滥用攻击所需要额外的系统开销几乎可以忽略不计.

4 结语

本文提出了一种检测 Android 应用程序权限滥用攻击的方法, 该方法的核心思想是通过修改 Android 系统的应用程序框架, 动态检测应用软件敏感权限的使用情况以及用户的操作情况, 并基于关联分析的基本思想, 计算软件行为与用户操作的置信度. 通过和良性应用的总体平均置信度相比较, 检测权限滥用攻击. 基于上述方法, 实现了一个原型系统 DroidDect, 并通过检测恶意应用样本, 验证了该系统可以有效检测权限滥用攻击, 并具有对 Android 系统代码的修改程度小, 系统额外开销低的优点.

当良性应用切换到后台运行状态时, 相比于在前台运行, 所消耗的网络流量和 CPU 负载均会大幅下降, 并维持在较低水平. 而存在权限滥用攻击的恶意应用则不然. 通过分析应用的网络流量和 CPU 负载数据的变化同用户操作之间的关联, 可以更加精确地检测出恶意应用权限滥用攻击及其发生的时间. 因此, 下一步的工作将着重研究如何加入网络流量、CPU 负载数据检测权限滥用攻击, 并进一步降低检测运行时的开销.

参考文献

1 Smartphone OS Market Share, Q1 2015. http://www.idc.com/prodserv/smartphone-os-market-share.jsp [2015-05-27].

2 Cybercriminals target Android platforms. http://www.av-comparatives.org/wp-content/uploads/2013/08/apkstores_investigation_2013.pdf [2013-08-26].

3 Mobile Threat Report Q1 2014. https://www.f-secure.com/documents/996508/1030743/Mobile_Threat_Report_Q1_2014.pdf [2014-05-06].

4 G Data Mobile Malware Report. https://public.gdatasoftware.com/Presse/Publikationen/Malware_Reports/G_DATA_MobileMWR_Q1_2015_US.pdf [2015-07-07].

5 Grace M, Zhou Y, Zhang Q, et al. Riskranker: Scalable and accurate zero-day android malware detection. Proc. of the 10th International Conference on Mobile Systems, Applications, and Services. ACM. 2012. 281−294.

6 Zhou Y, Jiang X. Dissecting android malware: Characterization and evolution. 2012 IEEE Symposium on Security and Privacy (SP). IEEE. 2012. 95–109.

7 Ristenpart T, Tromer E, Shacham H, et al. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. Proc. of the 16th ACM Conf. on Computer and Communications Security. ACM. 2009. 199–212.

8 Grace MC, Zhou Y, Wang Z, et al. Systematic Detection of Capability Leaks in Stock Android Smartphones. NDSS. 2012.

9 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.

10 Michalevsky Y, Boneh D, Nakibly G. Gyrophone: Recognizing speech from gyroscope signals. Proc. of the 23rd USENIX Security Symposium (SEC'14). USENIX Association. 2014.

11 Virus Profile: Android/NickiSpy. A. http://home.mcafee.com/virusinfo/virusprofile.aspx?key=554488 [2011-10-26].

12 Dendroid malware can take over your camera, record audio, and sneak into Google Play. https://blog.lookout.com/blog/2014/03/06/dendroid/ [2014-03-06].

13 Zhou Y, Jiang X. Dissecting android malware: Characterization and evolution. 2012 IEEE Symposium on Security and Privacy (SP). IEEE. 2012. 95−109.

14 Enck W, Gilbert P, Han S, et al. TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones. ACM Trans. on Computer Systems (TOCS), 2014, 32(2): 5.

15 Zhang Y, Yang M, Xu B, et al. Vetting undesirable behaviors in android apps with permission use analysis. Proc. of the 2013 ACM SIGSAC Conference on Computer & Communications Security. ACM. 2013. 611–622.

16 Yang Z, Yang M, Zhang Y, et al. Appintent: Analyzing sensitive data transmission in android for privacy leakage detection. Proc. of the 2013 ACM SIGSAC Conference on Computer & Communications Security. ACM. 2013. 1043–1054.

17 Camera. http://developer.android.com/guide/topics/media/camera. html [2013-06-04].

18 Contrary to what you've heard, Android is almost impenetrable to malware. http://qz.com/131436/contrary-to-what-youve-heard-android-is-almost-impenetrable-to-malware/ [2013-10-03].