doi:10.11959/j.issn.1000-436x.2017103

DroidBet：事件驱动的 Android 应用网络行为的自动检测系统

魏松杰，吴高翔，罗娜，时召伟，周紫阳
（南京理工大学计算机科学与工程学院，江苏 南京 210094）

摘 要：多数 Android 应用需要通过连入互联网与外界进行通信，所有与网络相关的活动都涉及网络流量，通过分析建模 Android 应用的网络流量，可以一定程度上掌握 Android 应用的网络行为。因此，设计了一个事件驱动的网络行为自动检测系统 DroidBet，来对 Android 应用进行自动测试评估。首先，建立一个场景模拟事件库，用来模拟应用程序运行过程中可能执行的事件，从而尽可能地触发应用程序的网络行为；然后，自动生成基于状态转移分析方法的测试序列，同时对应用程序测试过程中的网络行为进行动态收集；最后，采用机器学习方法对收集到的网络行为进行学习、训练，生成基于 BP 神经网络的网络行为模型，实现对未知的 Android 应用的行为检测。实验结果表明，DroidBet 能够有效地触发并提取应用程序的网络行为，并具有准确度高、系统资源开销低等优点。

关键词：Android；场景模拟；网络行为；自动化检测
中图分类号：TP393 **文献标识码**：A

DroidBet: event-driven automatic detection of network behaviors for Android applications

WEI Song-jie, WU Gao-xiang, LUO Na, SHI Zhao-wei, ZHOU Zi-yang
(School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China)

Abstract: The most Android applications connect to Internet to communicate with the outside world. Applications' network-related activities were reflected and described with network traffic. By analyzing and modeling network traffic of Android applications, network behaviors of Android applications could be subsequently characterized. Therefore, DroidBet: an event-driven network behavior automatic detection system was presented, to test and evaluate Android applications automatically. Firstly, a scenario simulation event library was built to simulate the events that applications may be executed in the process, so as to trigger the network behavior of the application as much as possible. Then, the test sequence based on the state transition analysis method was automatically generated, and the network behavior was dynamically collected during the application testing process. Finally, the machine learning method was used to learn and train the collected network behavior, and the network behavior model based on BP neural network was generated to detect the behavior of the unknown Android application. The experimental results show that DroidBet can effectively trigger and extract the network behavior of the application, which has the advantages of high accuracy and low resource cost.

Key words: Android, scenario simulation, network behavior, auto-detection

1 引言

随着以智能手机为代表的移动智能终端设备在经济生产、文化生活、教育娱乐领域的广泛普及，Android 系统成为目前用户最多、设备覆盖范围最广、应用程序最丰富的移动操作系统平台，得到了充分的发展与应用。据全球最具权威的 IT 研究与顾问咨询公司 Gartner 的数据显示，截至 2016 年底，已有 30 亿部智能设备使用 Android 系统[1]。与 IOS 操作系统不同，除了官方应用市场 Google Play，

收稿日期：2016-09-06；**修回日期**：2017-03-27
基金项目：国家自然科学基金资助项目（No.61472189）
Foundation Item: The National Natural Science Foundation of China (No.61472189)

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，缺少有效的安全审核和应用测评过程[2]，无法及时发现并过滤具有风险和恶意行为的应用程序。有数据显示，超过 95%的恶意应用是运行在 Android 平台上的；而且平均每 18 s 就有一个恶意 Android 应用诞生[2]。面对如此严峻的安全挑战，Android 应用的行为属性检测成为学者们的研究热点。

目前，针对 Android 应用的恶意检测已经取得了大量的研究成果，包括静态特征码匹配、动态特征码匹配等方法，一定程度上消除了恶意应用带来的威胁。但随着应用程序变异速度的加快（如重新打包、代码混淆），传统的静态匹配特征码方式已经无法满足需求，因此，动态行为检测方法已然成为分析与检测新兴的未知应用程序的重要研究手段。

网络流量是记录和反映应用程序网络行为的重要载体。通过和正常状态时的网络流量特征进行对比，能及时发现网络中的异常行为。基于以上特点，本文设计了一套基于事件驱动的移动应用网络行为自动检测方法 DroidBet（Android app behavior testbed）。通过构建应用程序运行过程中不同的行为触发事件序列，在测试平台上激发应用程序与外界网络环境之间受控的网络流量交换。系统自动抓取应用程序运行过程中产生的网络流量，而后对抓取到的网络流量进行统计分析和数据建模，进而通过跟踪与分析应用程序的网络行为，评估应用程序的恶意属性和风险程度。本文的主要研究成果和创新点如下。

1) 构建了一个基于事件驱动的 Android 应用网络行为的自动检测系统 DroidBet。

2) 建立了一个简单高效的场景模拟事件库，用于对不同场景事件进行仿真，以便尽可能地触发应用程序系统以及用户层面的网络行为。

3) 提出了触发事件测试序列的概念以及触发事件测试序列的生成方法，根据场景模拟事件的权重遍历状态转移图，生成触发事件组合测试序列。

4) 通过对应用程序的网络行为进行统计分析与数据建模，构建了基于机器学习的移动应用程序网络行为识别模型。

2　相关工作

2.1　自动检测方法

当前，移动应用程序的检测方法基本上都是自动化检测。由 Ajina Braham 等开发的 MSF[3]（mobile security framework）是一款智能一体化的开源移动应用自动渗透测试框架，其动态分析器可以在虚拟机或者经过配置的设备上运行程序，从而在运行过程中检测问题。动态分析器可以从抓取到的网络数据分组、解密的 HTTPS 流量、程序二进制文件 dump、程序日志、程序错误和崩溃报告、调试信息堆栈轨迹和程序的设置文件、数据库等方面进行进一步的分析。

由 Lin 在 Blackhat EU2015 发布的 AndroBugs[4] 则是在流行的 Androguard 基础上开发的检测工具，其修改了 Androguard 的内核，并通过其 DVM 引擎、Search 引擎以及漏洞向量实现漏洞挖掘。其生成的报告内容详细，检测效率较高，支持批量处理以及扩展。

2.2　Android 应用恶意检测

目前，针对 Android 应用的检测，根据检测流程和特征描述方法的不同，大致可分为基于签名和基于行为的检测方法。基于签名的恶意应用检测方法被学术界广泛使用，但其有一个致命的缺陷，它必须首先构造一个包含该类恶意应用的签名库，然后才能通过该签名库检测出该类恶意应用，因此，对于新近出现的未知恶意应用，该方法是无效的。

基于行为的 Android 应用检测主要采用动态和静态 2 种方法[5]。动态检测方法主要是指运行应用程序并收集应用程序运行过程中的行为信息。如 Sui 等[6]通过对应用程序运行过程中产生的网络流量进行分析，从而确定应用程序的传播行为。静态检测方法主要依靠强大的反汇编技术将应用程序的源代码反汇编为字节码，通过对字节码中敏感字符串的提取并与已有的恶意代码库进行比对，来检测应用程序恶意与否。如 ScanDal[7]以 Dalvik 虚拟机字节码为输入，检测 Android 应用程序的私密信息泄露。Chex[8]提出了一种基于组件的静态分析方法。ScanDroid[9]使用数据流分析方法进行静态分析，从应用程序的 AndroidManifest.xml 文件中提取权限，然后自动检测应用程序的数据流是否与这些权限一致。

Shabtai 等[10]曾提出过一种恶意应用检测框架，称之为“Andromaly”。该框架主要特征就是利用机器学习技术，同时，结合相应的检测算法对利用该框架采集的多种设备特征和事件等综合信息进行监控、判断应用的恶意与否。

Burguera 等[11]提出了一个 Android 应用程序检测系统 Crowdroid。该系统采用 C/S 架构模型，利用待检测应用对 Android 底层系统调用的次数等基础应用运行时的数据来区分正常应用与恶意应用。Dini 等[12]定义了一种基于多层信息的 Android 程序检测系统（MADAM，multi-level anomaly detector for android malware），其中所述的多层信息包括主要内核级别以及用户级别的信息，以此完成对恶意应用的检测。

2.3 网络行为分析

网络行为分析是指通过分析和提取网络中的行为模式并对网络行为发生的因素进行分析，以采取相应的应对措施。其核心思想是总结各种网络行为模式，当网络行为产生变化时，找出引起其变化的原因。

在目前主流的操作系统中，网络数据分组的收发都是基于 TCP/IP 协议的。网络行为分析方法与操作系统相对无关，具有较强的通用性。因此，可以通过分析相关协议进行网络行为的监测，从而检测恶意攻击。基于特征码的检测只能检测出已经出现过的恶意攻击，而基于网络行为的检测则可以实现检测以前未出现过的恶意攻击的效果。在实际操作使用中，不同类型的应用行为大不相同。因此，建立适用于所有应用的分析规则异常困难。目前，网络行为分析的分析方法大致分为基于概率统计的方法、基于机器学习的方法、基于神经网络的方法和基于数据挖掘的方法[13]，本文将采用基于机器学习的方法进行应用网络行为的建模与识别。

3 方案设计

本节基于网络流量行为，完整地描述了一个 Android 应用网络行为自动检测系统的设计。首先，系统自动安装、运行、收集 Android 应用的流量数据，然后对收集到的流量数据进行数据清洗、统计分析、数据建模，进而通过跟踪并揭示可疑的 Android 应用行为，评估应用程序的恶意属性和风险程度。

3.1 设计思想

随着互联网以及 4G 网络的快速发展，Android 智能移动设备得到了广泛的普及。几乎所有的 Android 设备都需要连接到网络，才能与外界进行联系。同时，所有与网络相关的活动都是与网络流量联系在一起的。网络流量是记录和反映网络及其应用程序活动的重要载体。网络流量的行为是网络行为的重要组成部分，通过对网络流量的统计分析，可以在一定程度上掌握应用程序的行为。

Android 应用在活动时都会引起网络流量的变化，如上传照片、视频下载。随着 Android 应用的不断增加，通过网络流量进行应用的分类与识别是一项很有意义的工作，它有助于趋势分析、动态访问控制，并且识别不同应用类型的流量也是网络安全和流量工作的重要依据。不同应用类型的网络流量的统计，反映了不同应用程序使用网络的行为，并且对流量进行分类也是发现入侵或恶意攻击的重要方法，同时，可以识别影响网络资源分布的新应用的出现。

因此，基于以上思想，本文设计了一个事件驱动的Android应用网络行为自动检测系统DroidBet。本文定义的网络行为指的是应用程序的网络流量数据，以及其中所反映的应用操作行为模式。

3.2 系统结构

本文系统由客户端和服务器端构成。客户端向服务器端上传需要检测的应用程序，同时，查询应用程序的属性；服务器端完成应用程序的启动、运行、流量行为收集、分析和存储等任务，其整体系统结构如图 1 所示。

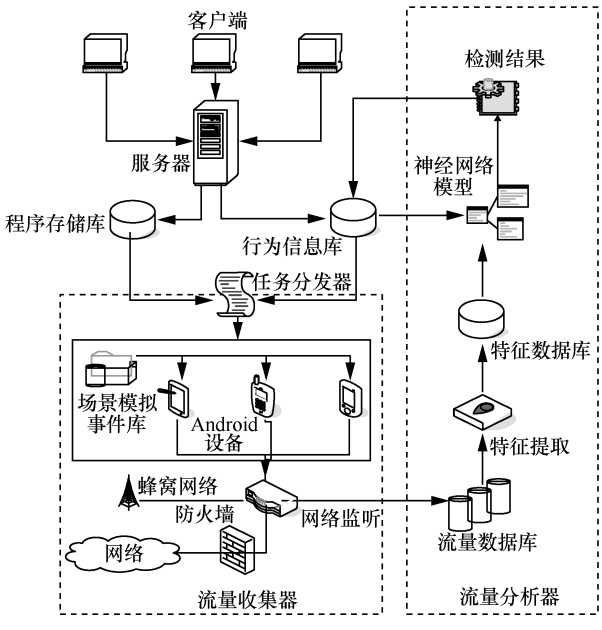


图 1 系统结构

客户端是一个通过 php 脚本语言实现的 Web 应用，主要用来上传应用程序，同时查询应用程序的属性等。

服务器端负责抓取、分发、运行、分析以及存储数据，后台需要多台 Android 设备支撑。其主要组成部分如下。

1) 程序存储库。接收来自客户端的应用程序，并将其存储在对应的位置。

2) 行为信息库。存储系统分析过的应用程序信息，具有任务分发器快速响应分析请求的能力，也可以作为流量分析器的训练数据。

3) 任务分发器。接收应用程序分析请求，并根据当前系统运行情况分发执行任务至流量收集器。

4) 流量收集器。基于 Android 设备，提供运行应用程序的功能，同时收集应用程序运行过程中的网络流量。此外，Android 设备运行过程中需要加入模拟交互策略，保证获取的网络流量能完整体现应用程序的网络行为。

5) 流量分析器。对 Android 设备收集到的网络流量进行行为统计分析和数据建模。

3.3 工作流程

系统在对 Android 应用程序进行自动检测的过程中，其数据流向如图 2 所示，主要工作流程如下。

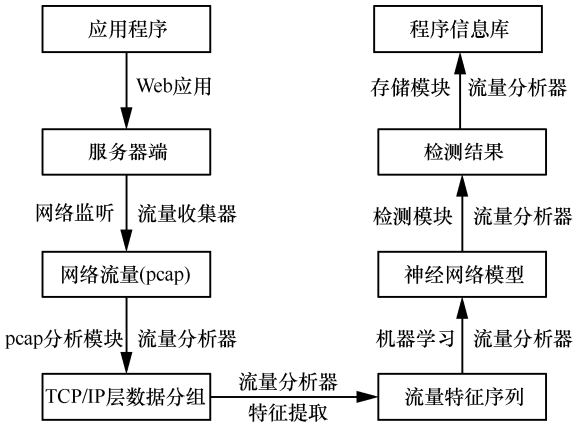


图 2　数据流向

步骤 1　客户端向服务器端发送应用程序分析请求，并将应用程序上传至服务器端。

步骤 2　任务分发器利用应用程序基本特征与行为信息库进行匹配查询，若查询到结果，直接将结果反馈给客户端，整个过程结束；若没有匹配结果，任务分发器将此任务分配给流量收集器。

步骤 3　流量收集器接到来自任务分发器的任务请求，根据系统运行情况，选择合适的 Android 设备运行应用程序，并收集应用程序运行过程中产生的网络流量。

步骤 4　流量分析器获得来自流量收集器采集到的应用程序的网络流量，并对其进行数据整理与统计分析，最后使用机器学习技术对提取的网络特征进行数据建模。

步骤 5　行为信息库存储流量分析器的最终分析结果，并且将结果反馈给客户端，整个过程结束。

4　网络行为的触发与检测

4.1 场景模拟

模拟交互技术是指通过程序的方法来模拟用户的行为。某些应用程序的网络行为必须在特定条件下才会触发。如一个音乐播放器应用需要按下播放按钮才能播放音乐，产生网络流量；还有一些应用程序，必须在 Android 4.4 以下才能表现出恶意网络行为。因此，模拟交互技术对于无人工干预下的程序自动运行过程至关重要。理想的模拟交互技术需适应当前应用程序，针对性地产生策略交互事件。本文设计并实现了 12 个场景模拟事件 E_1,E_2,…, E_{12}（如表 1 所示），用来模拟应用程序运行过程中所需要的环境，从而尽可能地触发应用程序的网络行为。根据应用程序触发行为的不同，本文将这 12 个场景模拟事件分为以下 4 类。

表 1　　场景模拟事件

事件编号	事件名称	事件描述	事件类型
E_1	启动	启动应用程序	应用改变
E_2	点击	单击应用程序主界面	人机交互
E_3	休眠	应用程序处于休眠状态	人机交互
E_4	解锁	解锁 Android 设备	人机交互
E_5	电池	Android 设备电池状态改变	环境改变
E_6	网络	Android 设备网络状态改变	环境改变
E_7	位置	Android 设备位置信息改变	环境改变
E_8	SIM 卡	Android 设备更换 SIM 卡	环境改变
E_9	时间	Android 设备系统时间改变	环境改变
E_{10}	相册	Android 设备相册内容改变	环境改变
E_{11}	通讯录	Android 设备通讯录改变	环境改变
E_{12}	短信	Android 设备运行过程中，接收到外来短信	外来事件

1) 应用改变，启动应用程序。本文通过调用 adb shell am start 来启动一个应用程序。

2) 人机交互，用来模拟用户与应用程序之间的交互动作。典型的交互动作包括单击应用程序的开始按钮、触摸应用程序屏幕和切换应用程序界面。人机交互不仅能够触发应用程序的网络行为，而且

是确保流量收集器正常运行不可或缺的一个环节。如很多应用程序在第一次启动的过程中，会出现广告界面，必须人为地滑过或单击“开始使用”按钮才能够进入应用程序主界面。本文采用 monkey 和 monkeyrunner 这 2 款工具模拟用户的动作，monkey 能够随机产生一系列的用户动作，如点击屏幕、滑动界面。monkeyrunner 允许用户编写 python 脚本来模拟指定的用户动作。

3) 环境改变，指 Android 设备系统环境的改变。如 Wi-Fi 状态由关闭到开启，电池状态由电量不足切换到正在充电状态。

4) 外来事件，Android 设备运行过程中，接收外界发来的信息。如 Android 设备测试过程中接收到一条短信。

4.2 基于状态转移分析方法的测试序列构建

状态转移分析方法将攻击表示成一系列被监控的系统状态迁移。攻击模式的状态对应于系统状态，并具有迁移到另外状态的条件断言。通过弧线将连续的状态连接起来以表示状态改变所需要的触发条件，允许事件类型被植入到模型并且无需同审计记录一一对应。测试序列的优劣将在很大程度上影响到应用程序的运行测试能否成功，典型的触发事件测试序列需适应当前应用程序，因此，需要针对性地产生策略模拟场景事件。

本文通过图形化的表示方法对应用程序状态之间的转移进行描述。状态转移图是一个四维向量 $<V,T,W,\boldsymbol{M}>$。

1) V 是应用程序状态节点的集合，表示应用程序在整个测试过程中的不同状态，可以将每个状态 V 定义为一组网络行为属性的集合.

2) $T\in V\times V$ 是节点之间有向边的集合。有向边 T 代表促使应用程序状态转移的场景模拟事件，当且仅当场景模拟事件发生，才会发生状态转移。

3) W 是有向边权重的集合，它是动态变化的，与前序列的状态序列有关，其中，$W=\{w_{ij}\mid i,j\in V\}$。

4) $\boldsymbol{M}$ 是当前应用程序状态下对应的状态转移矩阵。它的行和列与 V 相对应，$\boldsymbol{M}(i,j)$表示从 V_i 到 V_j 的有向边的权重。

在构建状态转移图前，需要确定状态转移图中有向边的集合 T，因此，本文设计了场景模拟事件－权限映射表，具体描述如表 2 所示。在场景模拟事件－权限映射表中，场景模拟事件与权限之间是一对一或一对多的关系，即某一个场景模拟事件发生可能需要多种权限的组合。通过应用程序申请的权限信息提取出应用程序的场景模拟事件，从而确定有向边的集合 T，其提取流程如下。

1) 在应用程序的 AndroidManifest 文件中，提取出应用程序申请的权限信息，根据提取的权限信息从场景模拟事件—权限映射表中获取相应的场景模拟事件。

2) 根据获取的场景模拟事件在表 2 中查找需要的权限。如果该场景模拟事件所需要的所有权限都满足，则表明这个场景模拟事件将出现在状态转移图中；否则，就舍弃这个场景模拟事件。

表 2 场景模拟事件—权限映射

字段名	字段类型	字段描述
id	int	表项编号
PermissionId	int	权限编号
PermissionDescribe	varchar(50)	权限描述
EventId	int	场景模拟事件编号
EventWeight	float	场景模拟事件权重

在整个状态转移图中，S_0~S_4 表示的是当前状态，w_i 是边的权重，箭头表示的是场景模拟事件，通过箭头方向来生成测试序列。如图 3 所示，它表示的就是 Android 应用在 4 个场景模拟事件（启动、短信、位置、相册）下的状态转换。在整个状态转移图中，需要对图进行遍历，具体的遍历规则如下。

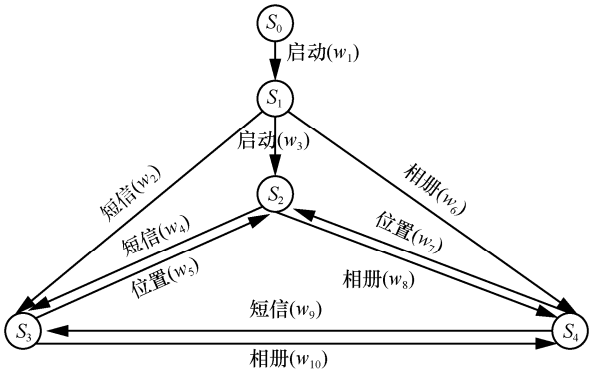


图 3 状态转移

1) 去除自回路。场景模拟事件并不是执行的次数越多测试就越完全，重复测试同一场景模拟事件不但意义不大，反而增大了测试的开销，对于这种自回路，应该先去除，后期单独处理。

2) 在一条测试序列中保证回路最多只出现一次。多个场景模拟事件构成的回路比较隐晦，不容易被发现，但它们的存在对生成测试序列并无益

处，对它们的执行最多能容忍一次。

3) 对产生自回路的场景模拟事件进行单独处理。对于自回路场景模拟事件，要对它们重复执行的情况进行测试，保证测试的正确性。

4) 从状态转移图的原点出发，根据场景模拟事件之间的约束关系，选择当前权重最大并且未被执行过的场景模拟事件执行，重复此过程，直到执行的场景模拟事件足够多。

4.3 测试序列的自动生成

在整个状态转移图中，存在着多条路径，如何在众多的路径中寻找到一条权重最大的路径，作为应用程序的测试序列，以便尽可能地来触发应用程序的恶意行为，是本文的创新点之一，也是难点之一。

本文首先获取每个应用程序的敏感权限，查询表 2，获取与应用程序相关联的场景模拟事件，并构建出应用程序状态转移图。根据状态转移图中边的权重不同，设计寻找路径算法。伪代码如下。

算法 1　测试序列生成算法

输入　状态转移图 *G*

输出　测试序列 *Sequence*

```
1)  #define MaxCount 5
2)  Struct G{
3)  int M[Max][Max];//状态转移矩阵
4)  int vertex;              //顶点
5)  int edge;                //有向边
6)  }
7)   FindRoad (graph G){
8)      if(NumOfErgodic(G.edge)> MaxCount){
9)      break;
10)     }
11)     else{
12)        NextEdge=MaxValue(G.edge);
13)        NextStartVertex=FindVertex(G.vertex);
14)            Update(G.M);
15)         FindRoad (G);
16)     }
17)           return Sequence;
18)  }
```

得到状态转移图后，测试序列的自动生成过程如算法 1 所示，具体解释如下。

第 1)~ 6)行对状态转移图进行定义以及初始化。

第 8)行和第 9)行统计已经遍历的有向边数量，如果已遍历边的数量超过了规定的最大值，结束对状态转移图遍历，返回结果，否则继续遍历。

第 12)行和第 13)行根据状态转移图中的状态转移矩阵，找到当前权重最大的边，并将该目标边的入度点作为下一次遍历开始的起始点。

第 14)行更新状态转移图的状态转移矩阵 $\boldsymbol{M}$，将已经遍历的边的权重变为 0。

第 15)行递归遍历过程，重复寻找当前状态转移图中权重最大的边，直到遍历结束。

第 17)行输出遍历权重最大的边的集合，作为应用程序的测试序列。

在应用程序状态转移的过程中，设应用程序的当前状态为 $S_n=\{Event_1,Event_2,Event_3,Event_4,Event_5\}$，状态转移矩阵为

$$\boldsymbol{M}=\begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & \ddots & \vdots \\ P_{n1} & \cdots & P_{nn} \end{bmatrix}$$

可以得到下一状态为 $S_{n+1}=S_n\times\boldsymbol{M}$，代入上述公式可得

$$S_{n+1}=\{Event_1,Event_2,Event_3,Event_4,Event_5\}\times\begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & \ddots & \vdots \\ P_{n1} & \cdots & P_{nn} \end{bmatrix}$$

在性能优化方面，为了避免每个顶点自成环对寻找过程的影响，本节将自成环的边的权重定义为 0，同时，为了避免某些场景模拟事件重复执行，本节在动态调整权重的过程中，采用了优先级调度策略，即已执行了的场景模拟事件权重将重置为 0，从而避免了某些场景模拟事件永远不被执行的情况发生。

4.4 网络行为特征提取

本文定义的网络行为指的是在应用程序运行过程中产生的网络流量行为。流量收集器将应用程序运行过程中产生的网络流量收集起来，并将收集到的结果保存为 pcap 文件。流量分析器通过 python 中的 pcap 模块来分析 pcap 文件，其中，pcap 模块主要获取的是 pcap 文件中 TCP/IP 层的信息。本文提取了 *S*、*R*、*C*、*A* 这 4 个特征来表示应用程序在模拟事件 *E* 下运行 30 s 的网络特征，其中，*S* 表示发送的字节数；*R* 表示接收的字节数；*C* 表示产生的连接数；*A* 表示泄露敏感信息的数量。此外，*S*、*R*、*C*、*A* 的取值为整数。本文定义了一系列的敏感信息，如表 3 所示。

表 3 Android 设备敏感信息

敏感信息内容	示例值
Android 设备 ID	3531505c0b421c4d
Android 设备型号	Huawei C8817l
Android 设备 IMEI 号	864036026855634
Android 设备 IMSI 号	310005123456789
操作系统版本	Android4.4
Android 设备位置	E118°46′N32°03′
Android 设备电话号码	15555215554

示例表示 Android 应用 com.tobyyaa. superbattery.apk 在点击事件 E_2 下的网络特征为

$$\{E_2,8\,543,5\,998,3,1\}$$

其中，E_2 是点击事件，该网络特征表明 Android 应用在点击事件下运行 30 s 后，产生了 3 个连接，向外界发送了 8 543 byte，同时，接收到了 5 998 byte，并且在与外界连接的过程中，泄露了 1 个敏感信息。本文先对 pcap 文件中的 HTTP 表单进行明文简析，再通过字符匹配来检测应用程序是否泄露了敏感信息。如图 4 所示，通过对 HTTP 表单中的 Android 设备电话号码进行匹配，能够匹配成功，说明应用程序泄露了 Android 设备电话号码这一敏感信息。

```
POST/a/p1 HTTP/1.1
Host: ade.woo.com.cn

Pit=1&so=16&ml=2&pn=15555215554&apn=com.t
Obyyaa.superbattery&mid=74HTTP/1.1 200 OK
Date: Wed, 28 Oct 2015 10:22:01 GMT
Connection: keep-alive
Content-Length:0
```

图 4 HTTP 表单

4.5 基于模式预测方法的行为收集

行为收集分为 2 个部分，一部分为收集应用程序运行过程中的场景模拟事件，另一部分为收集应用程序运行过程中产生的网络流量。

基于模式预测方法的前提条件是：事件序列不是随机发生的而是服从某种可辨别的模式，其特点是考虑了事件序列之间的相互联系[14]。Chen 等[15]给出一种基于时间的推理方法，利用时间规则识别用户正常行为模式的特征。通过归纳学习产生规则集，并动态地修改系统中的这些规则，使之具有较高的预测性、准确性和可信度。

本文模拟场景事件之间就存在着时间先后顺序，如启动这一事件必须发生在点击事件之前，才能触发应用程序的网络行为。本文对于每一个应用程序，其整个运行过程的网络行为特征定义为

$$\{E_x,S_x,R_x,C_x,A_x\},\{E_y,S_y,R_y,C_y,A_y\},\cdots$$

其中，E_x、E_y 分别表示启动和点击场景模拟事件，$\{S_x,R_x,C_x,A_x\},\{S_y,R_y,C_y,A_y\}$ 表示在相应的场景模拟事件下的网络行为。

在整个测试系列中，场景模拟事件之间存在着特定的逻辑关系，而且同一个场景模拟事件可以多次出现。如对于具有短信收发功能的应用，短信这一场景模拟事件就可以多次出现在其测试序列中。

4.6 基于 BP 神经网络的网络行为分析

BP（back propagation）神经网络是一种根据误差逆传播训练的多层前馈网络，是目前应用最广泛的神经网络模型之一。BP 神经网络能学习和存储大量的输入—输出模式映射关系，而无需事先揭示描述这种映射关系的数学方程。它的学习规则是使用梯度下降法，通过反向传播来不断调整网络的权值和阈值，使网络的误差平方和最小。BP 神经网络模型拓扑结构包括输入层（input layer）、隐层（hidden layer）和输出层（output layer）。输入层神经元的个数由样本属性的维度决定，输出层神经元的个数由样本分类个数决定。隐层的层数和每层的神经元个数由用户指定。本文将重构场景模拟事件 E 和提取的网络特征作为 BP 神经网络的输入层。由于本文定义了 12 个场景模拟事件，可以组合出无穷多个测试序列用于 BP 神经网络学习。

5 系统实现

整个测试过程都是通过 shell 以及 python 脚本语言来实现的，包括安装、启动、运行 Android 应用、收集应用在场景模拟事件触发下的网络流量，并对网络流量进行统计分析、数据建模。

5.1 客户端模块

本文设计的客户端模块的功能主要包括上传请求和信息查询。

5.1.1 上传请求

客户端通过 Web 应用来提交应用程序，并且告知服务器端自身所属的类别。其中，整个 Web 界面设计都是通过 php 脚本语言来实现的。服务器响应

客户端请求，同时查询行为信息库，判断当前应用程序是否已经在行为信息库中，如果存在，直接将应用程序的包名、大小和网络行为特征等信息反馈给客户端；否则，将应用程序的信息存入行为信息库中。其中，在行为信息库的设计中，分别设计了应用程序基本信息数据库、应用程序 API 调用数据库、应用程序权限使用数据库和应用程序测试序列数据库等。

5.1.2　信息查询

客户端通过一个 Web 应用来提交所要查询应用程序的 MD5 值或应用程序包名。服务器查询行为信息库将查询到的结果反馈给客户端。

5.2　服务器模块

服务器负责抓取、分发、运行、分析以及数据存储，后台需要多台 Android 设备支撑。其主要组成部分包括任务分发器、流量收集器、流量分析器和行为信息库。

5.2.1　任务分发器

任务分发器接收应用程序分析请求，并根据当前系统运行情况分发执行任务至流量收集器。本文在服务器端的 crontab 计划表中放置询问应用程序分析请求任务，如*/5 * * * * /home/testapk/request.sh 表示每隔 5 min 查询一次，一旦查询到新的分析请求，就通过 shell 脚本将分析请求分发给流量收集器。

5.2.2　流量收集器

流量收集器基于 Android 设备运行并收集应用程序的网络流量。流量收集器接收来自任务分发器下发的分析任务，并根据当前 Android 设备的运行情况，选择一台空闲的 Android 设备来运行应用程序。本文设计了一张 Android 设备运行情况数据库表 device，每当有新的分析任务时，通过 SQL 语言去查询库，来获取空闲的 Android 设备，如下所示。

select deviceid from device where status = “0”

获取空闲 Android 设备后，需要将应用程序在 Android 设备上运行测试，并收集其网络流量。收集应用程序网络流量可以总结为以下 6 个步骤。

1) 安装应用程序。
2) 开启 tcpdump。
3) 运行应用程序。
4) 各种场景模拟事件触发。
5) 归档收集应用程序的网络流量。
6) 停止卸载应用程序。

首先通过 adb install 命令将应用程序安装在 Android 设备上，然后在网络监听器上开启 tcpdump。tcpdump 是一款功能强大的网络数据采集分析工具，可以将网络中传送的数据分组完全截取下来。本文通过 am start 来启动应用程序，在程序的启动过程中，通过 monkey 来跳过刚开始的广告界面，其实现方法如下。

adb shell monkey –p package–throttle 1 000 –v 5

它表示启动指定的应用程序，并向其发送 1 000 个伪随机事件。通过对生成的 pcap 文件大小进行判断，一旦发现其大小为 0，就断定没有跳过应用启动时的广告界面，需要重新启动应用程序，再次使用 monkey 进行相同的操作，直到 pcap 文件大小非 0 为止。启动应用程序并跳过刚开始的广告界面后，针对性地产生策略场景模拟事件测试序列，尽可能地触发应用程序的网络流量。如一个新闻应用程序，一般需要先启动该程序，其才会与远程服务器建立起连接，并将一些文字和图片下载到本地，为了进一步获取到新闻的详细信息，必须单击新闻标题，进入新闻界面，只有这样才能够尽可能地触发应用程序的网络流量。对此类应用，本文通过 adb shell am start 来启动应用程序，通过 monkeyrunner 在屏幕的指定位置(x:250 y:250)单击，进入新闻内容界面，并收集应用程序的网络流量。

在归档收集网络流量的过程中，为了能够使应用程序的网络行为充分地显现出来，本文通过 tcpdump 监听应用程序在 5 种相对应的场景模拟事件下的网络流量，在每种场景模拟事件下运行 30 s，最终收集得到应用程序的网络流量，并将最终的结果保存为 pcap 文件。为了防止测试过程中用户的隐私（如电话号码、短信内容等）被泄露出去，本文在 Android 设备接入到互联网的过程中，设置了一道防火墙，对所有经过该防火墙流出的数据进行过滤，用来阻止数据泄露。

Android 设备运行完应用程序后，通过 adb uninstall 命令及时将应用程序从 Android 设备上卸载。卸载应用程序这一步骤有 2 个作用：1) Android 设备内存资源是十分有限的，及时卸载应用程序能避免因安装过多的应用程序造成 Android 设备运行效率降低的情况；2) 避免某些应用程序后台产生的网络流量对其他应用程序的检测结果造成影响。

5.2.3　流量分析器

流量分析器主要对流量收集器收集到的网络流量进行统计分析和数据建模，进而通过揭示并跟

踪可疑的应用程序网络行为，评估应用程序的恶意属性和风险程度。

流量分析器的工作流程如下。

1) 将已知分类的应用程序根据相对应的测试序列运行时间内抓取的网络流量文件（pcap 文件）存入行为信息库中以备查看和分析。

2) 分析 pcap 文件，提取 pcap 文件中的网络特征，作为机器学习的基础。

3) 借助 BP 神经网络的方法和工具，对已知分类的应用程序的网络特征进行训练、学习，生成训练数据集的 BP 神经网络模型。

4) 跟踪可疑的应用程序网络行为，评估应用程序的恶意属性和风险程度，并将结果存储到行为信息库中。

本文系统中采用 python 程序来完成 pcap 文件的统计分析和特征提取。最终得到的网络特征可以表示为

$$\{\{Event_1,S_1,R_1,C_1,A_1\},\{Event_2,S_2,R_2,C_2,A_2\},$$
$$\{Event_3,S_3,R_3,C_3,A_3\},\{Event_4,S_4,R_4,C_4,A_4\},$$
$$\{Event_5,S_5,R_5,C_5,A_5\},F\}$$

其中，S 表示发送的字节数，R 表示接收的字节数，C 表示产生的连接数，A 表示泄露敏感信息的数量，F 为应用程序自己定义的类别。当 F=0 时，表示该应用程序的类别是不确定的。图 5 给出了 3 条网络特征行为序列的例子，作为 BP 神经网络的输入示例。其中，$Event_1$~ $Event_5$ 列表示场景模拟事件的代号，最后一列 F 表示该网络特征行为所属的应用类别。其中，第 1 条序列属于新闻类，第 2 条序列属于视频类，其场景模拟事件组合与第 1 条序列相同，第 3 条序列属于恶意类，充分反映了不同场景模拟事件测试序列与应用程序网络特征行为之间的相关性。

第 1 条序列表示联网情况下启动（E_1）应用程序新浪新闻（sina.news），发送了 9 316 byte，接收了 82 891 byte，建立了 19 个连接，并且没有产生敏感数据泄露，接着模拟用户单击（E_2）新闻详情按钮，又产生一组网络行为数据，接下来更新 Android 设备的地理位置信息（E_7）、更改 Android 设备电池状态（E_5）、最后模拟接收到外界发送的短信（E_{12}），3 个场景模拟事件都未产生网络行为数据。第 2 条序列来自腾讯视频，场景模拟事件组合与第 1 条相同，这里就不再详细介绍。

第 3 条序列表示连网情况下用户启动恶意程序（t4t.power.management.malware），产生一组网络行为数据，接着关闭网络（E_6），模拟接收到外界的短信事件（E_{12}），都未产生网络行为数据，然后又打开了网络功能（E_6），模拟接收到外界的短信事件（E_{12}），产生了相应的网络行为数据。

对于确定类别的应用程序，本文将其测试得到的每一个测试序列结果作为 BP 神经网络的输入，通过训练与学习，生成训练后的 BP 神经网络模型，最后对客户端上传的未知类别的应用程序生成的特征属性结果进行检测，得出检测结果，并将结果反馈给客户端。

5.2.4 行为信息库

行为信息库用于对评估结果的管理和更新。评估结果的管理，可以通过应用程序的包名、应用程序的名称、MD5 值为关键字进行查询，确保通过应用程序的任意属性都可以查找到所需要的信息。行为信息库中，包含“应用程序的基本信息”、“应用程序的网络行为信息”和“应用程序检测结果”等信息。

6 实验和结果评估

6.1 实验设计

在实验过程中，选择华为 honor、虚拟机等多台 Android 设备进行测试，手机操作系统为 Android4.4，PC 端为一台 Xeron E5 CPU 和 4 GB 内存的计算机，系统环境为 Ubuntu14.04，用于存放和运行主要程序、下载并存放恶意应用程序、建立测试样本和测试结果数据库等核心内容。

本文实验采用 BP 神经网络模型进行应用程序的网络行为建模，模型由 1 个输入层、3 个隐层和 1 个输出层组成，其中，输入层的神经元个数为 25，每个隐层神经元个数为 20，输出层神经元个数为 5。

$Event_1$	S_1	R_1	C_1	A_1	$Event_2$	S_2	R_2	C_2	A_2	$Event_3$	S_3	R_3	C_3	A_3	$Event_4$	S_4	R_4	C_4	A_4	$Event_5$	S_5	R_5	C_5	A_5	F
1	9 316	82 891	19	0	2	3 055	60 976	5	0	7	0	0	0	0	5	0	0	0	0	12	0	0	0	0	新闻类
1	49 911	1 562 800	30	0	2	13 617	2 179 337	17	0	7	0	0	0	0	5	0	0	0	0	12	0	0	0	0	视频业
1	4 634	161 850	11	0	6	0	0	0	0	12	0	0	0	0	6	0	0	0	0	12	4 794	1 625	5	1	恶意类

图 5 网络特征行为序列

实验的输入为 1 条测试序列，包含连续的 5 个场景模拟事件与其触发的相应的网络行为特征，其中，每个场景模拟事件由 1 个触发事件与 4 个网络行为特征值组成，因此，输入层的神经元个数为 5×5=25。整个 BP 神经网络模型的架构如图 6 所示。

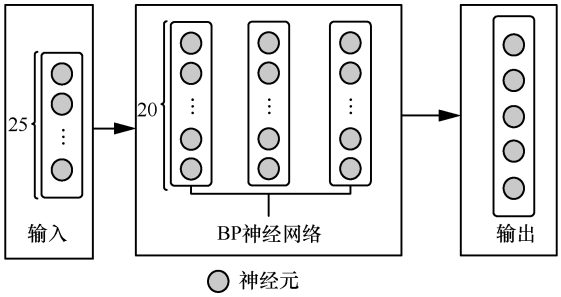


图 6　BP 神经网络模型架构

用于训练和测试的样本集合包括正常和恶意的测试样本，正常的样本从百度手机应用市场获取，包括新闻、购物、视频、云盘和工具 5 类共 568 个免费应用程序。恶意软件从 Contagio 等网站[16,17]获取，包括木马、病毒和后门等，共计 105 个样本。由于实验共计 5 类正常应用，因此，输出层神经元个数为 5。

为了确保分类模型的正确性，本文使用了 10 折交叉验证方法。首先将初始样本分割成 10 份子样本。在这 10 份子样本中，选择 1 份作为测试数据，剩下的 9 份作为训练数据。交叉验证重复 10 次，每个子样本验证 1 次。平均 10 次的结果，最终得到一个单一评估。

6.2　结果评估

本文系统不仅能够正常自动化运行，而且还能够对同一 Android 应用进行多次测试，抓取其网络流量，测试报告如图 7 所示。

测试报告

基本信息

应用程序名	com.tencent.news.apk
MD5值	303b5a960df4d6699089e25f6bad231a
应用程序大小	20.05 M
应用程序包名	com.tencent.news
应用程序类别	新闻

测试信息

测试1　测试2　测试3　测试4

开始时间	2016-07-25 15:45:15
结束时间	2016-07-25 15:47:45
Android设备	华为C88171
操作系统版本	Android4.4
测试序列	1->2->8->9->7
pcap文件	192.168.1.101/com.tencent.news.20160725_15.pcap
分析结果文件	192.168.1.101/com.tencent.news.20160725_15.csv

图 7　测试报告

为了检验系统的检测效果，本文采用准确率、误报率和漏报率这 3 个指标来衡量系统的效果。

表 4 是对 568 个正常样本的分类结果，表 5 是对 568 个正常样本和 105 个恶意样本集合的检测结果，其中，漏报率为 0.124，正确率为 0.897。

表 4　BP 神经网络分类结果

类别	正常样本个数	应用程序示例	正确率
新闻	161	新浪新闻	0.882
购物	105	淘宝	0.857
视频	170	爱奇艺视频	0.912
云盘	55	百度网盘	0.909
工具	77	antutu.powersaver	0.974

表 5　应用程序检测结果

正常样本	恶意样本	漏报率	误报率	正确率
568	105	0.124	0.099	0.897

从表 4 结果来看，本文基于 BP 神经网络的网络行为模型具有很强的代表性，能够区别各种类别的应用程序。通过将应用程序自定义类别与同类别的网络行为模型相对比，可以得出应用程序是否真的属于它自定义的那个类别，从而达到检测的目的。从表 5 的检测结果来看，本文系统能够对 Android 应用进行有效检测。

此外，学习率的设置对 BP 神经网络的收敛性有很大影响。学习率过小，误差波动小，但学习速度慢，往往由于训练时间的限制而得不到满意结果；学习率过大，学习速度加快，会引起网络出现摆动现象，导致训练不收敛，无法很好地拟合样本。因此，本文通过多次实验，调整学习率，来调整 BP 神经网络的收敛速度，如图 8 所示，不同的学习

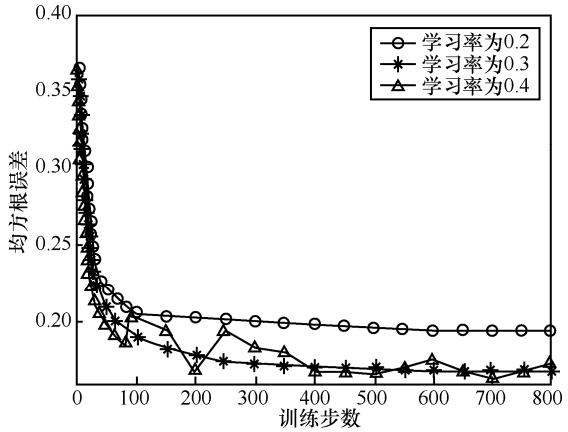


图 8　误差曲线

率对收敛速度有不同的影响。当学习率为 0.2 时，收敛速度较慢，而学习率为 0.4 时，收敛抖动严重。最终得到学习率在 0.3 以下整个模型的收敛性最好，训练步数在 400~600 之间即可较好地拟合样本特征。

6.3 性能评估

为了检测 DroidBet 系统的有效性，本文将提出的检测方法与近几年的相关工作进行了对比，具体对比情况如表 6 所示。

ComDroid[18]采用基于 Intent 的控制流，结合函数调用和组件调用，这种静态检测方法只能证明应用程序存在漏洞，但不能判断应用程序的恶意属性。Crowdroid[19]使用动态的方法对系统调用进行特征提取，并使用基于 *K*-means 聚类算法对应用程序进行检查，然而该工具只能够区分同一应用程序的多个变种是否为恶意程序。文献[20]中的工具动态提取应用程序运行过程中的 API 特征，并通过计算 API 的信息熵来对应用程序进行检测。AASandbox[21]只提出了一种动态提取应用程序系统调用特征的总体框架和思路，并没有给出具体的检测算法，也没有给出验证该思路正确的结果。在恶意行为触发方面，Crowdroid[19]模拟了固定的 6 种交互动作，而文献[20]和文献[21]单纯地使用 monkey 工具，随机模拟用户行为，难以有效触发恶意行为，最终导致部分恶意应用无法被检测出。

相比于以上 4 种检测工具，DroidBet 能够动态提取应用程序的网络特征，并构建基于 BP 神经网络的网络模型，对应用程序进行检测。在恶意行为触发方面，本文首先通过构建场景模拟事件库，用于对不同场景事件的模拟，以便尽可能地还原应用程序运行的真实环境，解决了 Jaesun 等[22]在论文中提到的对 SDK 版本、测试场景等条件严格要求的问题，并且根据场景模拟事件的权重自动生成测试序列。在检测过程中，可以同时运行 3 台 Android 设备，并发执行检测任务；检测系统运行速度快、吞吐量大、扩展性好，对每个移动应用的检测时间为 150 s，每小时可以对 72 个应用程序进行检测，而且可以根据实验需要自动地调整测试序列的长度。

7 结束语

本文描述了一个事件驱动的 Android 应用网络行为的自动检测系统 DroidBet。该系统基于应用程序的网络行为，通过场景模拟事件库触发，保证应用程序的网络行为完全显露出来。通过网络监听器自动化地收集应用程序网络行为，并对收集到的网络行为进行统计分析、数据建模，最终实现了对 Android 应用的检测。通过分析检测结果可知，本文系统检测效果良好。

本文下一步的工作将着重研究场景模拟事件库的记录与重放，同时，增加更多的训练数据和测试数据，以及监控应用程序内核层的行为。

参考文献：

[1] SORONCHO M F M, CHERUIYOT W, KIMANI S. Framework for vetting and identifying emulated Android mobile apps[J]. International Journal of Computer (IJC), 2016, 23(1): 35-41.

[2] 陈宏伟, 熊焰, 黄文超, 等. 基于关联分析的 Android 权限滥用攻击检测系统[J]. 计算机系统应用, 2016,25(4):36-42.
CHEN H W, XIONG Y, HUANG W C, et al. Association analysis based detection system for Android permission abuse attacks[J]. Computer Systems & Applications, 2016,25(4):36-42.

[3] BOWMAN-AMUAH M K. System, method and article of manufacture for security management in a development architecture framework: U.S. Patent 6,324,647[P]. 2001-11-27.

[4] TAYLOR V F, MARTINOVIC I. To update or not to update: insights from a two-year study of Android app evolution[C]//The 2017 ACM on Asia Conference on Computer and Communications Security. ACM, 2017: 45-57.

[5] 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(01):15-27.
YANG H, ZHANG Y Q, HU Y P, et al. A malware behavior detection system of android application based on multi-class features[J]. Chinese Journal of Computers, 2014, 37(01):15-27.

[6] SUI A F, GUO D F, GUO T, et al. Network behavior based mobile virus detection[C]//IEEE, International Conference on Communication

表 6 相关工作对比

检测工具	选取特征	动态/静态	算法	触发方式	测试过程	数据集
文献[18]中的 ComDroid	系统权限	静态	—	—	手动	100 个 APP
文献[19]中的 CrowDroid	系统调用特征	动态	*K*-means	特定交互触发	手动	2 类 APP
文献[20]中的工具	API	动态	—	monkey 随机触发	手动	1 600 个 APP
文献[21]中的 AASandBox	系统调用特征	动静结合	—	monkey 随机触发	手动	150 个 APP
DroidBet	网络特征	动态	BP 神经网络	测试序列触发	自动	673 个 APP

Technology. 2012:872-876.

[7] KIM J, YI K, et al. SCANDal: static analyzer for detecting privacy leaks in Android applications[J]. Mobile Security Technologies, 2012, 12.

[8] LU L, LI Z, WU Z, et al. CHEX: statically vetting Android apps for component hijacking vulnerabilities[C]//The 2012 ACM Conference on Computer and Communications Security. ACM, 2012: 229-240.

[9] FUCHS A P, CHAUDHURI A, FOSTER J S. Scandroid: automated security certification of Android[R]. 2009.

[10] SHABTAI A, KANONOV U, ELOVICI Y, et al. “Andromaly”: a behavioral malware detection framework for Android devices[J]. Journal of Intelligent Information Systems, 2012, 38(1): 161-190.

[11] BURGUERA I, ZURUTUZA U, NADJM-TEHRANI S. Crowdroid: behavior based malware detection system for Android[C]//The 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. ACM, 2011: 15-26.

[12] DINI G, MARTINELLI F, SARACINO A, et al. MADAM: a multi-level anomaly detector for Android malware[C]//International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security. Springer Berlin Heidelberg, 2012: 240-253.

[13] MAO C H, JENG A B, Lee H M, et al. Android malware detection via a latent network behavior analysis[C]// IEEE, International Conference on Trust, Security and Privacy in Computing and Communications. IEEE, 2012:1251-1258.

[14] 卿斯汉, 蒋建春, 马恒太,等. 入侵检测技术研究综述[J]. 通信学报, 2004, 25(7): 19-29.
QING S H, JIANG J C, MA H T, et al, Research on intrusion detection techniques：a survey[J]. Journal on Communications, 2004, 25(7): 19-29

[15] KRÜGEL C, TOTH T, KIRDA E. Service specific anomaly detection for network intrusion detection[C]//The 2002 ACM Symposium on Applied Computing. ACM, 2002: 201-208

[16] ZHOU Y, JIANG X. Dissecting Android malware: characterization and evolution[C]//Security and Privacy (SP), 2012 IEEE Symposium on. IEEE, 2012: 95-109.

[17] ALAZAB M, MOONSAMY V, BATTEN L, et al. Analysis of malicious and benign Android applications[C]//Distributed Computing Systems Workshops (ICDCSW), 32nd International Conference on IEEE, 2012: 608-616.

[18] CHIN E, FELT A P, GREENWOOD K, et al. Analyzing inter-application communication in Android[C]// International Conference on Mobile Systems, Applications, and Services. ACM, 2011: 239-252.

[19] BURGUERA I, ZURUTUZA U, NADJM-TEHRANI S. Crowdroid: behavior-based malware detection system for Android[C]//ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. ACM, 2011:15-26.

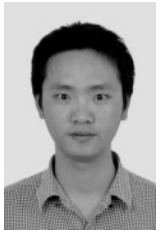
[20] 李舟军, 吴春明, 王啸. 基于沙盒的 Android 应用风险行为的分析与评估[J] 清华大学学报. 2016, 56(5): 453-460
LI Z J, WU C M, WANG X. Assessment of Android application's risk behavior based on a sandbox system[J]. Journal of Tsinghua University (Science and Technology), 2016, 56(5): 453-460.

[21] BLÄSING T, BATYUK L, SCHMIDT A D, et al. An Android application sandbox system for suspicious software detection[C]//International Conference on Malicious and Unwanted Software. IEEE, 2010:55-62.

[22] JANG J, YUN J, MOHAISEN A, et al. Detecting and classifying method based on similarity matching of Android malware behavior with profile[J]. SpringerPlus, 2016, 5(1).

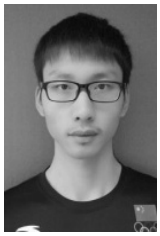
作者简介：

魏松杰（1977-），男，江苏南京人，南京理工大学副教授，主要研究方向为信息安全、无线网络与移动计算、智能计算与云计算等。



吴高翔（1990-），男，江西高安人，南京理工大学硕士生，主要研究方向为信息安全、移动互联网安全。

罗娜（1992-），女，福建福州人，南京理工大学硕士生，主要研究方向为信息安全、移动互联网安全。



时召伟（1992-），男，江苏南通人，南京理工大学硕士生，主要研究方向为信息安全、移动互联网安全。

周紫阳（1986-），男，江苏南京人，南京理工大学硕士生，主要研究方向为信息安全、移动互联网安全。