华中科技大学学报(自然科学版)
Journal of Huazhong University of Science and Technology(Natural Science Edition)
ISSN 1671-4512,CN 42-1658/N

《华中科技大学学报(自然科学版)》网络首发论文

题目：多层次特征的 Android 恶意程序静态检测方法
作者：刘晓建，雷倩，杜茜，刘柯宏
DOI：10.13245/j.hust.200215
收稿日期：2019-04-19
网络首发日期：2019-12-16
引用格式：刘晓建，雷倩，杜茜，刘柯宏. 多层次特征的 Android 恶意程序静态检测方法. 华中科技大学学报(自然科学版). https://doi.org/10.13245/j.hust.200215

网络首发：在编辑部工作流程中，稿件从录用到出版要经历录用定稿、排版定稿、整期汇编定稿等阶段。录用定稿指内容已经确定，且通过同行评议、主编终审同意刊用的稿件。排版定稿指录用定稿按照期刊特定版式（包括网络呈现版式）排版后的稿件，可暂不确定出版年、卷、期和页码。整期汇编定稿指出版年、卷、期、页码均已确定的印刷或数字出版的整期汇编稿件。录用定稿网络首发稿件内容必须符合《出版管理条例》和《期刊出版管理规定》的有关规定；学术研究成果具有创新性、科学性和先进性，符合编辑部对刊文的录用要求，不存在学术不端行为及其他侵权行为；稿件内容应基本符合国家有关书刊编辑、出版的技术标准，正确使用和统一规范语言文字、符号、数字、外文字母、法定计量单位及地图标注等。为确保录用定稿网络首发的严肃性，录用定稿一经发布，不得修改论文题目、作者、机构名称和学术内容，只可基于编辑规范进行少量文字的修改。

出版确认：纸质期刊编辑部通过与《中国学术期刊（光盘版）》电子杂志社有限公司签约，在《中国学术期刊（网络版）》出版传播平台上创办与纸质期刊内容一致的网络版，以单篇或整期出版形式，在印刷出版之前刊发论文的录用定稿、排版定稿、整期汇编定稿。因为《中国学术期刊（网络版）》是国家新闻出版广电总局批准的网络连续型出版物（ISSN 2096-4188，CN 11-6037/Z），所以签约期刊的网络版上网络首发论文视为正式出版。

网络首发时间：2019-12-16 16:05:16
网络首发地址：http://kns.cnki.net/kcms/detail/42.1658.N.20191216.1527.003.html

DOI：10.13245/j.hust.200215

多层次特征的 Android 恶意程序静态检测方法

刘晓建 雷 倩 杜 茜 刘柯宏

(西安科技大学计算机科学与技术学院 陕西 西安 710054)

摘要 提出一种基于多上下文特征的 Android 恶意程序检测方法，将敏感权限、广义敏感应用程序接口(API)和敏感系统广播三类敏感资源作为原始特征，并与其发生的上下文相结合形成程序特征，区分应用程序的良性和恶意行为．构造了基于回调函数的过程间控制流图，并定义了一组过滤压缩规则．该方法对 4 972 个应用程序进行检测分析，结果表明：随机森林算法在本文的特征集上表现效果最佳，准确率为 95.4%，召回率为 96.5%，本文方法比其他方法的检测效果更优．

关键词 恶意程序检测；静态分析；机器学习；多上下文特征；广义敏感 API

中图分类号 TP311.5 **文献标志码** A

Static detection approach for Android malware based on multi-level features

LIU Xiaojian LEI Qian DU Xi LIU Kehong

(School of Computer Science and Technology，Xi'an University of Science and Technology，Xi'an 710054，China)

Abstract The static approach to detecting Android malware was proposed based on multi-context features．Three raw features (the generalized sensitive application programming interfaces，permissions and system broadcasts) were combined with their contexts (callback methods，components and applications respectively) as program features to distinguish malware and benign applications．In order to extract the program features effectively，the inter-process control flow graphs of callbacks were constructed，and a set of filtering and compression rules was proposed．Our approach was evaluated on 4 972 samples．The comparison experiment shows that the random forest algorithm has the best performance on the selected feature set，with an accuracy rate of 95.4% and a recall rate of 96.5%，which is better than most of current methods．

Key words malware detection；static analysis；machine learning；multi-context features；generalized sensitive application programming interface (API)

由于 Android 平台的开放性和自身安全机制存在的漏洞，使其成为网络黑客的主要攻击对象，因此 Android 恶意程序检测成为信息安全领域须解决的重要问题．Android 恶意程序检测分为动态检测和静态检测．动态检测通过分析应用程序运行时的行为特征，对程序的安全性质作出判断[1-4]．静态检测不执行程序，直接对程序文本进行分析．静态分析具有代码覆盖率高，运行速度快，分析结果不受运行环境制约等优点．

机器学习已被广泛应用到恶意程序检测中[5-9]．机器学习检测方法的关键在于特征集的选择和特征数据的获取．权限和 API 是 Android 应用程序的两类重要特征，很多检测方法都以这两类特征作为判据[5-9]．但上述研究均以整个程序为粒度提取特征，丢失了特征发生的上下文信息及其所蕴含的重要语义信息，造成误报．

收稿日期 2019-04-19．

作者简介 刘晓建(1971-)，男，副教授，E-mail：780209965@qq.com．

基金项目 国家自然科学基金资助项目(61702408)；陕西省科技计划资助项目(2017JM6105)；教育部协同育人资助项目(2010918001)．

本文研究将 Android 恶意程序检测问题归结为二元分类问题，提出一种多层次特征的 Android 恶意程序静态检测方法．首先选取敏感系统广播、敏感权限和广义敏感 API 作为原始特征，并将原始特征与上下文结合，形成判断程序恶意性质的特征集.然后对特征进一步处理,并嵌入特征向量空间，采用随机森林机器学习算法设计和训练分类器．为了有效提取相关特征，构造了回调函数的过程间控制流图(iCFG)，并提出了对 iCFG 进行过滤、压缩的规则，得到 API 图，有效减小 iCFG 的规模，提高特征提取效率．最后通过对比实验，验证了方法的有效性.

1 特征选取

以广义敏感 API、敏感权限和敏感系统广播 3 类信息作为原始特征，并与上下文结合，形成判断程序恶意性质的一个特 $F =< feature, context\_set >$，其中，*feature* 为任意一个原始特征，*context_set* 为所有上下文集合.

1.1 原始特征

1.1.1 广义敏感 API

广义敏感 API 为所有对恶意行为检测有所贡献的 API,包含如下 4 种类型：**a.** 受权限保护的 API，此类 API 为 PScout[10]中给定的 API 列表；**b.** 与动态加载相关的 API，DexFile，ClassLoader，Method 和 Field[6]均包含此类 API；**c.** Source & Sink API，此类 API 预示信息的获取和泄露，也将其作为敏感 API，此类 API 可通过 SUSI[11]的 API 列表得到；**d.** 其他恶意应用常用 API.

1.1.2 敏感权限

本研究提取出恶意程序中使用最频繁的 20 个权限作为敏感权限，并与其在良性程序中发生的次数进行对比．结果表明：敏感权限在恶意程序与良性程序中的分布存在明显差异，敏感权限能够表征恶意程序的某些行为.

1.1.3 敏感系统广播

Android 恶意程序通常依赖系统广播来触发或启动恶意负载[12]，因此系统广播也可作为检测应用程序是否具有恶意性的重要依据．本研究采用文献[12]收集的 Android 恶意应用最感兴趣的 25 个系统广播作为敏感系统广播.

1.2 原始特征的上下文

恶意行为不仅与原始特征有关，还与发生的上下文有关．将原始特征和其上下文相结合能更好地表征程序的语义，提高检测的准确率，降低误报率．将上下文分为应用程序层、组件层和回调函数层,分别与敏感系统广播、敏感权限和广义敏感 API 相结合，共同组成程序的特征.

本研究将敏感权限与使用该权限的组件相结合,即把组件作为敏感权限的上下文，由于回调函数为 Android 程序执行的基本单位,因此将广义敏感 API 与调用它的回调函数相结合，能反映用户界面事件或系统事件所触发的广义敏感 API 调用.

值得注意的是，一个原始特征对应的上下文可能不止一个，一个敏感权限可能在多个组件中被使用，一个广义敏感 API 也可能在多个回调函数中被调用．因此一个原始特征对应一个上下文集合.

2 特征获取

本研究采用静态分析法，敏感系统广播及其上下文(应用程序)可通过 Manifest.xml 直接获取.敏感权限在 Manifest.xml 中定义的是整个应用程序使用的权限，并非每个组件使用的权限，因此必须通过程序分析得到每个敏感权限的上下文(组件).

2.1 特征获取过程

首先解析 apk 文件，获取反映程序中方法间调用关系的调用图(CG)和程序中所有方法(包括回调函数)的控制流图(CFG)；然后把调用图和控制流图相结合，构造每个回调函数的 iCFG；再对 iCFG 进行约简，仅保留广义敏感 API 节点和图的出入口节点，便可得到每个回调函数所调用的广义敏感 API 及其所在的上下文；最后利用 PScout 给出的 API 与权限的映射表，得到每个回调函数所使用的权限集，并根据回调函数所属的组件得到一个组件所使用的敏感权限集，即得到敏感权限及其上下文.

2.2 回调函数 iCFG 的构造

iCFG 是通过对程序调用图和每个类方法的控制流图进行综合形成的，调用图、控制流图的定义及 iCFG 的构造过程如下.

定义 1 程序的调用图定义为有向图，即 $CG =< N, S, E, start >$，其中，N 为程序中所有类方法的有限集 $\{m_1, m_2, \ldots, m_n\}$；$S$ 为方法调用点的有限集；$E \subseteq N \times S \times N$ 为带标记的有向边的有限集．每条边 $e = (m_i, s_k, m_j) \in E$ 为方法 m_i 在调用点 s_k 处调用了方法 m_j；$start \in N$ 为调用图唯一入口点.

定义 2 一个类方法 m 的控制流图定义为有向

图，即 $CFG=<N, C, E, head, tail>$，其中，$N$ 为一个类方法内程序语句的有限集 $\{s_1, s_2,\ldots, s_n\}$．程序语句包括赋值、条件判断、函数调用、调用返回等；C 为关于程序变量的一阶谓词公式的有限集，表示程序控制流转移的条件；$E\subseteq N\times C\times N$ 为边的有限集．每条边 $e=(s_i, c, s_j)\in E$ 表示控制流从节点 s_i 转移到节点 s_j，c 为控制流转移的条件，对于无条件控制流转移，可认为 c 恒为真；$head\in N$ 为唯一的入口节点；$tail\in N$ 为唯一的出口节点．

可进一步定义一个节点的前驱节点集合 Pred 和后继节点集合 Succ 分别为

$$\forall n\in N,\ \mathrm{Pred}(n)=\{n'\mid \exists c\in C\bullet(n', c, n)\in E\}$$

$$\forall n\in N,\ \mathrm{Succ}(n)=\{n'\mid \exists c\in C\bullet(n, c, n')\in E\}$$

得到程序调用图和类方法的控制流图后，将把二者综合起来形成一个类方法的过程间控制流图 iCFG．由于特征提取主要关注回调函数，因此须构造回调函数的 iCFG．首先利用 CG 获取类方法之间的调用关系，然后通过两个特定边(Call Edge 和 Return Edge)将每个调用点连接到被调用者的 CFG 上，Call Edge 将调用点连接到被调函数的入口点，Return Edge 将函数的出口点连接到紧跟在调用点之后的程序点．可认为 iCFG 为包含若干 Call Edge 和 Return Edge 的 CFG．

2.3　回调函数的 API 图

iCFG 的 API 图为包含该 iCFG 中的所有广义敏感 API 调用节点及出入口节点的无环子图．生成 API 图包括无关节点过滤和环的压缩两个步骤．由于本研究只关注应用调用的广义敏感 API，因此须进行无关节点过滤．为了在过滤中保持图结构的完整性，定义了图过滤规则．

定义 3　设 G 为 iCFG 的有限集，过滤规则 Rule 为图变换函数，$Rule: G\to G$．对于任一 $<N, C, E, head, tail>\in G$，$Rule$ 将其映射为 iCFG′，即

$$Rule(<N, C, E, head, tail>)=<N', C', E', head', tail'>$$

其中，$N'=N-\bar{N}$，$\bar{N}$ 为无关节点的集合；$E'=\{(s_i, c'_k, s_j)\in N'\times C'\times N'\mid (s_i, c_k, s_j)\in E\}\bigcup\{(s_i, c'_k, s_j)\in N'\times C'\times N'\mid \exists s_1, s_2,\ldots s_p\bullet(s_i, c_1, s_1)\in E\wedge(s_1, c_2, s_2)\in E\wedge\ldots\wedge(s_p, c_{p-1}, s_j)\in E\wedge c'_k=c_1\wedge c_2\wedge\cdots\wedge c_{p-1}\}$；$C'=\{c\mid \exists s_i, s_j\in N'\bullet(s_i, c, s_j)\in E'\}$；$head'=head$；$tail'=tail$．

Rule 的实现过程如图 1 所示，图中 m 为无关节点，A_1～A_3 为其余节点．

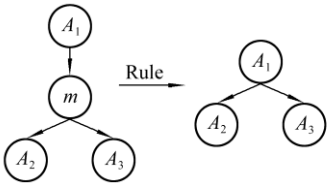

图 1　Rule 的实现过程

由于 iCFG 中自然循环的压缩须满足 iCFG 可归约[13]．因此采用文献[13]的环识别算法对自然循环进行识别和压缩．

2.4　获取广义敏感 API 和敏感权限特征

基于 API 图的特征提取过程如下．

输入　API-graphs//各回调函数对应的 API 图
API-Permission//PScout 的 API-敏感权限映射表

输出　<广义敏感 API，上下文集合>集合
<敏感权限，上下文集合>集合

```
foreach callback in API-graphs.keySet()
  nodes=dfn(APIs．get(callback))；//遍历 API 图，获取敏感 API 集合
  //提取 API 特征及其上下文
  foreach node in nodes except head and tail
    将 callback 加入 node 对应的 API 的上下文集合；
  end
  //提取权限特征及其上下文
  foreach node in nodes
    if(node 是受敏感权限保护的 API)
      权限=API-Permission．get(node)；//获取 node 对应的权限
      组件名=callback 所属的组件//获取回调函数所属的组件
      将组件名加入到权限的上下文集合中；
    end
  end
end
```

通过 Manifest.xml 文件中每个广播组件的<intent-filter>标签，提取<action android：name>子标签的内容获取系统监听的广播，从而提取出敏感系统广播特征．

3　特征变换

为了便于机器学习方法进行恶意程序检测，须将提取出的特征编码为特征向量，使用特征向量表示程序样本．

3.1　特征向量的定义

将特征向量定义为包含 1 197 个元素的数组．特

征向量中每个元素的序号 $i(1\le i\le 1197)$对应一个固定的原始特征，该元素的值 Vector[i]表示由该原始特征构成的程序特征的量化值.

特征向量中包含 1 152 个广义敏感 API 特征值(权限相关 API 814 个，动态加载相关 API 14 个，Source&Sink API 297 个，恶意应用常用 API 27 个)、20 个权限特征值和 25 个系统广播特征值.

设 i 对应的原始特征为 $feature'$，构成的特征为 $<feature', context\_set'>$，Vector[i] 由函数 $f(<feature', context_set'>)$得到，即 Vector[$i$]=$f(<feature', context\_set'>)$，其中将一个特征经过特征变换函数 f 变换后得到的值作为该特征的特征值.

3.2 特征值的计算

3.2.1 广义敏感 API 的特征值

根据敏感 API 的上下文计算其特征值. 良性应用程序通常在用户界面(UI)操作中调用敏感 API，恶意应用程序通常在系统后台，即 UI 无关的操作中调用敏感 API，因此将其上下文分为 UI 相关和 UI 无关能更好地区分广义敏感 API 的调用特征. 其特征变换函数定义如下.

定义 4 设 api 为广义敏感 API，构成的程序特征为<api，$context\_set$>，则其特征值由下式得到，

$$f(<api,\ context\_set>)=\sum_{k=1}^{|UI|}w_{11}+\sum_{k=1}^{|Non\_UI|}w_{12}$$

式中，UI 和 Non_UI 为集合 $context\_set$ 中与界面相关和无关的回调函数子集，且满足 $UI\cap Non\_UI=\phi,\ context\_set=UI\cup Non\_UI$. |UI| 和 |Non_UI|分别为两个集合的元素个数. $w_{1i}(1\le i\le 2)$为回调函数的权值. 其中 w_{11} 为界面相关回调函数的权值，w_{12} 为界面无关回调函数的权值.

由统计方法计算 w_{1i}. 以 300 个恶意应用和 300 个正常应用为样本，分别统计在 UI 相关和 UI 不相关的回调函数中广义敏感 API 的平均调用频次，统计结果如表 1 所示.

表 1 不同上下文中 API 的平均调用频次

UI 相关性	良性程序/个	恶意程/个
UI 相关	18.5	13.6
UI 不相关	9.3	29.6

根据 API 特征在不同上下文中出现频次的差异性为其设置权值. 差异性公式未

$$\varphi(p)=\frac{C_m^2(p)}{C_n(p)}=\frac{C_m(p)*C_m(p)}{C_n(p)} \qquad (1)$$

式中：$C_m(p)$为恶意应用中上下文为 p 的 API 特征出现的频次；$C_n(p)$为正常应用中上下文为 p 的 API 特征出现的频次.

表 2 为不同 API 上下文对应的 $\varphi(p)$值，对 $\varphi(p)$进行归一化处理，处理后的结果为 $\varphi'(p)$. $\varphi'(p)$为每类上下文中 API 特征对应的权值.

表 2 不同 API 上下文对应的 $\varphi(p)$ 值

上下文	UI 相关	UI 不相关
$\varphi(p)$	9.9	94.2
$\varphi'(p)$	0.1	0.9

3.2.2 敏感权限的特征值

根据敏感权限的上下文计算其特征值，由于 Android 应用程序由四大组件组成，每一类组件都有其使用场景，在不同组件中使用敏感权限能够反映出程序的不同意图，因此将其上下文(即组件)按照组件类型分为 Activity、Service、Receiver 和 Provider，特征变换函数定义如下.

定义 5 设 $permission$ 为敏感权限，其构成的程序特征为<$permission$，$context\_set$>，特征值通过如下式得到，

$$f(<permission,\ context\_set>)=\sum_{k=1}^{|Activity\_set|}w_{21}+\sum_{k=1}^{|Service\_set|}w_{22}+\sum_{k=1}^{|Receiver\_set|}w_{23}+\sum_{k=1}^{|Provider\_set|}w_{24}$$

其中，$w_{2i}(1\le i\le 4)$为每类组件类型的权值；Activity_set，Service_set，Receiver_set 和 Provider_set 分别为 $context\_set$ 中对应构成的互斥子集.

w_{2i} 通过统计方法求得. 在恶意样本和正常样本中以组件为粒度对 20 个敏感权限出现的平均频次做了统计，统计结果如表 3 所示. 由式(1)得到差异值 $\varphi(p)$，进行归一化处理得到 $\varphi'(p)$，如表 4 所示. 同样将 $\varphi'(p)$ 作为每类组件中权限特征对应的权值.

表 3 以组件为粒度的权限平均频次

组件	良性程序/个	恶意程序/个
Activity	4.142	6.147
Service	0.204	1.879
Receiver	0.283	2.954
Provider	0.042	0.290

表 4 组件对应的 $\varphi(p)$ 值

组件名	Activity	Service	Receiver	Provider
$\varphi(p)$	9	17	30	2
$\varphi'(p)$	0.15	0.30	0.52	0.03

3.2.3 敏感系统广播的权值设定

由于敏感系统广播的上下文为应用程序，本文对该类特征赋予的权值为 0 或 1. 即在一个待检测的应用程序中，若某敏感系统广播被该应用程序注册，则为其赋予权值 1，反之为 0.

综上所述，特征向量 Vector 若不存在某特征，

则特征向量的相应特征元素值为 0，否则特征元素值由下式计算

$$\begin{cases} a_i = \sum_{k=1}^{|UI|} 0.1 + \sum_{k=1}^{|Non\_UI|} 0.9 & 1 \le i \le 1152 \\ p_i = \sum_{k=1}^{|Activity集合|} 0.15 + \sum_{k=1}^{|Service集合|} 0.3 + \sum_{k=1}^{|Receiver集合|} 0.52 + \sum_{k=1}^{|Provider集合|} 0.03 & 1 \le i \le 20 \\ c_i = \begin{cases} 1 & c_i \text{ 广播被注册} \\ 0 & c_i \text{ 未广播被注册} \end{cases} & 1 \le i \le 25 \end{cases}$$

以恶意程序 Opfake 为例，其特征向量为：Vector={0，…，1.8，…，2，…，0.1，…，，0.9，…2.7，…，2.7，…，3.6，…，1.8，…，0.9，…，0.15，…，0.3，…，0.52，…，1，…，0}，省略号部分都为 0.

4　实现方法

本文通过调用 Soot 框架提供的获取 CG 和 CFG 的 API，然后再进行 iCFG 的合成、图变换、特征提取及分类器训练，具体实现流程如图 2 所示.

在特征提取阶段，从 API 图直接获取的是 API 调用及其发生的回调函数，因此需将回调函数进一步归类为 UI 相关和不相关. Android 官方文档提示，所有与 UI 相关的回调函数都在 android.view 类及其子类中定义，因此可通过判断回调函数的类名从而判断其 UI 相关性. 从 Android 官方文档中，共统计了 76 个与 UI 相关的回调函数所属的类.

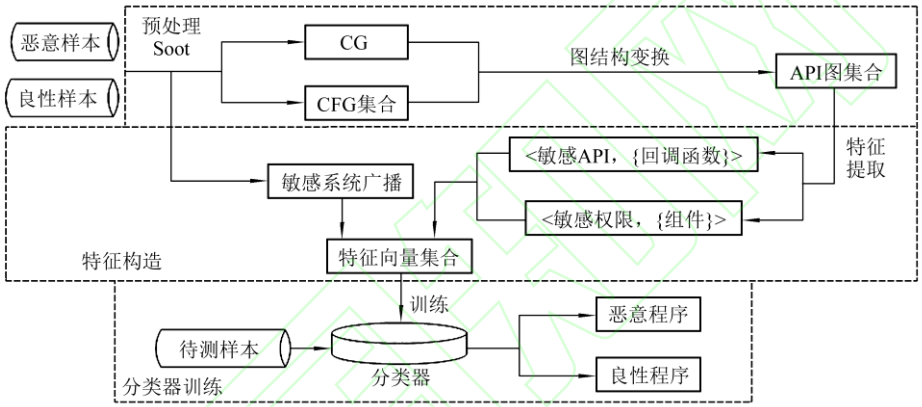

图 2　实现流程图

应用程序的所有 API 调用均以直接或间接的方式由 Android 组件的回调函数或生命周期函数完成，可通过回调函数所属的组件判断权限发生的组件.

5　实验与结果分析

5.1　数据集和度量指标

恶意样本集选自 Drebin 样本库、Virus Share 样本库及 DroidBench 测试集. 所选恶意样本集覆盖了各样本库中的所有恶意家族. 良性样本来自 Google Play，并预先通过检测以确保是良性应用. 表 5 为样本集的属性信息. 将样本集的 75%作为训练集，25%作为测试集. 每个样本的处理时间不超过 3 分钟.

表 5　样本集的属性信息

程序类别	样本库	样本数	家族分类
恶意程序	Drebin	1 358	179
	virusShare	1 160	80
	FlowDroid	142	未分类
良性程序	Google play	2 312	26

5.2　分类器训练

将本文特征集应用于朴素贝叶斯(NB)、K 邻近算法(KNN)、随机森林(RF)、逻辑回归(LR)、梯度提升树(GDBT)分类器进行训练和测试，不同分类算法的检测效果如表 6 所示. 可知：本文特征集在 RF 分类算法上的效果最好，召回率为 96.5%、准确率为 95.4%，误报率为 5.6%. 在 GBDT 算法上的效果最差.

表 6　不同分类算法的检测效果　　%

分类器	NB	KNN	RF	LR	GBDT
Accuracy	89.2	93.8	95.4	93.1	88.4
FPR	12.5	7.9	5.6	9.1	13.1
Recall	90.7	95.2	96.5	95	89.7
Precision	89.2	93.2	95.1	92.3	88.7

5.3　特征集的有效性

为了验证自定义的广义敏感 API 合乎逻辑，将敏感 API 分为 A1，A2，A3，A4. 其中 A1 为受权限保护的 API 集合，A2 为 Sources&sinks 函数集合，A3 为动态加载相关函数集合，A4 为恶意应用常用函数集合. G=A1∪A2∪A3∪A4. 分别以 G，A1，A1∪A2，A1∪A2∪A3 作为 API 特征，权限特征和

系统广播特征不变．将新的特征集应用于随机森林算法．不同 API 特征集的检测效果如表 7 所示．结果表明广义敏感 API 合理，可有效提高分类效果．

表 7　不同 API 特征集的检测效果

API 特征	分类效果/%		
	准确率	召回率	精确率
G	95.4	96.5	95.1
A1+A2+A3	93.1	93.9	92.4
A1+A2	90.7	92.1	89.3
A1	86.9	89.1	85.0

为了验证申请权限集合、基于应用程序粒度的实际使用权限集合和基于组件粒度的实际使用权限集合这三类特征集对检测效果的影响，使用 AAPT 工具从 Manifest.xml 文件中提取应用申请的权限集合，记为申请权限，然后基于应用程序粒度，提取整个应用使用的权限集合，记为粗粒度权限，将基于组件粒度的实际使用权限集合记为细粒度权限，广义敏感 API 特征和系统广播特征不变，将新的两类特征集应用于随机森林算法，其构造的分类器的准确率、召回率以及精确率如表 8 所示．结果表明从源码中提取的权限集合要比直接从 Manifest．xml 文件提取权限信息准确得多，而且对权限集合进行的细粒度划分也能有效提高分类器的性能．

表 8　不同权限集合在 RF 下的检测效果

权限	分类效果/%		
	准确率	召回率	精确率
细粒度	95.4	96.5	95.1
粗粒度	93.5	94.2	92.8
申请	89.0	90.7	88.1

为了验证系统广播对于 Android 恶意程序检测的有效性，将不包含系统广播的特征集应用于随机森林算法，并与原实验结果进行对比，结果如表 9 所示．结果表明：包含系统广播特征的特征集训练出的分类器效果更好，因此验证了该特征的有效性．

表 9　系统广播特征的有效性验证

广播特征	分类效果/%		
	准确率	召回率	精确率
包含	95.4	96.5	95.1
不包含	94.4	95.6	93.1

5.4　与相关工作对比

将提出的方法与相关工作进行对比，结果如表 10 所示．由表 10 可知：本文方法的准确率仅低于文献[5]方法，但本文方法的召回率比 DroidMat 文献[5]方法高 9.1%．

表 10　与现有的其他工作对比

检测工具	准确率/%	召回率/%
文献[5]	97.8	87.4
文献[8]	93.9	95.9
文献[7]	94.2	87.4
文献[9]	90.0	85.0
本文方法	95.4	96.5

VirusTotal 收集 69 种恶意软件检测工具，将数据集应用于该网站，采用检测工具对样本进行分类．结果表明本文检测效果比网站中的大部分工具效果好．本文分类器与 6 种检测工具的分类效果对比见表 11．

表 11　本文分类器与 6 种检测工具的分类效果对比

检测工具	分类效果/%	
	准确率	召回率
本文方法	95.4	96.5
Alibaba	87.6	83.1
AVG	87.4	99.0
Tencent	85.1	84.1
Microsoft	74.6	99.7
Baidu	63.2	60.5
Kingsoft	51.6	53.5

6　结语

本研究基于 Soot 提出一种多层次特征的 Android 恶意程序静态检测方法．实现了基于回调函数粒度的 iCFG 的构造及在 iCFG 上的无关节点过滤和环的压缩，生成 API 图，提高了特征提取效率；在 API 图上进行以回调函数和组件为单位的广义敏感 API 和权限的提取，构造了效果较好的分类器．准确率为 95.4%，召回率为 96.5%．

参　考　文　献

[1] ENCK W，GILBERT P，HAN S，et al．TaintDroid：an information-flow tracking system for realtime privacy monitoring on smartphones[J]．ACM Transactions on Computer Systems，2014，32(2)：99-106．

[2] 张家旺，李燕伟．基于 N-gram 算法的恶意程序检测系统研究与设计[J]．信息网络安全，2016(8)：74-80．

[3] 陈铁明，徐志威．基于 API 调用序列的 Android 恶意代码检测方法研究[J]．浙江工业大学学报，2018，46(2)：148-154．

[4] 荣俸萍，方勇．MACSPMD：基于恶意 API 调用序列模式挖掘的恶意代码检测[J]．计算机科学，2018，45(5)：132-138．

[5] WU D J，MAO C H，WEI T E，et al．DroidMat：android malware detection through manifest and API calls tracing[C]// Information Security．Tokyo：IEEE，2012：62-69．

[6] XIE N，ZENG F，QIN X，et al．Repassdroid：automatic detection of android malware based on essential permissions and semantic features of sensitive APIs[C]// Proc of 2018 International Symposium on Theoretical Aspects of Software Engineering (TASE)，Guangzhou：IEEE，2018：52-59．

[7] 杨欢，张玉清，胡玉璞，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)：16-26．

[8] DANIEL A，MICHAEL S，MALTE H，et al．Drebin：effective and explainable detection of android malware in your pocket[C]// Proc of 2014 Network and distributed system security symposium，San Diego：2014：201-215．

[9] 邵舒迪，虞慧群，范贵生．基于权限和 API 特征结合的 Android 恶意软件检测方法[J]．计算机科学，2017，44(4)：135-139．

[10] YEE A U，Kathy W．Pscout：analyzing the android permission specification[C]// Proc of 2012 ACM Conference on Computer & Communications Security．Raleigh：ACM，2012：217-228．

[11] RASTHOFER S，STEVEN A，ERIC B，et al．A machine-learning approach for classifying and categorizing android sources and sinks[C]// Proc of 2014 Network and distributed system security symposium．San Diego：2014：1-15．

[12] ZHOU Y J，JIANG X X．Dissecting Android Malware：Characterization and Evolution[C]// IEEE Symposium on Security and Privacy．San Francisco：IEEE，2012：95-109．

[13] WEI T，MAO J．A New Algorithm for Identifying Loops in Decompilation[C]// Static Analysis Symposium．New York：Springer，2007：170-183．