基于机器学习算法的 Android 恶意程序检测系统

张家旺，李燕伟

(国家计算机网络应急技术处理协调中心 实验室，北京 100029)

摘　要：对于传统的恶意程序检测方法存在的缺点，针对将数据挖掘和机器学习算法被应用在未知恶意程序的检测方法进行研究。当前使用单一特征的机器学习算法无法充分发挥其数据处理能力，检测效果不佳。文中将语音识别模型与随机森林算法相结合，首次提出了综和 APK 文件多类特征统一建立 N-gram 模型，并应用随机森林算法用于未知恶意程序检测。首先，采用多种方式提取可以反映 Android 恶意程序行为的 3 类特征，包括敏感权限、DVM 函数调用序列以及 OpCodes 特征；然后，针对每类特征建立 N-gram 模型，每个模型可以独立评判恶意程序行为；最后，3 类特征模型统一加入随机森林算法进行学习，从而对 Android 程序进行检测。基于该方法实现了 Android 恶意程序检测系统，并对 811 个非恶意程序及 826 个恶意程序进行检测，准确率较高。综合各个评价指标，与其他相关工作对比，实验结果表明该系统在恶意程序检测准确率和有效性上表现更优。

关键词：随机森林；恶意代码检测；多类特征；安卓应用；机器学习

中图分类号：TP

Malware detection system implementation of Android application based on machine learning

Zhang Jiawang, Li Yanwei

(Lab of National Computer Network Emergency Response Technical Team Coordination Center of China, Beijing 100029, China)

Abstract: For the weakness of traditional malware detection methods, this paper proposed a method in the detection of unknown malicious applications based data mining and machine learning algorithm. While a single feature of machine learning algorithms could not play the role of ability of data processing, detection effect. A method to combine speech recognition model with random forest algorithm was first proposed, which considered multi-class APK features in unknown malware detection. First, it combined a variety of ways to extract 3 classes which could reflect the behaviors of Android malware including sensitive permissions, DVM function calls and OpCodes characteristics; then, according to the characteristics of each type of N-gram model, each one could evaluate behaviors of malware independently; finally, 3 classes of feature model would join into a random forest learning algorithm, so as to detect the Android apps. It implemented an automated system based on this method to detect 811 non-malicious and 826 malicious apps with higher accuracy. Considering comprehensive evaluation of various indicators, the experimental results show that the malware detection system has a better performance than other related works on effective and accuracy.

Key Words: random forest; malicious code detection; multiple feature; android application; machine learning

0　引言

近年来，Android 系统手机使用率迅速增加，在给人们带来方便、快捷的同时也不断增加风险。由于 Android 系统的开放性，也成为了众多黑客攻击的对象。根据阿里移动安全的分析统计[1]，2015 年移动恶意代码数量和用户感染量虽然呈现出一定程度的下降趋势，但全年仍有 18%的设备感染过病毒，且病毒木马攻击手法的专业性较去年有了明显的提升。

由于恶意代码数量和种类的增加，加之技术的不断更新，使得基于签名和行为分析的典型分析方法失去了一定的时效性。由于签名方法需要事先获得恶意代码样本，然后将签名信息应用于其他程序的检查，因此无法检查未知恶意代码。另外基于行为的检测方法，虽然绕过了静态分析时的代码混肴和加密，但无法覆盖程序的全部有效路径，导致功能分析时不够全面。综合 Android 平台下传统恶意代码检测方法的缺点，本文提出一种基于多类行为特征并建立 N-gram 模型，再通过机器学

作者简介：张家旺（1981-），男，工程师，硕士研究生，北京人，主要研究方向为逆向分析、恶意软件分析、漏洞发现与利用（zhangjw@cert．org.cn）；李燕伟（1983-），男，河北人，博士研究生，工程师，主要研究方向为恶意代码分析，漏洞发现与利用．

习算法进行训练学习来检测恶意代码的方法，并以此方法为基础建立了一套检测系统。该系统采用多种方式提取多个 Android 应用程序的特征向量，并建立每类特征的 N-gram 模型，然后通过机器学习算法来进行训练，最后对大量未知应用程序进行检测来获得实际检测效果。

1 研究现状

恶意软件是Android系统面临的巨大威胁，由于恶意软件不断使用新技术、新方法，使得人们不断将各种算法和技术应用于Android恶意代码检测中。跟pc系统一样，最先出现的是基于签名和特征码[2]的检测方法，缺点是无法检测未知恶意代码。随着基于签名和特征码的静态检测方法的失效，越来越多的研究人员开始研究基于动态行为特征的检测方法[3]。

最近几年大数据相关技术不断成熟，也有研究者开始将数据挖掘和机器学习的方法引入到恶意代码检测中。在判别基于windows系统的恶意软件中，Ye等[4]使用API调用作为特征，数据集含有12214个良性程序以及1736个恶意软件，从API调用中提取出特征，在此基础上使用数据挖掘技术。李盟等[5]提出了一种恶意代码特征选取和建模的方法。

同样，针对 Android 恶意程序可以借助机器学习方法进行分类和检测[6][7][8][9][10]，如杨欢等设计了一套基于多个特征并采用三层混合系综算法的检测系统；Brandon Amos 等提出了 STREAM 框架，加速了 Android 恶意软件的机器学习分类的快速大规模验证；Sahs J 等提出了基于机器学习的检测系统，提取大量特征并采用离线的方式训练一类支持向量机。使用机器学习方法构成的分类器可以对 Android 应用行为进行模拟，区分应用是否为恶意软件。分类器通过分析应用的静态或动态行为特征做出判断，静态特征可以通过反编译 Android Dalvik 字节码获得，动态特征则通过监控程序运行时的行为来获取。通过提取申请的权限、反编译的代码或运行的行为，可以获取到各种不同类型的特征。常用的机器学习方法包括 k-Means、逻辑回归(logistic regression)、K-最近邻(K-nearest-neighbor，简称 KNN)、直方图(histogram)、决策树(decision tree)、贝叶斯网络(Bayesian network)、朴素贝叶斯(naïve Bayes)、随机树(random tree)、随机森林(random forest)、支持向量机(support vector machine，简称 SVM)等。采用哪一种机器学习算法、提取哪些特征以及如何提取的问题，并没有一个完美的解决方案。本文提取了权限、smali API 调用以及 native 代码 OpCodes 作为特征，采用随机森林算法构建分类器，取得了较好的分类与检测效果。

2 关键算法与技术

2.1 N-gram 模型

N-gram 是自然语言处理领域的概念，是大词汇连续语音识别中常用的一种语言模型。早期的语音识别技术和统计语言模型与它密不可分该模型基于这样一种假设，第 N 个词的出现只与前面 N-1 个词相关，而与其它任何词都不相关，整句的概率就是各个词出现概率的乘积。这些概率可以通过直接从语料中统计 N 个词同时出现的次数得到，常用的是二元的 Bi-Gram 和三元的 Tri-Gram。

该模型最早应用于恶意代码识别的想法最早由 Tony 等人[11]在 2004 年的论文中提出，他们的方法是基于 ByteCode 的。2008 年 Moskovitch 等人[12]提出利用 OpCode 代替 ByteCode，更加科学且检测效果更好。

2.2 特征向量选取

1）权限特征

Android 系统具有非常严格的权限管理机制，当用户使用系统中的某个功能或访问某些敏感数据时都要申请某个使用权限。一般应用程序在不使用短信功能时是不会申请短信发送、短信读写的权限。但恶意软件则不同，由于某些动机的存在，例如吸费类恶意应用会申请 SEND_SMS 权限发送短信。由于某些正常的程序也会申请短信功能使用权限，因此仅仅根据应用程序申请的权限判断一个应用程序是否存在恶意行为是不够的。但根据分析大量的恶意程序样本发现，存在恶意行为的应用往往会使用大量敏感权限，其中使用频率较高的包括：INTERNET 、 READ_PHONE_STATE 、 SEND_SMS 、 WRITE_EXTERNAL_STORAGE 、 READ_SMS 、 ACCESS_NETWORK_STATE 、 READ_CONTACTS 、 CALL_PHONE、RECEIVE_SMS 等等。由此可见，权限的使用从一个方面反映了恶意程序的某些行为特征，因此把权限特征作为本文中提取的特征向量之一。

2）API 调用序列特征

API 是操作系统为应用程序提供的服务性接口，应用程序在完成文件读写、网络访问以及其他重要资源的访问时都会调用 API。恶意程序在实现某个特征功能时同样会调用功能相似的 API 函数，因此通过提取 API 函数调用序列特征来识别恶意程序行为已成为通用一种做法，如 2004 年 J.Xu 等人在国际会议上的会议论文[13]中就提出了应用程序的 API 调用大致可以反映程序的行为。而 Ye 等人通过更进一步的研究，在他们的论文中使用 API 调用作为特征，并利用数据挖掘的方法对数据进行分析。

由于 Android 系统调用的复杂性、多样性，本系统中更加关注于 Android SDK 中的函数调用。参考 Min Zheng 等人[14]的做法，从 Android SDK 框架的 Android.jar 文件中提取全部 API 的 Class 路径和函数名称，并对每个 API 方法进行 hex 编码，最终形成 API 调用表，如 Android/accessibilityservice/AccessibilityService;-><init> 的编码为 0x0000，即 API 调用序列将由 16 进制数序列表示，从而降低了系统存储特征的空间。

3）OpCodes 特征

2008 年 Moskovitch 等人的论文《Unknown Malcode Detection Using OPCODE Representation》提出了使用 OpCodes 表达方式来反应一个可执行文件，他们主要针对的是 windows

PE 文件。在进行恶意程序分析的过程中，往往会将可执行文件进行反编译，从而获得由于机器语言指令集组成的汇编代码。OpCodes 是一个机器语言指令的第一部分，即执行的操作，如 MOVS R0,R5 其中 MOVS 就是一个 OpCodes。因此，应用程序要完成的功能是由一系列的指令构成，而能反映其具体操作的即为 OpCodes 集合。与 API 函数调用序列能够反映应用程序的行为类似，由 OpCodes 组成的集合同样可以反映行为特征，如下是在终端中打印字符串的汇编代码，提取的 OpCodes 特征向量{STMFD,ADR,BL, MOV,LDMFD}：

```
STMFD SP!, {R4,LR}
ADR R0, aHelloWorld ; "hello, world"
BL __2printf
MOV R0, #0
LDMFD SP!, {R4,PC}
```

与 PE 文件不同，Android 应用程序是由 java 语言开发，并运行在 Dalvik 虚拟机上，而虚拟机运行的是 Dalvik 字节码，并不存在上面所说的汇编代码。但 Android 应用程序常常会通过 JNI 接口调用原生程序，而原生程序则由 C 或 C++编译生成.so 库文件。该库文件是 ELF 格式的可执行程序，可以针对这部分文件提取 OpCodes 特征。另外，通过查询 arm 指令集手册，提取其中最常用的涉及数据移动、算术运算、逻辑计算、位移、比较、跳转等 23 类共计 157 个指令作为 OpCodes 特征集合，由于 OpCodes 本身只占很少的存储空间，因此并未对其进行编码处理。

2.3 决策树与随机森林算法

随机森林[15]是用随机的方式建立一个森林，森林里由多棵决策树组成。当新的输入样本进入，森林中的每一棵决策树分别进行判断，根据决策树投票决定样本属于哪一类。假设有 N 个样本，每个样本有 M 个 features，决策树们会随机地接受 n 个样本（对行随机取样）的 m 个 feature（对列进行随机取样），每颗决策树的 m 个 feature 相同。每颗决策树其实都是对特定的数据进行学习归纳出分类方法，而随机取样可以保证有重复样本被不同决策树分类，这样就可以对不同决策树的分类能力做个评价。

决策树是一个树结构（可以是二叉树或非二叉树）。其每个非叶节点表示一个特征属性上的测试，每个分支代表这个特征属性在某个值域上的输出，而每个叶节点存放一个类别。使用决策树进行决策的过程就是从根节点开始，测试待分类项中相应的特征属性，并按照其值选择输出分支，直到到达叶子节点，将叶子节点存放的类别作为决策结果。

由于随机森林算法自身的优点，如处理很高维度数据、训练速度快、分类准确度高等，比较适用于恶意程序的分类。国内外研究者已开始尝试使用相关算法进行 Android 应用的恶意程序识别与检测，刘阳[16]在其硕士毕业论文中使用随机森林和神经网络算法对 Android 平台下的恶意程序进行检测，但其提取的特征过于依赖 APK 自身信息，并未覆盖其他接口，如原生程序中的特征。本文中提取的 3 类特征是基于对大量恶意样本进行分析的总结，从不同维度描述恶意程序的行为，并将每类特征向量统一化建立为 N-gram 模型，再进行机器学习，具有一定的先进性。

3 系统设计与实现

该系统实现了样本收集，特征提取，数据统一建模，机器学习分类，各功能模块全部实现了自动化，系统结构图如图 1 所示。

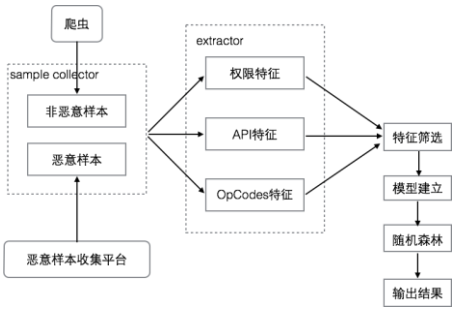

图 1 系统结构图

3.1 爬虫模块

爬虫模块实现了基于 python 语言中 urlib 库的爬虫功能，通过正则表达式方式匹配应用名称和下载链接，可以从指定的或第三方应用市场中下载 Android 应用程序。爬虫模块使用配置文件方式，可自由进行配置，定制下载应用类型、搜索起始和停止点、搜索页面深度等，以此满足下载需求。为了满足系统检测实验的要求，使用爬虫功能从某个第三方应用程序市场下载应用程序 811 个，涉及应用类型 16 类，涵盖影音媒体、通信、网络应用、系统工具、日常应用、聊天交友、电子阅读等。

3.2 提取特征向量

1）提取权限特征

使用静态分析的方法，对应用程序自身 APK 文件进行解析分析，提取应用程序在配置文件中申请的敏感权限信息。Androguard[17]是基于 python 的 Android 恶意应用程序检测工具，可以提取大量敏感信息，其中 androapkinfo.py 模块分析并列出应用程序的文件类型、权限、4 大组件、是否 NDK 反射等信息。本文利用 androapkinfo.py 提取 APK 申请的敏感权限，将每个应用的结果汇总为权限特征列表，提交数据统一建模模块。

2）提取 smali API 调用特征

Android 基于 Linux 内核，在架构上分为 5 个部分，包括 Linux Kernel、Android Runtime、Libraries、Application Framework 和 Application。Android 应用程序以 APK(Android Package)文件结构发布,APK 文件是一个压缩包，其结构如图 3 所示，META-INF 用于存放签名信息，用来保证 APK 包的完整性；res 存储资源文件,包括图片 、字符串等；AndroidManifest.xml 是应用程序的配置文件，其 中声明了应用程序的包名、SDK 版本、权限、组件等信息；classes.dex 则是 dvm 字节码文件，可运行于 Android 虚拟机 Dalvik 上。

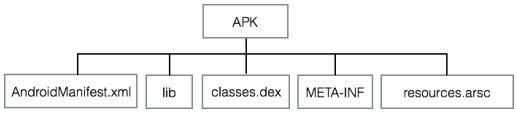

图 2 APK 文件结构

Dalvik 虚拟机有一套指令集，并制定了指令格式与调用规范，Dalvik 指令集组成的代码称为 Dalvik 汇编代码。目前 dex 可执行文件主流的反汇编工具有 Baksmali 与 Dedexer。提取 apk 应用的 API 调用序列是基于 dex 的反汇编代码，通过工具反汇编 apk 到指定工作目录，然后针对每个 smali 文件进行解析分析，提取其中的 API 调用序列，对比 Android SDK 中的 API 列表删除不重要的 API 调用，最后形成一个 apk 可执行程序的 API 调用序列。针对一个 apk 文件提取的基本流程如图 3 所示。

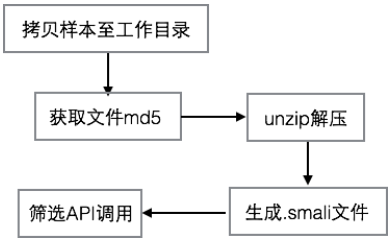

图 3 API 特征提取流程

基本步骤：a）拷贝 APK 文件到工作目录；b）获取文件的 md5 值；c）使用 unzip 命令解压文件到固定目录；d）使用返汇编工具 baksmali.jar 对 classes.dex 文件进行反汇编生成.smali 文件；e）遍历每个 smali 文件提取“invoke-”开头的函数调用并与 SDK API 调用表进行对比，包含在内的增加至 API 特征列表内。

3）提取 native 代码 OpCodes 特征

APK 文件结构中还包括 lib 目录，该目录下保存着应用程序通过 JNI 接口调用的 native 程序文件即.so 文件，虽然不是每个 APK 文件都包含原生程序，但通过大量分析恶意代码程序发现，存在恶意行为的应用程序一般会使用较为高级的方法实现隐藏等目的，即调用 active 程序，因此提取该部分代码的 OpCodes 特征具有一定的检测意义。针对一个 APK 文件提取的基本流程如图 4 所示。

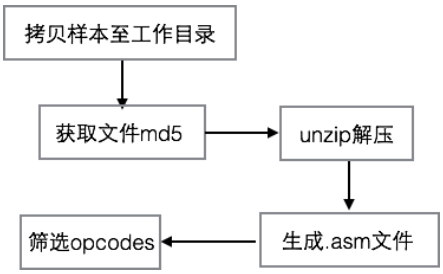

图 4 OpCodes 特征提取流程

基本步骤：a）拷贝 APK 文件到工作目录；b）获取文件的 md5 值；c）使用 unzip 命令解压文件到固定目录；d）查找 lib/armeabi/目录下的.so 文件，并使用 IDA 批量模式自动反汇编生成.asm 文件； e）扫描.asm 文件内容，逐行匹配 3.2 小节中提取的 OpCodes 指令集合，包含在内的增加至 OpCodes 特征列表。

3.3 建立 N-gram 模型

通过 3.2 节的特征提取，形成了 3 类特征的 3 个特征向量，将 3 个特征向量分别提交至数据统一建模模块进行 N-gram 建模。由于敏感权限特征不会在一个 APK 中反复申请，因此在建立权限模型时，选择 n=1，即建立 1-gram 模型。对于 API 和 OpCodes 特征选取 n=3，即建立 3-gram 模型，以 3.2 中的 OpCodes 特征集合{STMFD,ADR,BL,MOV,LDMFD}为例，3-gram 模型为{ STMFD,ADR,BL }，{ ADR,BL,MOV }，{ BL,MOV,LDMFD }。对每个子集出现的次数进行统计计数，并设置一个阈值，选择大于该阈值的特征集合作为机器学习的样本输入。对于每个特征向量进行建模时，将最终的结果调用 python 的 pandas.DataFrame()方法进行数据格式化，最终写入相应的 CSV 文件。

3.4 随机森林与分类

使用随机森林算法对 4.3 小节中建立的模型进行自动机器学习，最终输出分类结果。基本步骤：a）使用 python 的 pandas.merge()方法将提取的特征数据进行合并；b）生成测试矩阵；c）使用 cross_validation.train_test_split()将数据分割为训练数据和测试数据;d）设置决策树数量开始训练；e）输出测试集结果，并输出其他多个指标评价模型。

4 实验与结果分析

4.1 实验环境

本文设计的检测方法及系统全部使用 python 语言实现，系统中涉及的开源和商业工具包括：baksmali.jar、androguard、IDA Pro（Linux 版）。在检测该检测方法和系统有效性时，对现实中的 811 个非恶意程序和 826 个恶意程序进行检测。物理主机为 4GB 内存，处理器为 Intel Core i5-2400，系统开发环境为 32 位 ubuntu 15.10。

4.2 实验结果与分析

为了保证实验的可靠性和高覆盖率共计使用样本 1637 个，811 个非恶意程序来源于 16 类不同的应用类别，而 826 个恶意样本来源于第三方样本收集平台 virusshare.com[18]。将所有样本提取特征建立模型后，使用 60%的样本进行训练，而 40%（655 个）样本作为独立测试集。

3.2 节中提取的 3 类特征是基于对恶意程序的人工分析，每一类都可以单独作为特征集合对应用程序进行检测，为了体现本文所提方法的有效性，首先使用每个特征向量进行检测，实验结果如表 1 所示。从结果可以看到，每类特征进行独立检测时准确率各不相同，其中 API 特征准确率居中，敏感权限最高，OpCodes 最低，符合人工分析的结果，主要原因是 OpCodes 特征并非每个 APK 文件都具备，而 API 特征对于混肴代码具有一定的局限性。为了弥补相互的不足，使用 3 种特征的集合来进行机器学习并分类，获得了满意的效果，准确率为 95.3%。

表 1 特征检测率对比

Type	Accuracy(100%)
API feature	0.895
Opcodes feature	0.690
Permission feature	0.945
Triple feature	0.953

下面使用 TPR 、FPR、MA、precision（精确度）、recall（召回率）、f1-score 等指标来衡量本文提出的检测方法的有效性。其中，召回率反映了被正确判定的正例占总的正例的比重。f1-score，即 F1 分数，是统计学中用来衡量二分类模型精确度的一种指标，它同时兼顾了分类模型的准确率和召回率，是模型准确率和召回率的一种加权平均。实验结果如下表所示。

表 2 综合指标

Class	precision	recall	f1-score	support	Confusion matrix	Accuracy
1:nomarl	0.92	0.99	0.95	315	[TN:312 FP:3]	0.95
2:malfare	0.99	0.92	0.95	340	[FN:28 TP:312]	

TPR （true positive rate）表示所有恶意代码被检测出的概率，用公式表示即为 TPR=TP/(TP+FN)，其中 TP 为正确检测到的恶意代码数量，FN 表示被判定为良性程序的恶意代码数量。FPR（False Positive Rate ）表示良性程序被检测为恶意程序的概率，用公式表示即 FPR=FP/(FP+TN)，其中 FP 为被检测为恶意程序的良性程序的数量，TN 为被正确认定为良性程序的数量。漏警概率（Missing Alarm）MA = FN/(TP + FN) 反映有多少个正例被漏判。

表 3 检测指标

TPR	FPR	MA
0.92	0.0095	0.082

由表 2、3 可以看出，本文提出的 Android 恶意程序检测方法，不论是从综合指标还是检测指标来看都具有较好的表现，较 KolbitschC 等[19]使用的行为图建模的方法或 AlazabM 等[20]使用的方法检测效率、准确率更优。

5 结束语

本文根据大量分析经验，提取了 APK 文件的权限特征、DVM API 调用序列特征以及 native 程序的 OpCodes 特征，并建立每个特征向量的 N-gram 模型，最后使用基于随机森林的机器学习算法进行判断分类。创新点在于提出了对 APK 文件多类行为特征进行 N-gram 建模，且应用机器学习进行检测的方法，并实现了该方法的检测系统。在检测实验中对 811 个非恶意程序和 826 个恶意程序进行检测，结果表明以该方法实现的检测系统在各项指标上表现良好且优于其他相关工作。下一步工作包括针对使用混淆、repack 等技术的恶意代码的检测、分析，以及采用更好的方式模拟人的行为并监控应用程序动态行为特征，将动态行为特征加入特征向量中，进一步提高检测准确率，以完善检测系统。

参考文献:

[1] http: //www. freebuf. com/articles/terminal/97563. html[EB/OL].

[2] Christodorescu M, Jha S. Static analysis of executables to detect malicious paterns[R]. [S. l.]: Wisconsin Univ-Madison Dept of Computer Sciences, 2006.

[3] 徐曾春, 卢洲, 叶坤. 一种检测可疑软件的 Android 沙箱系统的研究与设计[J]. 南京邮电大学学报: 自然科学版, 2015, 35(4): 104-109.

[4] Ye Y, Wang D, Li T, etal. An inteligent PE malware detection system based on association mining[J]. Journal in Computer Virology, 2008, 4(4): 323 334.

[5] 李盟, 贾晓启, 王蕊, 等. 一种恶意代码特征选取和建模的方法[J]. 计算机应用与软件. 2015, 32(8).

[6] 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 2015, 37(1).

[7] Amos B, Turner H, White J. Applying machine learning classifiers to dynamic Android malware detection at scale[C]//Proc of the 9th IEEE Int'l Wireless Communications and Mobile Computing Conf. 2013: 1666-1671.

[8] Sahs J, Khan L. A machine learning approach to Android malware detection[C]//Proc of the European Intelligence and Security Informatics Conf. 2012: 141-147.

[9] Peiravian N, Zhu X. Machine learning for Android: Malware detection using permission and API calls[C]//Proc. of the 25th IEEE Int'l Conf on Tools with Artificial Intelligence. 2013: 300-305.

[10] Shabtai A, Fledel Y, Elovici Y. Automated static code analysis for classifying Android applications using machine learning[C]//Proc. of IEEE Int'l Conf. on Computational Intelligence and Security. 2010: 329-333.

[11] Abou-Assaleh T, Cercone N, Kešelj V, *et al*. N-gram-based Detection of New Malicious Code[C]//Proc of IEEE Computer Software & Applications Conference. 2004: 41-42.

[12] Moskovitch R, Feher C, Tzachar N, etal. Unknown malcode detection using opcode representation: intelligence & security informatics[C]// Proc of the 1st European Conference on Intelligence and Security Informatics. 2008: 204-215.

[13] Xu J Y, Sung A H, Chavez P, etal. Polymorphic malicious executable scanner by API sequence analysis[C]// Proc of the 4th International Conference on Hybrid Intelligent Systems. 2004: 42-45.

[14] Zheng Min, Sun Mingshen, Lui J. DroidAnalytics: a signature based analytic system to collect, extract, analyze and associate Android malware[C]// Proc of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications. 2013: 163-171.

[15] Breiman L. Random forest[J]. Machine Learning Journal, 2001, 45(1): 5-32.

[16] 刘阳. 应用随机森林与神经网络算法检测与分析 Android 应用恶意代码[D]. 北京: 北京交通大学, 2015.

[17] Peng Hao, Gates C, Sarma B, etal. Using probabilistic generative models for ranking risks of Android APPs[C]// Proc of ACM Conference on Computer and Communications Security. 2012: 241-252.

[18] https: //virusshare. com/[EB/OL].

[19] Kolbitsch C, Comparetti P M, Kruegel C, etal. Effective and efficient malware detection at the endhost[C]// Proc of USENIX Security Symposium. 2009: 351-366.

[20] Alazab M, Layton R, Venkataraman S, *et al*. Malware detection based on structural and behavioural features of API cals[C]// Proc of the 1st International Cyber Resilience Conference. 2010.