计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2016.20160348
53(10):2299-2306,2016

Maldetect:基于 Dalvik 指令抽象的 Android 恶意代码检测系统

陈铁明　杨益敏　陈　波
(浙江工业大学计算机科学与技术学院　杭州　310023)
(tmchen@zjut.edu.cn)

Maldetect: An Android Malware Detection System Based on Abstraction of Dalvik Instructions

Chen Tieming, Yang Yimin, and Chen Bo
(*College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou*, 310023)

Abstract　A novel static Android malware detection system Maldetect is proposed in this paper. At first, the Dalvik instructions decompiled from Android DEX files are simplified and abstracted into simpler symbolic sequences. N-Gram is then employed to extract the features from the simplified Dalvik instruction sequences, and the detection and classification model is finally built using machine learning algorithms. By comparing different classification algorithms and N-Gram sequences, 3-Gram sequences with the random forest algorithm is identified as an optimal solution for the malware detection and classification. The performance of our method is compared against the professional anti-virus tools using 4 000 malware samples, and the results show that Maldetect is more effective for Android malware detection with high detection accuracy.

Key words　malware; Android; Dalvik instruction; N-Gram; machine learning

摘　要　提出了一个 Android 恶意代码的静态检测系统 Maldetect,首先采用逆向工程将 DEX 文件转化为 Dalvik 指令并对其进行简化抽象,再将抽象后的指令序列进行 N-Gram 编码作为样本训练,最后利用机器学习算法创建分类检测模型,并通过对分类算法与 N-Gram 序列的组合分析,提出了基于 3-Gram 和随机森林的优选检测方法.通过 4 000 个 Android 恶意应用样本与专业反毒软件进行的检测对比实验,表明 Maldetect 可更有效地进行 Android 恶意代码检测与分类,且获得较高的检测率.

关键词　恶意代码;安卓;Dalvik 指令;N-Gram;机器学习

中图法分类号　TP309

随着移动互联网的发展,移动智能终端越来越普及,移动应用的种类与数量都呈现高速增长,智能手机已经成为网民最常用的上网工具.来自 Gartner 统计数据显示,2015 年第 4 季度全球智能手机的销售量为 4 亿多台,其中 Android 系统占据了 80.7%[1].截止 2016 年 2 月 1 日,仅 Android 官方应用市场 Google Play 上的应用数量就接近 200 万[2].同时,Android 应用安全也面临着两大威胁:应用漏洞和恶意应用.Android 应用开发者往往缺少代码审计的能力和代码安全规范检查的习惯,容易

收稿日期:2016-05-18;**修回日期**:2016-08-12

基金项目:国家自然科学基金项目(U1509214);浙江省自然科学基金项目(LY16F020035)
This work was supported by the National Natural Science Foundation of China (U1509214) and the Natural Science Foundation of Zhejiang Province of China (LY16F020035).

在软件生命周期的各个阶段中产生安全漏洞,一旦被攻击者所利用会对软件的完整性、机密性和可用性造成损害[3]. Android 恶意应用通常是在一些热门的 Android 应用中插入一段攻击代码,并在安全管理较差的第三方应用商店进行发布与传播[4]. 根据阿里聚安全发布的 2015 移动安全病毒年报,18%的 Android 设备感染过病毒,95%的热门移动应用存在仿冒应用,恶意应用类型呈现越来越多的趋势[5]. 常见手机恶意应用的恶意行为包括恶意扣费、信息窃取、短信劫持等,可严重损害手机用户的利益,危害不容忽视.

主流的 Android 恶意代码检测方法主要有基于特征代码和基于行为的检测. 基于特征代码的检测方法通过检测文件是否拥有已知恶意软件的特征代码来判断其是否为恶意软件,具有快速、准确率高等特点,但无法检测未知的恶意代码,且通常需要维护病毒特征数据库. 国外著名的 Android 恶意代码检测工具 Androguard[6] 就是基于特征代码实现的.

基于行为的检测方法则通过程序的行为与已知恶意行为模式进行匹配,判断目标文件是否包含恶意代码. 误报率虽然并不理想,但可实现对未知恶意代码或病毒的检测,可弥补基于特征代码检测的不足. 基于行为的分析又可进一步分为动态和静态 2 种分析方法[7]. 动态分析方法是指利用"沙盒或虚拟机"来模拟运行程序,通过拦截或监控的方式分析程序运行时的行为特征,一定程度上可绕过代码混淆等代码保护机制,但是计算资源和时间消耗较大,且代码覆盖率低. 相对于重量级的动态分析,静态分析则相对属于轻量级的方法,通常是通过逆向工程抽取程序的特征,分析函数调用、程序指令等序列,具有快速高效、代码覆盖率高等特点.

目前,大部分的静态分析方法主要从 AndroidManifes. xml 文件、lib 库文件(. so 文件)、Java 源文件(通过反编译 APK 文件获得)等进行特征的提取与分析,而针对 Dalvik 指令序列的特征研究相对较少,而本文则基于 Dalvik 指令序列的抽象模型再结合机器学习方法进行研究. 本文的主要贡献为:

1) 简化 Dalvik 指令集,用指令符号抽象一类指令的操作码,并提出了一种基于 N-Gram 序列的特征模型;

2) 针对抽象的 Dalvik 指令符号的 N-Gram 序列特征,结合机器学习分类算法,有效实现了 Android 恶意代码的检测与分类,与常见反病毒软件相比较,获得较高的检测率.

1 相关工作

针对 PC 平台的恶意代码检测技术已相对成熟,其中利用 N-Gram 提取特征的方法也获得了较好的应用效果. 例如,文献[8]提出利用 OpCode N-Gram 代替 ByteCode N-Gram,有更高的精度,并对 x86 平台下的汇编指令进行了实验的验证. 文献[9]使用了一种 OpCode 序列频率的表达式来检测和分类恶意代码,并结合了数据挖掘等方法,也有较高的检测率.

Android 代码静态分析方法基本都从 AndroidManifes. xml 文件、lib 库文件(. so 文件)、反编译得到的 Java 源文件中进行特征的提取与分析. 例如,文献[10]通过反编译 APK 得到 Java 源代码,再从 Java 源码提取调用方法名作为特征,并结合 N-Gram 等方法进行恶意软件分类,但该方法没有对混淆过的 Java 源代码进行处理;文献[11]设计了一种轻量级的 Android 恶意代码检测方法,将申请的权限、可疑的 API 函数调用和网络地址等作为静态分析的特征,通过 5 560 个恶意样本和 123 453 个正常样本的实验分析,表明有较高的准确率,但是该方法特征的提取依赖于反汇编后的 Java 源码,混淆过的 Java 源码会导致该方法失效;文献[12]提出了一种适合恶意代码分析的简化 Dalvik 中间语言(SDIL),并结合 MOSS 算法进行了实验测试,验证了分析方法的有效性,但该方法在代码分析、特征提取时还需加入大量的人工操作,且提取的特征数量与涵盖的恶意代码种类较少.

综上所述,为了提高检测性能,本文首先研究 Android 的 Dalvik 指令抽象与简化,提取 Dalvik 指令的操作码,再借鉴 OpCode N-Gram 方法提取抽象指令符的序列特征,最后实现针对 Android 的一种快速有效的恶意代码静态检测方法.

2 本文方法

2.1 系统模型

本文提出了一种 Android 恶意代码静态检测系统. 该系统可以快速有效地检测 Android 应用程序,识别恶意应用且能够给出恶意家族的类别信息.

整个系统分为 2 部分:1)恶意代码检测模型和恶意家族分类模型的训练,如图 1 所示;2)恶意代码

检测模型和恶意家族分类模型的测试,如图 2 所示:

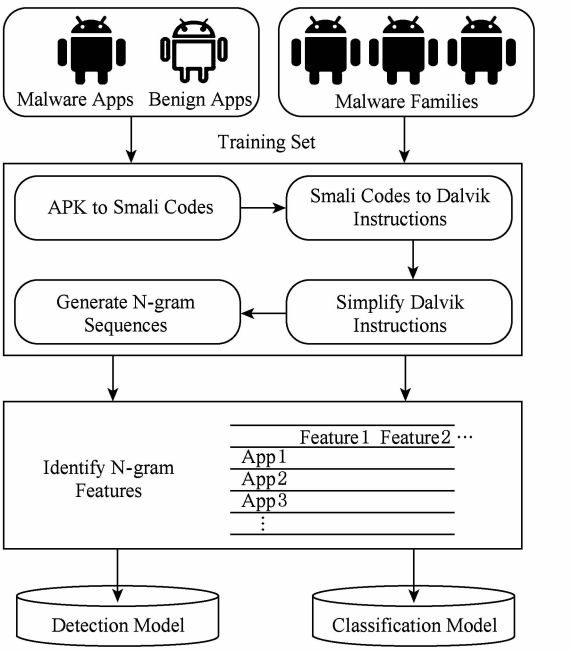

Fig. 1 The detection and classification model.

图 1 恶意代码的检测与分类模型训练

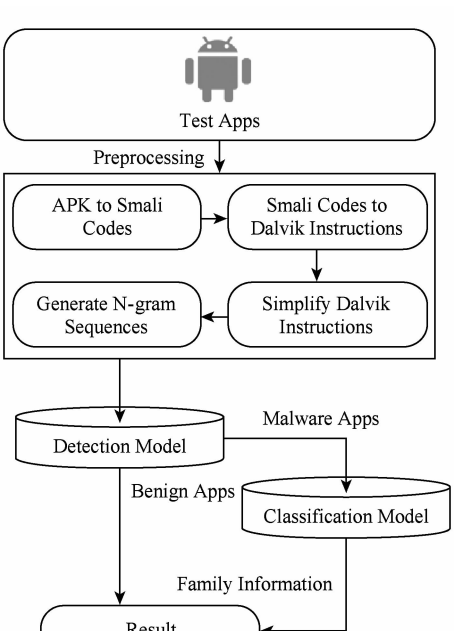

Fig. 2 The process of predicting malware.

图 2 恶意代码的检测与分类过程

首先,需要确定训练恶意代码检测模型的训练集.训练集分为 2 个子集:1)恶意 APK 样本集合;2)非恶意样本 APK 集合. APK 文件格式通常都包含一个 classes.dex 文件,该 DEX(Dalvik executable format)文件封装了可被 Dalvik 虚拟机执行的 Dalvik 字节码.利用工具 Apktool[13] 反汇编 APK 文件,就能得到一个包含 smali 源码的文件目录,smali 目录结构对应着 Java 源码的 src 目录. smali 是对 Dalvik 字节码的一种解释,所有语句都遵循一套标准的语法规范.从 smali 文件中提取出 Dalvik 操作码并进行抽象简化为指令符号,再针对抽象的 Dalvik 指令符号的 N-Gram 序列特征进行统计与归一化处理,最后采用机器学习的分类算法建立恶意代码检测模型.同理,按照恶意家族的类型划分多个训练子集,按照上述类似的处理过程,可建立一个恶意家族分类模型.

利用模型检测与分类时,将待测 APK 文件先进行预处理步骤,提取出 Dalvik 指令特征并作同样的抽象简化与 N-Gram 序列化处理,再通过恶意代码检测模型的检测,就可判断出是否为恶意代码,如果不是就直接给出检测结果,如果是则需要进一步通过恶意家族分类模型来获得该恶意代码家族类型.

2.2 Dalvik 指令特征与 N-Gram

根据文献[14]表明官方 Dalvik 指令有 230 条,分别包括方法调用指令、数据操作指令、返回指令等十几种类型.文献[12]提出了一种适合恶意代码分析的简化 Dalvik 中间语言(SDIL),对 218 种指令简化成了 13 种,并分别统计了指令的频率.基于 Dalvik 指令集与上述工作,本文另行设计了一种指令符号方案:在指令集中提取出能够正确反映程序语义的指令和操作码,对功能相近的指令进行归类,并将该类的指令集合抽象为指令符号.与原有的 Dalvik 的指令集相比,更加简化精要,且便于后续的分析与特征提取.表 1 中给出了最终的设计方案,在 Dalvik 指令集中提取出的 107 种代表性指令并分成 10 大功能相近的类,并且每类指令抽象为一个特定的指令符号,后续 N-Gram 编码将直接使用这种指令符号代替完整的 Dalvik 指令.

N-Gram 算法经常用于自然语言处理领域,但是它也经常用来处理恶意代码的分析.对 Dalvik 指令符号进行 N-Gram 编码,同样可以用于分析 Android 的恶意代码. N-Gram 算法假设第 m 个词的出现只是与前面的 $m-1$ 个词相关,现假设有 Dalvik 指令符号序列为{M,R,G,I,T,P,V},3-Gram 编码后提取的特征为[{M,R,G},{R,G,I},{G,I,T},{I,T,P},{T,P,V}].表 2 中给出了当 3-Gram 编码时一个 APK 应用中包含的 1 000 种不同 N-Gram 特征的频率统计.

Table 1 Symbolic Instruction Set Definition

表 1 指令符号集合的定义

Symbols	Semantics	Quantity	Representative Opcode Prefixes
C	comparison	5	cmpl-float|cmpg-float|cmpl-double|cmpg-double|cmp-long
D	definition	11	const|const/4|const/16|const-wide|const/high|const-string
M	manipulation	13	move|move-wide|move-object|move-result|move-exception
R	return	4	return|return-void|return-wide|return-object
L	monitor	2	monitor-enter|monitor-exit
G	jump	3	goto|goto/16|goto/32
I	judgment	12	if-eq|if-ne|if-lt|if-ge|if-gt|if-le|if-eqz|if-nez|if-ltz|if-gez|if-gtz|if-lez
T	reading	21	aget|iget|sget|aget-wide|aget-object|aget-boolea|aget-byte|aget-char
P	writing	21	aput|iput|sput|aput-wide|aput-object|aput-boolea|aput-byte|aput-char
V	method call	15	invoke-virtual|invoke-super|invoke-direct|invoke-static

Table 2 Illustration for the Frequency Statistics of 3-Gram Features

表 2 3-Gram 特征频率的统计

APK Set	CCC	CCD	CCM	CCR	…	VVV
DroidKungFu. apk	0.032 016	0.010 601	0.019 507	0.036 822	…	0.071 454
BaseBridge. apk	0.059 207	0.011 801	0.018 162	0.032 164	…	0.047 486
Plankton. apk	0.030 508	0.006 391	0.017 376	0.038 064	…	0.053 952
⋮	⋮	⋮	⋮	⋮	⋮	⋮

2.3 机器学习方法及评估标准

分类算法是一种典型的机器学习方法. 文献[15]介绍了最常见的衡量分类算法性能的评估参数. 度量分类算法的有效性最基本的 4 个指标是:1)真阳性(true positive, TP),表示正确分类为恶意软件的恶意样本个数;2)假阳性(false positive, FP),表示错误分类为恶意软件的良性样本个数;3)真阴性(true negative, TN),表示正确分类为良性软件的良性样本个数;4)假阴性(false negative, FN),表示错误分类为良性软件的恶意样本个数. 由这 4 个指标可以构成如表 3 所示的混淆矩阵,用于分析分类算法的优劣性.

Table 3 Confusion Matrix

表 3 混淆矩阵

Prediction	Malicious	Benign
Malicious	TP	FN
Benign	FP	TN

由上述的基本指标可以构成下面的一些常用指标:

$$TPR=Recall=\frac{TP}{TP+FN},$$

$$FPR=\frac{FP}{FP+TN},$$

$$Precision=\frac{TP}{TP+FP},$$

$$Accuracy=\frac{TP+TN}{TP+TN+FP+FN},$$

$$F\text{-}Measure=\frac{2\times Recall\times Precision}{Recall+Precision}.$$

真阳性率(TPR)亦即检测率(detection rate),有时也称为召回率(recall rate);假阳性率(FPR)亦即误报率(false alarm rate);正确率 $Precision$ 指在真正的恶意样本在分类为恶意样本的比例;分类精度 $Accuracy$ 是所有正确分类的样本与所有样本之比;$F\text{-}Measure$ 指准确率与召回率的调和平均值. ROC 曲线(接收者运行特征)是一种显示分类算法的 TPR 和 FPR 之间折中的图形化方法. AUC(area under the curve)是 ROC 曲线下方的面积,如果完全准确的模型,面积为 1.

3 实验与分析

3.1 实验数据与实验环境

测试所采用的恶意样本来源于德国哥廷根大学

Drebin 项目的恶意样本数据库[16],正常的样本通过安卓官方的 Google Play 应用商店[17]随机下载获得.文献[18]指出在 Google Play 应用商店中 52 208 个安卓应用中只有 2 个为恶意的,所以基本可以认为 Google Play 应用商店中的安卓应用均为非恶意应用.

本文利用机器学习工具 Weka(Waikato environment for knowledge analysis)对实验数据得到的特征用机器学习算法训练分类模型.Weka 是一款免费的基于 JAVA 环境下开源的机器学习软件,集成了大量的机器学习算法,且具有高效准确的特点,所以本文使用它作为机器学习的工具.

3.2 分类算法与 N-Gram 序列的优选

本节先采用 270 个恶意样本和 330 正常样本作为实验样本集,根据 2-Gram,3-Gram,4-Gram 编码分别提取了不同长度的 Dalvik 指令符号序列,然后再分别使用 4 种分类算法包括随机森林算法(Random Forest)、支持向量机算法(SVM)、*K*-最近邻算法(*K*NN)、朴素贝叶斯算法(Naive Bayes)对 3 种编码的序列采用 10 折交叉验证进行测试,结果如表 4～6 所示.

Table 4 The Detection Results Based on 2-Gram Sequences

表 4 基于 2-Gram 序列的测试结果

Method	*TPR*	*FPR*	*Precision*	*Recall*	*F-Measure*	*AUC*
Random Forest	0.915	0.106	0.876	0.915	0.895	0.970
SVM	0.852	0.173	0.801	0.852	0.826	0.840
*K*NN	0.930	0.136	0.848	0.930	0.887	0.897
Naive Bayes	0.870	0.221	0.763	0.870	0.813	0.888

Table 5 The Detection Results Based on 3-Gram Sequences

表 5 基于 3-Gram 序列的测试结果

Method	*TPR*	*FPR*	*Precision*	*Recall*	*F-Measure*	*AUC*
Random Forest	0.956	0.079	0.908	0.956	0.931	0.982
SVM	0.922	0.130	0.853	0.922	0.886	0.896
*K*NN	0.952	0.148	0.840	0.952	0.892	0.903
Naive Bayes	0.867	0.170	0.807	0.867	0.836	0.903

Table 6 The Detection Results Based on 4-Gram Sequences

表 6 基于 4-Gram 序列的测试结果

Method	*TPR*	*FPR*	*Precision*	*Recall*	*F-Measure*	*AUC*
Random Forest	0.956	0.103	0.884	0.956	0.918	0.984
SVM	0.944	0.106	0.879	0.944	0.911	0.919
*K*NN	0.904	0.088	0.894	0.904	0.899	0.909
Naive Bayes	0.900	0.103	0.877	0.900	0.888	0.917

选用 *AUC*,*TPR*,*FPR*,*Precision*,*Recall*,*F-Measure* 等指标作为实验的评估标准.其中,*AUC* 是最重要的标准,它可以正确地反映 *TPR* 和 *FPR* 之间的关系,*AUC* 值越高代表检测的综合表现越优.*FPR* 也是一项重要的指标,*FPR* 值越低代表误报率越低.在表 4 中,随机森林算法的 *AUC* 值最高,*FPR* 值最低,而 *TPR* 值略不如 *K*NN 算法.在表 5 中,随机森林算法的各项指标都表项最优.在表 6 中,随机森林算法的 *AUC* 值最高,*K*NN 算法的 *FPR* 值最理想,但是随机森林算法的 *FPR* 值与 *K*NN 算法的 *FPR* 值差距很小.综合分析,在 2-Gram,3-Gram,4-Gram 指令符号序列中,随机森林算法的表项是最优的.

以随机森林算法为基础进一步寻找最优的 N-Gram 序列.综合比较表 4～6,不难发现,随机森林算法 *AUC* 值按从优到劣的顺序为:4-Gram,3-Gram,2-Gram;随机森林算法 *FPR* 值按从优到劣的顺序为:3-Gram,4-Gram,2-Gram;2-Gram 序列表现都是最劣的,可以不再考虑.4-Gram 序列的随机森林算法 *AUC* 值比 3-Gram 序列的随机森林算法 *AUC*

值仅高出了 2%,而 4-Gram 序列的随机森林算法 *FPR* 值比 3-Gram 序列的随机森林算法 *FPR* 值却高出了 30%.综合分析,3-Gram 序列的随机森林算法是一种较优的方法.

接下来,分析样本数量对准确性的影响.除了上述实验的 600 个样本作为小样本集合,再增加一个 1 100 个样本作为大样本集合作为对比实验,按 3-Gram 的随机森林算法,采用 10 折交叉验证对它们进行测试.实验结果如表 7 和图 3 所示,不难看出,大样本集的所有评估指标都要优于小样本集,这表明样本数量大小对检测效果有一定的影响,训练样本数量越大则综合表现越优.

Table 7 The Results with Different Size of Samples

表 7 样本数量对比的结果

Sample Size	*TPR*	*FPR*	*Precision*	*AUC*
600 samples	0.956	0.086	0.908	0.982
1100 samples	0.980	0.079	0.917	0.992

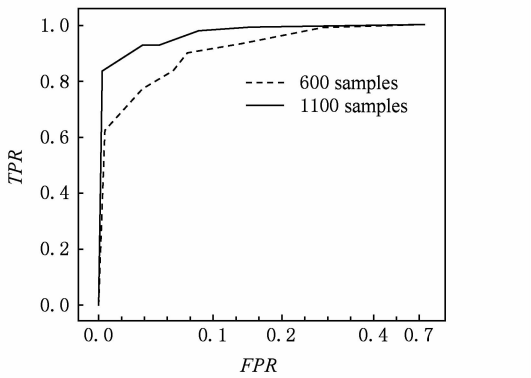

Fig. 3 ROC curves with different size of samples.

图 3 不同样本数量的 ROC 曲线对比图

3.3 MaldetectVS 专业反病毒软件

根据 3.2 节的实验结果,以实验效果较优的 3-Gram 序列的随机森林算法作为基础,通过计算信息增益来选取 200 个区分度最高的特征作为特征选择方法,我们实现了一个 Android 恶意代码检测系统 Maldetect.本节实验我们分别在 2 000 个样本和 4 000 个样本的 2 个数据集上进行实验,并按照 60%数据作为 Maldetect 的训练样本库,40%数据作为测试样本库.作为实验对照,我们也将测试样本发送到 ThreatBook 多引擎文件检测服务[19]然后得到了 9 款常见的反病毒软件(Avira,Dr. Web,AVG,Kaspersky,ESET,GDATA,Rising,MSE,Avast)的检测结果,最后统计出各系统的检测率与误报率.

根据表 8 显示的本次实验结果,我们发现大部分的反病毒软件的检测率都超过了 90%,但是也存在一些检测率甚至低于 50%的反病毒软件,可能这些反病毒软件并不适用于安卓平台的检测.在 2 000 个样本数据集上,Maldetect 以 95.3%的检测率在 10 款反病毒软件中排第 3;在 4 000 个样本数据集,Maldetect 以 98.6 %的检测率在 10 款反病毒软件中排第 2.这表明随着样本训练样本数量增加,Maldetect 的精确度呈上升趋势.另外,实验所用的恶意样本已经公开了一段时间,大多数反病毒软件的特征库已经加入了这些恶意样本的特征,在检测未公开的恶意样本时,Maldetect 的机器学习方式则更加能体现优势.

专业的反病毒软件的误报率基本在 0%~1%之间,Maldetect 的误报率比它们要稍微高一点.另外,表 9 也表明随着训练样本数量的增加,Maldetect 的误报率呈较明显的下降趋势.

Table 8 Detection Rates of Maldetect and Antivirus Scanners

表 8 Maldetect 与反病毒软件的检测率比较 %

Sample Size	Maldetect	AV1	AV2	AV3	AV4	AV5	AV6	AV7	AV8	AV9
2 000 samples	95.3	100.0	97.0	94.0	91.5	91.0	66.0	49.5	15.5	11.0
4 000 samples	98.6	99.1	92.7	92.3	78.7	90.0	42.9	45.0	7.5	10.2

Table 9 False Positive Rates of Maldetect and Antivirus Scanners

表 9 Maldetect 与反病毒软件的误报率比较 %

Sample Size	Maldetect	AV1	AV2	AV3	AV4	AV5	AV6	AV7	AV8	AV9
2 000 samples	5.0	0.5	0.5	0	0	0	0.5	0.5	0	0
4 000 samples	1.4	1.6	0.9	0	0	0	0	1.2	0	0

3.4 恶意家族分类实验

除了检测恶意代码,另一个重要方面,我们需要分类恶意家族并进行性能评估.本节实验我们精选样本数量最多、最常见的 9 个恶意家族共计 900 个样本作为实验样本,其中 60%数据作为训练集,40%数据作为测试集.用 Maldetect 进行实验测试,结果如表 10 所示.从表 10 可知,9 类恶意家族的 *AUC* 值都不低于0.97,*FPR* 值都不高于 0.037,该方法对恶意家族分类有较好的效果,也完全适用于恶意家族的分类.

Table 10 The Results on Classification of Malware Family Using Maldetect

表 10 类恶意家族分类的结果

Malware Family	*TPR*	*FPR*	*Precision*	*Recall*	*F-Measure*	*AUC*
Plankton	0.949	0.003	0.974	0.949	0.961	0.999
DroidKungF	0.833	0.003	0.972	0.833	0.897	0.996
GinMaster	1	0.037	0.755	1	0.86	0.997
FakeDoc	0.976	0	1	0.976	0.988	1
FakeInstalle	0.865	0.006	0.941	0.865	0.901	0.978
Opfake	1	0.019	0.882	1	0.938	0.997
BaseBridge	0.816	0	1	0.816	0.899	0.995
Kmin	1	0	1	1	1	1
Iconosys	1	0	1	1	1	1

4 总 结

本文提出了一种 Android 静态检测方法,采用逆向工程将 DEX 文件转化为 Dalvik 指令并对其进行简化抽象,再将抽象后的指令序列进行 N-Gram 编码作为样本训练,通过机器学习的分类算法来创建分类检测模型.在实验环节,对不同分类算法与 N-Gram 序列分别进行了比较,提出了基于 3-Gram 和随机森林的优选检测方法.最后,采用 2 000 个样本与 4 000 个样本 2 组样本集,与专业的反毒软件进行对比实验,实验结果表明该方法可有效地进行 Android 恶意代码检测与分类,且获得较高的检测率.下一步工作将研究在提高训练样本数量的情况下,用聚类的方法选择出典型样本,去除噪声数据,进一步提高准确率、降低误报率.

参 考 文 献

[1] Egham. Gartner Says Worldwide Smartphone Sales Grew 9.7 Percent in Fourth Quarter of 2015 [EB/OL]. (2016-02-18) [2016-04-01]. http://www.gartner.com/newsroom/id/3215217

[2] AppBrain. Android Statistics: Google Play Stats [EB/OL]. (2016-02-01) [2016-03-01]. http://www.appbrain.com/stats

[3] Zhang Yuqing, Fang Zhejun, Wang Kai, et al. Survey of Android vulnerability detection [J]. Journal of Computer Research and Development, 2015, 52(10): 2167-2177 (in Chinese)
(张玉清, 方喆君, 王凯, 等. Android 安全漏洞挖掘技术综述[J]. 计算机研究与发展, 2015, 52(10): 2167-2177)

[4] Zhang Yuqing, Wang Kai, Yang Huan, et al. Survey of Android OS security [J]. Journal of Computer Research and Development, 2014, 51(7): 1385-1396 (in Chinese)
(张玉清, 王凯, 杨欢, 等. Android 安全综述[J]. 计算机研究与发展, 2014, 51(7): 1385-1396)

[5] Alibaba. 2015 Security Report [EB/OL]. 2016 [2016-03-01]. http://jaq.alibaba.com (in Chinese)
(阿里巴巴. 2015 年安全报告[EB/OL]. 2016[2016-03-01]. http://jaq.alibaba.com)

[6] Androguard Team. Androguard [EB/OL]. [2016-03-01]. http://code.google.com/p/androguard

[7] Qing Sihan. Research progress on Android security [J]. Journal of Software, 2016, 27(1): 45-71 (in Chinese)
(卿斯汉. Android 安全研究进展[J]. 软件学报, 2016, 27(1): 45-71)

[8] Shabtai A, Moskovitch R, Feher C, et al. Detecting unknown malicious code by applying classification techniques on opcode patterns [J]. Security Informatics, 2012, 1(1): 1-22

[9] Santos I, Brezo F, Ugarte-Pedrero X, et al. Opcode sequences as representation of executables for data-mining-based unknown malware detection [J]. Information Sciences, 2013, 231(9): 64-82

[10] Dhaya R, Poongodi M. Source code analysis for software vulnerabilities in Android based mobile devices [J]. International Journal of Computer Applications, 2014, 93(17): 10-14

[11] Arp D, Spreitzenbarth M, Hubner M, et al. DREBIN: Effective and explainable detection of Android malware in your pocket [C] //Proc of the 21st Annual Network and Distributed System Security Symposium. San Diego, California: Internet Society, 2014

[12] Dong Hang, He Nengqiang, Hu Ge, et al. Malware detection method of Android application based on simplification instructions [J]. Journal of China Universities of Posts & Telecommunications, 2014, 21(Suppl 1): 94-100

[13] GitHub. Apktool: A tool for reverse engineering Android apk files [EB/OL]. [2016-03-01]. https://github.com/iBotPeaches/Apktool

[14] Paller G. Dalvik opcodes [EB/OL]. [2016-03-01]. http://pallergabor.uw.hu/androidblog/dalvik_opcodes.html

[15] HAN J. Data Mining [M]. San Francisco: Morgan Kaufmann, 2011

[16] Arp D, Spreitzenbarth M, Hubner M, et al. The Drebin Dataset [EB/OL]. [2016-03-01]. https://www.sec.cs.tu-bs.de/~danarp/drebin/index.html

[17] Google Inc. GooglePlay [EB/OL]. [2016-03-01]. https://play.google.com

[18] Grace M, Zhou Y, Zhang Q, et al. RiskRanker: Scalable and accurate zero-day Android malware detection [C] //Proc of the 10th Int Conf on Mobile Systems, Applications, and Services. New York: ACM, 2012: 281-294

[19] ThreatBook Inc. VirusBook [EB/OL]. [2016-03-01]. https://x.threatbook.cn/

Chen Tieming, born in 1978. PhD. Professor in the Zhejiang University of Technology. His main research interests include network and information security.

Yang Yimin, born in 1985. Master candidate in the Zhejiang University of Technology. His main research interests include network and information security.

Chen Bo, born in 1971. Associate professor in the Zhejiang University of Technology. His main research interests include network and information security.