

基于操作码的安卓恶意代码多粒度快速检测方法

张雪涛, 孙蒙, 王金双

(陆军工程大学指挥控制工程学院, 江苏 南京 210001)

摘要: 基于操作码的检测方式被广泛用于安卓恶意软件检测中, 但存在特征提取方法复杂、效率低等问题。针对此类问题, 提出一种基于操作码的安卓恶意软件多粒度快速检测方法, 其中多粒度指以词袋模型为基础、函数为基本单位提取特征, 通过逐级聚合特征获得 APK 多层级信息, 通过对数长度表征函数规模; 并基于 Dalvik 指令集中操作码语义上的相似性对其进行压缩映射以提升效率, 构建相应分类模型。测试表明所提方法在性能和效率上均有明显优势。

关键词: 操作码; 压缩映射; 多粒度; 快速检测; 卷积神经网络

中图分类号: TP393

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2019064

Multi-granularity Android malware fast detection based on opcode

ZHANG Xuetao, SUN Meng, WANG Jinshuang

Institute of Command Control Engineering, Army Engineering University, Nanjing 210001, China

Abstract: The detection method based on opcode is widely used in Android malware detection, but it still contains some problems such as complex feature extraction method and low efficiency. In order to solve these problems, a multi-granularity fast detection method based on opcode for Android malware was proposed. Multi-granularity refers to the feature based on the bag of words model, and with the function as basic unit to extract features. By step-by-level aggregation feature, the APK multi-level information is obtained. The log length characterizes the scale of the function. And feature can be compressed and mapped to improve the efficiency and construct the corresponding classification model based on the semantic similarity of the Dalvik instruction set. Tests show that the proposed method has obvious advantages in performance and efficiency.

Key words: opcode, compression map, multi-granularity, rapid detection, convolutional neural networks

收稿日期: 2019-03-11; 修回日期: 2019-06-14

通信作者: 王金双, siyeshishuang@163.com

论文引用格式: 张雪涛, 孙蒙, 王金双. 基于操作码的安卓恶意代码多粒度快速检测方法[J]. 网络与信息安全学报, 2019, 5(6): 85-94.

ZHANG X T, SUN M, WANG J S. Multi-granularity Android malware fast detection based on opcode[J]. Chinese Journal of Network and Information Security, 2019, 5(6): 85-94.

1 引言

近年来, 安卓平台恶意软件数量呈大幅增长的态势。相关报告显示^[1], 2018 年全年安卓平台可检测的恶意软件增加了 283.1 万个。安卓平台恶意软件种类繁多, 来源多样, 如挖矿木马、勒索软件等恶意软件多为单独开发, 隐私窃取、恶意扣费等软件则多是通过对于良性软件进行恶意代码插入获得^[2-3]。

机器学习方法已被应用于安卓恶意软件检测, 常基于敏感权限、敏感 API 调用、控制流图 (CFG, control flow graph)、APK 的操作码 (opcode) 等特征进行识别。但对于由普通安卓软件改造而来的恶意软件, 由于插入代码量少, 其恶意特征可能被其他无关代码混淆掩盖, 影响检测速度与效果。同时传统的检测方法严重依赖于专家经验以及人为制定的规则, 导致其检测效率低。而安卓恶意软件的数量不断增加, 加快了对恶意软件的查杀速度、提高了检测效率, 成为恶意软件检测系统发展的重要方向。相关检测方法对重打包等类型恶意软件检测效果较差、受其他无关变量影响较大, 以及模型训练耗时长, 且现有的检测方法常将单个安卓软件作为整体进行检测, 导致无关因素影响了恶意代码特征的显著性。为解决这一问题, 本文提出了一种以 opcode 为特征、基于机器学习技术的安卓恶意软件分段多层次快速检测方法^[4]。

本文提出将反汇编得到的 APK 文件以类中的函数为基本单位, 以词袋模型为基础进行基础特征向量的提取, 随后将提取出的多个向量进行组合再提取, 经过多次特征组合与提取获得 APK 文件由细粒度到粗粒度的特征信息, 并将其作为最终 APK 的特征值送入分类模型进行训练与检测。同时为弥补词袋模型对函数规模信息的缺失, 在特征中加入了 loglength 以补充函数规模信息, 用以提升检测准确性。

传统的基于 opcode 的相关工作常使用 Dalvik 虚拟机指令集中的全部 opcode 作为词典进行特征提取, 但由于传统汇编指令种类繁多, 导致特征空间维度过高, 特征数据量过大, 最终导致严重的运算负荷, 限制了安卓恶意软件的分类速度。为降低特征数据量, 加快模型训练与检测效率, 本文基于汇编指令语义上的相似对其进行多对一的压缩映射, 从而减少特征空间的维度, 提高安卓软件的检测效率。同时针对本文提取的特征数据的特点, 重新设计了基于 CNN 的检测模型, 通过多深度卷积核匹配更多粒度上的 opcode 信息, 增强了模型的泛化能力、提升了检测精度。

本文的主要创新如下。

1) 提出了一种基于 opcode 的安卓恶意软件多粒度特征提取方法, 同时兼顾程序的局部特征以及总体特征, 有较强的程序表征能力。

2) 为提升系统的检测效率, 在特征提取过程中使用 opcode 压缩词典, 通过语义相似性对训练数据进行压缩精简, 降低了特征空间维度, 显著提高了模型的训练与检测效率。

3) 为补足传统特征提取方法中仅包含 opcode 比例、丢失程序规模信息的缺陷, 本文提出 loglength 参数, 用于对函数规模进行评定, 有效补全了特征中的缺失信息, 提高了检测准确率。

4) 基于卷积神经网络设计了轻量化的检测模型, 通过不同深度的卷积核对数据进行挖掘, 在保证检测精度的同时降低了训练耗时。模型分类 F-score 达到 0.98, 分类精度达到 0.99, 在本文实验环境下达到了每秒 540 个安卓软件的检测效率, 相较于其他相近工作拥有更优的分类性能。

2 相关工作

安卓恶意软件检测主要可以分为静态检测与动态检测两大类, 使用的特征分别为程序静态特征或程序运行时产生的动态特征。静态分析主要

包括对 `AndroidManifest.xml` 等配置文件进行解析, 或对程序进行反汇编, 获取 APK 的函数、API 调用、程序流程图、二进制信息等静态特征。例如, Shabtai 等^[5]使用反汇编得到的类作为识别特征。动态分析则常包括沙箱模拟安卓运行 APK, 提取程序执行的动作特征。例如, Lindorfer 等^[6]使用 APK 在运行过程中的系统调用作为特征进行了恶意软件与良性软件的分类工作。动态分析可以较好地表征恶意程序的行为特征, 并在一定程度上克服程序混淆对检测带来的困难; 但相较于静态分析, 其资源开销更大, 难以对大量恶意软件进行快速分析。故安全研究人员对基于静态特征的安卓恶意软件检测开展了广泛的研究, 例如, Yerima 等^[7]使用了 API 调用、权限等多种静态信息进行检测。基于权限、敏感 API 等静态特征的分析方法常依赖于人为定义的规则, 而基于反汇编的二进制特征则无须人为设定条件, Jerome 等^[8]以操作码作为特征, 将其向量化后送入分类器进行检测, 并获得了良好的检测效果。该种检测方法的优点是检测速度快、人为因素少, 但同时容易受到无关程序片段或混淆程序片段的影响。

传统的基于 opcode 的安卓恶意代码检测方法多使用 N-Gram 模型对 opcode 序列进行处理, 如 Kang 等^[9]使用该模型对程序进行处理, 并将 N-Gram 片段作为基本单位进行统计, 获得了良好的检测效果; Yewale 等^[10]对 opcode 进行统计

挖掘, 在相应数据集上获得了 0.98 的准确率。但由于 Dalvik 指令集数量繁多, 组成序列后特征空间维度呈指数增长, 导致特征数据量过大, 影响检测效率, 故同样存在一定程度的局限性。本文针对上述问题, 提出使用程序分段的方法提升目标恶意代码在检测域中所占的相对比例, 并通过对语义相似的汇编指令进行压缩以提高检测效率。

3 基于 opcode 的多粒度快速检测方法

3.1 检测框架

本文提出的检测方法整体流程如图 1 所示。首先由 APKtool 对 APK 文件进行解包反编译, 并结合基于语义相似性的压缩映射词典以及词袋模型对 smali 文件进行特征提取; 在得到每个函数的特征信息后, 再对特征进行多轮整合再提取; 最终将特征向量送入分类模型进行判定。

3.2 基于语义的 opcode 压缩编码

造成基于 opcode 的检测系统数据量庞大的根本原因在于指令集词典空间较大, Dalvik 架构对应的指令集包含 200 余个汇编指令, 复杂多样的 opcode 可组成更加复杂的 opcode 序列。过大的特征空间会导致数据量的膨胀, 进而导致处理效率低。为应对此类问题, Chen 等^[11]提出了轻量化安卓恶意软件检测系统 TinyDroid, 在该检测系统中基于语义对 opcode 进行了类别合并, 从而降低特征空间维度并提高处理效率; Bakhshinejad

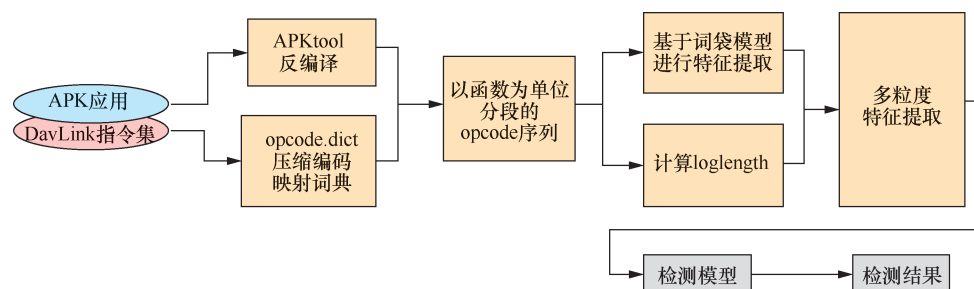


图1 检测方法整体流程

等^[12]在其提出的安卓恶意软件检测系统中基于PPM压缩算法对opcode序列进行压缩,从而提升了检测效率;Dong等^[13]在其工作中指出为提升基于操作码的检测系统的检测效率,首先需要对数据进行压缩处理。为降低特征数据量,去除冗余信息,本文基于opcode指令之间的相似性对其进行了多对一的压缩编码,以实现操作码特征空间的压缩进而提高处理效率,压缩原理如图2所示。

基于操作码的语义相似性,将所有opcode分为44组,并为每组指定编号,形成多对一的映射关系。如move、move-wide、move-wide/from16等针对不同数据类型的move指令被合并为同一类。经过压缩编码,opcode子序列被压缩为由低维数词典构成的向量,同构降低特征空间的维度,达到控制数据量的效果。为去除包含垃圾指令的子序列,本系统设计了对应的筛选机制,以过滤

干扰项。由于篇幅限制,表1中仅列出了部分按照语义进行划分的opcode及映射关系。

基于语义相似性的映射关系,将原始opcode特征空间的维度由原始的267下降至44,有效缩减特征的数据量,提高检测速度。

3.3 多粒度特征提取

为提取基于opcode的APK文件特征,本文方法首先对APK文件进行反汇编,通过正则表达式从smali文件中对opcode进行匹配获取。以函数为基本单元,按照上述压缩词典中的映射关系对APK中的opcode进行记录。为保留恶意软件的局部特征,避免过多无关代码对恶意代码检测产生影响,本文设计了以函数为基本单位进行多粒度特征提取的方法。

APK中包含的所有函数被分为若干组进行基础特征向量的提取,每组中包含函数的数量决定了特征提取过程中的起始粒度大小。为保证基

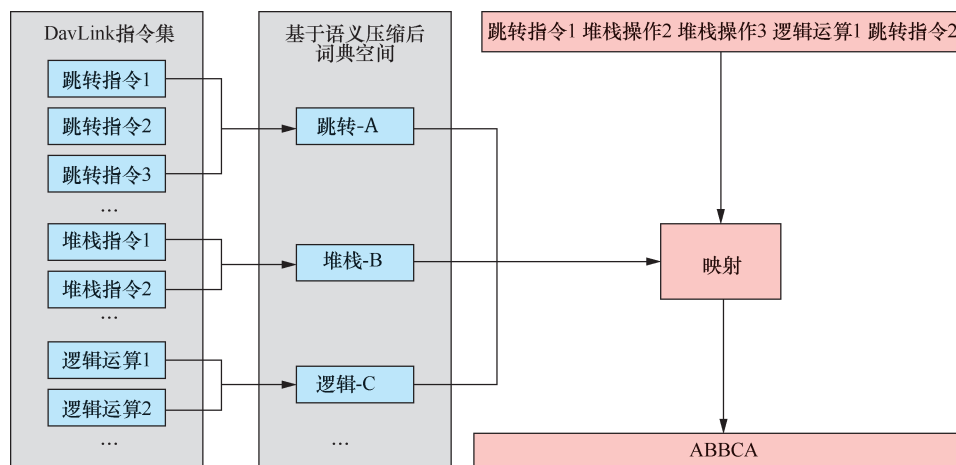


图2 opcode压缩编码

表1

opcode压缩编码对应关系

opcode 指令	编码
move, move/from16, move/16, move-wide, move-wide/from16, move-wide/16, move-object, move-object/from16, move-object/16, move-result, move-result-wide, move-result-object, move-exception	0x01
return-void, return, return-wide, return-object	0x02
const/4, const/16, const, const/high16, const-wide/16, const-wide/32, const-wide, const-wide/high16, const-string, const-string/jumbo, const-class	0x03
if-eq, if-ne, if-lt, if-ge, if-gt, if-le, if-eqz, if-nez, if-ltz, if-gez, if-gtz, if-lez	0x12
add-int, add-long, add-float, add-double, add-int/2addr, add-long/2addr, add-float/2addr, add-double/2addr, add-int/lit16, add-int/lit8	0x20

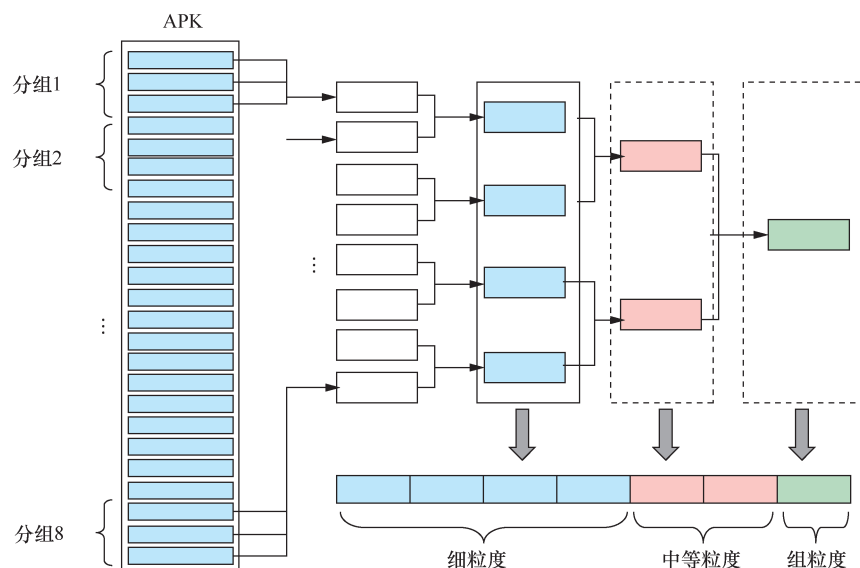


图3 多粒度特征提取

本单位特征，以两两聚合的方式进行多次合并，并在最后一次聚合后获得表征整个 APK 的最终特征向量，需要保证基本特征数目可被 2 进行 n 次整除，即需要将原始程序中所有函数分为 2^n 组以进行特征提取。如图 3 所示，假设原始 APK 中含有 24 个函数，当 $n=3$ 时，需要提取出的基础特征数量为 $2^3=8$ 个，故将原始 APK 中的函数按照每 3 个一组的形式进行共 44 类的 opcode 数目统计。当 APK 中的函数数量不能被指定的分组数量整除时，最后一组的函数数量可少于平均数量，直接进行统计。统计出的数目在最后进行归一化，以反映 opcode 在指定区域内的分布信息。

随后对基础特征向量进行两两聚合，聚合方式如图 4 所示。

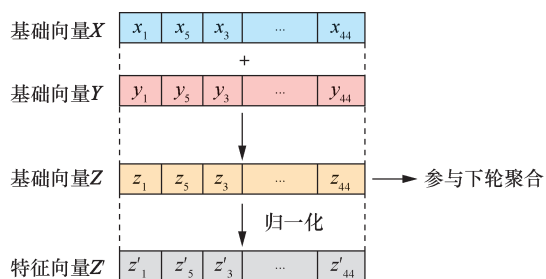


图4 基础向量聚合

假设现有基础向量 X 、 Y ，每个基础向量均包

含 44 维数据，每个维度中数值对应相应类别的 opcode 数量。为体现在指定的观察粒度下函数包含 opcode 的平均数量以及分布比例信息，聚合操作将两个子基础向量中对应类别的 opcode 数量相加并求均值，如在第一维中有 $z_1 = \frac{x_1 + y_1}{2}$ 。由于聚合操作是对操作码数量进行累计，随着观察粒度的增大，opcode 数量将会产生大幅度增长，导致大粒度信息覆盖包含 opcode 数量较少的小粒度信息。为避免这一问题，在特征提取过程中将会对特征向量 Z 进行归一化以平衡数值，将最终获得的向量 Z' 作为特征项量的组成部分。而基础向量 Z 保留了上层基础向量 opcode 的数值信息，便于进行下一轮聚合操作，故取 Z 送入下一轮迭代。循环上述过程即可完成所有特征的聚合。

以图 3 为例，通过对 8 个基础特征进行逐层聚合，分别获得 4、2、1 个高层向量，随后将前 4 层共 15 ($8+4+2+1$) 个不同粒度的特征向量进行合并，最终得到维度为 15×44 的特征矩阵，该矩阵包含 APK 文件不同区域的 opcode 分布信息，以及由分组到整体的不同粒度的特征信息。函数分组数目越大，则基础特征对于 APK

含 44 维数据，每个维度中数值对应相应类别的 opcode 数量。为体现在指定的观察粒度下函数包含 opcode 的平均数量以及分布比例信息，聚合操作将两个子基础向量中对应类别的 opcode 数量相加并求均值，如在第一维中有 $z_1 = \frac{x_1 + y_1}{2}$ 。由于聚合操作是对操作码数量进行累计，随着观察粒度的增大，opcode 数量将会产生大幅度增长，导致大粒度信息覆盖包含 opcode 数量较少的小粒度信息。为避免这一问题，在特征提取过程中将会对特征向量 Z 进行归一化以平衡数值，将最终获得的向量 Z' 作为特征项量的组成部分。而基础向量 Z 保留了上层基础向量 opcode 的数值信息，便于进行下一轮聚合操作，故取 Z 送入下一轮迭代。循环上述过程即可完成所有特征的聚合。

的表征粒度越细致,但同时会带来更大的数据量,造成模型训练与判定时间增长,为取得检测效率与准确率之间的平衡点,测试中会对不同分组数目进行测试,以寻求最优的基础特征数量。

为平衡不同粒度下获得的特征向量,在特征提取过程中对其进行了归一化。但归一化后的特征向量仅能表示在指定的程序范围内 opcode 分布的相对比例信息,丢失了函数规模信息(即原有函数中包含的 opcode 数量信息)。为增强特征的表征能力,保留函数相对应的规模信息,在原有基础特征后加入 $\loglength$,用于表示函数规模,其计算方法如式 1 所示:

$$\loglength = \frac{\min \left\{ \max \left\{ \begin{matrix} 8 \\ 1 \end{matrix} \log_8 \log_8 \text{func_size} \right\} \right\}}{10} - 1 \quad (1)$$

式(1)中的 func_size 为基础单位向量中含有 opcode 的数目,首先通过对数函数将数值较大的 func_size 映射到较小的空间中,若计算结果小于 1,则取 1;若函数值超过 8,则函数尺寸取 8,计算得出的数值向下取整并减 1,最终获

得区间在 0~7 之间的整数,数值越大,代表函数规模越大。其中,对数函数以 8 为底,同时将函数规模分为 8 个级别为实验测得的最佳结果,可对函数规模进行较为均衡的评估。对于粗粒度的特征向量,计算 $\loglength$ 时使用的是平均 func_size ,即总 opcode 数量除以包含的函数数目。

由于上述过程中提取的特征向量均进行了归一化,其数值在 0~1.0 之间,为避免区间在 0~7 之间的函数规模信息在数值上对特征造成扰动,在计算 $\loglength$ 时令相对应的整数除 10,令其区间也落在 0~1.0 之间,作为特征向量的第 45 维加入。经过上述处理,最终获得的特征矩阵规模由

$$\sum_0^n 2^n \cdot 44 \text{ 扩展至 } \sum_0^n 2^n \cdot 45。$$

3.4 基于卷积神经网络的检测模型

在文本分类等领域,卷积神经网络(CNN)发挥了良好的分类性能,Yoon Kim 等^[14]在其工作中提出了使用 CNN 进行文本分类,获得了良好的检测效果。如图 5 所示,与分类任务相类似,本文方法使用一维卷积作为分类网络的基本单元。

如图 5 所示,APK 的特征数据输入后,首先

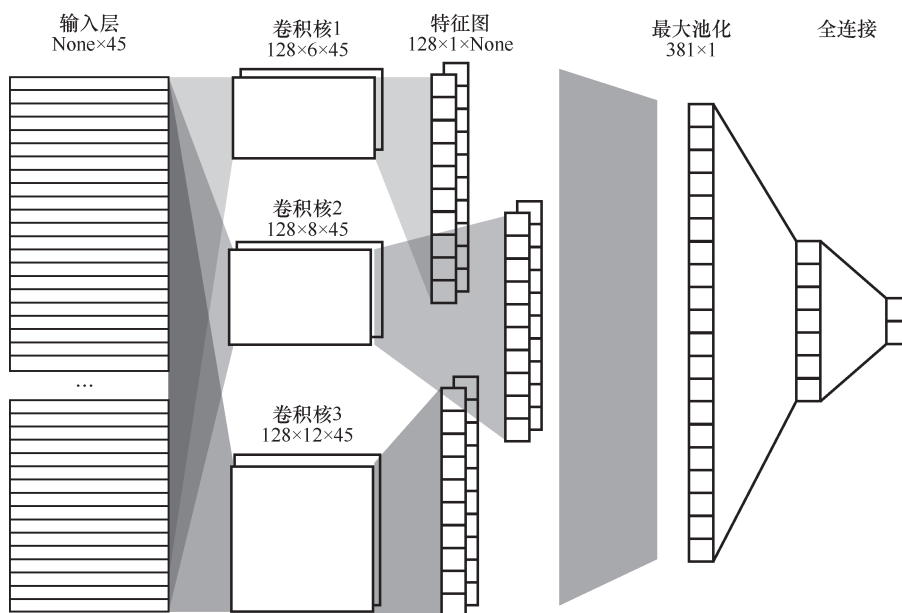


图 5 分类模型结构

使用3个拥有不同尺寸卷积核的卷积层对矩阵进行卷积操作。模型使用一维卷积，卷积核宽度与输入数据宽度一致，每个卷积层包含128个卷积核，卷积核深度分别为6, 8, 12。通过卷积核深度差异覆盖不同范围的基础向量，从而进一步挖掘隐藏信息。随后将得到的特征图进行连接合并，基于最大值池化将特征整合为381维的向量，最后通过全连接输出到一个二维向量，即模型判定样本分别属于正类以及负类的概率。为防止模型产生过拟合现象，模型中加入了DropOut层用于防止模型过拟合。近期Sepp Hochreiter提出的selu激活函数^[15]在模型训练以及梯度回传方面表现优异，本模型中采用selu作为激活函数实现二分类问题。

检测系统通过反汇编及opcode静态特征提取，以分组再提取的方式增强了特征对于APK的描述能力，通过压缩编码的方式对数据量进行压缩并提升检测效率，无须针对应用程序动态执行且检测精度高，有利于应用于大规模应用检测系统的部署。

4 实验结果与分析

4.1 检测模型训练与测试

模型训练流程如下。

1) APK应用反编译，使用APKTools对应用程序进行反编译，并使用正则表达式从中进行opcode的提取，并将每个APK对应的提取结果以单个文件的形式保存。

2) 特征提取，基于Dalvik字节码语义相似性的压缩编码词典，以词袋模型为基础进行字节码分布统计，并加入loglength表征函数规模信息，最终单个样本对应规模为31×45的特征矩阵。

3) 模型训练与测试，使用上文所述的基于CNN分类模型进行训练，使用Tensorflow以及Tflearn实现分类模型中的相关结构，并将网络输出的二维向量作为分类结果。

通过实验，评估本文所提算法在以下3个方面的有效性。

1) APK文件使用的单位特征向量数目与分类效果的关系，找到在满足精度条件下需要的最少特征向量数目，以提高分类效率。在实验过程中分别使用前1~6层基础向量，并分别记录对应情景下模型的分类精度以及召回率，用以对模型分类的准确性进行验证评估。

2) 使用原始opcode映射关系进行检测，对比使用压缩词典后的训练耗时以及分类精度之间的关系，以验证本文使用的基于语义相似性的映射词典的有效性。

3) 与其他相关工作进行对比，证明本文所提方法在检测效率以及检测精度上的优势。

4.2 数据、平台和评价指标

本文实验环境为CPU Intel(R) E5, 16 GB内存，GPU为GTX1080，硬盘为120 GB SSD，使用Ubuntu16.04操作系统。实验数据集中恶意APK应用来自VirusShare病毒数据库，良性样本来源于Drebin安卓恶意软件检测数据集^[16]，以及小米应用商城爬取，其中良性样本共计5 031个、恶意样本共计5 003个，样本总量为10 034，并将其按照9:1的比例随机划分为训练集以及测试集。

模型的分类性能主要使用准确率(Precision)、召回率(Recall)、F-score以及漏报率(FNR)这4个参数进行衡量，参数的计算方法如下。

$$recall = \frac{TP}{TP + FN} \quad (2)$$

$$precision = \frac{TP}{TP + FP} \quad (3)$$

$$F-score = \frac{2precision \cdot recall}{precision + recall} \quad (4)$$

其中，TP为真阳率，FP为假阳率，TN为真阴率，FN为假阴率。而在上式计算的参数中，漏报率接近0说明检测效果越好，反之，Recall越接近1，

则说明模型分类性能越强。

4.3 结果分析

首先为获得最小特征数据量与最优检测效果的平衡点：在测试中分别使用前 1~6 层基础特征，即 n 值分别为 0、1、2、3、4、5 时，对模型的分类精度、召回率、FNR 以及 F-score 这 4 项参数进行测试；测试分为包含 `loglength` 函数规模信息、不包含函数规模信息两组分别进行，以验证 `loglength` 的有效性。测试结果如表 2 及表 3 所示。

表 2 包含 `loglength` 特征向量下模型测试结果

n 值	基础特征数量	<i>precision</i>	<i>recall</i>	<i>FNR</i>	<i>F-Score</i>
5	32	0.992 7	0.976 1	0.023 9	0.984 4
4	16	0.981 1	0.983 4	0.016 6	0.982 2
3	8	0.982 8	0.957 0	0.043 0	0.969 8
2	4	0.973 5	0.930 9	0.069 1	0.951 7
1	2	0.977 4	0.911 0	0.089 0	0.943 0
0	1	0.971 1	0.893 6	0.106 4	0.930 7

其次以不包含 `loglength` 的特征值为基础，对模型进行上述 4 项参数的测试，相应结果如表 3 所示。

表 3 未包含 `loglength` 特征向量下模型测试结果

n 值	基础特征数量	<i>precision</i>	<i>recall</i>	<i>FNR</i>	<i>F-Score</i>
5	32	0.975 0	0.958 2	0.041 8	0.966 5
4	16	0.964 8	0.962 5	0.037 5	0.963 7
3	8	0.959 4	0.974 8	0.025 2	0.967 0
2	4	0.954 5	0.938 8	0.061 2	0.946 6
1	2	0.949 4	0.931 4	0.068 6	0.940 3
0	1	0.943 1	0.927 0	0.073 0	0.935 0

由于模型为二分类模型，使用 F-score 可以较为全面地对模型的分类性能进行衡量，综合表 2 以及表 3 中的数据，将基于两种特征的 F-score 测试结果进行绘制，结果如图 6 所示。

由图 6 中数据进行分析可知，由于 `loglength` 包

含函数规模信息，相较于不使用该项特征的分类模型，F-score 值有一定程度的提升，说明模型拥有更低的漏报率与更好的分类性能。对图中数据走势进行分析可以得到：随着使用的基础特征数量增加，模型分类性能在不断上升，但随着数量不断增长，基础特征数量为 16、 $n=4$ 时分类模型获得最好效果，随后随着基础特征数量不断增长，模型分类性能的提升并不明显，并有可能出现分类效果不稳定的现象。出现该现象的主要原因是无关函数数量过多对分类结果带来了影响。综上所述，在初始阶段将 APK 中所有函数分为 16 组，基于每组提取 1 个基础特征向量。在获得 16 个基础特征向量后，按照上文所述聚合方式进行逐层聚合处理，最终特征总维度为 31×45 时，可以获得最优的检测效果以及在不牺牲精度前提下的最高效率，即 $n=4$ 时可以获得检测效果与检测效率之间的均衡。

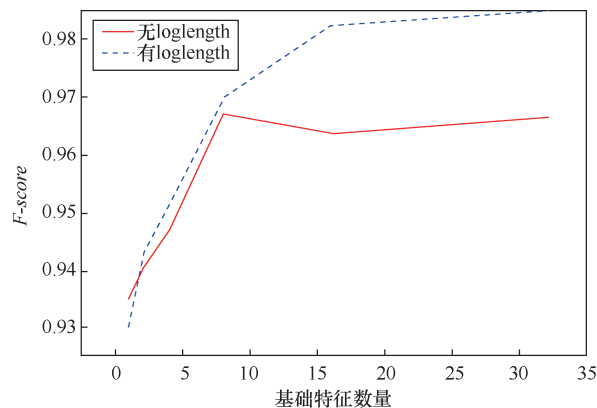


图 6 基于两种特征的 F-score 对比

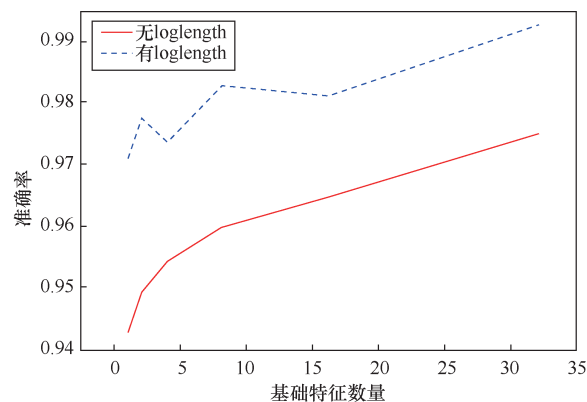


图 7 基于两种特征的准确率对比

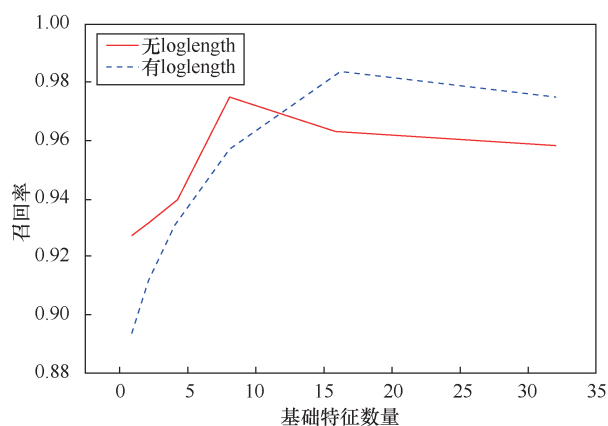


图8 基于两种特征的召回率对比

对使用两种特征的召回率以及精度进行绘制,如图7及图8所示。可以得出, `loglength` 有效提升了判定精度,并在APK误判方面有更优表现。本文方法最优判定精度可达到0.99。

McLaughlin等^[17]提出了一种新型基于Dalvik操作码的恶意APK检测方法。该方法针对传统基于N-Gram及opcode检测方法流程复杂的问题,提出了一种直接使用opcode进行分类的恶意APK检测方法,提高了对于软件的检测效率。该方法中同样使用opcode进行恶意软件的检测,与本文方法不同的是,该方法将APK作为一个整体提取所有的opcode信息,并使用一对一映射,即每个opcode均对应唯一的标识符,故特征空间较大,并使用卷积对提取出的opcode序列进行学习训练,最终获得分类结果。

基于相同的数据集,本文与该方法进行了对比测试,将该方法最优成绩与本文方法最优成绩进行对比,测试结果如表4所示。

表4 对比测试结果

检测方法	<i>precision</i>	<i>recall</i>	<i>FNR</i>	<i>F-Score</i>
文献[17]	0.99	0.95	0.05	0.97
本文方法	0.99	0.97	0.03	0.98

从表4中的数据分析对比可以得到,本文提出方法相较于文献[11]方法拥有更好的分类性能以及

误判率,各项参数指标均优于该方法。同时对本方法的测试效率进行统计,本方法对单个样本进行判别需要耗费0.00185s,即每秒可判定样本540个,故使用了压缩映射的opcode词典,本方法有较高恶意软件检测效率。

为验证本检测系统相较于同类工作在训练以及判定方面的时间优势,本文从训练集中随机抽取了2000个样本用于模型训练耗时测试:分别对Arp D等^[16]提出的安卓恶意软件检测框架Drebin, McLaughlin等^[17]提出的基于opcode的检测方法进行训练,模型训练耗时以及检测精度记录如表5所示。

表5 模型训练耗时及检测精度

检测方法	<i>T/s</i>	<i>precision</i>
文献[16]	54	0.97
文献[17]	35	0.98
本文方法	17	0.98

同类型的基于opcode的检测工作常以N-Gram模型为基础进行特征组织,这导致数据量较大,进而导致模型在训练与判定过程中的开销增大。故Deep Android Malware Detection检测系统在训练过程中耗时较高。而本文提出方法由于多粒度的组织方法,增强其表达能力,同时基于压缩词典对数据规模进行缩减,在同等检测精度的条件下,相较于同类模型训练效率显著提高。

5 结束语

本文针对传统的基于opcode的安卓恶意软件检测系统特征维度过高、检测效率较低且无法表征APK局部信息等缺点,提出了一种新型基于opcode的多粒度特征提取方式:通过将APK分段提取opcode特征,并将特征进行组合提升模型与检测效率,并加入loglength对函数规模进行表示,补全了词袋模型对于函数长度信息表

征方面的缺失。同时为提高系统的分类与检测效率,本文采用了压缩映射编码的方法对原始 Dalvik 指令集进行了压缩映射,通过降低特征空间维度有效减少了特征数据量,降低训练时间并提高了检测效率;针对基于 opcode 的特征特点,基于 CNN 对分类模型进行了设计。

通过对模型进行多项参数测试,验证了本文提出算法的有效性,证明了本检测框架相较于同类型工作在检测精度、检测效率等方面的优势。但本检测框架在未来仍有提高效率的空间,可通过信息增益等方式对 opcode 进行进一步筛选,而进一步压缩特征空间的维度以提高检测效率;并在未来工作中对分类器模型进一步完善,提升检测性能并增强模型的泛化能力。

参考文献:

- [1] 360 互联网安全中心. 2018 中国手机安全生态报告[R]. 2018.
360 Internet Security Center. 2018 China mobile phone security ecological Report [R]. 2018.
- [2] ZHOU W, ZHOU Y J, JIANG X X, et al. Detecting repackaged smartphone applications in third-party Android marketplaces[C]//The Second ACM Conference on Data and Application Security and Privacy. 2012: 317-326.
- [3] 汪润,唐奔霄,王丽娜. DeepRD: 基于 Siamese LSTM 网络的 Android 重打包应用检测方法[J].通信学报, 2018, 39(8): 69-82.
WANG R, TANG B X, WANG L N. Deeprd: Android repackaging application detection method based on Siamese LSTM network [J]. Journal on Communications, 2018, 39(8): 69-82.
- [4] TIAN K, YAO D D, RYDER B G, et al. Detection of repackaged android malware with code-heterogeneity features[J]. IEEE Transactions on Dependable and Secure Computing, 2017:1-1.
- [5] 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
- [6] LINDORFER M, NEUGSCHWANDTNER M, PLATZER C. MARVIN: efficient and comprehensive mobile app classification through static and dynamic analysis[C]// IEEE Computer Software & Applications Conference. IEEE Computer Society. 2015.
- [7] YERIMA S Y, SEZER S, MUTTIK I. High accuracy android malware detection using ensemble learning[J]. Iet Information Security, 2016, 9(6): 313-320.
- [8] JEROME Q, ALLIX K, STATE R, et al. Using opcode-sequences to detect malicious Android applications[C]//2014 IEEE International Conference on Communications (ICC). 2014: 914-919.
- [9] KANG B J, YERIMA S Y, MCLAUGHLIN K, et al. N-opcode analysis for android malware classification and categorization[C]//2016 International Conference on Cyber Security and Protection of Digital Services (Cyber Security). 2016: 1-7.
- [10] YEWALE A, SINGH M. Malware detection based on opcode frequency[C]//IEEE International Conference on Advanced Communication Control & Computing Technologies. 2017.
- [11] CHEN T, MAO Q, YANG Y, et al. TinyDroid: a lightweight and efficient model for android malware detection and classification[J]. Mobile Information Systems, 2018: 1-9.
- [12] BAKHSHINEJAD N, HAMZEH A. A new compression based method for android malware detection using opcodes[C]//IEEE Artificial Intelligence and Signal Processing Conference. 2018: 256-261.
- [13] DONG H, HE N Q, HU G, et al. Malware detection method of android application based on simplification instructions[J]. The Journal of China Universities of Posts and Telecommunications, 2014, 21: 94-100.
- [14] KIM Y. Convolutional neural networks for sentence classification[C]//The 2014 Conference on Empirical Methods in Natural Language Processing. 2014.
- [15] KLAMBAUER G, UNTERTHINER T, MAYR A, et al. Self-normalizing neural networks[C]//Advances in Neural Information Processing Systems 30 (NIPS 2017)2017: 971-980.
- [16] ARP D, SPREITZENBARTH M, HUBNER M, et al. Drebin: effective and explainable detection of android malware in your pocket[C]//NDSS. 2014: 23-26.
- [17] MCLAUGHLIN N, RINCON J M D, KANG B J, et al. Deep android malware detection[C]//ACM on Conference on Data & Application Security & Privacy. 2017.

[作者简介]



张雪涛(1995-), 男, 河北保定人, 硕士, 主要研究方向为网络安全以及恶意软件检测。



孙蒙(1984-), 男, 山东齐河人, 博士, 陆军工程大学副教授, 主要研究方向为人工智能和网络安全。



王金双(1978-), 男, 黑龙江佳木斯人, 博士, 陆军工程大学副教授, 主要研究方向为系统安全, 机器定理证明。