



研究与开发

基于字节码图像和深度学习的 Android 恶意应用检测

陈铁明, 项彬彬, 吕明琪, 陈波, 江颀

(浙江工业大学计算机科学与技术学院, 浙江 杭州 310023)

摘要: 提出一种将字节码转换成彩色图像后, 再采用深度学习模型的新型 Android 恶意应用检测方法。首先将 Android 应用的字节码文件映射为三通道的 RGB 彩色图像, 同时计算局部信息熵值, 并将其作为 Alpha 通道, 与 RGB 图像融合为带透明度的 RGBA 彩色图像, 最后利用卷积神经网络方法对图像进行分类, 实现一个 Android 恶意应用检测原型系统。通过对 8 种恶意代码家族进行分类实验验证, 并与灰度图等其他同类可视化成像方法进行对比, 发现该方法具有检测速度快、精确度高等优点。

关键词: 恶意代码检测; 字节码图像; 香农信息熵; 深度学习; 分类

中图分类号: TP309

文献标识码: A

doi: 10.11959/j.issn.1000-0801.2019022

Android malware detection method based on byte-code image and deep learning

CHEN Tieming, XIANG Binbin, LV Mingqi, CHEN Bo, JIANG Xie

College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China

Abstract: A new Android malware detection method based on byte-code image and deep learning was proposed. Firstly, Android malware byte-code files were mapped to RGB colorful images which had three channels. Also, the Shannon entropy as Alpha channel of images were calculated, and then merged with RGB images into RGBA images. Finally, the convolutional neural network as classifier was employed to classify aforementioned images. According to the experiment on malware of eight malicious families and compared this method with the method which mapping the byte-code to gray image, the result shows that the method using RGBA images has good performance not only in speed, but also in accuracy.

Key words: malware detection, byte-code image, Shannon entropy, deep learning, classification

1 引言

Android 开源系统的出现迅速推动了全球互

联网应用的发展, 同时也引发了诸多安全问题。

参考文献[1]中指出 2017 年 Android 平台平均每天新增恶意软件 2.1 万个, 平均每天恶意软件感染量

收稿日期: 2018-05-25; 修回日期: 2019-01-09

基金项目: 国家自然科学基金资助项目 (No.61202282, No.61772026); 国家自然科学基金与浙江省政府联合项目 (No.U1509214)

Foundation Items: The National Natural Science Foundation of China (No.61202282, No.61772026), The Joint Project of National Natural Science Foundation and Zhejiang Provincial Government (No.U1509214)



约为 58.8 万人次, 监测到 Android 用户感染恶意软件 2.14 亿个, 其中超过 80% 的恶意软件以资费消耗的形式牟取不正当经济利益。当前恶意软件使用了多种新技术以达到破坏和绕过检测的目的, 例如新技术针对系统运行库攻击、利用 Telegram 软件协议进行远程控制操作等。同时, 报告指出具备自动化和对抗能力的恶意软件工厂正在量产恶意 App, 这种技术能探测杀毒软件的检测模型, 推断杀毒软件所使用的特征和检测规则, 结合探测的信息和混淆技术构造特定的 APK (Android package, Android 安装包), 以绕过杀毒软件检测。恶意软件生成与检测方法形成对抗式发展, 恶意软件层出不穷, 检测方法也需推陈出新。

当前大部分安全厂商使用基于特征代码的检测方法。该类方法通过检测文件的特征代码来判断其是否为恶意软件。它的优点是速度快、准确率高且误报率低, 但是需要及时更新特征库, 难以抵御短时间爆发的恶意应用。同时该类方法易受代码混淆和加密的干扰。恶意软件可以通过加壳、变形等技术手段阻止研究员逆向程序, 进而增加检测的难度和成本。以目前被认为最安全的 Google Play 商店为例, 有很多研究指出已有大量的恶意软件绕过 Google Play 商城的安全检测。

基于行为的检测方法不易受代码混淆和加密的影响, 但是需要收集恶意应用大量信息, 例如权限、函数调用、运行行为等。该类方法对未知的恶意应用检测效果优于基于特征代码的检测方法, 但是需要消耗大量的计算资源和时间。现阶段亟待提出一种检测速度快、时效性高、能抗混淆的检测方法。

本文提出一种利用恶意代码可视化的 Android 检测方法。从 APK 中提取 Dalvik 字节码文件, 计算字节码文件的局部信息熵值, 将信息熵和字节码文件结合, 映射为包含四通道的 RGBA 彩色图像, 利用卷积神经网络的图像识别

能力, 对其进行检测和分类。

本文主要的贡献如下: 将恶意应用可视化为代码图像, 省略代码反编译过程, 结合香农信息熵降低代码混淆带来的影响, 弥补代码图像来源单一的缺陷, 最后尝试了利用卷积神经网络提取图像特征进行恶意应用代码分类, 最终实现自动化检测。

2 相关工作

2.1 基于代码图像的 Android 检测

大部分恶意应用仅是在某个恶意应用的基础上增加少量模块的变种, 两者之间有大量的复用代码, 具有明显的家族特征^[2]。恶意代码可视化方法旨在将恶意代码通过可视化方法转化为图片, 结合代码同源性分析, 提取图像特征, 使用机器学习方法进行分类。2011 年, Nataraj 等^[3]提出一种将桌面端恶意代码映射为灰度图, 并从灰度图中提取 GIST 特征, 通过 K 最近邻算法进行分类的方法。Kancherla 等^[4]使用 Gabor 滤波器对灰度图进行信号处理获得特征, 使用 SVM 进行分类。韩晓光等^[5]将恶意代码映射为无压缩灰阶图片, 运用基于纹理的分割算法对图像进行分块, 通过算法提取各块的纹理特征作为纹理指纹, 同时建立纹理指纹索引, 最后采用加权综合多分段纹理指纹相似性匹配方法检测。

2.2 基于深度学习的 Android 检测

在计算机视觉和语言处理方面, 深度学习因其卓越的效果被大量地应用。参考文献[6]将 DEX 文件映射为 RGB 图像, 将百万级别训练数据送入深度学习 Inception-v3 模型进行训练; DeepDroid 结合 Android 的动态和静态特征^[7], 使用深度置信网络 (DBN) 作为训练模型进行检测; 参考文献[8]将代码图像作为静态特征, 收集恶意代码活动生成向量空间, 利用栈式自编译模型提取特征并分类。

上述工作运用多种图像处理技术提取图像特征, 使用多种深度学习模型进行建模分类。本文

主要研究恶意代码图像结合香农信息熵的检测方法,即将恶意代码结合香农信息熵转化为代码图片,运用卷积神经网络模型,并将其应用在Android恶意检测方面。

3 本文方法

3.1 系统模型

本文的基本思路如下:首先将恶意APK分类解压,得到DEX文件。该DEX文件包含有Dalvik字节码,能够被Dalvik虚拟机理解并运行,同时该文件包含应用程序运行的主要逻辑和数据。将所得到的DEX文件转换为RGB彩色图像,同时计算DEX文件的局部信息熵,将信息熵作为透明度,即融合代码图像和信息熵为一张RGBA图像,最后利用卷积神经网络对图像进行检测。同理,按照恶意代码家族的分类建立训练集,按照上述处理过程,可以建立一个恶意代码家族的分类模型。

在利用该系统进行检测与分类时,先对待检测的APK进行预处理,提取DEX文件,通过上述可视化方法生成彩色代码图像,再通过恶意代码检测模型进行检测。若结果为恶意应用,则使用恶意代码家族分类模型对其进行分类,以获知该恶意应用的恶意代码家族类型。

系统模型如图1所示,其中实线所表示的流程为模型的训练过程,虚线表示的流程为样本的检测分类过程。

3.2 字节码图像处理

Android应用通常以“.apk”结尾,其包含的文件目录如图2所示。重要文件主要有两个,其中“AndroidManifest.xml”是Android应用程序的全局配置文件,在该文件中保存了组件配置信息和程序所需的权限。classes.dex文件是运行在Dalvik虚拟机上的可运行文件,该文件包含了整个程序的运行逻辑和变量空间分配,因此是整个程序的核心。由于图片转换操作十分消耗CPU和内存,且转换后的图片文件比源文件更大。因此

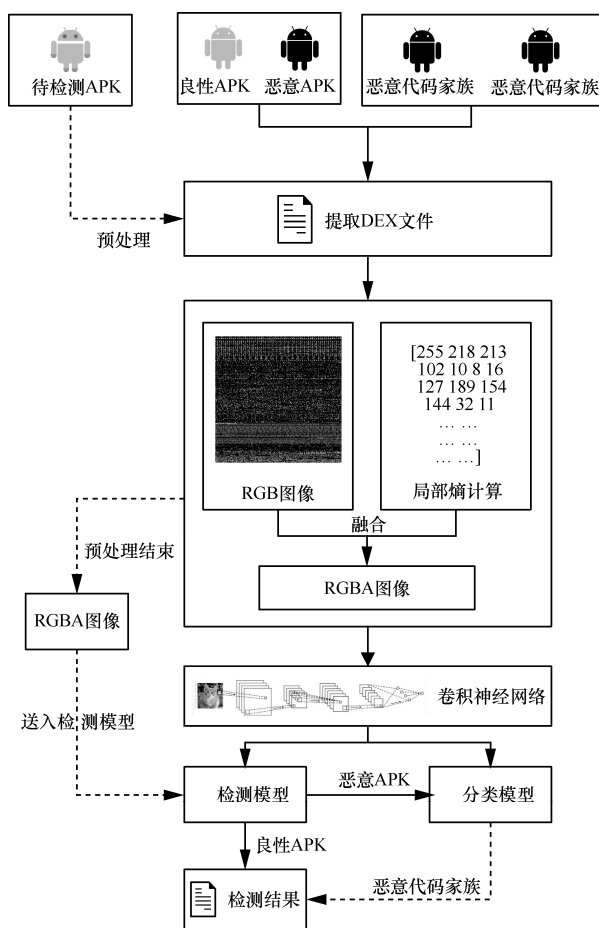


图1 系统模型

本文挑选核心的DEX文件,仅将其转换为图片。DEX文件被分为三大块:文件头、索引区和数据区。文件头记录DEX文件的一些基本信息以及大致的大致数据分布;索引区有整个DEX文件中的字符串、类型、方法声明、字段以及方法的信息等的索引,并且索引区中的偏移最终都指向数据区^[9]。

通常彩色图像有红绿蓝(RGB)3个颜色通道,每个通道有8位(256种色级)。因此将DEX文件按字节码读取,每个字节包含8bit,取值的范围为0x00~0xFF,即0~255。将DEX文件划分为3byte长度的数据块,将数据块映射为图片的RGB值(例:FFB6C1=R:255 G:182 B:193),长度不足的地方使用0进行填充,即可以将DEX文件转化为RGB彩色图像,并将其存储为PNG图片。图3是DEX文件结构与RGB图像的对应关系图,由

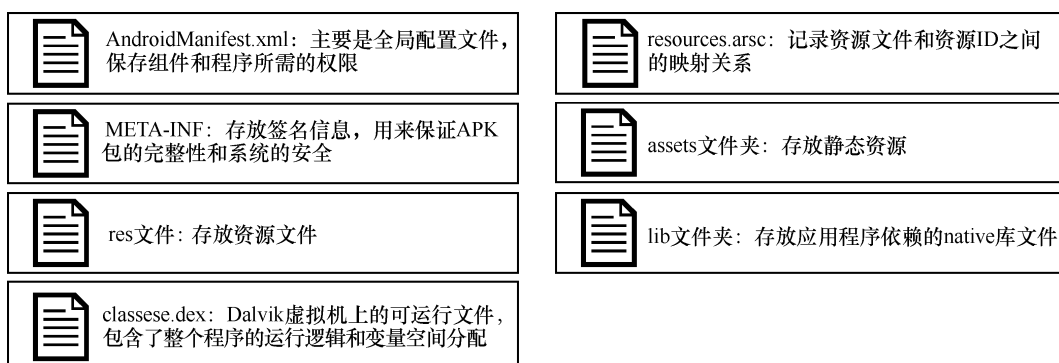


图2 APK逆向文件目录

图3可以看出, 相同的家族之间存在一定的相似性, 不同的家族之间有较大差异。基于图像的映射方法能有效地避免码混淆所造成的漏报, 同时由于相同家族的图像呈现一定程度的相似性, 因此代码图像能作为一种分类特征^[10]。相比较于灰度图只拥有单通道, 彩色图像拥有RGB三通道, 因此RGB彩色图像的有效信息量是灰度图的3倍, 能为之后的检测提供更多更准确的特征。

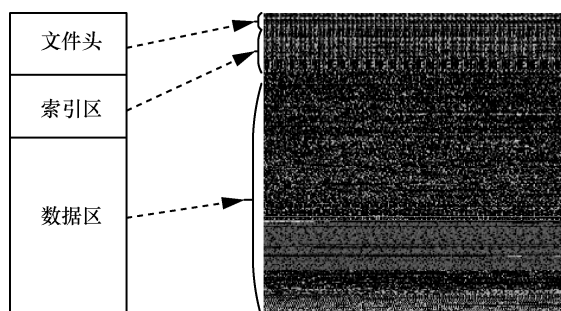


图3 DEX文件结构与RGB图像对应关系

3.3 信息熵处理

信息是用来消除随机不确定的东西, 香农在1948年提出的信息熵是用来描述这种随机变量的不确定性的, 表示无序的信息^[11]。如果在一个数据集中, 某个元素出现的频率几乎为100%, 则该数据集的信息熵几乎为0, 反之, 若数据集中的元素均匀出现, 则信息熵的值最大。因此, 信息熵经常被用于自然语言处理中的数据划分。逆向工程需要关注信息熵, 因为有的恶意代码存在信息加密或信息压缩, 在这种情况下, 信息熵值一般会比较高^[12]。

局部信息熵的具体计算方法如下。

步骤1 将DEX文件预处理, 按长度每256 byte进行划分, 末尾不足256 byte则用0进行填充。

步骤2 对每个数据块使用式(1)计算信息熵值, 其中, j 表示数据块中出现的特定字节, 即0x00~0xFF, p_j 表示字节 j 出现的概率。

$$\text{entropy} = - \sum_{j=0}^{255} j \cdot \text{lb} p_j \quad (1)$$

步骤3 通常熵值的计算结果 entropy 在[0,8], 而Alpha通道的取值范围在0~255, 因此将熵值以指数形式放大, 放大后的熵值范围为0~255, 和Alpha通道的区间范围一致^[13]。

步骤4 由于熵值是以每256 byte作为数据块进行的划分, 因此每个熵值代表85个或86个像素点的透明度, 按此规则映射为完整的Alpha通道的矩阵。

步骤5 将RGB图像与Alpha矩阵相结合, 生成新的RGBA图像, 图4为不同家族的RGBA图。

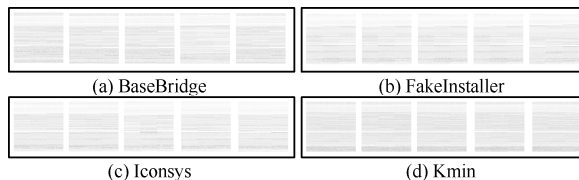


图4 RGBA不同恶意家族图

生成RGBA图像的具体实现如下。

算法1 RGBA图像生成

输入: Android字节码文件 F

输出: RGBA 图像 P

Begin

$M \leftarrow \text{hex}(\text{readFile}(F));$

$MS \leftarrow \text{split}(M, 3);$

For each S in MS : /*彩色图映射*/

$r \leftarrow \text{Mapper}(S, \text{RED});$

$g \leftarrow \text{Mapper}(S, \text{GREEN});$

$b \leftarrow \text{Mapper}(S, \text{BLUE});$

$MK \leftarrow \text{split}(M, 256);$

For each K in MK : /*局部信息熵计算*/

$T \leftarrow \text{calculateEntropy}(k);$

$T \leftarrow \text{intRound}(T8)$

$Z \leftarrow \text{concatArray}(Z, T, 85);$ /*扩展熵值, 与 RGB 图相符*/

$P \leftarrow \text{SavePic}(Z);$

$p \leftarrow \text{resizePic}(P, 256, 256);$

End

3.4 深度学习算法

卷积神经网络是一种受到了生物神经网络“激活”现象的启发, 引入激活函数和卷积核而构建的机器学习模型。卷积神经网络通常包含以下几层: 卷积层、线性整流层、池化层和全连接层。卷积层的目的是提取输入的不同特征, 第一层的卷积层可能只提取低级特征, 例如边缘、线条等, 深层的卷积层可能提取更高级、更复杂的特征。卷积层会产生许多维度较大的特征, 用池化层可以很好地对特征进行降维, 最常用的有最大池化法和均值池化法。全连接层在整个模型中起到分类器的作用^[14]。在卷积神经网络中, 卷积核大小、

卷积核数量和网络深度等参数对分类的效果和训练时间有至关重要的影响^[15]。

图 5 为本实验使用的卷积神经网络模型。通过卷积运算, 可以增强特征, 同时减少噪声。不同的卷积核能提取不同的特征图 (feature map), 因此模型使用两层 32 个 3×3 卷积核进行特征提取, 并采用 ReLU 作为激活函数。通过 2×2 的池化层将特征图进行压缩, 从而减少计算量。最后通过全链接层送入 softmax 分类器进行分类。

3.5 机器学习方法及评估标准

参考文献[16]介绍了常见的分类算法性能评估参数。一般在分类算法中, 有基本指标真正例 (true positive, TP)、假正例 (false positive, FP)、真负例 (true negative, TN) 以及假负例 (false negative, FN), 并由这些基本指标构成其他更能准确表达分类器效果的指标。由这 4 个基本指标可以构成表 1 所示的常用指标。真正例率 (true positive rate, TPR) 代表能将正例正确分类的概率, 即命中率, 有时也称为召回率。假正例率 (false positive rate, FPR) 代表将正例分类错误的概率, 即误报率。精确度 (precision) 代表正确分类的正样本占总正样本的数量。F-measure 代表精确度和召回率的调和均值。准确率 (accuracy) 为所有正确分类样本与所有样本之比。

4 实验与分析

4.1 实验数据

本文实验所使用的实验数据来源于德国哥廷根大学 DREBIN 项目^[17]的公开数据集。DREBIN

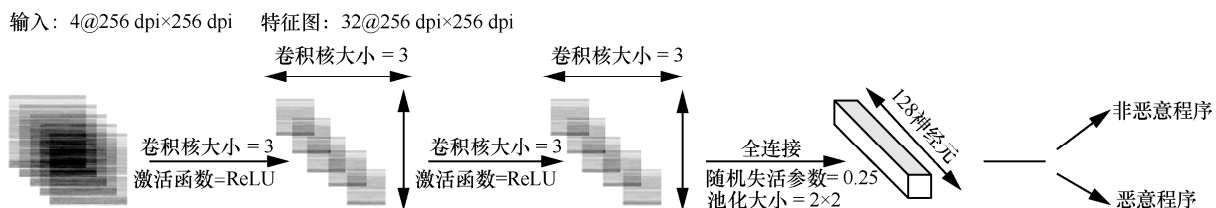


图 5 卷积神经网络模型



表 1 评估指标

名称	计算式
TPR	$TP/(TP+FN)$
FPR	$FP/(FP+TN)$
精确度	$TP/(TP+FP)$
召回率	$TP/(TP+FN)$
F-measure	$2 \times \text{precision} \times \text{recall} / (\text{precision} + \text{recall})$
准确率	$(TP+TN)/(TP+TN+FP+FN)$

数据集一共包含有 5 560 个恶意应用, 涵盖 179 个恶意代码家族。本文实验使用数据集中样本数量较多的几大恶意代码家族作为恶意应用样本。同时从谷歌商店、360 软件商城等 Android 软件市场获取良性应用作为良性样本。

实验设备环境见表 2。

表 2 实验设备环境

硬件	说明	软件	说明
GPU	GTX1060 6GB	CUDA	9.0
CPU	Intel i3-2420	cuDNN	7.02
RAM	8 GB	编程语言	Python

4.2 多种图像生成方式对比

图像的生成方式是决定分类效果的关键, 因此本实验将与其他几种图像生成方式进行对比。

(1) GrayImage (灰度图): 将 DEX 文件映射为灰度图, 图像大小为 256 dpi×256 dpi。

(2) RGBImage (彩色图): 将 DEX 文件映射为 RGB 三通道彩色文件, 图像大小为 256 dpi×256 dpi。由于彩色图像拥有三通道, 因此训练的总参数量为灰度图的 3 倍。

(3) ShannonImage (信息熵图): 计算 DEX 文件的局部信息熵, 将熵值进行指数方法放大, 将相邻信息熵作为 X 轴、Y 轴, 形成信息熵图。

所形成图像如图 6 所示。

本文随机选取 1 000 个样本作为训练集, 500 个样本作为测试集, 采用相同的卷积神经网络模型分类, 使用表 2 介绍的指标作为评估指标。实验结果见表 3。

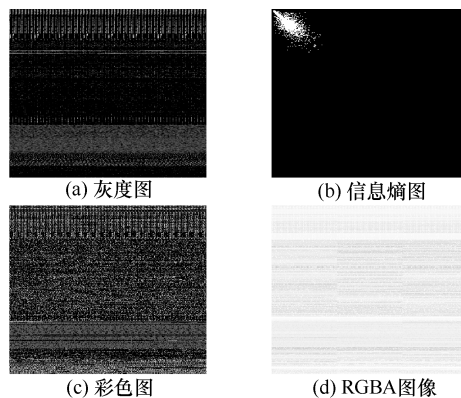


图 6 4 种代码成像

表 3 不同图像实验对比结果

图像生成方式	精确度	召回率	F-measure	准确率
灰度图	90.8%	99.0%	94.7%	94.5%
彩色图	95.1%	97.0%	96.0%	96.1%
信息熵图	94.1%	97.0%	95.6%	95.5%
RGBA 图	96.0%	98.4%	97.2%	97.2%

从实验结果可以看出, RGBA 图像相对于其他几种图像, 在各个指标均获得较优结果。因为其包含更多的信息, 因此能提供更多特征, 从而获得更好的分类效果。

接下来分析样本数量对准确率的影响。实验分别使用 500 个训练样本、1 000 个训练样本、1 500 个训练样本和 2 000 个训练样本进行对比实验。实验结果如图 7 所示, 可以得出结论: 随着样本数的增加, 刚开始对准确率有较大的提升, 但是超过一定数量之后提升的幅度不明显, 而训练时间随着样本的增加呈现线性的增加趋势。

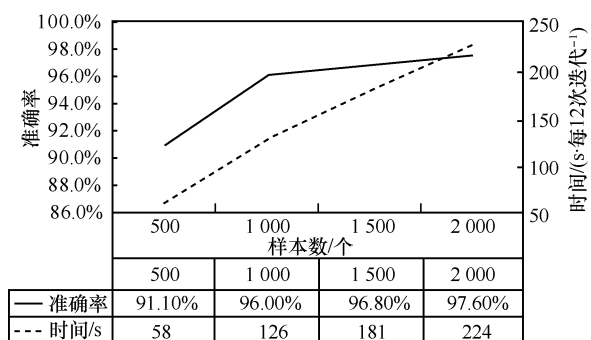


图 7 实验样本数量影响

4.3 恶意代码家族分类

对恶意代码家族进行分类并对分类效果进行评估。由于每个家族恶意样本较少,因此选取8个恶意样本较多的家族进行实验,每个家族取50个训练样本和25个测试样本。实验结果见表4,总体上平均检测率高于90%,平均误报率为1.1%,表明该方法对恶意代码家族分类有较好效果。

4.4 相关工作比较

本文在样本量、检测率、误报率、准确率和

检测时间方面与 DroidSieve^[18]收集的方法进行性能对比,对比结果见表5。由表5可知,在小样本情况下,本文提出的方法与其他方法相比具有较高的检测率和准确率以及较低的误报率。本文方法在检测时主要的耗时操作为生成图片,相对于提取静态API类和语义分析类方法,本文方法无需进行反编译等耗时操作,因此在检测时间方面耗时较短。综上所述,本文方法在Android恶意检测方面具有较好的效果。

表4 RGBA图像家族分类结果

家族名称	TPR	FPR	精确度	召回率	F-measure	准确率
BaseBridge	80%	0.6%	95.2%	80%	87.0%	97%
DroidDream	80%	2.9%	80%	80%	80%	95%
FakeDoc	100%	2.8%	83.3%	100%	90.9%	97.5%
Geinimi	84%	2.3%	84%	84%	84%	96%
Iconosys	100%	0	100%	100%	100%	100%
Kmin	100%	0.5%	96.1%	100%	98%	99.5%
Opfake	100%	0	100%	100%	100%	100%
Plankton	92%	0	100%	92%	95.8%	99%

表5 各个方法检测对比

年份	方法	样本量/个	检测率/误报率	准确率
2014年	Dendroid ^[19]	1 247	—	94%
2014年	DroidAPIMiner ^[20]	3 987	99%/2.2%	—
2014年	DroidMiner ^[21]	2 466	95.3%/0.4%	92%
2014年	Drebin ^[17]	5 560	94.0%/1.0%	—
2014年	DroidSIFT ^[22]	2 200	98.0%/5.15%	93%
2014年	DroidLegacy ^[23]	1 052	93.0%/3.0%	98%
2015年	AppAudit ^[24]	1 005	99.3%/0.61%	—
2015年	Mudflow ^[25]	10 552	90.1%/18.7%	—
2015年	Marvin ^[26]	15 741	98.24%/0	—
2015年	RevealDroid ^[27]	9 054	98.2%/18.7%	93%
2016年	DroidScribe ^[28]	5 246	—	94%
2016年	Madam ^[29]	2 800	96%/0.2%	—
2017年	DroidSieve ^[18]	16 141	99.3%/0	99%
2017年	R2-D2 ^[30]	829 356	96%/9%	93%
2018年	本文方法	2 000	97.6%/2.4%	98.6%



5 结束语

本文研究了恶意代码可视化方法, 不同于 InsDal^[31]所提出的方法, 该方法对 DEX 文件进行语义化分析, 在相关方法处插入调用提示, 从而实现动态分析。本文方法并未对 DEX 文件进行语义化分析, 而是通过将 DEX 文件映射为 RGB 图, 结合香农信息熵, 生成 RGBA 四通道恶意代码图像, 并使用卷积神经网络进行特征提取和分类。本文使用相同的训练集和测试集对不同的图像生成策略进行比较, 得出 RGBA 拥有更高的准确率的结论。通过对常见的 8 种恶意代码家族样本进行分类测试, 结果表明, 本文提出的方法在家族分类方面具有较高的准确率和较低的误报率。

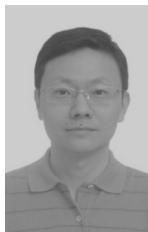
下一步工作将尝试结合更多可视化方法, 将 Android 应用的权限文件 (AndroidManifest.xml 文件) 也加入可视化范围, 尝试使用其他有效特征信息, 优化现有的深度学习模型, 增加网络深度和卷积核的数量以达到更好的分类效果。同时尝试收集更多样本, 使用更大的数据集进行试验, 用以进一步地提高准确率、降低误报率。

参考文献:

- [1] 360 烽火实验室. 2017 年 Android 恶意软件专题报告[Z]. 2018. 360 Fiber Home Lab. 2017 Android malware special report[Z]. 2018.
- [2] 蔡林, 陈铁明. Android 移动恶意代码检测的研究概述与展望[J]. 信息安全, 2016(9): 218-222.
CAI L, CHEN T M. Research review and outlook on Android mobile malware detection[J]. Netinfo Security, 2016(9): 218-222.
- [3] NATARAJ L, KARTHIKEYAN S, JACOB G, et al. Malware images: visualization and automatic classification[C]//8th International Symposium on Visualization for Cyber Security, July 20, 2011, Pittsburgh, Pennsylvania, USA. New York: ACM Press, 2011.
- [4] KANCHERLA K, MUKKAMALA S. Image visualization based malware detection[C]//2013 Computational Intelligence in Cyber Security, April 16-19, 2013, Singapore. Piscataway: IEEE Press, 2013.
- [5] 韩晓光, 曲武, 姚宣霞, 等. 基于纹理指纹的恶意代码变种检测方法研究[J]. 通信学报, 2014, 35(8): 125-136.
HAN X G, QU W, YAO X X, et al. Research on malicious code variants detection based on texture fingerprint[J]. Journal on Communications, 2014, 35(8): 125-136.
- [6] HUANG H D, KAO H Y. R2-D2: ColoR-inspired convolutional neural network (CNN)-based Android malware detections[J]. arXiv:1705.04448v5, 2017.
- [7] 苏志达, 祝跃飞, 刘龙. 基于深度学习的安卓恶意应用检测[J]. 计算机应用, 2017(6): 1650-1656.
SU Z D, ZHU Y F, LIU L. Android malware application detection using deep learning[J]. Journal of Computer Applications, 2017(6): 1650-1656.
- [8] 罗世奇, 田生伟, 禹龙, 等. 基于纹理指纹与活动向量空间的 Android 恶意代码检测[J]. 计算机应用, 2018, 38(4): 1058-1063.
LUO S Q, TIAN S W, YU L, et al. Android malware detection based on texture fingerprint and malware activity vector space[J]. Journal of Computer Applications, 2018, 38(4): 1058-1063.
- [9] ELENKOV N. Android security internals: an in-depth guide to Android's security architecture[M]. [S.l.:s.n.], 2014.
- [10] 杨益敏, 陈铁明. 基于字节码图像的 Android 恶意代码家族分类方法[J]. 网络与信息安全学报, 2016, 2(6): 38-43.
YANG Y M, CHEN T M. Android malware family classification method based on the image of bytecode[J]. Chinese Journal of Network and Information Security, 2016, 2(6): 38-43.
- [11] SHANNON C E. A mathematical theory of communication[J]. Bell System Technical Journal, 1948, 27(4): 379-423.
- [12] 任卓君, 陈光. 熵可视化方法在恶意代码分类中的应用[J]. 计算机工程, 2017(9): 167-171.
REN Z J, CHEN G. Application of entropy visualization method in malware classification[J]. Computer Engineering, 2017(9): 167-171.
- [13] 任卓君, 韩秀玲, 孔德凤, 等. 像素归一化方法在恶意代码可视分析中的应用[J]. 计算机工程与应用, 2016, 52(21): 121-125.
REN Z J, HAN X L, KONG D F, et al. Pixel normalization method applied in malware visualization analysis[J]. Computer Engineering and Applications, 2016, 52(21): 121-125.
- [14] LECUN Y, BOTTOU L, BENGIO Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.
- [15] KARBAB E M B, DEBBABI M, DERHAB A, et al. Android malware detection using deep learning on API method sequences[J]. arXiv: 1712.08996, 2017.
- [16] HAN J, KAMBER M. Data mining: concepts and techniques[J]. Data Mining Concepts Models Methods & Algorithms Second Edition, 2011, 5(4): 1-18.
- [17] ARP D, SPREITZENBARTH M, HÜBNER M, et al. DREBIN: effective and explainable detection of Android malware in your pocket[C]//2014 Network and Distributed System Security Symposium, February 23-26, 2014, San Diego, California, USA. [S.l.:s.n.], 2014.
- [18] SUAREZ-TANGIL G, DASH S K, AHMADI M, et al.

- DroidSieve: fast and accurate classification of obfuscated Android malware[C]//2017 ACM Conference on Data and Application Security and Privacy, March 22-24, 2017, Scottsdale, Arizona, USA. New York: ACM Press, 2017.
- [19] SUAREZ-TANGIL G, TAPIADOR J E, PERIS-LOPEZ P, et al. Dendroid: a text mining approach to analyzing and classifying code structures in Android malware families[J]. Expert Systems with Applications, 2014, 41(4): 1104-1117.
- [20] AAFER Y, DU W, YIN H. DroidAPIMiner: mining API-level features for robust malware detection in Android[J]. Lecture Notes of the Institute for Computer Sciences Social Informatics & Telecommunications Engineering, 2013(127): 86-103.
- [21] YANG C, XU Z, GU G, et al. DroidMiner: automated mining and characterization of fine-grained malicious behaviors in Android applications[C]//2014 European Symposium on Research in Computer Security, September 6-10, Wroclaw, Poland. Heidelberg: Springer, 2014.
- [22] ZHANG M, DUAN Y, YIN H, et al. Semantics-aware Android malware classification using weighted contextual API dependency graphs[C]//The 2014 ACM SIGSAC Conference on Computer and Communications Security, November 3-7, 2014, Scottsdale, Arizona, USA. New York: ACM Press, 2014: 1105-1116.
- [23] DESHOTELS L, NOTANI V, LAKHOTIA A. DroidLegacy: automated familial classification of Android malware[C]//2014 ACM Sigplan on Program Protection and Reverse Engineering Workshop, January 22-24, 2014, San Diego, CA, USA. New York: ACM Press, 2014.
- [24] XIA M, GONG L, LYU Y, et al. Effective real-time Android application auditing: security and privacy[C]//2015 IEEE Symposium on Security and Privacy, May 17-21, 2015, San Jose, CA, USA. Piscataway: IEEE Press, 2015.
- [25] AVDIENKO V, KUZNETSOV K, GORLA A, et al. Mining Apps for abnormal usage of sensitive data[C]//2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, May 16-24, 2015, Florence, Italy. Piscataway: IEEE Press, 2015: 426-436.
- [26] LINDORFER M, NEUGSCHWANDTNER M, PLATZER C. MARVIN: efficient and comprehensive mobile App classification through static and dynamic analysis[C]//2015 IEEE Computer Software and Applications Conference, July 1-5, 2015, Taichung, China. Piscataway: IEEE Press, 2015.
- [27] GARCIA J, HAMMAD M, MALEK S. Lightweight, obfuscation-resilient detection and family identification of Android malware[C]//2018 IEEE/ACM 40th International Conference on Software Engineering, May 27-Jun 3, 2018, Gothenburg, Sweden. Piscataway: IEEE Press, 2018.
- [28] DASH S K, SUAREZ-TANGIL G, KHAN S, et al. DroidScribe: classifying Android malware based on runtime behavior[C]//2016 IEEE Security and Privacy Workshops, May 22-26, 2016, San Jose, CA, USA. Piscataway: IEEE Press, 2016.
- [29] SARACINO A, SGANDURRA D, DINI G, et al. MADAM: effective and efficient behavior-based Android malware detection and prevention[J]. IEEE Transactions on Dependable & Secure Computing, 2018(99): 1.
- [30] HUANG H D, KAO H Y. R2-D2: ColoR-inspired convolutional neural network (CNN)-based Android malware detections[C]//OWASP AppSec USA 2017, September 19-22, 2017, Orlando, FL, USA. [S.l.:s.n.], 2017.
- [31] LIU J, WU T, DENG X, et al. InsDal: a safe and extensible instrumentation tool on dalvik byte-code for Android applications[C]//IEEE International Conference on Software Analysis, Evolution and Reengineering, Feb 20-24, 2017, Klagenfurt, Austria. Piscataway: IEEE, 2017: 502-506.
- [32] 陈昊, 卿斯汉. 基于组合式算法的 Android 恶意软件检测方法[J]. 电信科学, 2016, 32(10): 15-21.
- CHEN H, QING S H. Android malware detection method based on combined algorithm[J]. Telecommunications Science, 2016, 32(10): 15-21.

[作者简介]



陈铁明 (1978-), 男, 博士, 浙江工业大学计算机科学与技术学院教授, 主要研究方向为网络空间安全与大数据智能分析。



项彬彬 (1994-), 男, 浙江工业大学计算机科学与技术学院硕士生, 主要研究方向为信息安全。



吕明琪 (1981-), 男, 博士, 浙江工业大学计算机科学与技术学院副教授, 主要研究方向为数据挖掘与普适计算。

陈波 (1971-), 男, 浙江工业大学计算机科学与技术学院副教授, 主要研究方向为信息安全、移动开发和图像处理。

江颀 (1972-), 女, 博士, 浙江工业大学计算机科学与技术学院副院长、副教授, 主要研究方向为信息安全和电子服务。