

基于程序基因的恶意程序预测技术

肖达^{1,2}, 刘博寒^{1,2}, 崔宝江^{1,2}, 王晓晨^{1,2}, 张索星^{1,2}

(1. 北京邮电大学网络空间安全学院, 北京 100876;

2. 移动互联网安全技术国家工程实验室, 北京 100876)

摘要: 随着互联网技术日益成熟, 恶意程序呈现出爆发式增长趋势。面对无源码恶性未知可执行文件, 当前主流恶意程序检测多采用基于相似性的特征检测, 缺少对恶性性来源的分析。基于该现状, 定义了程序基因概念, 设计并实现了通用的程序基因提取方案, 提出了基于程序基因的恶意程序预测方法, 通过机器学习及深度学习技术, 使预测系统具有良好的预测能力, 其中深度学习模型准确率达到99.3%, 验证了程序基因理论在恶意程序分析领域的作用。

关键词: 程序基因; 动态分析; 基本块; 恶意程序预测

中图分类号: TP309.5

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2018069

Malware prediction technique based on program gene

XIAO Da^{1,2}, LIU Bohan^{1,2}, CUI Baojiang^{1,2}, WANG Xiaochen^{1,2}, ZHANG Suoxing^{1,2}

1. School of Cyberspace Security, Beijing University of Post and Telecommunications, Beijing 100876, China

2. National Engineering Lab for Mobile Network Security, Beijing 100876, China

Abstract: With the development of Internet technology, malicious programs have risen explosively. In the face of executable files without source, the current mainstream malware detection uses feature detection based on similarity, with lack of analysis of malicious sources. To resolve this status, the definition of program gene was raised, a generic method of extracting program gene was designed, and a malicious program prediction method was proposed based on program gene. Utilizing machine learning and deep-learning algorithms, the forecasting system has good prediction ability, with the accuracy rate of 99.3% in the deep-learning model, which validates the role of program gene theory in the field of malicious program analysis.

Key words: program gene, dynamic analysis, basic block, malware prediction

1 引言

在互联网技术日益成熟的今天, 网络安全态势却不乐观, 各类恶意程序数量呈井喷式增长, 各种信息安全事件屡见不鲜。2017年5月和6月, 名为“WannaCry”和“Petya”的勒索软件利用操

作系统漏洞感染了全球100多个国家和地区的数十万台主机系统, 造成的损失难以预计。根据《腾讯安全2017年度互联网安全报告》数据显示, 腾讯电脑管家在2017年全年共拦截病毒30亿次, 发现6.3亿台主机感染病毒或木马^[1]。相似的, 根据瑞星公司发布的《2017年中国网络安全报告》

收稿日期: 2018-06-20; 修回日期: 2018-07-20

通信作者: 崔宝江, cuibj@bupt.edu.cn

基金项目: 国家自然科学基金资助项目 (No.U1536122, No.61502536)

Foundation Item: The National Natural Science Foundation of China (No.U1536122, No.61502536)

显示,瑞星“云安全”系统在2017年全年共截获病毒样本总量5 003万个,病毒感染次数29.1亿次,病毒总体数量比2016年同期上涨15.62%^[2]。此外,恶意程序还呈现出变种多、抗检测技术更新快的发展情况。当前主流的恶意程序识别技术已经难以满足恶意程序迅猛发展的安全性需求,如何对未知二进制程序进行系统性分析,准确快速识别程序的恶意性成为当前研究的热点。目前主流的恶意识别技术主要是静态分析技术和动态分析技术。

静态分析技术主要是通过二进制程序静态反汇编技术,提取程序的反汇编指令、文件格式等特征。Santos等^[3]将操作码序列的长度与出现的频率的乘积作为特征向量,通过计算两向量之间的余弦相似度识别恶意代码变体。孙博文等^[4]通过字符串等离散信息,提取程序大小、可见字符串等特征,并利用特征融合和集成学习对恶意软件进行分类,取得了较好的准确率。白金荣等^[5]针对PE文件引用的API、资源目录表、节区头属性等特征,采用Bagging等分类算法构建模型分类器,达到了99%的准确率。但静态分析技术具有一定的局限性,如无法识别用户函数、无法抵御恶意程序中常用的对抗加壳、多态、混淆等抗分析技术。

动态分析技术主要利用QEMU^[6]、Pin^[7]等动态插桩工具沙箱获取系统调用、指令流序列程序行为。曹梦晨^[8]利用Pin插桩平台实现了无感知沙箱可提取程序的网络行为、文件行为、注册表行为等信息。Li等搭建了CSS(crystal security sandbox)^[9]沙箱对Adware、Trojan和Worm三大恶意软件记录恶意软件的文件、注册表等恶意行为,最后使用SVM分类模型建立恶意软件识别分类器。Ivan等^[10]利用开源的沙箱Anubis对恶意程序进行程序行为监控,在沙箱生成的XML日志文件中提取特征再建立稀疏矢量模型,最后使用KNN、SVM、多层感知器神经网络(MIP)等分类算法建立分类模型,准确率最高达96.8%。动态分析技术具有对常见抗静态分析技术的规避能力,但由于程序具体执行环境复杂,提取信息繁多,很难从海量的指令数据流中提取出具有识别作用的特征数据。

上述研究大多数针对恶意程序具有的结构

相似性或恶意程序系统调用的相似性来对恶意程序进行分析,分析结果一般是程序的恶意性与否或执行的危险行为列表,并未从程序行为来源进行分析。2014年,Chan P P F等^[11]提出可以利用程序运行时的堆内存信息检测。Java程序功能的相似性,并称之为动态软件胎记(software birthmark making)。Liu K等^[12]提出使用系统调用序列与程序依赖数据来作为软件基因,以对抗软件程序的盗版技术,检测算法的抄袭。此时,软件基因主要用于检测软件的同源性。2018年,韩金等针对Android操作系统的恶意软件提出了恶意软件基因^[13]的概念,该研究提出从同源性的角度看待程序恶意行为,利用静态分析的方法提取了资源段和代码段恶意软件基因,并采用机器学习方法构建恶意程序分类器对恶意程序分类。该理论针对程序基因的提取,基于对程序资源段和代码段静态分析,但PC端程序的资源段并不固定,且PC端的程序代码段中多采用加壳、混淆等抗分析技术,因此该理论不适用于对PC端恶意程序的分析。

由于当前PC端程序大多数为无源码、恶意性未知的可执行文件,本文针对该类程序提出程序基因的概念,从程序真实执行的汇编指令层面定义并研究程序基因。程序基因旨在研究程序基因与程序行为间的关系,即程序基因具有的行为表达能力、遗传性质等,希望对无源码的可执行文件提取程序基因,基于程序基因对程序行为及同源性进行数据挖掘和人工智能分析,以解决当前日益严峻的程序恶意性、恶意程序分类、APT攻击溯源等问题。本文从程序恶意性入手,通过对千余个Windows平台的恶意程序及正常程序提取程序基因,建立恶意程序预测模型,以验证程序基因理论在分析恶意程序中的佐证作用。本文的创新点如下。

1) 针对无源码的二进制可执行文件,定义了通用的汇编指令层程序基因概念,并基于程序动态监控技术,实现了通用的程序基因提取方法,提取的指令是程序实际执行的汇编指令,规避了加壳等反分析手段的影响,与程序行为更加贴近。

2) 提出了基于程序基因的恶意程序预测方

法,采用多种机器学习模型和深度学习模型,具有较高的预测能力。

2 程序基因与提取方法

2.1 程序基因的定义

为研究无源码、恶意性未知程序的行为及程序行为来源问题,本文提出了程序基因概念,并将程序基因定义为具有一定功能、能反映某些行为的汇编指令片段,是控制程序行为的基本遗传单位。

基因在生物学上的定义是控制生物性状的基本遗传单位,而程序的性状反映在程序表现出的独特行为上,二进制程序实质上执行的是大量的汇编指令,汇编指令的顺序、类型决定了程序行为。与生物学上的基因相比,程序基因与其具有极大的相似性。首先,程序基因稳定存储于程序文件中,并在程序执行过程发挥作用实现程序自身的功能行为。其次,程序基因可以通过程序的更新、二次开发、引用等一系列行为在程序引用链中遗传,具有该程序基因的程序在执行时会产生特定的行为。从分子组成上来看,基因由4种脱氧核糖核苷酸组成,脱氧核糖核苷酸中碱基的排列顺序构成了生物的多样性;而程序基因由多种排列各异的汇编指令组成,不同的汇编指令组合,反映了程序的不同行为。通过研究程序基因有助于分析程序功能,对程序的相似性与同源性判定也有一定作用。

2.2 程序基因的提取方法

根据程序基因的定义,程序基因是具有一定功能能反映某些行为的汇编指令片段。在一个二进制程序中,汇编指令片段从粒度上可分为:镜像文件、程序节段、函数、指令流、基本块和汇编指令6个层次^[7]。其中,镜像文件包含二进制程序相应的所有数据结构,包括程序自实现代码、加载的动态链接库等全部代码;程序节段是PE文件中的程序段,如代码段、数据段、资源段等;函数是实现程序某项特定功能的代码片段;指令流是单入口多出口的指令序列;基本块是单入口单出口的指令序列,是顺序无跳转的最大汇编指令序列;汇编指令是程序执行最基本的单元,由操作码和操作数组成。出

于对程序基因稳定性和遗传性的考虑,本文以基本块为程序基因的提取粒度。由于本文研究的样本包含大量恶意程序,其中可能存在修改注册表、加密磁盘等不可逆恶意行为,因此借助项目组之前研究的PinFWSandBox^[14]无感知沙箱对程序行为及指令流进行监控,该沙箱可以有效地监控并回滚恶意行为以保护主机安全性。

2.2.1 基于Pin的程序指令流快照提取

Pin是Intel公司提供的-一个程序插桩平台,能在可执行程序的任意位置插入代码,以便对程序进行动态监控分析^[7]。Pin平台为用户提供了之前提及镜像文件等6个粒度的API,通过调用这些API,编写动态链接库(DLL, dynamic link library)文件类型的插件,称为Pintool。Pin平台根据Pintool中编写的插桩代码,对可执行程序翻译并执行,实现Pintool中编写的功能。其具体工作流程如图1所示。

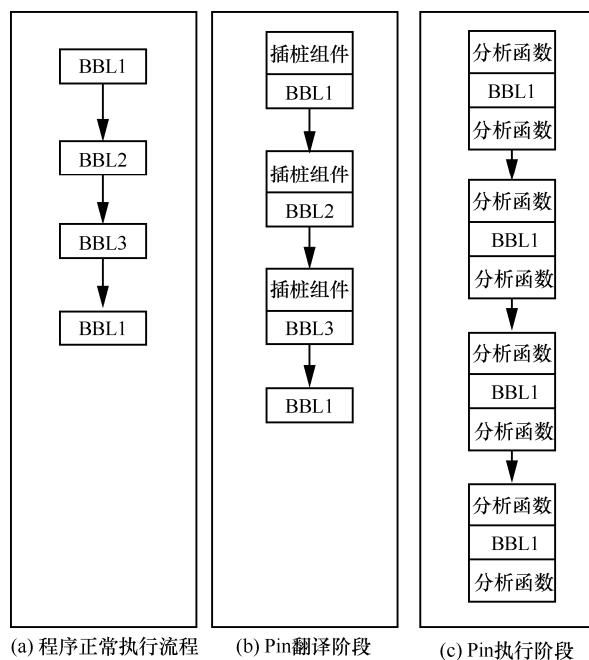


图1 Pin平台工作流程

可以看出Pin的工作流程可分为翻译阶段和执行阶段。在翻译阶段,Pin不会对相同的被插桩代码段重复翻译,仅在第一次翻译时根据Pintool中的函数干涉程序的执行流程。在程序的执行阶段,在每一次被插桩代码段执行前后都会调用Pintool中的分析函数,以便在执行时得到相应的寄存器状态、指令操作、内存读写情况等行

为信息。在本文的指令流快照提取模块中, Pintool 以 Pin 平台提供基本块插桩 API, 达到在基本块粒度提取指令流快照的目的。

2.2.2 基于 Pin 的指令流快照存储

由于程序实际执行时汇编指令的数量庞大, 重复性高, 若详细记录每一条汇编指令会造成大量存储空间的浪费。本文采用以基本块为基本存储单位, 每个基本块仅在第一次执行时记录其全部汇编指令, 其余情况下仅以索引号的形式记录基本块执行顺序, 以节省存储空间, 并通过重放执行基本块流的方式恢复程序执行的实际指令流。通过编写 Pintool 插件中的分析函数代码生成程序指令流快照日志, 日志共分为 3 个部分。

1) 程序加载的镜像文件日志, 命名格式为“待提取程序名称_image_list.fw”, 该日志记录了加载的全部自实现函数及动态链接库的起始地址、结束地址和名称。

2) 基本块执行序列日志, 命名格式为“待提取程序名称_process.fw”, 该日志以基本块索引号的形式记录了程序在实际运行过程中的基本块执行顺序。

3) 基本块指令序列日志, 命名格式为“待提取程序名称_bbl_list.fw”, 该日志在 Pin 平台的翻译阶段内进行记录, 详细记录了每一个 BBL 的索引号、所在的镜像文件号、指令地址及包含的全部汇编指令。

经过上述两部分的提取, 可得到待提取程序实际运行的全部汇编指令流, 以此可作为“原始程序基因库”。由于“原始基因库”中包含了大量程序基本块数据, 其中一部分基本块是每一个二进制程序都会存在的, 如函数对栈的初始化、函数结束调用时对栈的销毁等, 这部分基因对判定程序的行为起到作用很小, 因此, 可根据基本块调用频率设定阈值对这部分基因进行剔除, 剔除以后可以得到程序基因库。

3 恶意程序预测模型构建

3.1 恶意程序预测模型简介

利用第 2 节构建的程序基因库, 本节针对程序恶意性进行研究, 构建了恶意程序预测模型,

实现对未知二进制程序恶意性的预测, 从而验证程序基因在对程序行为研究的可行性。

该模型的输入是程序基因库提取得到的程序基本块数据。首先, 利用 n -gram 序列提取方法, 构造步长不同的操作码序列。其次, 计算各操作码序列对整体的信息增益, 从而提取出贡献信息量较多的操作码特征。然后, 利用 Word2vec 方法对操作码特征进行抽象编码, 以得到具有良好语义相关性的特征词。最终利用多种机器学习算法及深度学习算法实现了基于程序基因的恶意程序预测模型。

本模型创新性在于模型对动态提取得到的大量汇编指令级数据进行分析, 通过信息增益方法解决了动态分析技术中数据量庞大不利于分析的问题; 类文本挖掘技术, 本模型将程序基因作为文本处理, 采用语义相关性较好的 Word2vec 抽象编码方法提取特征词, 利用上述特征构建了基于多种机器学习方法的分类器, 取得了良好的准确性。

3.2 基于 n -gram 的操作码序列提取

程序基因库中包含大量汇编指令语句, 在汇编指令语句中, 操作码决定了操作执行的行为, 而汇编指令的逻辑顺序对执行结果有很大影响, 因而本项目采用相对受编译器影响较小的操作码作为研究的特征, 选取 n -gram 预测模型提取操作码。

在程序指令流快照提取过程中, 并未完全记录全部汇编指令执行序列, 因此需要对记录的日志进行重放, 以获取完整的操作码顺序, 流程如下^[15]。

1) 读取基本块指令序列日志, 并以字典的方式记录该日志内容, 将基本块序号及基本块汇编指令以键值对方式存储。

2) 遍历基本块执行序列日志的全部内容, 并按照存储的逻辑结构从 1) 生成的字典中提取汇编指令的操作码, 并按照阈值对基本块进行筛选。

3) 根据 n -gram 模型中 n 值的大小分别对 2) 中生成的操作码序列进行提取, 并记录 n -gram 提取到的每一条操作码序列项出现的次数, 本文 n 的取值分别为 1、2、3、4。

经过上述 3 个步骤, 可以提取得到基于

n -gram 的 4 类操作码序列。

3.3 基于信息增益的特征选择

通过 3.2 节中的操作码序列提取过程将得到大量的操作码序列,但由于操作码数量过于庞大,不利于后续的计算分析,本文利用各操作码的信息增益对特征进行筛选,得到信息增益较大的操作码特征,在保证预测结果准确的前提下,尽可能减少计算量。

信息增益是特征选择中常用的一种评价标准,描述的是特征为预测系统贡献信息量的多少,贡献信息量越多的特征,对预测系统越重要。在信息论中,特征 X 的信息增益定义为系统熵与固定特征 X 后的条件熵之差。其中,系统熵用来描述特征 X 的不确定性,对于该预测系统,设样本空间为 $\{C_i, i=1,2,\dots,n\}$, 则该预测系统的熵为^[16]

$$H(C) = -\sum_{i=1}^n P(C_i) \log P(C_i)$$

对于预测系统的特征 X , 其样本空间为 $\{x_i, i=1,2,\dots,n\}$, 预测系统的条件熵为

$$H(C|X) = -\sum_{i=1}^n \sum_{j=1}^m P(C_i, x_j) \log(P(C_i | x_j))$$

则对于任意特征 x_i , 预测系统的条件熵为

$$H(C|x_i) = -\sum_{j=1}^n P(C_j | x_i) \log(P(C_j | x_i))$$

从而可以得到

$$H(C|X) = \sum_{i=1}^n P(x_i) H(C|x_i)$$

对于本预测系统,由于特征 X 只有出现与否 2 种可能,假设 X 出现的概率为 $P(x)$, 不出现的概率为 $P(\bar{x})$, 则可将上式简化为

$$H(C|X) = P(x)H(C|x) + P(\bar{x})H(C|\bar{x})$$

根据信息增益的定义,得到对于特征 X 的信息增益公式为

$$\begin{aligned} IG(X) = & -\sum_{i=1}^n P(C_i) \log P(C_i) + \\ & P(x) \sum_{i=1}^n P(C_i | x) \log(P(C_i | x)) + \\ & P(\bar{x}) \sum_{i=1}^n P(C_i | \bar{x}) \log(P(C_i | \bar{x})) \end{aligned}$$

为了统计各特征的信息增益,首先分别对每个 n -gram 序列特征在恶意和非恶意程序中出现的次数和出现的样本数进行统计,然后利用信息增益公式求得每个 n -gram 序列特征值,从而按照阈值筛选特征。

3.4 特征语义抽象表示方法

为了方便对 3.3 节中筛选得到的特征进行计算,需要对特征进行语义抽象表示以转化成为向量。本文采用了 Word2vec 方法, Word2vec^[17] 是一种由“输入层—隐藏层—输出层”构成的 3 层神经网络模型,可将输入的自然语言中的词转换为任意维度的行向量,并将语言文本转换为相同维度向量空间的向量运算,在 Word2vec 内部采用了 2 种人工神经网络的分类算法——CBOW (continuous bag of words)^[18] 和 Skip-gram^[19] 模型,最终得到每一个词的最优向量,且在多个特征词间具有良好的语义相似度。本文中 Word2vec 的处理对象是由 3.2 节筛选后的 4 种 n -gram 操作码序列,使用 python 语言的 gensim 模块,首先遍历全部样本中的特征,为其生成维数为 300 的词向量字典,再利用该字典将全部样本转换为向量并保存,为后续模型训练与测试提供数据支持。

3.5 基于机器学习的恶意程序分类器

利用 3.4 特征抽象节中抽象表示得到的二维向量样本,则整个样本集可视为一个三维向量。但由于机器学习算法限制,需保证机器学习算法的输入必须为二维向量。因此,在构建分类器前要将得到的三维向量降维处理,利用前文提到的 Word2vec 具有良好的特征词间语义相似度性质,可对各样本的二维向量进行简单的运算操作降为一维向量。本文采用对每个样本的词向量特征累加求平均值的方法,可在最大限度上保证原文本特征不丢失。在将样本空间降维成为二维向量后,本文利用 python 的 Scikit-learn 库分别构建了基于向量机、朴素贝叶斯、决策树、随机森林、 K 近邻和逻辑回归 6 种分类模型的恶意程序分类器。

在模型构建中,为充分评估模型的预测性能,减小过拟合问题,引入了 k 折交叉验证 (k -fold cross validation)^[20] 方法。 k 折交叉验证通过将全部样本分组训练并取平均的方法来减少检验方差,降低性能对数据划分的敏感性。本文中采取

的 k 折交叉验证方法如下。

第1步：不重复地将样本集中的样本随机分为5份。

第2步：每次选取其中一份作为测试集，其余4份作为训练集以训练预测模型。

第3步：重复第3步5次，使每个子集都有一次机会作为测试集，其余作为训练集。针对每次训练得到的模型测试，保存相应的测试结果。

第4步：计算5组测试结果的平均值作为该预测模型的评估指标。

在得到测试集样本后，由于其本身同样具有恶意性属性标签，可验证本预测模型的准确性，将预测结果与标签比较可计算出准确率 (accuracy)、精确率 (precision)、召回率 (recall)、 F_1 分数 (F_1 score) 为模型性能指标。其中，分类结果参数定义如表1所示。

表1 分类结果参数定义

样本来源	检测结果	
	恶意样本	非恶意样本
恶意样本	TP	FN
非恶意样本	FP	TN

准确率标识恶意样本中被预测为恶意样本的比例；精确率为在全部被预测为恶意样本中真正是恶意样本的比例；召回率标识被正确判定的恶意样本占有所有恶意样本的比例； F_1 分数是准确率和召回率的加权调和平均。得到计算公式如下：

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

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

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

$$F_1 = 2 \frac{precision \cdot recall}{precision + recall}$$

3.6 基于深度学习的恶意程序分类器

本文另选取卷积神经网络^[21] (CNN, convolutional neural network) 模型对提取得到的 n -gram 操作码序列特征进行训练学习，以构成基于深度学习的恶意程序分类器。卷积神经网络是

神经网络的一种，由输入层、卷积层、池化层、输出层4个基本元素构成，相比于其他神经网络，CNN使用卷积运算代替矩阵运算，并以通过稀疏交互 (sparse interactions)、参数共享 (parameter sharing)、等变表示 (equivariant representations) 3个重要思想帮助改进学习系统。

对于本文提取的 n -gram 操作码序列，首先需要为 n -gram 操作码建立索引，并存储操作码与索引之间的键值关系，该步骤本文采用 keras 预处理模块 preprocessing 中的 Tokenizer 来实现，可得到单词索引序列。

在 CNN 模型工作流程中，模型会对输入的文本进行向量转换，称为 embedding 层，而在 3.4 节中 n -gram 操作码序列已通过 Word2vec 抽象成为词向量，因此可用 3.4 节中得到的词向量矩阵代替模型的 embedding 层继续训练模型。以 1-gram 操作码序列为例，本文采用的 CNN 模型结构如图2所示。

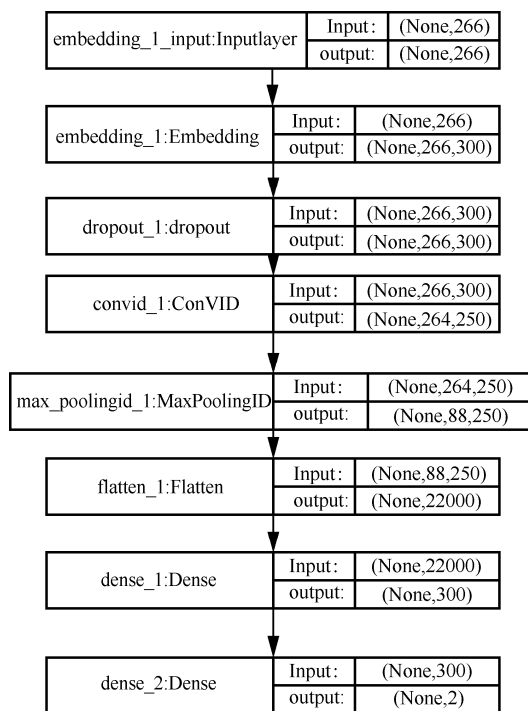


图2 1-gram CNN模型具体结构

4 实验分析

依据第2、3节构建的程序基因提取方法及恶意程序预测分类器构建方法，本文以Python语言实现了上述流程，并使用互联网上

公开的恶意及非恶意样本对实现的分类器进行性能检验。

4.1 实验流程

首先,采集使用的原始样本数据,其中恶意样本来源于恶意样本网站 <https://www.virusign.com/>,正常程序来源于 Windows XP、Windows 7 64 位操作系统的系统程序,利用网络资源爬取系统共收集恶意程序 1 038 个、正常程序 901 个,根据收集样本来源将样本标记为正常程序和恶意程序。接下来,使用基于 Pin 的指令流快照提取系统提取得到程序基因库,并使用 n -gram 方法对基因库中的操作码部分提取形成基于 n -gram 的操作码序列。然后,使用 Word2vec 对操作码序列进行抽象语义描述转换为特征向量。最后分别输入到基于机器学习和基于深度学习的恶意程序预测分类器中,进行训练和测试。在结果评价方面,使用 k 折交叉验证的结果展示各恶意程序预测分类器的性能。

4.2 基于机器学习的恶意程序分类器性能测试

在此实验中,对支持向量机、朴素贝叶斯、决策树、随机森林、 K 近邻和逻辑回归 6 种模型构成的分类器进行测试,以 n -gram 的提取长度为 1、2、3、4 得到的操作码序列进行 4 次实验,实验结果如表 2~表 5 所示。

将表 2~表 5 转换成为图形表示可更形象地突出各机器学习模型的优势,以准确率为标准,各预测模型对比如图 3 和图 4 所示。

表 2 $n=1$ 时基于机器学习的恶意程序分类器分类能力

模型	准确率	精确率	召回率	F_1 分数
SVM	0.865	0.880	0.930	0.905
朴素贝叶斯	0.860	0.873	0.931	0.901
随机森林	0.899	0.936	0.915	0.925
决策树	0.895	0.925	0.923	0.923
K 近邻	0.904	0.931	0.929	0.930
逻辑回归	0.874	0.893	0.929	0.910

表 3 $n=2$ 时基于机器学习的恶意程序分类器分类能力

模型	准确率	精确率	召回率	F_1 分数
SVM	0.898	0.897	0.970	0.932
朴素贝叶斯	0.720	0.868	0.821	0.852
随机森林	0.886	0.920	0.922	0.921
决策树	0.870	0.906	0.914	0.910
K 近邻	0.894	0.910	0.946	0.928
逻辑回归	0.896	0.913	0.947	0.930

表 4 $n=3$ 时基于机器学习的恶意程序分类器分类能力

模型	准确率	精确率	召回率	F_1 分数
SVM	0.913	0.999	0.834	0.910
朴素贝叶斯	0.907	0.979	0.841	0.905
随机森林	0.921	0.943	0.886	0.914
决策树	0.915	0.980	0.838	0.912
K 近邻	0.915	0.999	0.839	0.912
逻辑回归	0.915	0.961	0.836	0.911

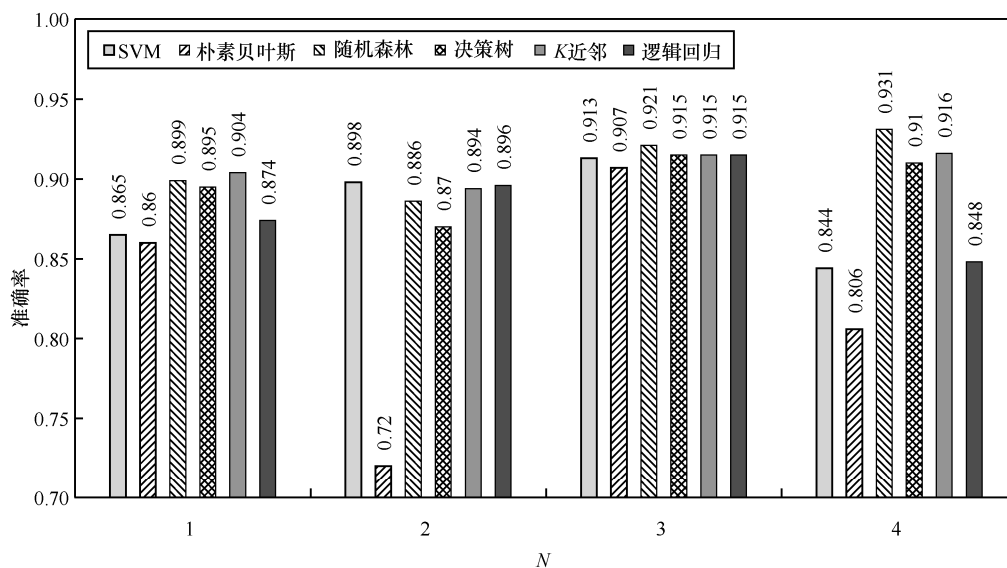


图 3 预测模型准确率对比

表5 $n=4$ 时基于机器学习的恶意程序分类器分类能力

模型	准确率	精确率	召回率	F_1 分数
SVM	0.844	0.940	0.752	0.836
朴素贝叶斯	0.806	0.864	0.872	0.868
随机森林	0.931	0.923	0.947	0.935
决策树	0.910	0.873	0.954	0.912
K 近邻	0.916	0.893	0.954	0.922
逻辑回归	0.848	0.917	0.780	0.843

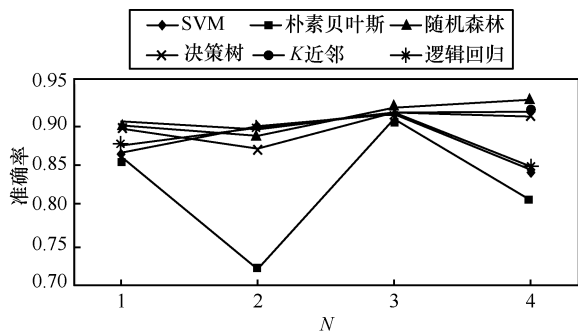


图4 预测模型准确率趋势

4.3 基于深度学习的恶意程序分类器性能测试

以 n -gram 的提取长度为 1、2、3、4 得到的操作码序列对深度学习预测模型进行 4 次实验，实验结果如表 6 所示。

为了便于分析深度学习的性能，对比本文构建的 2 种恶意程序预测模型，选取基于决策树、随机森林、 K 近邻和逻辑回归模型的机器学习分类器与深度学习进行对比，结果如图 5 和图 6 所示。

表6 基于深度学习的恶意程序分类器分类能力

n	准确率
$n=1$	0.939
$n=2$	0.993
$n=3$	0.907
$n=4$	0.916

4.4 实验总结

在基于机器学习的恶意程序分类器性能测试中，除朴素贝叶斯模型外，其余模型的准确率均超过 85%；当 n -gram 步长为 3 时，取得最优效果，全部模型准确率均超过 90%。在此处取得最优的原因是本文所研究的程序基因中各操作码排列具有很强的相关性，在步长较短时会丢失部分相关性特征，从而降低准确率；而当步长为 4 时，可能出现部分过拟合现象，导致在步长为 3 时取得最优效果。

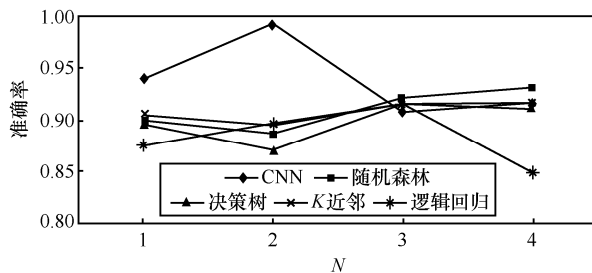


图6 深度学习模型准确率趋势

在基于深度的恶意程序分类器性能测试中，分类器在步长为 2 时取得最优值，准确率高达 99.3%，

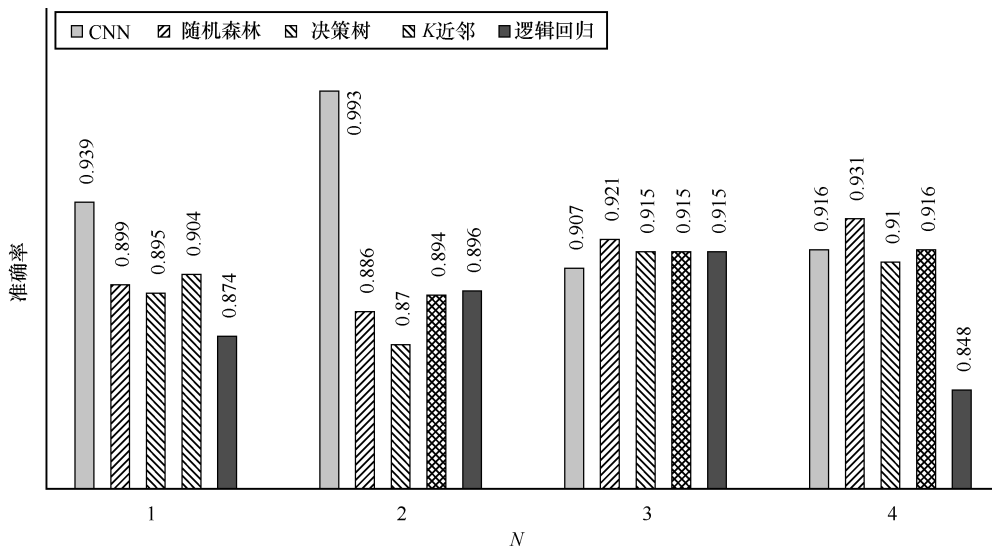


图5 深度学习模型准确率对比

其余情况准确率均超过 90%, 总体上来讲, 基于深度学习的分类模型优于基于机器学习的分类模型, 其原因是在构建分类器输入时, 由于机器学习分类器的输入是二维数组, 从而对得到的特征向量采用平均值法降维处理, 虽然利用 Word2vec 得到的特征词具有良好的语义相关性, 但仍有部分特征丢失, 导致深度学习模型整体上优于机器学习模型。

通过上述实验数据分析本文提出的 2 种基于程序基因的恶意程序预测模型在准确率上均表现出了较强的分类能力, 可以证明基于程序基因对程序恶意性预测是可行有效的。

5 结束语

本文针对二进制无源码程序, 首次提出了程序基因的概念, 并基于项目组前期研发的无感知沙箱基础上, 设计了一套系统化的程序基因提取方法。针对定义的程序基因, 本文提出了基于程序基因的恶意程序预测技术, 实现了基于程序基因的恶意程序预测系统原型, 通过对千余个恶意程序样本、正常程序样本预测实验, 基于程序基因的恶意程序预测模型具有很强的识别能力, 可以为程序恶意性分析、程序恶意行为检测提供佐证。在后续工作中, 本项目组将进一步完善程序基因的理论体系, 探索更多、更细致的识别程序行为的基因; 在恶意程序预测方面, 增加更多可供判断程序属性的特征, 优化当前使用的模型, 增大样本库容量; 在其他领域, 积极探索程序基因理论在恶意程序同源性、恶意程序行为识别分析等相关领域的应用。

参考文献:

- [1] 2017 年度互联网安全报告[EB/OL]. <https://slab.qq.com/news/authority/1708.html>
- [2] 2017 Internet security report [EB/OL]. <https://slab.qq.com/news/authority/1708.html>
- [3] 阮斌. 《2017 年中国网络安全报告》发布[J]. 计算机与网络, 2018(5). RUAN B. 2017 China cyber security report released[J]. Computer & Network, 2018(5).
- [4] SANTOS I, BREZO F, NIEVES J, et al. Idea: opcode-sequence-based malware detection[C]// International Conference on Engineering Secure Software and Systems. Springer-Verlag, 2010:35-43.
- [5] 孙博文, 黄炎裔, 温俏琨, 等. 基于静态多特征融合的恶意软件分类方法[J]. 网络与信息安全学报, 2017, 3(11):68-76. SUN B W, HUANG Y Y, WEN Q K, et al. Malware classification method based on static multiple-feature fusion[J]//Chinese Journal of Network and Information Security, 2017, 3(11):68-76.
- [6] 白金荣, 王俊峰, 赵宗渠. 基于 PE 静态结构特征的恶意软件检测方法[J]. 计算机科学, 2013, 40(01):122-126. BAI J R, WANG J F, ZHAO Z Q. Malware detection approach based on structural feature of PE file[J]//Computer Science, 2013, 40(01): 122-126.
- [7] QEMU [EB/OL]. https://wiki.qemu.org/Main_Page.
- [8] Pin-a dynamic binary instrumentation tool[EB/OL]. [https:// software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool](https://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool)
- [9] 曹梦晨. 基于沙箱指令流快照的恶意程序智能识别技术研究[D]. 北京邮电大学, 2017. CAO M C. Research on malware intelligent recognition technology based on sandbox instruction flow snapshot[D]// Beijing University of Posts and Telecommunications, 2017.
- [10] LI H J, TIEN C W, TIEN C W, et al. AOS: An optimized sandbox method used in behavior-based malware detection[C]//International Conference on Machine Learning and Cybernetics. IEEE, 2011: 404-409.
- [11] FIRDAUSI I, LIM C, ERWIN A, et al. Analysis of machine learning techniques used in behavior-based malware detection[C]// Second International Conference on Advances in Computing, Control and Telecommunication Technologies. 2010:201-203.
- [12] CHAN P P F, HUI L C K, YIU S M. Dynamic software birthmark for java based on heap memory analysis[M]// Communications and Multimedia Security. Springer Berlin Heidelberg, 2011:94-107.
- [13] LIU K, ZHENG T, WEI L. A software birthmark based on system call and program data dependence[C]//Web Information System and Application Conference. IEEE, 2015:105-110.
- [14] 韩金, 单征, 赵炳麟, 等. 基于软件基因的 Android 恶意软件检测与分类[J]. 计算机应用研究, 2019(6). HAN J, SHAN Z, ZHAO B L, et al. Detection and classification of Android malware based on malware gene[J]//Application Research of Computers, 2019(6).
- [15] 赵晶玲, 陈石磊, 曹梦晨, 等. 基于离线汇编指令流分析的恶意程序算法识别技术[J]. 清华大学学报(自然科学版), 2016(5): 484-492. ZHAO J L, CHEN S L, CAO M C, et al. Malware algorithm recognition based on offline instruction-flow analyse. Journal of Tsinghua University(Science and Technology), 2016, 65(5): 484-492.
- [16] 陈晨. 基于操作码行为为深度学习的恶意代码检测方法[D]. 黑龙江: 哈尔滨工业大学, 2013. CHEN C. Malicious code detection based on opcode behavior deep learning[D]. Heilongjiang: Harbin Institute of Technology, 2013.

- [16] BERRAR D, DUBITZKY W. Information gain[M]. Springer New York, 2013.
- [17] RONG X. Word2vec parameter learning explained[J]. Computer Science, 2014.
- [18] MIKOLOV T, CHEN K, CORRADO G, et al. Efficient estimation of word representations in vector space[J]. Computer Science, 2013.
- [19] MIKOLOV T, SUTSKEVER I, CHEN K, et al. Distributed Representations of Words and Phrases and their Compositionality[J]. NIPS'13 Proceedings of the 26th International Conference on Neural Information Processing Systems 2013. 3111-3119.
- [20] 周志华. 机器学习: = Machine learning[M]. 北京: 清华大学出版社, 2016.
- ZHOU Z H. Machine learning[M]. Beijing: Tsinghua University Press, 2016.
- [21] CHUA L O, ROSKA T. CNN paradigm[J]. IEEE Transactions on Circuits & Systems I Fundamental Theory & Applications, 1993, 40(3):147-156.

[作者简介]



肖达(1981-), 男, 黑龙江宁安人, 北京邮电大学大学讲师, 主要研究方向为信息安全和人工智能。



刘博寒(1995-), 男, 辽宁锦州人, 北京邮电大学硕士生, 主要研究方向为系统安全、漏洞利用和恶意行为检测。



崔宝江(1973-), 男, 山东东营人, 博士, 北京邮电大学教授, 主要研究方向软件安全、漏洞挖掘、大数据安全和区块链安全。



王晓晨(1996-), 女, 山东阳信人, 北京邮电大学研究生, 主要研究方向为信息安全。

张索星(1994-), 男, 江苏苏州人, 北京邮电大学硕士生, 主要研究方向为恶意程序检测、逆向工程和漏洞利用。