一种基于深度学习的恶意软件家族分类模型

郑 锐[1,2], 汪秋云[2], 傅建明[1], 姜政伟[2,3], 苏日古嘎[1,2], 汪姝玮[2]

[1]空天信息安全与可信计算教育部重点实验室, 武汉大学国家网络安全学院 武汉 中国 430072

[2]中国科学院信息工程研究所 北京 中国 100093

[3]中国科学院大学网络空间安全学院 北京 中国 100049

摘要 恶意软件的家族分类问题是网络安全研究中的重要课题, 恶意软件的动态执行特征能够准确的反映恶意软件的功能性与家族属性。本文通过研究恶意软件调用 Windows API 的行为特点, 发现恶意软件的恶意行为与序列前后向 API 调用具有一定的依赖关系, 而双向 LSTM 模型的特征计算方式符合这样的依赖特点。通过设计基于双向 LSTM 的深度学习模型, 对恶意软件的前后 API 调用概率关系进行了建模, 经过实验验证, 测试准确率达到了 99.28%, 所提出的模型组合方式对恶意软件调用系统 API 的行为具有良好的建模能力, 为了深入的测试深度学习方法的分类性能, 实验部分进一步设置了对抗样本实验, 通过随机插入 API 序列的方式构造模拟对抗样本来测试原始参数模型的分类性能, 对抗样本实验表明, 深度学习方法相对某些浅层机器学习方法具有更高的稳定性。文中实验为深度学习技术向工业界普及提供了一定的参考意义。

关键词 深度学习; 恶意软件; 家族分类; 鲁棒性

中图法分类号 TN915.08 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2020.01.01

A Novel Malware Classification Model Based on Deep Learning

ZHENG Rui[1,2], WANG Qiuyun[2], FU Jianming[1], JIANG Zhengwei[2,3], SU Riguga[1,2], WANG Shuwei[2]

[1] School of Cyber Science and Engineering WuHan University, Wuhan 430072, China

[2] Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

[3]School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

Abstract Family classification of malicious software is an important issue in research of computer system security. The dynamic execution feature of malicious software always reflect the functionality and family attributes of malware. By studying the behavior characteristics of malware system API call, we observer that malicious behavior of malware has a certain dependence on sequence forward and backward API call. Bidirectional LSTM can cover such dependency characteristics. By designing a deep learning model based on Long Short-Term Memory network, model the relationship of forward and backward. The experimental results show that the test accuracy reaches 99.28%. The proposed model combination method has good modeling ability for the behavior of malicious software invocation system API. To evaluate the classification performance of deep learning method, in the experimental part, we further add the adversarial examples experiment, and construct the simulated adversarial examples by inserting the adversarial sequence randomly in test samples to test the classification performance of the original parameter model. The adversarial examples experiment shows that the deep learning model is more robust than shallow machine learning methods. The experiment in this paper provides some reference for the popularization of deep learning technology to industry

Key words deep learning; malicious software; family classification; robust

1 前言

恶意样本的识别与家族分类是网络安全领域的重要研究领域。根据 360 发布的安全报告[1], 2018 年旗下安全产品每天检测出 PC 端恶意代码 75.2 万个, 其中绝大部分都是 Windows 系统平台的恶意软件。

通讯作者: 傅建明, 博士, 教授, Email: jmfu@whu.edu.cn。

本课题得到国家自然科学基金项目(No.61972297, No.U1636107), 基础加强计划(No.2017-JCJQ-ZD-043-01-00), 国家重点研发计划(No.2016QY06X1204, No.2018YFC0824801)资助。

收稿日期: 2019-09-05; 修改日期: 2019-12-09; 定稿日期: 2019-12-16

恶意软件的分类检测除了检测出待测样本是否具有恶意性之外, 检测出其家族分类也同样具有重要意义。恶意软件的家族分类往往能够指示出恶意软件的恶意行为类别与执行目的, 这些恶意目的的分类也为恶意软件的危险程度等信息提供了重要的参考, 并且恶意软件家族分类的检测有利于快速跟踪恶意软件家族发展, 以便对网络空间安全形势进行快速评估。

另一方面常见的恶意软件家族的免杀方法一般采用特征值编辑的方法逃避杀软检测, 包括特征值撞击修改, 软件加壳加密等方法。这些方法操作简单, 容易规避查杀工具的检测, 而动态沙箱测试则可以绕过这些对抗手段的混淆。更进一步的, 家族分类往往以恶意软件功能作为分类标准, 动态检测正是针对恶意软件的功能进行识别。所以从理论上说采用动态方法的恶意软件家族分类可以获得更加健壮的家族分类结果。

长期以来恶意软件家族分类研究面临很多现实瓶颈。在传统的恶意软件家族分类实践中, 恶意软件的家族分类方法是一个开放性的问题[2], 杀毒软件厂商对恶意软件的命名方式不尽相同, 即使针对相同的软件, 相同的恶意功能, 不同的反病毒软件厂商也可能会给出不同的家族名称。目前针对恶意样本的标签问题, 学术界通用的方法是使用 virustotal.com 的分类结果设定数量阈值作为恶意软件的恶意性分类标准[3,4], 但是在家族分类的标签标定方法上, virustotal 给出了各种杀软厂商的家族分类结果, 这些分类结果同样面临家族分类标准难以统一的问题。如何标定恶意软件的家族分类标签制约了恶意软件家族多分类问题研究的探索。除此之外, 恶意软件的分析检测作为一种安全关切的领域, 恶意软件检测结果的抗干扰和鲁棒性也深受学术界和工业界的重视。

针对以上的问题, 本文通过改进一种家族标签聚合方法, 提出了一种基于深度学习的恶意软件家族分类解决方案, 比较了深度学习方法和与浅层机器学习方法分类性能, 并对这些分类方法进行了鲁棒性测试。本文的主要贡献如下:

1. 根据恶意软件调用 Windows API 的特点, 提出了使用双向长短时记忆网络(BiLSTM)对恶意软件的 API 调用行为进行建模, 并运用该模型实现恶意软件家族分类, 实验结果验证了模型组合的合理性和准确性。

2. 提出了一种基于 API 注入的恶意软件家族分类模型的鲁棒性测量方法.其实验结果证明了采用深度学习的分类模型具有更强的鲁棒性, 为深度学习模型用于恶意软件家族分类的实际部署提供了一定的实验支持与可行性支撑。

2 相关工作

本节将对学术界与工业界的恶意软件家族分类使用的普通机器学习方法与深度学习方法做概括。对比我们工作的创新性。

机器学习模型的训练与使用如图 1 所示, 训练数据与测试数据均使用相同的特征处理方法, 先后输入到模型中训练和测试分类模型的性能。机器学习的模型构建需适应于特征数据的类型, 例如循环神经网络(RNN)结构更易于处理序列数据, 卷积神经网络(CNN)更易于提取图片数据的特征, 而恶意软件的动态检测与静态检测提取的特征形式差别较大。在本节中我们分别从静态检测与动态检测的角度梳理恶意软件家族分类使用的特征方法与模型特点。

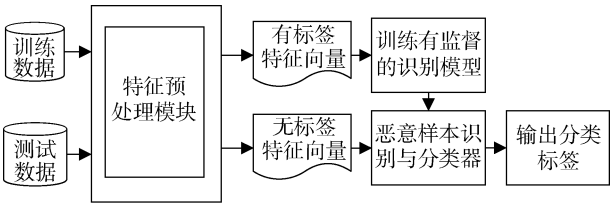

图 1 恶意代码家族分类的系统框架图

Figure 1 Malware family classification system framework

2.1 恶意软件的静态检测与家族分类

静态检测技术提取的特征种类丰富, 恶意代码的结构信息, IAT 导入表信息等都可以作为恶意软件的特征向量训练恶意软件分类器。

Saxe 等[5]提出了一种恶意软件的统计特征, 通过分别计算恶意软件二进制文件中字节数值的熵值, 恶意软件的可打印字符, 调用的函数表等, 构造固定维度的恶意软件特征向量, 输入到深度神经网络中训练恶意软件分类器。从得到的结果来看, 恶意软件的检测准确率达到 90%以上, 且误报率低于千分之一。Zhang 等[6]使用 opcode 作为特征向量训练了基于卷积神经网络的恶意软件分类模型, 实验结果表明, 基于卷积神经网络的分类模型效果要大幅度优于 KNN 等普通机器学习方法。Hardy 等[7]使用了 IAT 表中的导入 API 信息来判断软件的恶意性, 但是这样的判别方法忽视了 API 序列之间的相关性。

除了传统的抽取二进制文件静态特征的方法,

文献[8-10]使用卷积神经网络的方法, 对恶意软件二进制文件转换为的图片进行分类, 结果表明基于卷积神经网络的与特征图片的方法对恶意软件的家族分类检测可以达到 98%以上。

Raff 等[11]在恶意软件识别中不使用 PE 文件中代码与数据域部分, 而是提取 PE 头信息, 对 PE 头的各个字符数据向量化, 并使用深度网络训练恶意软件识别模型, 通过在 A, B 两个测试集中测试模型性能, 深度学习方法能够在 B 测试集中取得较好的效果, 作者认为深度学习模型更能应对数据训练集与测试集分布差异的情况。

可以看出, 在静态检测中以卷积神经网络为代表的深度学习方法在静态特征工程的帮助下可以实现更好的恶意软件识别效果。但是静态特征很容易受到加壳混淆等方法的干扰而导致分析无法进行。

2.2 恶意软件的动态检测与家族分类

恶意软件的动态检测技术主要提取恶意软件在运行时产生的特征, 通过沙箱技术对恶意软件运行时的系统动作进行捕获, 这些系统动作包括注册表修改, 网络通信, 文件创建, 写入, 删除, 传输等。在大多数的研究中 API 调用往往作为最重要的动态特征来识别恶意软件的恶意性和家族种类。

Huang 等[12]研究人员通过沙箱技术提取恶意软件与正常软件的系统 API 调用序列, 通过将恶意软件无字符终结符与 API 序列做降维处理构建了一个 5000 维的特征样本集。论文使用了多层感知机网络作为分类模型, 设置一个多任务的学习机制, 在网络的设计和训练的过程中论文加入了一些深度学习方法, 例如神经元丢弃(Dropout)和矫正线性单元激活函数(ReLu)。分别识别恶意软件的恶意性与恶意软件的类别, 经过实验验证, 论文中提出的方法获得了较低误报率, 并且在实验中作者发现识别效果会随着分类网络中隐含层的层数增加而提升。

Rosenberg 等[3]使用恶意软件执行之后的 API 调用序列训练了一种恶意软件恶意性识别方法, 获得了较好的效果。Kolosnjaji 等[13]使用动态测试方法, 设计了基于长短时记忆网络的深度学习方法检测恶意软件的家族属性, 但是这种方法忽略了序列中前后双向的依赖关系, 本文在对比实验中对比了这种方法的性能。

3 模型方法理论分析

在本节, 我们将论证论文提出方法的合理性, 介绍深度学习模型的网络层计算原理。

3.1 基于动态检测的恶意软件家族分类

恶意软件的动态测试主要提取恶意软件执行时的行为特征。API 函数调用是现代操作系统的结构基础, Windows 操作系统上所有的软件都需要调用 Windows 系统层面的 API 函数执行操作。本文提取 API 的调用序列特征作为恶意软件家族分类的特征向量。例如 Wanacry 蠕虫病毒恶意软件, 在感染被检测主机的过程中会有一系列的行为操作, 恶意软件的主程序会分别释放恶意可执行文件, 首先会更改系统注册表, 注册本地服务向内网中其他资产探测和移动, 然后执行文件加密进程, 对疑似有价值的文件执行加密操作并删除原文件, 强制重启电脑之后显示勒索窗口。完整的恶意行为具有鲜明的 API 调用特点。

在序列的执行中, 前后的 API 调用不只包含后者依赖前者的情况, 后执行的 API 同样可以推算出前置 API 的概率。所以在这种情况下, 理论上覆盖双向依赖关系的模型可以达到更好的识别效果。

在双向的 API 依赖中, 还是以 Wanacry 勒索软件为例, 恶意软件在执行一系列的创建文件和覆盖原有文件的操作时, 需要对注册表的权限信息进行修改, 此时, 单向的依赖关系提取注册表修改到文件操作的概率分布, 而后向的依赖关系在计算文件操作的概率分布时也会强化注册表修改与文件操作之间的概率依赖特征, 双向的信息模式会强化模型输出的概率特征, 所以对双向的依赖信息进行建模会优于对单向的因果关系建模。另外, 基于已有的验证可知[14], 2017 年大规模爆发的 Wanacry 样本具有明显的阶段执行特性, 两个阶段之间的依赖关系是灵活的, 也就是说探测网域内资产并横向移动与勒索进程加密目标文件之间没有先后的关系, 所以在不同的变种, 不同的受害主机中会产生不一样的执行顺序, 此时前后向的信息建模更容易捕获勒索软件的双向动作的不确定性特征, 确定家族分类。

3.2 深度学习技术与模型选取

深度学习采用多种方法降低模型参数增多之后的过拟合风险, 并在降低过拟合风险的基础上, 构建新的分类系统, 适应不同的数据类型和概率分布关系。应用在恶意软件的恶意性识别与家族分类, 深度学习技术获取了不错的效果。文献[13]主要采用了卷积层与长短时记忆网络组合的网络方式, 通过恶意软件的 API 调用序列特征来识别恶意软件的家族种类。

根据上一小节的分析, 恶意软件 API 调用的反

向依赖关系可以加强恶意软件家族分类的特征表达, 从而提升分类效果, 参考已有的自然语言处理所使用的文本分类模型, 我们选取了四种深度学习模型作为测试模型, 比较不同的网络层组合对分类结果的影响。四种网络层分别为: LSTM, biLSTM, CNN+LSTM, CNN+biLSTM。其中文献[13]中采用 CNN+LSTM 方法识别恶意软件的家族类别。在实验中我们将之作为对照组, 比较新方法的性能。以下分别对模型中用到的网络层结构进行介绍。

卷积层, 卷积层可以降低特征维度, 为后来的长短时记忆神经网络层的处理提供更加健壮的特征。

单向 LSTM 层与双向 LSTM 层, 是特征处理的关键层, 长短时记忆网络能够压缩序列数据特征, 并与全连接层连接输出计算结果, 双向的 LSTM 层通过添加一个反向的传输层提取反向依赖特征。

全连接层, 是一种通用的是神经网络方法, 在分类中用于处理前置神经网络的信息来输出类别概率。

所使用的四种神经网络层结构及其参数如下:

- LSTM: Input > LSTM(150) > Dense(9*1) > Result
- BiLSTM: Input > biLSTM(150) >Dense(9*1) > Result
- CNN+LSTM: Input > Conv1D(20*1) > LSTM(150) > Dense(9*1) > Result
- CNN+BiLSTM: Input > Conv1D(20*1) > biLSTM(150) > Dense(9*1) > Result

3.3 数据集选取与预处理

动态检测所使用的样本文件均为可执行的二进制文件, 由于文件本身具有感染风险和免杀之后的传播危险性, 目前很少有论文研究者公布自身的动态测试原始样本。在实验中我们采用了 virustotal.com 公司面向教育学者公布的恶意样本数据集, 数据集包含了 PDF, DOC 等格式的样本及各种语言编写的一些源代码文件。数据的收集范围为 2017 年和 2018 年。Virustotal.com 官方没有给出这些数据收集标准, 只是在数据包中附带了样本的检测报告。在样本集合中我们一共得到了 154570 个 Windows 系统平台的 EXE 可执行恶意样本。在实际的操作中, 数据的标签难以确定, 为了合理的取得恶意样本的标签数据, 我们借助文献[15]公布的恶意样本标签标定方法 AVclass, 聚合方法主要是将 virustotal.com 报告的家族标签数据进行统计, 并且根据统计结果将偏差修正为恶意软件样本家族标签。为了兼顾恶意软件行为特征的独立性, 我们从样本中挑选了 6681 个样本作为实验数据样本集。实验数据的样本集类别如表 1 所示:

表 1 数据集各类样本数量分布

Table 1 The distribution of malware samples quantity

序号	家族名称	数量
1	AdWare	663
2	AutoRun	228
3	Blocker	575
4	Neshta	1199
5	Sality	711
6	Scar	349
7	ShipUp	880
8	Vobfus	976
9	Wabot	1100

本文数据集与微软在 kaggle 的恶意软件分类大赛中的公开数据集[8]做对比, 本文数据集在数量上略少于微软恶意软件分类比赛中训练集数据的个数。

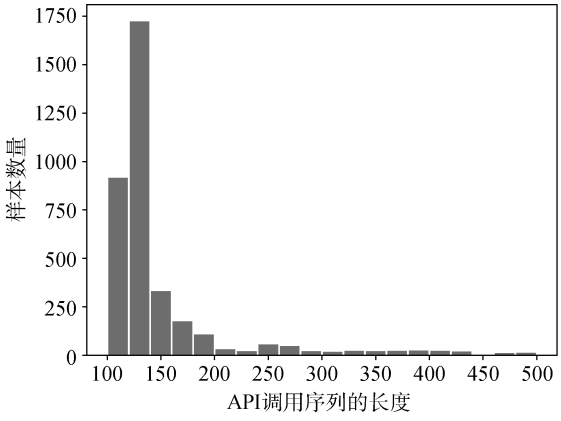

图 2 去重之后的 API 长度统计

Figure 2 API length with deduplication statistics of malware samples

4 数据集处理与实验

4.1 数据集与性能指标

恶意软件的动态特征提取依靠自动化测试工具进行, 将恶意软件放置于虚拟的沙箱环境, 当恶意软件运行时, 虚拟化环境设置的 hook 机制在恶意软件调用 windows API 时自动触发, 记录恶意软件调用 API 的时间以及传递的参数。目前开源的沙箱工具中, cuckoo sandbox 融合了众多的恶意软件测试功能, 可以提取恶意软件运行时的 windows API 调用序列, 恶

意软件网络通信数据流量等恶意软件的行为数据。

本文采用 cuckoo sandbox 平台, 设置 25 台虚拟机, 每台虚拟机安装了 64 位 windows 7 操作系统, cuckoo sandbox 的系统参数配置采用原始的参数, 恶意软件的测试时间阈值为 3 分钟, 当恶意软件上传至虚拟机中运行结束或者到达 3 分钟阈值时, 终止虚拟机运行, 输出恶意软件测试报告。在实验中, 我们一共使用了两种 API 长度作为测试样本的固定长度。

经过统计恶意软件共使用了 183 种系统 API 函数。对 API 函数采用了 one-hot 编码输入到深度神经网络中, 每条样本提取了前 220 个 API 调用函数组成特征向量, 不足的 API 调用采用补 0 的方式凑足 220 个调用动作。400 长度的 API 序列数据使用相同的截断和补零操作。

在 Cuckoo Sandbox 的测试中, 恶意软件会因为各种原因反复调用特定的 API 函数, 有些是正常的行为, 有些则是在非预期条件下导致 API 函数重复调用。为了抵消在调用数据上的非预期行为, 增大训练模型在其他数据集上的泛化能力, 我们针对前后重复的 API 调用序列进行去重, 图 2 为去重后对数据集中 API 序列长度的统计。根据图 2 的 API 长度统计结果, 我们分别取 220 和 400 作为序列长度测试深度学习模型的分类效果。

4.2 实验平台配置与性能指标选取

深度学习技术需要 GPU 加速器支持, 缺少计算加速器将会严重减慢实验效率。本文采用了 GPU 服务器作为实验平台, 具体的硬件与软件配置如下。

表 2 实验平台硬件与软件库配置

Table 2 Hardware and software library configuration of experiment platform

关键设备与软件	参数
CPU 中央处理器	2U 双通道 10 核
GPU 加速器	NVIDIA Tesla V100
内存	128GB
动态测试虚拟机	Cuckoo Sandbox with Windows7 X64
加速器计算库	CUDA 9.0 and cuDNN7.1
编程语言及版本	Python 3.5.2
深度学习库	Tensorflow 1.12.1
数据分析	Numpy Pandas

为了衡量深度学习模型对恶意软件家族分类的效果, 选取精确率(Precise Rate)、召回率(Recall Rate)、F1-score、准确率(Accuracy)作为模型性能的衡量标准, 由于家族分类属于多分类问题, 为了更好的平衡不同类别之间的数量和性能差异, 在实验中使用加权平均的方式计算各个类别指标。四个指标的计算公式如下所示:

$$macro(\text{Precision}) = \frac{1}{n}\sum_{n=1}^{n} w\left(\frac{TP_n}{TP_n + FP_n}\right) \quad (1)$$

$$macro(\text{Recall}) = \frac{1}{n}\sum_{n=1}^{n} w\left(\frac{TP_n}{TP_n + FN_n}\right) \quad (2)$$

$$macro(\text{F1-score}) = \frac{1}{n}\sum_{n=1}^{n} w\left(\frac{2 * \text{Prcision} * \text{Recall}}{\text{Prcision} + \text{Recall}}\right) \quad (3)$$

$$\text{Accuracy} = \frac{TP_1 + TP_2 + \cdots + TP_n}{\text{number of all samples}} \quad (4)$$

其中, TP_n 代表得到的正确分类出的正样例, FN_n 代表错误分类为负样例的正样例, FP_n 代表被错误分类为正样例的负样例。n 为类别的种类与编号, w 为类别的权重, 即标签数据个数与总的数据个数的比值。

以上指标计算方式均为交叉验证一次的性能指标计算方式, 实验最终的性能指标取五折交叉验证取得结果的平均。

4.3 实验结果与分析

实验分别测定了 3.2 节所述几种深度学习模型的准确率性能, 为了对比深度学习准确率性能的高低, 在实验中我们设置了两种浅层机器学习方法作为实验对照组。为了从多个维度衡量深度学习技术的性能, 我们也测试了模型的测试效率, 训练效率。模型健壮性测试同样能够为模型的部署提供更强的参考意义。

在实验结果中我们分别展示了模型的精确率(Precision)、召回率(Recall)、F1-score、整体的准确率(Accuracy)四组数据, 四组数据的计算方式见 4.1 节的公式(1-4)。如 4.1 节说明, 为了多方面的衡量深度学习模型在恶意软件动态分析技术中的效果, 将单个实验样本的序列长度值记作 L, 下面分别给出 L=220 和 L=400 的实验效果。

表 3 模型取得的检测效果(L=220)

Table 3 Classification performence of the machine learning model (L=220)

实验算法	精确率	召回率	F1-score	Accuracy
朴素贝叶斯	0.7475	0.7120	0.6993	0.7488
随机森林	**0.9760**	0.9410	0.9551	0.9698
LSTM	0.9380	0.9335	0.9301	0.9859
biLSTM	0.9719	**0.9689**	**0.9677**	**0.9933**
CNN+LSTM[13]	0.9494	0.9467	0.9451	0.9840
CNN+biLSTM	0.9715	0.9681	0.9671	0.9931

从表3的分类结果可以看出, 双向LSTM网络分类模型效果相对于其他方法可以获得更好的分类性能, 但是随机森林算法获取了最高的精确率, 随机森林算法的计算原理与序列数据接近, 这可能时随机森林获得高性能的原因。

表 4 模型取得的检测效果(L=400)

Table 4 Classification performence of the machine learning model (L=400)

实验算法	精确率	召回率	F1-score	Accuracy
朴素贝叶斯	0.6931	0.7050	0.6686	0.7517
随机森林	**0.9745**	0.9387	0.9531	0.9693
LSTM	0.9479	0.9437	0.9417	0.9826
biLSTM	0.9697	0.9668	0.9654	0.9927
CNN+LSTM[13]	0.9275	0.9250	0.9196	0.9826
CNN+biLSTM	0.9711	**0.9678**	**0.9668**	**0.9928**

如表4的分类结果所示, 卷积层与双向LSTM层相结合的分类模型取得了更优的分类结果, 且在总的准确率上, CNN+biLSTM 方法获得了更高的测试性能。

对比文献[13]采用的 CNN+LSTM 方法, 本文在实验中加入的 BiLSTM 层取得了更佳的效果, 相似的网络结构在精确率, 召回率与 F1-sorce 的指标上提升了至少 2%。

深度学习技术对长序列样本的建模能力更强处理方式更加灵活。深度学习技术在处理序列问题上凭借网络的优势可以提取更多维度更长时间跨度的序列特征。双向的 LSTM 网络层不光从前向对数据进行建模, 还能够提取从后往前的反向依赖关系, 这在恶意软件的 API 调用的序列分析中更加实用。

浅层机器学习方法也能够取得较好的效果, 随机森林算法在各项指标上也能达到 0.9 以上的性能, 因为随机森林算法的计算结构与序列数据非常接近所以能够得到更好的效果。但是随机森林算法在算法性能上会由于参数增加而出现过拟合, 导致算法在在序列长度增加的背景下准确率降低。这在序列长度的验证上得到了结果。并且随后的健壮性实验也从模型稳定性的角度展示了随机森林可能出现了过拟合的现象。

除了模型的准确率指标外, 实验还对模型的执行效率进行对比。我们分别给出模型五折交叉验证的总时间平均之后的单次训练时间。

由于模型长度会造成一定的参数增多, 导致训练时间和识别时间相对增多。序列长度为 400 时的时间消耗如表 6 所示。

表 5 序列长度 220 时, 模型单折训练时间

Table 5 Time consumption of machine learning model one-fold cross validation when L=220

实验算法	时间(h)
朴素贝叶斯	**0.00001**
随机森林	0.00003
LSTM	0.60036
biLSTM	0.98883
CNN+LSTM	0.61138
CNN+biLSTM	0.96360

表 6 序列长度 400 时, 模型单折训练时间

Table 6 Time consumption of machine learning model one-fold cross validation when L=400

实验算法	时间(h)
朴素贝叶斯	**0.00002**
随机森林	0.00005
LSTM	1.10685
biLSTM	1.75233
CNN+LSTM	1.09099
CNN+biLSTM	1.73542

由于浅层机器学习使用的时间消耗较少, 所以精确到了小数点后五位, 从表中的对比可知, 当序列长度增加时, 恶意软件家族分类的深度学习模型的训练时间和训练时间会大幅上升。

虽然深度学习模型在训练的精确度方面具有一定的优势, 但是通过对比可以发现, 深度学习模型的训练时间成本要远远超过浅层机器学习模型的时间成本。

4.4 模型鲁棒性测量

深度学习技术具有黑盒性质, 在安全关切的领域, 比如恶意代码检测, 自动驾驶等, 模型的可靠性问题或者说模型的抗攻击性能对识别模型的部署具有重大意义。为了研究 4.3 节深度学习模型相对浅层机器学习方法的鲁棒性, 专门设置了鲁棒性测试的对比实验。

在实践中, 恶意软件的动态执行序列的修改可以通过 hook 特定的 API 来实现, 理论上只要投入足够的时间任何的 API hook 机制都可以建立。本文作为实验验证设计了一种随机插入 API 的方法, 通过插入无用的 API 调用(no-operation)来保证恶意软件功能前提下, 实现 API 序列的更改。

为了衡量对抗的 API 序列的修改程度, 我们使用修改率 C 作为衡量指标, C 的计算公式如下

$$C = \frac{insert\_num}{L} \tag{5}$$

其中分子为插入 API 调用动作的个数, 分母为实验中 API 的总长度。插入 API 必须保证不对程序原有的执行结果造成影响, 我们选用四个 API 作为插入对象 [NtCreateFile, NtWriteFile, NtOpenFile, NtReadFile], 在原有的测试样本的执行序列中随机插入这个 no-op 序列。

我们分别衡量了插入 0 组, 1 组, 2 组, 3 组, 4 组, 5 组 API 序列之后, 模型性能指标的变化情况。序列长度为 220 时, 插入 1 组, 2 组, 3 组, 4 组, 5 组对抗 API 序列修改率分别为 0.018, 0.036, 0.054, 0.072。序列长度为 400 时, 修改率分别为 0.01, 0.02, 0.03, 0.04, 0.05。

如图 3 所示, 图片上排为序列长度为 220 时, 分别对测试样本插入 0 组, 1 组, 2 组, 3 组, 4 组, 5 组对抗序列之后, 普通机器学习方法与深度学习方法性能指标的损失情况。从图形中可以看出, 插入对抗序列之后明显会降低恶意软件家族分类的性能, 但是相对传统的朴素贝叶斯(NB)与随机森林(RF)方法, 深度学习方法可以获得更加稳定的性能, 良好的实现恶意软件家族检测的分类。序列长度为 400 时, 深度学习模型同样表现较好。

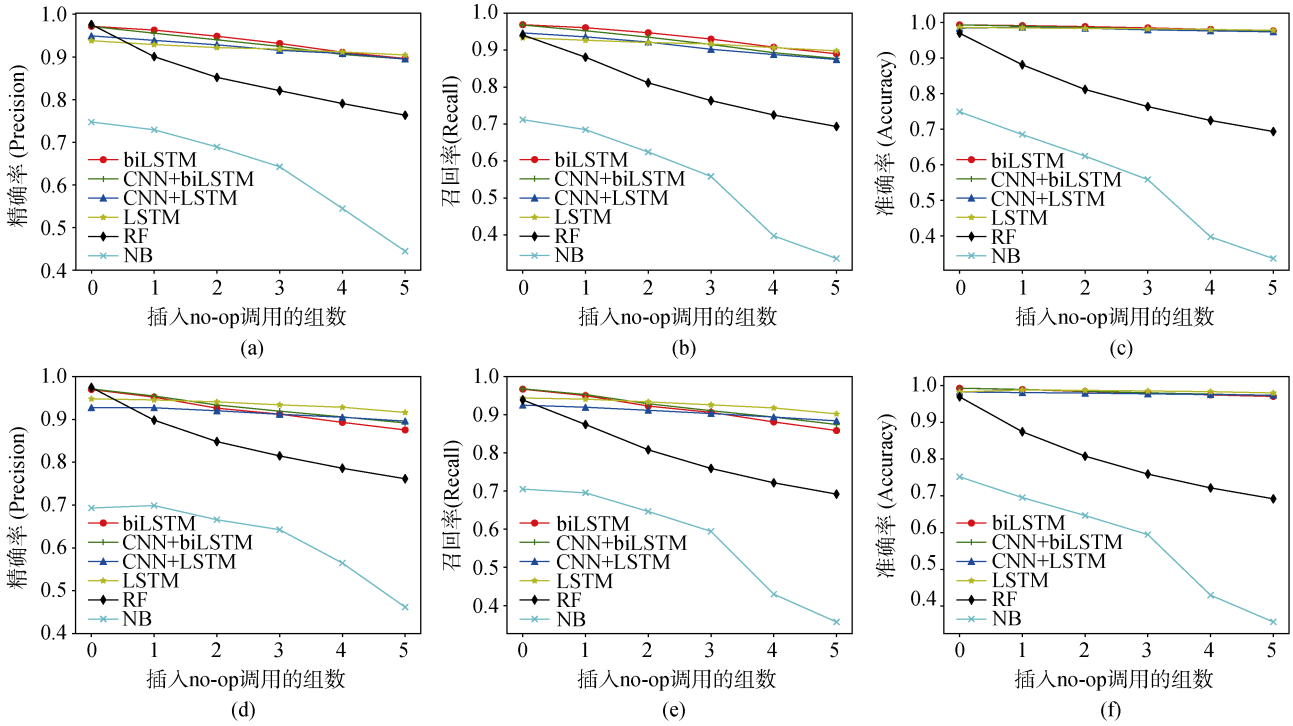

图 3　无用 API 插入时模型的性能损失(*L* 表示 API 序列的长度), (a)*L*=220 的精确率损失变化, (b) *L*=220 的召回率损失变化, (c) *L*=220 的准确率损失变化, (d) *L*=400 的精确率损失变化, (e) *L*=400 的召回率损失变化, (f) *L*=400 的准确率损失变化

Figure 3　Performance loss when no-op API inserted(*L* denote API length), (a) precision rate change *L*=220,(b)recall rate change *L*=220, (c)accuracy rate change *L*=220, (d) precision rate change *L*=400,(e)recall rate change *L*=400, (f)accuracy rate change *L*=400

测试样本序列长度为 400 时, 可以看到五角星线条的 LSTM 方法相对其他的方法下降趋势较小。但是序列长度为 220 时则没有这样的趋势, 可以推断, 这是因为序列长度为 400 的 LSTM 识别模型参数较多, 而 LSTM 参数较少的情况下具有更高的鲁棒性。

5　结论

本文探索了深度学习模型对恶意软件家族的动态特征的分类性能, 通过研究恶意软件 API 动态调用规律, 从前向依赖关系与后向依赖关系的分类中引入双向 LSTM 网络作为网络层模型。从实验结果可以看出, 带有双向 LSTM 网络层的深度学习方法在家族分类的实验中获取了更好的效果, 提出的方法相对文献[13]的深度学习方法获得了较大的性能提升。本文首次在样本家族的多分类问题中测试了样本分类的鲁棒性特点。经过实验验证, 序列的插入会导致分类模型的分类性能下降, 且深度学习方法相对传统的浅层机器学习方法性能损失要小。也就是说深度学习模型在样本家族的多

分类问题中能够一定程度的抵抗对抗样本的攻击。虽然本文鲁棒性测试采用的序列插入方法较简单，但是对于深度学习在实际环境中的部署仍然具有一定的参考意义。

参考文献

[1] 360 互联网安全中心 2018 年中国互联网安全报告(个人安全篇) http://zt.360.cn/1101061855.php?dtid=1101062370&did=610132397,Apr 2019.

[2] “Microsoft. Malware names,” Microsoft, https://docs.microsoft.com/en-us/windows/security/threat-protection/intelligence/malware-naming, Jun 2019.

[3] Rosenberg I, Shabtai A, Rokach L, et al. Generic Black-Box End-to-End Attack Against State of the Art API Call Based Malware Classifiers[M]. Research in Attacks, Intrusions, and Defenses. Cham: Springer International Publishing, 2018: 490-510.

[4] Al-Dujaili A, Huang A, Hemberg E, et al. Adversarial Deep Learning for Robust Detection of Binary Encoded Malware[C]. *2018 IEEE Security and Privacy Workshops (SPW)*, May 24, 2018. San Francisco, CA. Piscataway, NJ: IEEE, 2018:76-82.

[5] Saxe J, Berlin K. Deep Neural Network Based Malware Detection Using Two Dimensional Binary Program Features[C]. *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, October 20-22, 2015. Fajardo, PR, USA. Piscataway, NJ: IEEE, 2015:11-20.

[6] Zhang J X, Qin Z, Yin H, et al. IRMD: Malware Variant Detection Using Opcode Image Recognition[C]. *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, December 13-16, 2016. Wuhan, China. Piscataway, NJ: IEEE, 2016: 1175-1180.

[7] Yanfang Ye，Lingwei Chen，Shifu Hou, et al. DL4MD: A Deep Learning Framework for Intelligent Malware Detection[C]. The International Conference on Data Mining(PICDM) , 2016: 61-70.

[8] Ronen R, Radu M, Feuerstein C, et al. Microsoft Malware Classification Challenge[EB/OL]. 2018: arXiv:1802.10135[cs.CR]. https://arxiv.org/abs/1802.10135.

[9] Yue S Q. Imbalanced Malware Images Classification: A CNN Based Approach[EB/OL]. 2017: arXiv:1708.08042[cs.CV]. https://arxiv.org/abs/1708.08042.

[10] Kim J Y, Bu S J, Cho S B. Zero-day Malware Detection Using Transferred Generative Adversarial Networks Based on Deep Autoencoders[J]. *Information Sciences*, 2018, 460/461: 83-102.

[11] Raff E, Sylvester J, Nicholas C. Learning the PE Header, Malware Detection with Minimal Domain Knowledge[C]. *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security - AISec '17*, November 3, 2017. Dallas, Texas, USA. New York, USA: ACM Press, 2017: 121-132.

[12] Huang W Y, Stokes J W. MtNet: A Multi-Task Neural Network for Dynamic Malware Classification[M]. Detection of Intrusions and Malware, and Vulnerability Assessment. Cham: Springer International Publishing, 2016: 399-418.

[13] Kolosnjaji B, Zarras A, Webster G, et al. Deep Learning for Classification of Malware System Call Sequences[M]. AI 2016: Advances in Artificial Intelligence. Cham: Springer International Publishing, 2016: 137-149.

[14] “安天针对勒索蠕虫“魔窟”(WannaCry)的深度分析报,” 安天实验室, https://www.antiy.com/response/wannacry.html, Jun. 2017.

[15] Sebastián M, Rivera R, Kotzias P, et al. AVclass: A Tool for Massive Malware Labeling[M]. Research in Attacks, Intrusions, and Defenses. Cham: Springer International Publishing, 2016: 230-253.

郑锐 于 2017 年在昆明理工大学获得硕士学位，现在武汉大学网络空间安全专业攻读工学博士学位。研究领域为人工智能在网络空间安全中的应用。研究兴趣包括：恶意代码分析，对抗机器学习应用与防范。
Email: zr_12f@whu.edu.cn

汪秋云 于 2014 年在北京信息科技大学获得硕士研究生学位，现任中国科学院信息工程研究所第六研究室工程师，研究领域为高级威胁发现与溯源取证。
Email: wangqiuyun@iie.ac.cn

傅建明 于 2000 年获得博士学位，现为武汉大学国家网络安全学院教授，博士生导师。主要研究领域为恶意代码分析，软件漏洞挖掘与利用。主要研究兴趣为恶意代码检测，漏洞检测与防御。
Email: jmfu@whu. edu.cn

姜政伟 于 2014 年在中国科学院大学计算机应用技术专业获得博士学位，现任中国科学院信息工程研究所高级工程师。研究领域为威胁情报与威胁发现。研究兴趣包括威胁情报智能处理、对抗样本自动分析。
Email: jiangzhengwei@iie.ac.cn

苏日古嘎 于2019年在武汉大学获得硕士学位。现在武汉大学网络空间安全专业攻读博士学位。研究兴趣包括: 恶意代码分析、深度学习等。
Email: mys0gar@whu. edu.cn

汪姝玮 于 2016 在国防科技大学计算机科学与技术专业获得硕士学位。现任中国科学院信息工程研究所工程师三级, 研究领域为恶意代码、深度学习等, 研究兴趣包括恶意代码同源分析等。
Email: wangshuwei@iie.ac.cn