基于递归神经网络的恶意程序检测研究

王乐乐[1] 汪斌强[1] 刘建港[2] 张建辉[1] 苗启广[3]
(国家数字交换系统工程技术研究中心 郑州 450000)[1]
(南京信息技术研究院 南京 210000)[2] (西安电子科技大学计算机学院 西安 710071)[3]

摘 要 针对传统恶意程序检测判定效率低及自动分析恶意程序能力不足的问题,在深度学习环境下,研究利用递归神经网络进行恶意程序的检测分类的问题。首先,用快速模拟器(Quick Emulator,QEMU)捕获到恶意程序运行时所调用的API及其参数序列,经过行为抽象,形成恶意程序的特征序列。然后使用对数化的双线性模型(Hierarchical Log-bilinear Language Model,HLBL)将特征序列映射成固定长度的词向量,并将这些词向量合成递归神经网络(Recursive Neural Network,RNN)所需要的输入矩阵。通过对递归神经网络模型的训练,建立恶意程序的多层语义聚合模型,完成对恶意程序的分类检测。实验数据表明,递归神经网络模型在恶意程序检测分类中能够有效地检测出恶意程序,与传统机器学习算法相比,其检测率提高了17%。特别是在引入张量(Tensor)的概念,采用递归张量神经网络(Recursive Neural Tensor Network,RNTN)模型后,通过降低整体的参数数量和计算量,使检测率较RNN模型又提高了7%。实验数据充分说明,采用递归神经网络模型完全可以完成大数据环境下恶意程序的检测分类任务。

关键词 QEMU,HLBL,词向量,递归神经网络,多层语义聚合模型
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.07.013

Study on Malicious Program Detection Based on Recurrent Neural Network

WANG Le-le[1] WANG Bin-qiang[1] LIU Jian-gang[2] ZHANG Jian-hui[1] MIAO Qi-guang[3]
(National Digital Switching System Engineering and Technological Research Center,Zhengzhou 450000,China)[1]
(Nanjing Information Technology Institute,Nanjing 210000,China)[2]
(Department of Computer Science,Xidian University,Xi'an 710071,China)[3]

Abstract In view of the low efficiency of traditional malicious program detection and the lack of automatic analysis of malicious programs,this paper studied to use recurrent neural networks to detect and classify malicious programs in deep learning environment. First,the QEMU is used to capture the API and its parameter sequence that are called when the malicious program runs,after the behavior abstraction,the characteristic sequence of the malicious program is formed. Then the feature sequence is mapped to a fixed length word vector by using a logarithmic bilinear model (HLBL),and these word vectors are synthesized into an input matrix of a recursive neural network (RNN). Through the training of the recursive neural network model,a multi-layer semantic aggregation model of malicious programs is established to complete the classification detection of malicious programs. The experimental data show that the recursive neural network model can detect malicious program effectively in the classification of malicious program detection. Compared with the traditional machine learning algorithm,its detection rate has increased by 17%. In particular,when the concept of tensors is introduced,after using the Recursive Neural Tensor Network (RNTN) model,the detection rate is increased by 7% compared to the RNN model by reducing the overall number of parameters and the amount of calculations. The experimental data fully show that the recursive neural network model can complete the detection and classification of malicious programs in big data environment.

Keywords Quick emulator,Hierarchical log-bilinear language model,Word vector,Recursive neural network,Multi-level semantic aggregate model

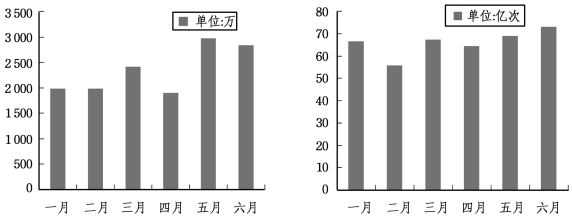
1 引言

恶意程序包括病毒、蠕虫、木马等,它们利用用户主机、网络、服务器或其他设备的漏洞,肆意地窃取网络用户的敏感信息和隐私数据,造成不可估量的损失。当前,随着互联网和移动终端技术的飞速发展,恶意程序已经成为威胁网络安全的

到稿日期:2018-09-08 返修日期:2018-11-10

王乐乐(1985-),女,博士生,主要研究方向为信息安全,E-mail:635718080@qq.com;**汪斌强**(1963-),男,教授,博士生导师,主要研究方向为网络安全;**刘建港**(1968-),男,研究员,主要研究方向为信息安全;**张建辉**(1977-),男,博士,副研究员,主要研究方向为宽带信息网络,E-mail:ndsczjh@163.com(通信作者);**苗启广**(1972-),男,博士,教授,CCF会员,主要研究方向为机器学习、高性能计算。

重大隐患，是我们必须解决的重大安全问题。2018 年上半年，360 互联网安全中心累计截获新增恶意程序样本 1.4 亿个，平均每天截获新增恶意程序样本 79.5 万个。其中新增 PC 端恶意程序样本 14099.8 万个，平均每天截获新增恶意程序样本 77.9 万个，同比 2017 年上半年（6 617.2 万个）上升 113%。累计拦截恶意程序攻击 396.5 亿次，同比（2017 年 315.6 亿次）上升 25.6%，如图 1 所示[1]。另一方面，攻击者使用加壳、反调试、反跟踪、反虚拟化等技术来逃避安全软件的检测，使得从海量可疑文件中提取真正的恶意程序特征进行分析的恶意程序检测工作异常艰巨，给传统的恶意程序分类检测方法带来了巨大的挑战。



(a)新增量　　(b)拦截量

图 1　2018 年上半年 PC 端恶意程序新增量和云查询拦截量[1]

Fig. 1　PC-side malicious program additions and cloud query interceptions in first half of 2018

将深度学习技术应用于恶意程序检测可以准确、有效地发现未知恶意程序。因为深度学习能够对复杂的数据进行有效的特征提取，通过深度学习提取出的特征能够得到有效的特征数据，这些有效的特征数据经过对应的分类算法可以进行有效的分类。

2　相关研究

2.1　深度学习

深度学习（Deep Learning）是机器学习研究中新兴的一个研究领域，由 Hinton 等于 2006 年提出[2]，近年来在很多应用场合都取得了很好的成绩。深度学习通过组合底层特征形成更加抽象的高层表述属性类别或特征，以发现数据的分布式特征表示。其动机在于建立、模拟人的大脑进行分析和学习，即模仿人脑的机制来解释数据。

深度学习网络是一种分层结构，包括输入层、隐藏层和输出层。相邻的层之间有连接，跨层和同层之间没有连接。深度学习首先利用自下而上的非监督学习对每一层进行逐层预训练（Pre-Training）来学习特征；每次单独训练一层，将训练的结果作为更高层的输入；最上层采用自顶向下的监督学习，通过带标签的数据进行训练，误差自顶向下传输，对深度网络进行微调（Fine-Tune）。

近年来，随着深度学习的兴起，多种深度学习模型已逐步应用到恶意程序的分类检测中。其本质是通过对原始数据进行一些非线性的模型转换，将其转换成更高层和更抽象的形式，这些高层和抽象的形式一方面能够加强原始数据的分类能力，另一方面可以去除分类中不相关的特征。研究人员利用深入学习分层预训练的特征学习方式，设计从最低层到最高层的特征检测器，并构建最终的分类模型[3]，将深度学习技术引入恶意程序的检测研究中。Cui 等[4]根据深度学习在图像识别方面的优异性能，提出将恶意代码转换为灰度图像，利用卷积神经网络（Convolutional Neural Network，CNN）对图像进行识别和分类，自动提取恶意软件图像的特征。DING 等[5]利用深度置信网络（Deep Belief Networks，DBN）进行恶意代码的检测，DBN 利用标记数据对多层生成模型进行预训练，更好地反映了样本数据的特征。

深度学习的模型有很多，目前研究人员最常用来进行恶意程序分类检测的深度学习模型与架构包括：自动编码器（Auto Encoder）、深度置信网络（DBN）、卷积神经网络（CNN）、循环神经网络（Recurrent Neural Network，RNN）等。使用这几种模型进行恶意程序分类检测的原理及优缺点如表 1 所列。

表 1　深度学习模型在恶意程序分类检测中的应用

Table 1　Application of deep learning model in classification detection of malicious programs

模型	原理	优缺点
自动编码器	多层前传神经网络，把具体的特征向量转化为抽象的特征向量，达到对高维数据进行特征降维的目的，从而得到低维的特征向量	用于对高维数据的降维。只对数据特征进行表达，无法分类，需在输出时添加分类器才能对恶意程序进行分类
深度置信网络	主要由限制玻尔兹曼机模型（RBM）和 BP 神经网络组成	其中用层叠 RBM 组成深度神经网络进行训练，最后一层用 BP 神经网络进行分类
卷积神经网络	一种深度前馈神经网络，包括卷积层和池化层。卷积层主要用来抽取特征，池化层用来降维	该模型在分类上具有相当优势，多被设计用来进行多维数据的处理，能够准确提取特征的局部相关性，提高特征提取的准确度
循环神经网络	一类具有反馈结构的神经网络，神经网络会对前面的信息进行存储并应用到当前输出的计算中。输出与当前输入和网络权值有关，而且与之前的网络输入有关	RNN 模型有一定记忆功能，对于恶意程序的特征学习和训练有一定优势，可以提高检测的效率

2.2　恶意程序检测

对于恶意程序检测，根据检测的位置目前分为两种方法：基于网络的检测方法[6]和基于主机的检测方法。基于网络的检测方法包括基于蜜罐的方法[7]和基于深层数据包的检测；基于主机的检测方法包括基于校验和的方法、基于签名的方法，以及基于数据挖掘的方法。基于数据挖掘的方法[8]是采用机器学习的方法，其对未知恶意程序的有效检测是通过学习、比较恶意程序与正常程序的各种特征来进行的。恶意程序检测使用到的机器学习方法包括朴素贝叶斯、决策树、支持向量机等。Mahindru 等[9]对 11 000 个安卓恶意程序进行研究，从分类精度和性能等方面详细评估了机器学习的几种分类算法，包括贝叶斯、决策树、随机森林、K 最邻近等。

上述方法虽然在恶意代码检测方面取得了一定的成果，但仍存在一些问题：1）特征提取不合适，检测率和检测精度不高，且算法复杂度高；2）对恶意程序的语义刻画能力不足，无法有效表示语义之间的聚合和传递关系，严重制约了恶意程序的分析性能。基于传统恶意程序检测中出现的问题，本文从深度学习这一角度出发，设计了基于深度学习的恶意程序检测模型。

3　基于递归神经网络的恶意程序检测模型

3.1　模型框架

基于递归神经网络的恶意程序分类检测模型如图 2 所

示，主要包括三大模块：数据预处理、特征提取、递归神经网络。

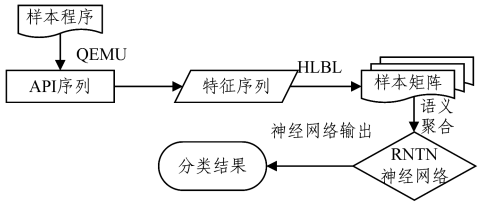


图 2 基于递归神经网络的恶意程序检测模型

Fig. 2 Malicious program detection model based on recursive neural network

3.2 数据预处理

数据预处理主要包括两个方面：1）通过快速模拟器（Quick Emulator，QEMU）[10]捕获到恶意程序运行时所调用的API及其参数序列；2）将API及参数序列抽象成基本行为。

用QEMU模拟器实现对恶意程序的分析。QEMU是一款开源的模拟器，它采用动态二进制翻译技术实现了多源多目标的仿真，是一种应用广泛的开源模拟器。QEMU可以实现对多源多目标的仿真，其内部模块划分清晰，系统架构易于扩展，仿真功能强大。

QEMU系统由以下几部分组成：解释部件、翻译部件、翻译缓存部件、控制核心部件等。其中解释器也为前端解码器，它是一个switch_case结构，用于识别出每一条指令并生成该指令对应的中间代码。翻译器包括中断分析优化器和后端翻译器。后端翻译器实现由中间代码向目标代码的翻译。前端解码器、后端翻译器都是体系结构相关的模块。通过扩展这两个模块可以实现源指令体系结构和目标指令体系结构的灵活扩展。前端解码器、中端分析优化器、后端翻译器构成了QEMU的指令翻译模块TCG（Tiny Code Generator）。

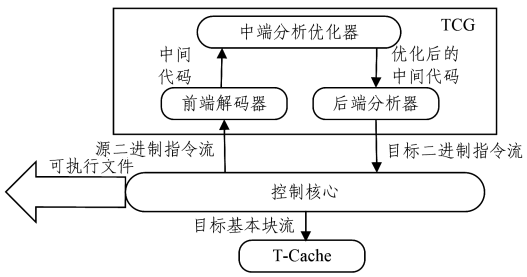


图 3 QEMU系统架构

Fig. 3 QEMU system architecture

QEMU的系统架构如图3所示，控制核心负责控制整个系统的流程，负责取指、翻译、执行3个阶段中的上下文切换。翻译模块（TCG）完成源指令向目标指令的翻译。翻译缓存（T-Cache）实现了已有翻译块的重用。

恶意程序样本在QEMU中运行时，首先会对样本运行时的主进程进行识别，然后通过枚举当前进程，形成未监控进程列表和受监控进行列表，划分出恶意程序样本进程的最初边界。通过监控进程演化列表的不断演化，识别出所有子进程，最终所有的子进程也被纳入监控范围。接着捕获每一个受监控的进程运行时所调用的函数及其参数序列，得到包含进程和参数信息的API调用序列。

API序列的抽象层次程度过低，并且含有大量冗余信息，数据格式也不一致。因此，需要将API调用序列抽象成容易处理的数据形式，得到恶意程序运行时的特征序列，供算法使用。

3.3 特征提取

使用递归神经网络进行恶意程序的分类检测，首先需要将行为抽象得到的特征序列转换成递归神经网络所要求的输入形式。本文是将行为抽象得到的特征序列映射为固定长度的词向量（Word Embedding）[11]，按照特征序列的调用顺序，将词向量合成一个矩阵，作为递归神经网络的输入矩阵。这一过程称作词向量映射。

词向量映射模型主要有n-gram模型、NNLM模型[12]、Log-linear模型、Log-Blinear模型[13]、层次化Log-Blinear模型[14]、CBOW模型、GloVe模型[15]。从计算复杂度等方面综合考虑，本文选取层次化的对数双线性模型（Hierarchical Log-bilinear Language Model，HLBL）进行词向量的映射。

HLBL模型的第一个组成部分是一棵带有叶子节点的二叉树。假定词汇表中的每个词都在一个确切的叶子上，这样每个词就可以用从二叉树的根节点到叶子节点的一条路径来唯一确切表示。这条路径本身可以用二进制的字符串进行编码并在每个节点上进行决策。例如“$d_i=1$”表示访问当前节点的左孩子，字符串“10”对应的路径就是从根节点开始，先访问左孩子节点，再访问左孩子节点的右孩子节点。这样每一个词都可以用一个二进制的字符串来表示，我们把这个二进制的字符串称作词的编码。

HLBL模型的第二个组成部分是为每个节点进行决策的概率模型，这是LBL模型的一个修改版本。在HLBL模型中，使用实值特征向量来表示上下文词。二叉树上的每一个非叶节点也有一个特征向量与之相关联，用于区分其是该节点的左子树上的词还是右子树上的词。被预测的词使用二叉树所确定的二进制编码来表示。这种表示方法相当灵活，节点的特征向量决定了编码中每一个二进制数字对在节点处所进行的决策。

HLBL模型其中的Log Bilinear部分可简要表示为：

$$\hat{r}=\sum_{i=t-1}^{t-n+1}C_i r_{w_i} \tag{1}$$

$$P(w_t=w\mid w_{t-1:t-n+1})=\frac{\exp(\hat{r}^{\mathrm{T}}r_w+b_w)}{\sum_j \exp(\hat{r}^{\mathrm{T}}r_j+b_j)} \tag{2}$$

其中，r_{w_i}是第w_i个词的实际特征向量，$\hat{r}$是预测的特征向量，其值由式(1)计算，C_i是位置i的权重矩阵。

在HLBL模型中，下一个词是w的概率是指根据上下文由词编码所决定的二进制决策序列的概率。由于在节点做出决策的概率仅仅取决于由上下文确定的预测特征向量和该节点的特征向量，因此可以将下一个词的概率表示为二进制决策概率的乘积：

$$P(w_n=w\mid w_{1:n-1})=\prod_i P(d_i\mid q_i,w_{1:n-1}) \tag{3}$$

其中，d_i是词的编码中的第i个数字，q_i是二叉树结构中编码路径上第i个节点对应的特征向量。每个节点的概率由式(4)给出：

$$p(d_i=1|q_i,w_{1:n-1})=\sigma(\hat{r}^{\mathrm{T}}q_i+b_i) \tag{4}$$

其中，$\sigma(x)$是对数函数，b_i 是第 i 个节点的偏置。$P(w_n=w|w_{1:n-1})$表示对 w 所有编码方式求和。

3.4　递归神经网络模型

用递归神经网络模型解决恶意程序的分类检测问题，是从恶意程序的语义本身出发考虑的。实际上，恶意程序的语义是从底层向高层逐层聚合的。在底层，通常由若干关联于同一句柄的 API 构成聚合关系，完成对某一特定系统资源的单次原子操作。中间层是针对某一特定系统资源，由恶意程序生存期内的操作流程构成的聚合关系，表明了对某系统资源的完整操作意图。最高层完成业务逻辑（攻击意图）的聚合。

这种从底层到高层的多层聚合关系描述了恶意程序完成具体恶意行为的途径，即具有类似树形的递归结构，因此本文采用递归神经网络（Recursive Neural Network，RNN）[16] 构建恶意程序语义聚合模型。从树形结构的叶子节点开始，自底向上计算每个叶节点的父节点，父节点同时作为输入继续向上计算其父节点。在自下而上的计算中，根据语义的近似程度，语义相近的先组合，这样从词开始逐步完成语义的聚合，直到得到整句话的语义，在组合的过程中形成自底向上的树形结构。

考虑到整体的计算量和参数量，对递归神经网络中的参数进行优化，每个节点除了用向量描述词的本身含义外，还要用矩阵描述该词语是如何影响其相邻词语的含义，同时引入张量（Tensor）的概念来表示向量和矩阵的乘积，用基于张量的组合函数来代替原来的线性函数。所有节点共用相同的张量参数，以降低整体的参数数量和计算量。

具体到模型来讲，本文选用递归神经张量网络（Recursive Neural Tensor Network，RNTN）[17] 来构建恶意程序的语义聚合模型，利用基于张量积的复合函数完成两个词的合成，如图 4 所示。

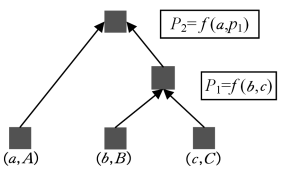


图 4　RNTN 结构

Fig. 4　RNTN structure

计算 (p_1,P_1)，向量 b 和 c 及对应的矩阵进行合成：

$$p_1=f\left\{\begin{bmatrix}b\\c\end{bmatrix}^{\mathrm{T}}V^{[1:d]}\begin{bmatrix}b\\c\end{bmatrix}+W\begin{bmatrix}b\\c\end{bmatrix}\right\} \tag{5}$$

计算(p_2,P_2)，向量矩阵对(p_1,P_1)与 a 进行合成：

$$p_2=f\left\{\begin{bmatrix}a\\p_1\end{bmatrix}^{\mathrm{T}}V^{[1:d]}\begin{bmatrix}a\\p_1\end{bmatrix}+W\begin{bmatrix}a\\p_1\end{bmatrix}\right\} \tag{6}$$

$$P_2=f_M(A,P_1) \tag{7}$$

其中，$V^{[1:d]}\in R^{2d\times 2d\times d}$是定义多个双线性形式的张量，$a$，$b$ 为词向量，A 为词向量 a 的参数矩阵，W 为词向量映射的矩阵，f 为激活函数。自下而上递归计算每一个节点，直至得到顶点节点的向量和矩阵对，获得语义合成的最终结果。

利用 RNTN，使用共同的张量参数，可以简化网络训练，减少所需学习的参数，同时符合恶意程序的语义聚合特性。另一方面，张量层能够保证语义聚合的拟合能力。

RNTN 网络的训练是通过训练每个节点的 softmax 分类器来预测各个类别的分布，表示形式为：

$$y^a=softmax(W,a) \tag{8}$$

其中，W 为分类矩阵，单词大小为 $|V|$，$W\in R^{|V|\times d}$，a 表示 softmax 分类器当前词向量。对于每个节点的测试分类 $y^i\in R^{c*l}$和目标分类 $t^i\in R^{c*l}$，RNTN 网络的训练目标是使交叉熵 $E(y,t,\theta)$最小（C 表示类别的数目）。

RNTN 模型训练交叉熵的定义如下：

$$E(y,t,\theta)=\sum_i\sum_j t_j^i\log y_j^i+\lambda\|\theta\|^2 \tag{9}$$

其中，j 表示第 j 个节点，定义 θ 为参数集(W,W_M,W^{label},L,L_M)，λ 为规范化的先验分布参数，L 为单词的向量集，L_M 为单词的矩阵集。

在神经网络的参数学习中，参数 W，V 的完整求导过程是求解所有非叶节点的导数之和。$V^{[K]}$的求导过程是将每个非叶节点导数相加：

$$\frac{\partial E}{\partial V^{[k]}}=\frac{E^{p_2}}{\partial V^{[k]}}+\delta_k^{p_1,com}\begin{bmatrix}b\\c\end{bmatrix}\begin{bmatrix}b\\c\end{bmatrix}^{\mathrm{T}} \tag{10}$$

可以采用同样的方法对 W 求导，使用该方法使交叉熵最小化以达到 RNTN 模型的训练目的，这里利用张量使计算过程得到简化，提高运算效率。

经过训练得到的递归神经网络不仅可以帮助我们研究恶意程序的恶意性程度和恶意类型在多层语义聚合过程中的传递关系，还可以精确估计程序的恶意性程度和恶意类型，从而完成对恶意程序的检测分类任务。

4　实验设计及分析

4.1　实验步骤

本文实验步骤如下。

1）将样本程序上传至 QEMU 中进行动态分析和运行，得到每个样本程序在虚拟机环境中运行时所调用的 API 序列。

2）用层次化的对数双线性模型进行词向量的映射。

3）用 RNTN 网络进行样本分类。

4）参照实验，在相同数据集上，采用递归神经网络模型（RNN）和机器学习中的支持向量机（SVM）这两种方法对样本进行分类。

4.2　实验环境

使用 Python 语言（Python 2.7），开源 Python 分词工具 Jieba，深度学习框架 Tensorflow 进行实验。在 QEMU 上模拟的系统为 Windows XP，样本运行时调用的 API 为 Win32 API。

本实验所采用的样本来源为互联网及笔者日常工作的积累。样本分为训练样本和测试样本两类，共 2 000 个样本程序，涵盖 10 个家族的恶意程序，这 10 类程序在训练样本和测试样本中都存在。其中恶意样本程序和正常样本程序各 1 000 个。在 1 000 个恶意程序样本中，包含 900 个训练恶意样本程序，100 个测试恶意样本程序；在 1 000 个正常程序样本中，包含 900 个训练正常样本程序，100 个测试正常样本程序。

4.3 评价标准

本文采用恶意样本程序检测率 F 和正常样本程序误检率 W 作为评价标准。其中：

$$\text{恶意样本程序检测率 } F=\frac{\text{被检测出为恶意程序}}{\text{全部的恶意程序样本}}\times 100\%$$

$$\text{正常样本程序误检率 } W=\frac{\text{被检测出为恶意程序}}{\text{全部的正常程序样本}}\times 100\%$$

一般地，检测率越高，误检率越低的实验方法，被认为是较好的实验方法。

4.4 实验结果及分析

样本集上的实验结果如表 2 所列。

表 2 样本集上的实验结果

Table 2 Experimental results on sample set

（单位：%）

分类方法	检测率	误检率
SVM	67	6
RNN	84	12
RNTN	91	16

从实验中可以看出，与传统的机器学习方法相比，深度学习模型在恶意代码检测任务上获得了不错的效果。特别是 RNTN 模型，其准确率达到了 91%，比机器学习中的 SVM 模型的准确率提高了 24%，与 RNN 模型相比提高了 7%。因此，在恶意代码检测领域，递归神经网络模型更具有潜力。

对于实际的恶意代码检测任务来说，误检率也是一个重要的评价指标。误检是指将正常程序误检为恶意程序，如果正常程序是重要文件，误检而导致其被删除将造成严重后果。因此，控制误检率具有实际意义，但是过度地控制也会使检测率降低，失去对恶意程序检测的意义。实验中，采用 RNN 和 RNTN 这两个深度学习模型进行检测分类的误检率高于机器学习方法的误检率，这与样本集的选取有很大的关系，因此在下一步的研究中，要增加样本集的数量并丰富样本的类型，通过多次实验来降低实验结果对样本集的依赖性。

结束语 本文通过 QEMU 模拟器对运行中的恶意程序进行监控，获得其 API 调用序列，经过行为抽象得到其特征序列。在深度学习的环境下，用对数化的双线性模型 HLBL 对 API 调用序列进行词向量映射，把每个 API 序列映射成词向量，再整理成大小固定的矩阵，作为递归神经网络的输入。最后利用递归张量神经网络建立恶意程序的多层语义聚合模型，描述恶意程序的多层语义聚合和传递关系，为恶意程序分类检测提供基础。实验测试表明，用递归张量神经网络模型进行恶意代码的检测分类获得了较高的检测率，达到了预期的设计目标。下一步，一方面将对现有的递归张量神经网络的检测方法进行改进，提高模型的训练速度；另一方面，将继续探究新的恶意程序检测方法，以提高检测效率。

参 考 文 献

[1] 360 互联网安全中心. 2018 年上半年互联网安全报告[EB/OL]. www.anquanke.com/post/id/156689.

[2] HINTON G, OSINDERO S, WELLING M, et al. Unsupervised discovery of nonlinear structure using contrastive backpropagation [J]. Cognitive Science, 2006, 30(4): 725-731.

[3] LV Y, DUAN Y, KANG W, et al. Traffic Flow Prediction With Big Data: A Deep Learning Approach [J]. IEEE Transactions on Intelligent Transportation Systems, 2015, 16(2): 865-873.

[4] CUI Z, XUE F, CAI X, et al. Detection of Malicious Code Variants Based on Deep Learning [J]. IEEE Transactions on Industrial Informatics, 2018, 14(7): 3187-3196.

[5] DING Y, ZHU S. Malware detection based on deep learning algorithm [J]. Neural Computing & Applications, 2017(1): 1-12.

[6] IDIKA N, MATHUR A P. A survey of malware detection techniques[R]. Purdue University, 2007.

[7] PEREVOZCHIKOV V A, SHAYMARDANOV T A, CHUGUNKOV I V. New techniques of malware detection using FTP Honeypot systems[C]// Young Researchers in Electrical and Electronic Engineering. IEEE, 2017: 204-207.

[8] YE Y, LI T, ADJEROH D, et al. A survey on malware detection using data mining techniques [J]. ACM Computing Surveys (CSUR), 2017, 50(3): 1-40.

[9] MAHINDRU A, SINGH P. Dynamic Permissions based Android Malware Detection using Machine Learning Techniques [C]// Innovations in Software Engineering Conference. ACM, 2017: 202-210.

[10] BELLARD F. QEMU, a fast and portable dynamic translator [C]// Conference on Usenix Technical Conference. USENIX Association, 2005: 41.

[11] HINTON G E. Learning distributed representations of concepts [C]// Eighth Conference of the Cognitive Science Society. 1989.

[12] BENGIO Y, VINCENT P, JANVIN C. A neural probabilistic language model [J]. Journal of Machine Learning Research, 2003, 3(6): 1137-1155.

[13] MNIH A, HINTON G. Three new graphical models for statistical language modelling[C]// International Conference on Machine Learning. ACM, 2007: 641-648.

[14] MNIH A, HINTON G. A scalable hierarchical distributed language model[C]// International Conference on Neural Information Processing Systems. Curran Associates Inc. 2008: 1081-1088.

[15] PENNINGTON J, SOCHER R, MANNING C. Glove: Global Vectors for Word Representation[C]// Conference on Empirical Methods in Natural Language Processing. 2014: 1532-1543.

[16] SOCHER R, MANNING C D, NG A Y. Learning continuous phrase representations and syntactic parsing with recursive neural networks[C]// Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop. 2010: 1-9.

[17] SOCHER R, PERELYGIN A, WU J, et al. Recursive deep models for semantic compositionality over a sentiment treebank [C]// Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. 2013: 1631-1642.