基于行为特征的 PowerShell 恶意代码检测模型

雷鑫焱[1] 高 见[1,2] 王凯悦[1]
LEI Xinyan GAO Jian WANG Kaiyue

摘 要 随着网络攻击技术的发展，PowerShell 恶意代码被广泛应用于无文件攻击中。为了有效检测 PowerShell 恶意代码，提出一种基于行为特征的 PowerShell 恶意代码检测模型。首先，通过搭建 CAPE 沙箱运行分析 PowerShell 脚本提取 API 序列。随后，使用一维卷积层提取获取 API 序列的短距离依赖关系，在应用 Bi-LSTM 获取 API 序列之间的时序依赖关系后，利用 Transformer 编码器捕获序列间的长距离依赖和全局关系。最后，使用全连接层实现恶意性检测。实验结果表明，模型能够有效检测 PowerShell 恶意代码。

关键词 PowerShell；恶意代码检测；动态分析；API 序列；深度学习

doi：10.3969/j.issn.1672-9528.2024.04.012

0 引言

随着互联网的广泛应用，恶意代码种类和数量开始快速增长。为应对这些恶意代码对计算系统的威胁，安全专家开发了各种检测工具以识别和防御潜在的安全风险。然而，随着攻击者采用混淆技术来隐藏其恶意脚本的真实目的，以规避检测工具的监控，传统的静态分析技术识别恶意代码变得更加困难。静态分析主要是分析已知的恶意代码，面对新出现的攻击技巧往往力不从心。相比之下，动态分析技术通过监测和分析恶意代码在运行时的行为，例如系统调用、文件操作和注册表的变更等，能够揭示恶意代码的实际意图，有效地补充了静态分析的不足。

早在 2016 年，赛门铁克就已经发现使用 PowerShell 工具进行网络攻击的趋势[1]，这些攻击包括金融威胁、网络钓鱼邮件、勒索软件、加密挖掘和高级持续威胁攻击等。根据赛门铁克的报告[2]，2020 年在赛门铁克客户环境中，使用 LOTL 攻击技术的恶意攻击的执行次数从 2019 年的 59 亿次增加到了 2020 年的 76 亿次。从 2021 年 1 月初到 2021 年 9 月底，该类型恶意攻击的执行次数与 2020 年同期相比增加了 44.9%。在这些网络攻击中，恶意使用 PowerShell 的事件占比达 39%。

本文提出了一种基于行为特征的 PowerShell 恶意代码检测模型，使用深度学习方法深入挖掘 PowerShell 脚本的动态特征。首先，搭建 Cape 沙箱虚拟环境运行 PowerShell 脚本以获取 API 调用序列，并将 API 序列转换为向量形式。然后，使用一维卷积层进行处理，以捕获 API 序列中短距离的依赖关系。随后，应用双向长短期记忆网络挖掘脚本执行过程中 API 序列之间的时间序列依赖关系，使用 Transformer 模块进一步增强模型对长距离依赖关系的识别能力，聚焦重点动态行为信息。最后，使用全连接层决策实现恶意性检测。

1 模型整体框架

模型框架主要包括三个部分，分别为特征提取、关系挖掘和分类，如图 1 所示。

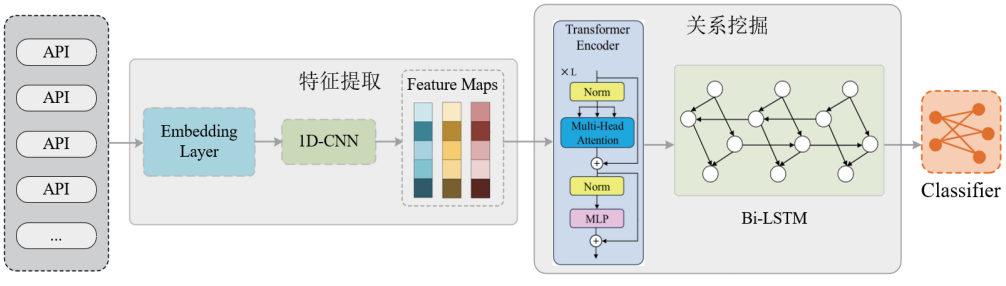


图 1 模型整体框架

（1）特征提取：首先使用嵌入层将 API 调用序列转换为向量表示，然后采用一维卷积神经网络（1D-CNN）作为特征提取机制，从原始 API 特征中学习到有用的表示。

1. 中国人民公安大学信息网络安全学院 北京 100038
2. 安全防范与风险评估公安部重点实验室 北京 102623

（2）关系挖掘：先利用双向长短期记忆网络（Bi-LSTM）捕捉 API 调用之间的时序依赖关系，随后应用 Transformer 进一步挖掘关键特征。其中，Bi-LSTM 的强大记忆力适于处理长距离的时序依赖关系，而 Bi-LSTM 的双向结构使模型可以同时整合过去和未来状态的关系信息，增强时序特征的表现能力。Transformer 的自注意力机制和位置编码能够增强模型的全局依赖关系的捕获能力。

（3）分类：通过多个全连接层（Linear）逐步压缩特征空间，使用 Sigmoid 激活函数输出脚本是否的预测结果。

1.1 特征提取模块

嵌入层是一种广泛应用于自然语言处理领域的特征表示方法[3]。在本文提出的框架中，应用嵌入层是为了将每个 API 名称转换为 k 维密集特征向量。如图 2 所示，若设 $a_i \in \mathbb{R}^k$ 为 API 序列中第 i 个 API 对应的 k 维特征向量，由 n 个 API 组成的 API 序列就可以表示为：

$$A_{1:n} = a_1 \oplus a_2 \oplus \cdots \oplus a_n \tag{1}$$

式中：$\oplus$ 表示串联操作，$A_{1:n} \in \mathbb{R}^{n \times k}$ 表示一个 $n \times k$ 的序列矩阵。

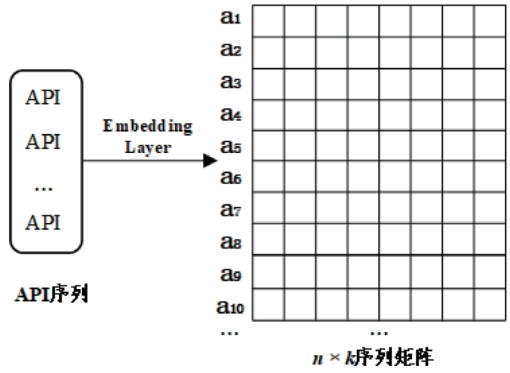


图 2 API 序列嵌入表示

1.2 关系挖掘模块

一个 API 代表一个软件操作，多个操作代表一个脚本的行为，恶意脚本操作之间的前后顺序往往更能体现其真实意图，挖掘不同操作之间的时序关系可以有效地分辨该脚本的行为特征。模型采用 Bi-LSTM 模块和 Transformer 深入挖掘 API 序列中的时序依赖关系和全局关键特征。

LSTM 是一种递归神经网络结构，能够借助几个控制信息传输状态的门捕获长期上下文信息[4]。双向长短期记忆网络（Bi-LSTM）使用两个 LSTM 互相连接，每个输入的特征会从正反方向经过循环神经网络。Bi-LSTM 共享权重以将特征映射到向量空间上，前向 $\overrightarrow{LSTM}$ 和后向 $\overleftarrow{LSTM}$ 在时刻 t 的输入处理如公式（2）和（3）所示：

$$\overrightarrow{h_t} = \overrightarrow{LSTM}(x_t, \overrightarrow{h_{t-1}}) \tag{2}$$

$$\overleftarrow{h_t} = \overleftarrow{LSTM}(x_t, \overleftarrow{h_{t-1}}) \tag{3}$$

自注意力机制（self-attention mechanism）是一种在深度学习模型中用于处理序列数据的技术，通过对特征的不同部分设置不同的权重，能够在特征中选出关键信息，从而提高模型的准确性及效率。

Transformer 是自注意力机制中的一种变形，其在步骤长距离依赖上的并行处理能力出色，于 2017 年由 Vaswani A 等人提出[5]。Transformer 模型的架构图如图 3 所示，可以看出，模型主要由编码器和解码器组成。其中，编码器（encoder）由多头注意力机制（multi-head attention）、残差和标准化（add & norm）、前馈层（feed forward）三个小模块组成，而解码器的组成部分多了一个掩码多头注意力机制（masked multi-head attention）。

借助 Transformer 在全局特征提取上的出色性能对 API 特征进行聚焦，深入挖掘特征之间的关联，能实现对脚本行为恶意性的识别。

1.3 分类

在 Transformer 模块之后，使用多层感知机（multi-layer perceptron，MLP）分类模块进行决策。模型采用 2 个 Dense 层和 2 个 dropout 层作为 MLP 的隐藏层，每个 Dense 层之后，使用 dropout 层来减少过拟合。然后，在最后一个 Dense 层应用 Sigmoid 激活函数输出脚本恶意性的预测概率。在整体训练中，使用 Adam 作为优化器，并使用 [0,1] 标注每个输入序列，使用二元交叉熵函数计算训练阶段的损失，函数定义为：

$$L = -\left(\sum_{i=1}^{n} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - y^{(i)}) \right) \tag{4}$$

模型整体网络结构图如图 4 所示。

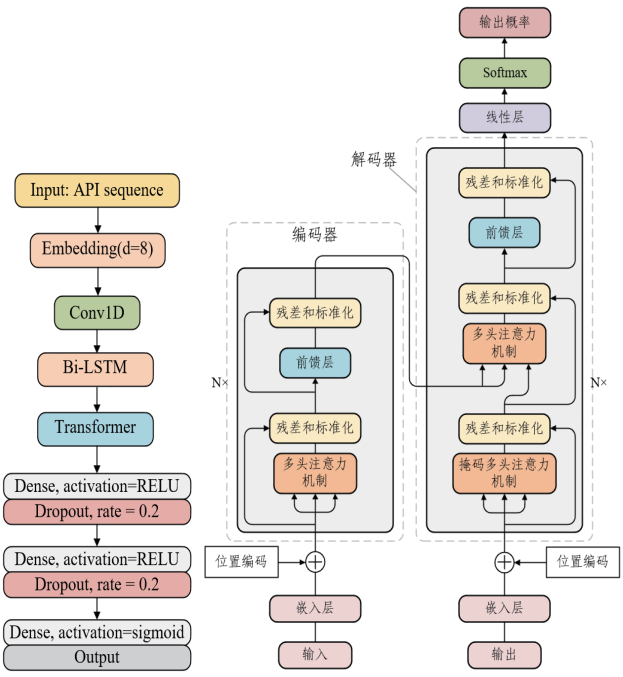


图 3 Transformer 架构图　　图 4 模型整体网络结构

2 动态分析环境

动态分析需要通过在真实环境中运行脚本，监视脚本，获取其真实的目的，以抵抗脚本混淆技术。要实现对脚本的动态分析，需要构建一个安全的分析环境。本文使用 CAPE 沙箱 [6] 进行动态分析实验，在 VMware 中构建 Ubuntu 20.04 系统的虚拟机进行系统嵌套安装，在 Ubuntu 虚拟机中安装 VirtualBox 软件和 CAPE 沙箱，使用 VirtualBox 搭建 Windows7 64 位系统的虚拟机作为感染主机。

在本文中，每个样本的最大运行时间设置为 4 分钟。当分析完成时，CAPE 服务器将生成关于该脚本的运行日志，并从中提取 API 序列。由于不同脚本形成的 API 序列长度不同，模型对提取的 API 序列进行规整化，使用前 1000 个不连续重复的 API 作为模型的输入序列。图 5 为使用 CAPE 沙箱提取 PowerShell 脚本的 API 序列的过程。

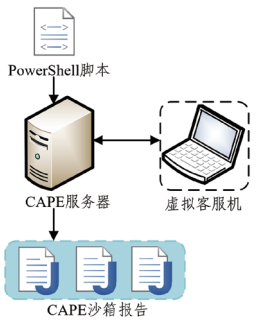


图 5 动态分析过程

3 实验设计

3.1 数据集

由于 PowerShell 缺乏基准数据集，本文使用多个数据集组成实验数据集，分别包括 Jeff White 在 Github 上公开的 PowerShell 脚本数据集 [7]，以及方勇、柴华君在其论文中所使用的 PowerShell 脚本数据集 [8-9]。删除重复样本和无法运行的样本后，如表 1 所示，获得了共 7603 个 PowerShell 恶意样本和 4316 个良性样本。

表 1 实验数据集来源

Source	Malicious	Benign
Jeff White	2968	0
Fang et al.	4202	4316
Chai et al.	3346	0
Our dataset	7603	4316

3.2 评价指标

在本文的实验中，混淆矩阵将被应用于评估模型的性能。此外，还计算了准确率、精确率、召回率和 F_1 值，公式见（5）～（8）。评估分类有效性的指标的定义如下。

（1）真阳性 TP：在分类结果中被预测为正样本，实际为正样本的样本数量。

（2）假阳性 FP：在分类结果中被预测为正样本，实际为负样本的样本数量。

（3）真阴性 TN：在分类结果中被预测为负样本，实际为负样本的样本数量。

（4）假阴性 FN：在分类结果中被预测为负样本，实际为真样本的样本数量。

$$ACC=\frac{TP+TN}{TP+TN+FP+FN} \quad (5)$$

$$PRE=\frac{TP}{TP+FP} \quad (6)$$

$$Recall=\frac{TP}{TP+FN} \quad (7)$$

$$F1=2\times\frac{PRE\times Recall}{PRE+Recall} \quad (8)$$

3.3 实验环境

本文实验环境配置见表 2。

表 2 实验环境软硬件配置

实验环境	配置
操作系统	Windows 11
CPU	11th Gen Intel(R) Core(TM) i5-11400H @ 2.70 GHz 2.69 GHz
GPU	NVIDIA GeForce RTX 3050 Ti Laptop GPU
内存	32 GB
编程语言	Python3.9
IDE	PyCharm
其他工具	Cuckoo Sandbox, VMware, VirtualBox

3.4 对比实验

由于缺少对 PowerShell 动态分析的研究，为了有效比对模型性能，本文选取部分机器学习模型进行性能对比。将样本中每个 API 调用的使用频率作为特征输入到机器学习模型中，获取检测结果，本文选取的机器学习模型包括：朴素贝叶斯（naive bayes，NB）、K 近邻（K-Nearest Neighbor，KNN）、决策树（decision tree，DT）、随机森林（random forest，RF）、支持向量机（support vector machine，SVM）和极限梯度提升（extreme gradient boosting，XGBoost）[10]。为了验证模型各模块对性能的影响，设计了相关消融实验。

4 实验结果与分析

4.1 基线实验

本文选用机器学习（machine learning，ML）模型作为基线进行对比实验，实验结果如表 3 所示。

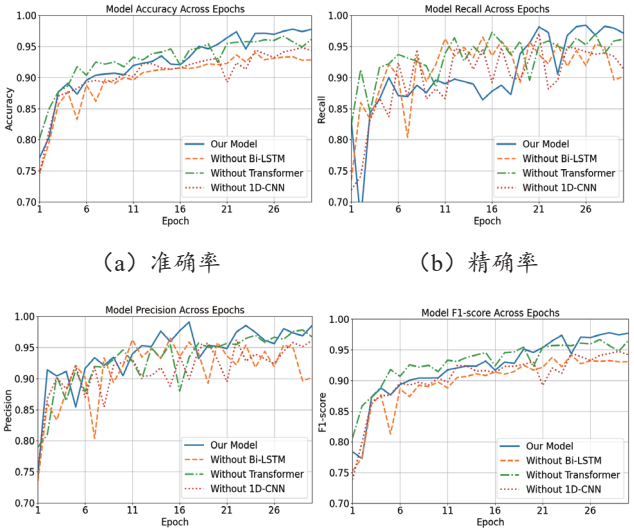
表 3 与基于频率统计的机器学习模型效果对比

方法	准确率	精确率	召回率	F_1 值
Naive Bayes	74.62%	73.46%	77.10%	0.752 3
KNN	85.62%	83.31%	89.10%	0.861 1
Decision Tree	89.05%	89.36%	88.65%	0.890 1
Random Forest	93.27%	92.74%	93.90%	0.933 1
SVM	90.10%	93.12%	86.60%	0.897 4
XGBoost	94.30%	93.05%	95.75%	0.943 8
Our model	**97.75%**	**98.51%**	**97.15%**	**0.970 1**

本文提出的模型在实验数据集上的准确率为 97.75%，精确率为 95.46%，召回率为 98.90%，F_1 值为 0.971 5。根据表 3 结果，本文提出的模型明显优于 ML 基线模型，这表明基于频率统计的 ML 模型对 API 序列的特征挖掘明显不足。

4.2 消融实验

为了评估每个模块在实验性能上的贡献，从模型体系结构中单独移除每个模块，并进行实验以评估模型的性能。本文提出的模型主要有三个模块，分别是 1D-CNN 模块、Bi-LSTM 模块、Transformer 模块，去除后的实验结果使用折线图进行展示，如图 6 所示。



（a）准确率　（b）精确率

（c）召回率　（d）F1 值

图 6 消融实验结果对比

从图 6 可以看出，在去除各模块之后，模型的表现性能都有所下降，其中影响最大的是 Bi-LSTM 模块，去除该模块后，模型的各性能指标表现都最差；影响第二大的便是 1D-CNN 模块，然后是 Transformer 模块。

5 讨论

本文设计了一个基于行为特征的 PowerShell 恶意代码检测模型，模型使用 CAPE 沙箱动态分析 PowerShell 脚本获取 API 序列，随后使用 1D-CNN 层提取获取 API 序列的短距离依赖关系，应用 Bi-LSTM 和 Transformer 获取 API 序列之间的时序依赖关系和长距离依赖关系后，使用全连接层实现恶意性检测。该模型补充了 PowerShell 恶意代码动态分析的研究空白，最终在真实数据集上取得了 97.75% 的准确率。

参考文献：

[1]SYMANTEC.Security center white papers[EB/OL].(2020-08-08)[2023-12-21]. https://symc.ly/2TlKph.

[2]SYMANTEC THREAT HUNTER TEAM. The threat landscape in 2021[EB/OL].(2021-12-17)[2023-12-21]. https://symantec.drift.click/Threat_Landscape_2021_Whitepaper.

[3]KETKAR N, SANTANA E. Deep learning with Python[M]. Berkeley, CA: Apress, 2017.

[4]PICHOTTA K, MOONEY R J. Learning statistical scripts with LSTM recurrent neural networks[C]//Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence and the Twenty-Eighth Innovative Applications of Artificial Intelligence Conference,v.4. Palo Alto:AAAI Press, 2016: 2800-2806.

[5]VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[EB/OL].(2017-06-12)[2024-02-20].https://arxiv.org/abs/1706.03762.

[6]KEVIN O, ANDRIY B. CAPEv2[EB/OL].(2024-03-18)[2024-02-20].https://github.com/kevoreilly/CAPEv2.

[7]WHITE J.Pulling back the curtains on encoded command powershell attacks[EB/OL].(2017-03-10)[2024-02-18]. https://unit42.paloaltonetworks.com/unit42-pulling-back-the-curtains-on-encodedcommand-powershell-attacks.

[8]CHAI H, YING L, DUAN H, et al. Invoke-deobfuscation: AST-based and semantics-preserving deobfuscation for powershell scripts[C]//Proceedings of the 2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). Piscataway:IEEE,2022:295-306.

[9]FANG Y, ZHOU X, HUANG C. Effective method for detecting malicious PowerShell scripts based on hybrid features[J]. Neurocomputing, 2021,448:30-39.

[10] 杨剑锋，乔佩蕊，李永梅，等．机器学习分类问题及算法研究综述 [J]. 统计与决策 ,2019,35(6):36-40.

【作者简介】

雷鑫焱（2000—），男，江西南昌人，硕士研究生，研究方向：恶意代码识别、机器学习等。

高见（1982—），男，山东菏泽人，副教授，博士，研究方向：网络安全、恶意代码、僵尸网络。

王凯悦（1998—），男，江苏徐州人，硕士研究生，研究方向：恶意流量检测、深度学习等。

（收稿日期：2024-03-24）