# 基于特征组合的 Powershell 恶意代码检测方法

刘 岳[1,2], 刘宝旭[1,2], 赵子豪[1,2], 刘潮歌[1,2], 王晓茜[1,2], 吴贤达[1,2]

[1] 中国科学院信息工程研究所 北京 中国 100093
[2] 中国科学院大学网络空间安全学院 北京 中国 100049

**摘要** 近年来, Powershell 由于其易用性强、隐蔽性高的特点被广泛应用于 APT 攻击中, 传统的基于人工特征提取和机器学习方法的恶意代码检测技术在 Powershell 恶意代码检测中越来越难以有效。本文提出了一种基于随机森林特征组合和深度学习的 Powershell 恶意代码检测方法。该方法使用随机森林生成更好表征原始数据的新特征组合, 随后使用深度学习神经网络训练并进行分类识别。该方法可以弥补人工特征工程经验不足的问题, 更好表征原始数据从而提高检测效果。本文实验结果显示, 利用本文提出方法构建的 Powershell 恶意代码检测系统性能良好, 在真实数据集中的召回率、准确率均在 99%以上, 可以对 Powershell 恶意代码进行有效的检测识别。

**关键词** Powershell; 恶意代码; APT; 深度学习; 随机森林
**中图法分类号** TP393.08 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2021.01.04

# Powershell malware detection method based on features combination

LIU Yue[1,2], LIU Baoxu[1,2], ZHAO Zihao[1,2], LIU Chaoge[1,2] , WANG Xiaoxi[1,2] , WU Xianda[1,2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

**Abstract** In recent years, powershell is widely used in APT attack due to its ease of use and high concealment. Traditional malicious code detection technology based on artificial feature extraction and machine learning method is more and more difficult to be effective in the detection of malicious code in PowerShell. For this reason, this paper proposes a malicious Powershell code detection method based on random forest features combination and deep learning. This method uses random forest to generate new features which better characterize the original data, and uses deep neural network to build classifiers for classification and recognition. This method can make up for the lack of experience in artificial feature engineering, and characterize the original data better, so as to improve the detection effect. The experimental results in this article show that this method has a good performance, high recall rate and accuracy rate, which can effectively detect and identify malicious Powershell code.

**Key words** Powershell; malicious code; APT; deep learning; random forest

## 1 引言

随着互联网技术的发展, 人们的生产和生活方式发生了巨大的改变[1], 网络安全问题也愈发严峻。据赛门铁克发布的互联网安全威胁报告显示[2], 2019 年其在全球超过 157 个国家和地区的 1.23 亿个监测终端, 平均每天报告拦截的网络攻击次数达 1.42 亿次。

在网络攻击中, APT(高级可持续威胁, Advanced Persisten Threat)攻击是指对特定目标进行长期持续性网络攻击的一种攻击方式。APT 攻击的全链路[3-4]如图 1 所示。

由于 APT 攻击过程中的隐蔽性需要, 无文件恶意代码[5]越来越多地被应用在了 APT 攻击链路中。具有代表性的有, 2012 年最早出现的无文件恶意代码 Lurk[6], 2015 年的 Duqu2.0[7], 2017 年的 poshspy[8], 2020 年 Donoff 系列的 TrojanDownloader[9]。在无文件恶意代码的攻击过程中, 存在于 Windows 操作系

**通讯作者:** 刘宝旭, 博士, 研究员, Email:liubaoxu@iie.ac.cn。
本课题得到国家自然科学基金项目(No.61902396), 中国科学院战略性先导科技专项项目(No.XDC02040100), 中国科学院网络测评技术重点实验室和网络安全防护技术北京市重点实验室资助。
收稿日期: 2020-09-29; 修改日期: 2020-11-16; 定稿日期: 2020-11-26

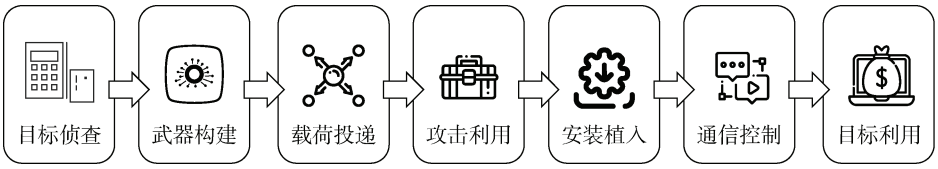


**图 1 APT 攻击链路**

**Figure 1 APT attack link**

统中的 Powershell 是攻击者常使用的攻击方式。据安全公司 Macfee 2020 年 7 月发布的网络安全威胁报告[9]数据显示, 与上一季度相比, 2020 年第一季度的新 Powershell 恶意软件数量增长了 689%。Powershell 是一种命令行 shell 程序, 微软默认将其安装于 Windows 系统中。由于 Powershell 可以直接调用.NET 和 Windows API, 并且允许使用者可以从其他系统登录下载代码并远程执行的特性, Powershell 被广泛应用于配置管理和任务自动化, 但其便捷性使其成为了大量开源渗透工具的利用对象。通过利用 Powershell 下载 shellcode、内存运行、修改注册表以及 WMI(Windows 管理规范, Windows Management Instrumentation)等方式, 攻击者可以实现无本地文件驻留并持久化, 从而达到攻击利用的目的。以 GorgonAPT 为例, 攻击者诱导受害者点击 Word 文件, 通过宏代码解密执行 Powershell 下载, 随后执行 vbs 脚本, 再通过新释放的 Powershell 脚本执行 payload 加载 njrat 远控木马。在该过程中, 通过宏执行解密的 Powershell 代码如下:

"$T='*EX'.replace('*','I');sal M $T;'(&(GCM'+'*W-O*)'+'Net.'+'Web'+'Cli'+'ent)'+
'.Dow'+'nl'+'oad'+'Stri'+'ng("http://19*.***.***.***/documento/cdt.jpg")'|M|M"。

该段 Powershell 代码的作用是从远程服务器上下载 cdt.jpg 文件到本地解析并执行, 从而为后续攻击行为做铺垫。

为了在 Windows 系统上防御以 Powershell 为代表的脚本攻击, Microsoft 发布了用于检测恶意代码的扫描接口(Antimalware Scan Interface, AMSI)[10], 该接口通过检查脚本引擎执行的代码为 Windows 系统提供防御, 但 AMSI 本身未提供 Powershell 恶意代码的检测方案。在传统的恶意代码检测技术中, 根据指纹、特征库、黑白名单等的检测方式难以达到高准确率、高召回率、低误报率的效果。这也使得可以从大量数据中自主学习的机器学习技术被越来越多地应用在了恶意代码检测领域, 但目前在 Powershell 恶意代码检测领域仍应用较少。在恶意代码检测问题中, 安全研究人员常常依据手动抽取的特征建立原始特征数据集, 利用贝叶斯、支持向量机、决策树等算法训练模型, 以此用于恶意代码分类识别[11-12]。在上述方法的研究实验中, 依赖人工的特征选取是使用机器学习进行恶意代码检测方法中普遍存在的问题, 在特征较多的情况下, 研究人员很难根据经验决定哪种特征是对于恶意代码检测更有效的特征, 而目前对于优化原始特征表达的方式主要以降维、压缩和特征选择为主, 多存在信息损耗和丢失的问题, 此时便需要一种有效方法对原始数据进行更有效的特征表达。

本文为解决上述 Powershell 恶意代码检测中存在的特征选取困难和检测效果差的问题, 提出一种基于随机森林特征组合和深度学习的 Powershell 恶意代码检测方法, 称为多特征组合 Powershell 恶意代码检测方法(Features Combination Powershell Malware Detection System, FC-PSDS)。具体的, FC-PSDS 采用集成学习思想, 利用随机森林对原始特征进行非线性变化和特征组合得到更好表征原始数据的新特征组合, 再利用深度学习算法进行检测识别。本文的三个主要贡献如下:

1) 提出一种 Powershell 恶意代码检测方法 FC-PSDS, 该方法首先利用随机森林, 通过集成多个决策树来自动化生成对于原始数据更好特征表达的组合, 再应用深度神经网络学习特征并分类, 实现了在测试数据中 99%以上的检测准确率。

2) 提出一种基于随机森林特征组合的重表达原始数据的方法, 验证了随机森林在特征组合表达上的有效性。该方法根据输入特征的不同维度, 使用随机森林将其组合生成更能表征原始数据的新特征组合, 而非传统的简单线性选择方式, 再将其传入深度神经网络中训练, 实现了对原始数据的更优表达。

3) 实现了基于 Powershell 恶意代码检测方法 FC-PSDS 的检测系统, 实验结果表明该系统拥有高召回、低误报, 时间性能良好的特点, 能够用于 Powershell 恶意代码检测。

本文余下内容, 第 2 节介绍了 Powershell 恶意代码检测研究的现状, 第 3 节介绍了 FC-PSDS 的方法

框架, 第 4 节介绍了 FC-PSDS 在真实数据中的检测效果和与其他方法的对比, 第 5 节为讨论, 第 6 节为总结。

## 2 相关工作

目前, 以 Powershell 恶意代码为代表的无文件恶意代码检测领域, 具有代表性的科研工作总结如表 1 所示。

Powershell 恶意代码的研究可以根据检测方式主要分为利用自然语言处理技术的检测识别方法、基于异常调用的异常检测方法、基于特征识别的检测识别方法。

**表 1 Powershell 检测相关技术研究**

**Table 1 Research on Powershell malware detection**

| 研究方向 | 论文题目 | 研究团队 | 研究内容 | 存在问题 |
|---|---|---|---|---|
| 基于文本识别的检测技术 | Detecting Malicious Powershell Commands using Deep Neural Networks | Ben-Gurion University of the Negev,Microsoft | 提取代码的起始固定长度使用 CNN 网络检测 | 仅使用固定头部信息，导致模型检测泛化效果差 |
| | Detecting Malicious Powershell Scripts Using Contextual Embeddings | Ben-Gurion University of the Negev,Microsoft | 使用 CNN、LSTM 网络对经词嵌入后特征向量进行识别 | 仅使用词空间特征使得模型的召回率在测试数据上较低 |
| 基于异常调用的检测技术 | 一种无实体文件恶意代码的检测方法及系统 | 安天 | 通过遍历系统内的进程和对应路径形成记录，并进行深度检测 | 仅支持程序运行时的动态检测，黑名单扫描容易被绕过 |
| | 一种检测 Powershell 恶意代码的方法及系统 | 安天 | 通过将 Powershell 代码执行的行为与明文特征库进行匹配并检测 | |
| 基于特征分类的检测技术 | AST-Based Deep Learning for Detecting Malicious Powershell | StandFord/MIT | 使用抽象语法树深度和节点数结合随机森林检测 Powershell 脚本 | 训练模型少，泛化能力差 |
| | Powershell-based Malware Detection Method Using Command Execution Monitoring and Deep Learning | 高丽大学 | 使用 CNN 从 Powershell 代码的命令中提取特征，通过 RNN 进行检测 | 使用深度学习网络直接提取特征导致可解释性差，泛化效果差 |

基于自然语言处理技术的检测方法主要利用基于自然语言处理的文本识别方法对 Powershell 恶意代码进行分类检测。自然语言处理技术已经被大量的应用到了情感分析[13-14]、恶意代码检测问题[15-17], 但使用自然语言处理技术检测 Powershell 恶意代码仍存在较大的研究空间。文献[18]对 Powershell 代码的命令语法构建了基于前缀树的词干分析器, 将词干内容向量化后, 使用支持向量机、深度神经网络等监督学习分类器进行 Powershell 恶意代码分类。文献[19]将 Powershell 命令作为文本进行头部固定长度的 one-hot 编码处理, 然后将其传入卷积神经网络进行特征学习和分类, 该方法在 66388 个样本的数据集中精确率达到 95%以上, 该数据集拥有 6290 个恶意样本和 60098 个良性样本。文献[20]将 Powershell 恶意代码进行文本处理, 采用自然语言处理技术, 分别将经 one-hot 编码的字符级数据和使用 FastText 预训练的令牌级表示作为输入, 使用卷积神经网络和长短期记忆网络进行特征学习和分类, 该方法在测试集中的召回为 89.4%。基于深度学习自然语言处理方式的检测方法在目前的研究中大多通过分词的方式将处理后的数据直接传入神经网络, 该种方式仅采用 Powershell 恶意代码本身的程序语言特征, 容易在训练过程中产生过拟合的现象, 从而影响测试的召回和泛化效果。

在基于异常调用的检测方法上, 文献[21]利用系统日志、系统计划任务列表、启动项的调用信息, 通过检测是否包含敏感调用来判断是否存在 Powershell 攻击。文献[22-24]提出了检测 Powershell 恶意代码的方法, 一种方式为在 Powershell 代码执行时进行调用函数和进程的监听, 检测隐藏的 Powershell 窗口行为、代码加密行为, 将其与预设特征库进行匹配进行恶意性的识别; 另一种检测方式为, 在执行中检测 Powershell 的进程链, 将其调用关系与预设的黑白名单做比对, 从而实现对于 Powershell 代码恶意性的决策。此种检测方法针对执行中的 Powershell 恶意代码, 优点是可不考虑基于字符串等基础混淆方式的影响, 但基于执行中特征码扫描的方式存在容易被绕过和无法实现离线检测的问题。

在基于特征分类的检测方法上, 机器学习和深度学习技术被广泛应用在恶意代码检测领域[25-27]。在 Powershell 恶意代码检测中, 文献[28]利用机器学习方法对 4100 个 Powershell 恶意代码家族分类, 研究人员利用 Powershell 代码的抽象语

法树的深度和节点个数特征，使用机器学习中的随机森林算法，抽取特征学习训练分类检测模型，获得 85%的分类精确度。文献[29]利用决策树、随机森林等机器学习方法，对经专家系统提取的特征通过机器学习分类器进行识别。文献[30]通过使用卷积神经网络从 Powershell 代码中提取命令的特征，并将经其提取的特征传递给递归神经网络做分类识别，在实验数据集上达到召回率 97%，误报率 1%的效果。基于纯特征检测方法的检测效果受特征提取的有效性影响较大，且不同特征对于分类结果的权重不同，对于不同特征的有效性缺少评价的体系，手动提取特征的冗余性和有效性直接决定了模型的效果。

与上述现有的工作成果不同，相比于纯动态检测的高耗时、无法离线检测，纯自然语言处理、特征分类的易过拟合和纯手动提取特征耗时、繁琐且存在冗余的问题。本文提出的 FC-PSDS 方法结合随机森林和深度神经网络，通过随机森林特征组合生成对原始数据更优的特征表达，并通过深层神经网络抽象并自动学习经特征组合生成的新特征，从而实现对 Powershell 恶意代码的有效识别，同时该方法可用于离线检测。

## 3 模型设计

FC-PSDS 方法设计的目标是使用随机森林和深度神经网络检测恶意 Powershell 代码。具体的，通过混淆去除和标准化从 Powershell 代码中抽取特征组成原始特征集，根据集成学习思想使用原始特征集生成对原始数据表达更有效的新特征组合，并据此训练深度神经网络分类器模型，以此解决传统方法对于 Powershell 恶意代码的检测困难，以及冗余特征导致的模型检测能力较差的问题。本文提出的 FC-PSDS 总体框架图如图 2 所示。其中，FC-PSDS 的检测过程如下所示:

1) **原始数据处理。**FC-PSDS 首先对数据进行处理，通过混淆检测框架和混淆去除对数据进行标准化操作，然后对标准化后的 Powershell 代码抽取初始特征，再对数据进行切分。

2) **特征组合。**由于手动抽取的初始特征均为离散型，且不同特征的量级存在差异，因此为保证分类模型可以有更好的拟合和泛化效果，FC-PSDS 利用集成学习的思想，使用多决策树分类器组合的随机森林对初始特征进行特征组合生成可以更好表达原始数据的新特征。

3) **模型训练与分类识别。**FC-PSDS 将经过特征组合后生成的新特征数据作为模型的输入，并使用深度神经网络作为攻击检测的模型。再通过预设的效果评估指标衡量 FC-PSDS 的检测效果。

### 3.1 原始数据处理

FC-PSDS 的数据处理部分包括对原始 Powershell 代码的去混淆和标准化，对原始特征的提取和数据切分。

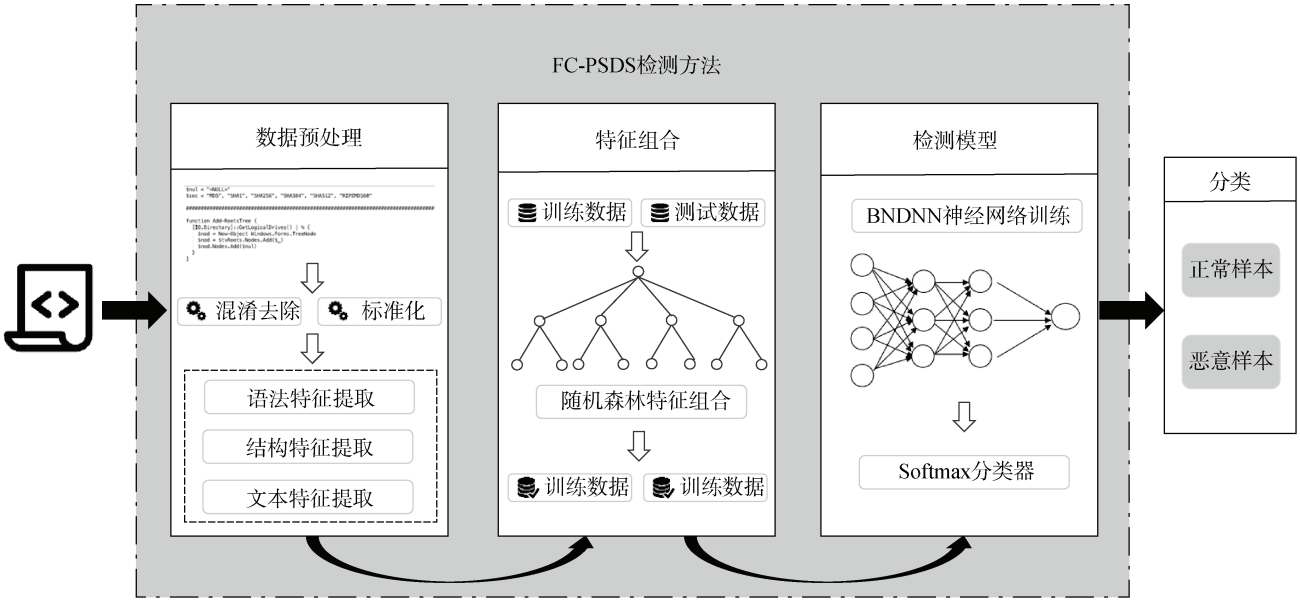


**图 2 FC-PSDS 检测方法框架示意**

**Figure 2 Proposed FC-PSDS framework structure**

### 3.1.1 去混淆和标准化

**混淆去除。**本文方法采用的数据原始格式为文本格式, 由于 Powershell 代码中存在大量的混淆, 需要在特征提取前对样本进行混淆去除, 否则会对提取特征的有效性产生干扰。Powershell 代码中常见的混淆类型包括字符操作、编码、加密和压缩。以获取并终止 Notepad 进程的 Powershell 代码为例, 经过混淆后的部分效果如表 2 所示。

对于本文使用的数据, 首先使用混淆检测框架 Revoke-Ofuscation[31]对样本进行混淆检测, 该框架可以对 Powershell 代码进行混淆识别, 包括基于字符串、编码、加密、压缩的混淆等。对于经 Revoke-Ofuscation 检测存在混淆的样本, 根据论文[29]的方法, 通过执行该代码并记录其日志调用记录、基于正则的静态混淆去除等方式来实现对于混淆的去除。具体的, 该方法通过动静结合的混淆去除方式, 先执行该 Powershell 代码, 收集其运行的脚本日志信息, 该方式可以绕过除字符类编码之外的混淆, 若检测到字符类编码的混淆, 则将执行后日志中收集到的中间码, 利用正则表达式去混淆。若为非字符类混淆的其他方式混淆的代码, 在经动态执行后, 首先利用正则表达式对于获得脚本日志进行重排、换行、字符切割等方式的混淆去除, 对于该步混淆去除后的得到的内容, 若检测存在 invoke-Expression 命令, 则去除该命令后再执行, 以此获得执行后的结果。通过上述方式便可以获得混淆去除后的代码, 该方式优势在于, 这种动静结合的方式可以解决部分混淆难以通过正则等静态方式去除的问题, 从而提高了去混淆的效果。

**标准化。**由于 Powershell 允许使用者使用参数补全和命令别名, 用户在 Powershell 中设置的参数具有随机性, 这会导致从代码文本的角度提取特征存在困难, 在提取 Powershell 代码的语法树结构前, 需要将参数标准化, 以降低提取难度, 提高检测召回效率。本文采用论文[29]的标准化方法: 面向 Powershell 的 Cmdlet 参数对 Powershell 构建每条命令的参数前缀树(Trie)。构建的前缀树中的节点内容为参数的所包含字符, 同时为记录当前字符是否为命令的最后一个字符, 在节点中设置了布尔值用于表示当前位置。使用字典结构, key 为当前所在位置字符, value 为子节点。在进行对样本数据进行特征提取时, 对 Powershell 的抽象语法树使用 Findall 函数抽取代码中的命令和参数, 根据创建好的命令参数前缀树, 在前缀树中搜索以当前参数为前缀的标准参数, 以此达到 Powershell 代码标准化的目的。

**表 2 Powershell 代码混淆示意**

**Table 2 Powershell code obfuscation**

| 类型 | 混淆效果 |
|---|---|
| / | Get-Process -Name notepad \| foreach-object{$_.Kill()} |
| 编码 | &((gET-varIable '*mDR*').naMe[3,11,2]-jOiN") (-JoIN((47 ,65,74 , '2d', 50 ,72 , '6f' , 63,65 ,73 , 73, 20 , '2d', '4e' , 61 ,'6d', 65 , 20, '6e' , '6f',74 , 65 ,70 , 61 ,64 ,20 ,'7c' ,20, 66,'6f',72, 65, 61 ,63,68, '2d', '6f',62 , '6a',65,63 , 74, '7b', 24, '5f' , '2e','4b',69 , '6c' , '6c' , 28 , 29, '7d' ) \|% { ([ChaR]([conVert]::toiNT16( ($_.TosTRinG() ) ,16 ) ))} ) ) |
| 加密 | ( [RuNtiME.IntErOpserVICES.mARshaL]::([RuntIme.iNTERopSErVICes.mArShal].GeTMeMBeRs()[5].NAmE).InVOke([RuNtIMe.INtErOPsErViCeS.marshAl]::SECUrEStrIngtOBStr($('76492d1116743f0423414AEEAN.....lADkAMBjADgEAYwA='\| cOn-verTTo-seCUrEsTRiNG -ke 88,114,126,40,139,174,228,132,80,129,46,192,37,148,19,153,67,91,37,20,207,127,28,187,102,13,193,206,83,96,221,54) ) ) ) \| .( $EnV:cOMSpEC[4,26,25]-JOiN") |
| 压缩 | (NEw-oBJeCT iO.COmPreSSioN.dEFLAtEstReam( [io.MeMOrYstReAM] [COnvErt]::frOmbaSe64StRInG( 'c08t0Q0oyk9OLS5W0PVLzE1VyMsvSS1ITFGoUUjLL0pNTM7QzU/KSk0uqVaJ1/POzMnR0KwFAA==' ) , [SYS-teM.iO.comPRessiOn.ComprESSionmODe]::deCOmPResS )\| ForeacH-obJEcT{NEw-oBJeCT SYStem.io.stReAMREADer( $_ , [Sys-tem.TExT.encoDing]::AsCii ) } \|ForEaCH-ObjeCT{$_.ReADToenD()}) \|& ( $sHELLiD[1]+$sHeLliD[13]+'X') |
| 字符连接 | (' ('+'N'+'Ew-oBJe'+'CT i'+'O.C'+'OmPreSS'+'ioN.'+'dEFL'+'At'+'Es'+'tR'+'e'+'am'+'( [io.'+'MeMOrYst'+'Re'+'AM] [COn-vEr'+'t]::fr'+'O'+'mb'+'aSe64S'+'t'+'RInG( mR'+'sc'+'08t0Q0oy'+'k9'+'OLS5'+'W0'+'PVL'+'zE1'+'VyMsv'+'S'+'S'+'1'+'IT'+'FG'+'o'+'UUj'+'LL'+'0pN'+'TM7Q'+'z'+'U/KS'+'k0u'+'qV'+'a'+'J1'+'/P'+'O'+'zMn'+'R0KwFAA'+'==m'+'Rs ) , '+'[SY'+'SteM.i'+'O'+'.co'+'mPRes'+'siOn.'+'Co'+'mp'+'rES'+'S'+'io'+'nmODe'+']:'+':'+'deCOm'+'PResS )'+'jHO Fore'+'a'+'cH'+'-objEc'+'T{NEw-'+'oBJeC'+'T SY'+'Stem'+'.io'+'.st'+'ReA'+'MR'+'EADe'+'r'+'('+' kYc_'+' , '+'[Sys'+'te'+'m'+'.'+'TExT'+'.e'+'ncoDing]::A'+'sCii'+' ) '+'}'+' jHO'+'Fo'+'rEaCH-ObjeC'+'T{k'+'Yc_'+'.Re'+'AD'+'ToenD()}) jHO&'+' ( kYcsHELL'+'iD['+'1]+kYcsHe'+'LliD['+'13]+mRsXmRs'+')').RePLAce(([cHAr]106+[cHAr]72+[cHAr]79),[sTRInG][cHAr]124).RePLAce(([cHAr]109+[cHAr]82+[cHAr]115),[sTRInG][cHAr]39).RePLAce('kYc',[sTRInG][cHAr]36) \|.( $EnV:CoMsPeC[4,26,25]-jOin") |

### 3.1.2 原始特征提取

在经过对原始数据的混淆去除和标准化后, 对处理后的样本进行原始特征的提取, 本文结合现有的 Powershell 恶意代码中常见的提取的特征基础上[29], 在 Powershell 5.0 版本的背景下, 从 Powershell 语法树节点数、语法树宽度、语法树深度, Powershell 语句长度、语句个数、变量数量、Powershell cmdlet 命令、

cmdlet 参数等 8 个维度提取共计 1657 个离散型特征。对原始文本格式的 Powershell 恶意代码样本, 经特征提取后映射为 1657 维的原始特征向量。表 3 所示为 1 个经特征提取后的 Powershell 样本的部分特征向量。

**表 3　经特征提取后的 Powershell 样本**

**Table 3　Powershell sample after feature extraction**

| 特征名 | 特征值 | 特证名 | 特征值 |
|---|---|---|---|
| get-process | 1 | AST_Node | 144 |
| add-member | 4 | AST_Depth | 15 |
| add-type | 0 | AST_Width | 37 |
| commend | 0 | Variable_Count | 31 |
| compress | 0 | Statement_Length | 683 |
| … | … | Statement_Count | 47 |

### 3.1.3　交叉验证

为了使 FC-PSDS 方法具有更好的泛化能力和更有效的评估方法效果, 本文对 FC-PSDS 方法进行了交叉验证, FC-PSDS 对数据采用 *k*-折交叉验证(*k*-fold cross validation), 以此构造实验使用的数据集。*k*-折交叉验证在实验过程中将原始数据分为 *k* 组, 在训练的过程中每个子集分别成为一次验证集, 其余 *k*-1 组子集作为模型训练集。

本文 FC-PSDS 方法采用的方法为 10-折交叉验证, 该方法将数据集均分为 10 份, 即 9 份用作训练, 1 份用作测试, 训练时根据测试的结果优化下一轮的迭代。

## 3.2　基于随机森林的特征组合

对于经过预处理得到的原始特征数据, 由于经人工提取的不同特征对模型的分类效果影响不同, 维度较高的原始特征数据若直接传入分类模型中会导致其存在大量无关和冗余特征。直接使用其训练分类器模型会导致检测效果、泛化能力差。为了使分类模型可以更好更快收敛, 此时便需要对原始的经特征工程提取的特征数据进行更优化的表达。

**常用降维方法。**当前研究中应用较多的方式比如通过主成分分析(Principal Component Analysis, PCA)、自编码器(Auto-Encoder, AE)等对高维数据进行降维, 以自编码器为例, 基于自编码器的降维方式属于基于深度学习的降维方法, 可以从原始数据中通过压缩和还原的操作, 从中间层得到经非线性计算的压缩后的数据。自编码器包含编码器(Encoder)和解码器(Decoder), 若用 $\beta$ 和 $\gamma$ 表示对应的映射, 则自编码器降维的原理如公式(1)所示。

$$\begin{aligned}&\beta: X \to h\\&\gamma:\ h \to X'\end{aligned} \tag{1}$$

除该种方法外, 对原始特征数据的表示优化还可以通过特征选择和特征组合的方式进行。

**特征选择方法。**特征选择方法的思想为在原始的所有 *M* 个特征的特征集合中, 选取由 *N* 个特征组成的子集, 从而使得该子集在评价指标中的效果最优[32]。该方式通过选取最优特征子集的方式可以减少原始特征数据中冗余特征对于分类效果的影响, 从而减少训练所需的时间。但不论是降维还是特征选择的方式, 对原始数据的压缩和特征减少会对原始数据的表示存在损耗。

**特征组合方法。**对原始特征数据进行重表示的另一种方式是特征组合, 特征组合是广告系统中进行点击率预估的常用方法[33], 在广告系统的点击率预估(Click-Through-Rate, CTR)问题中, 由于预估模型的输入样本量非常大, 为了能够使模型更快速的训练, 在输入模型前需要对数据进行更有效的特征工程, 好的特征工程不但可以减少训练的时间也可以提高后续检测模型的分类效果。CTR 预估问题中常用到的梯度提升树(Gradient Boosting Decison Tree, GBDT)便是用于特征组合的常用方法, 除此之外随机森林、XGBoost 也可用于特征组合。使用梯度提升树或者随机森林进行特征组合的原理为, 当利用其构建新的训练数据时, 树的每个弱分类器有且只有一个叶子节点可以输出分类预测的结果, 对于一个有 *f* 个叶子节点的决策树, 便可以将一个进入决策树的数据变换为一个 *f* 维向量的新数据, 也就是原始特征经决策树处理后的新特征组合数据。当有 *k* 棵决策树构成随机森林时, 输入的原始数据便可以变换成 $k \times f$ 维度的新特征组合。通过上述集成学习思想的特征组合方式, 随机森林特征组合的方法能够得到对于原始数据的最优表示, 并可以增加检测模型的鲁棒性和有效性。相比于传统的特征选择方式, 本文采用基于随机森林的特征组合可以得到更有助于分类器逼近全局最优解的特征组合。

**随机森林特征组合。**本文采用了多决策树集成的随机森林作为特征组合的方法。集成学习是一种有监督的机器学习技术, 通过组合多个监督模型得到一个更全面的强监督模型[34], 以此解决单分类器性能不稳定、易过拟合的问题。随机森林算法[35]是一种基于决策树的集成学习方法, 其由多个决策树组成, 决策树是一种用于解决分类问题的二叉树, 树中的每一个节点表示一个数据集和对这个数据集的一次决策, 树的每条边表示一种选择, 随机森林的结构如图 3 所示。通过决策树可以将原本的 *N* 维特征转换为节点数量 *M* 维的新特征。在从 *N* 维扩增到 *M* 维新特征组合的建立过程中, 由于即使一个叶子节点只有一条数据, 从以一个根节点到该叶子节

点的路径也会被决策树认为构成一种决策方式, 该种决策方式会导致决策树在分类过程中易发生过拟合, 从而影响后续训练分类模型的精度。通过集成多个决策树形成随机森林, 根据投票的结果作为分类结果, 就可以很好的避免过拟合的发生。

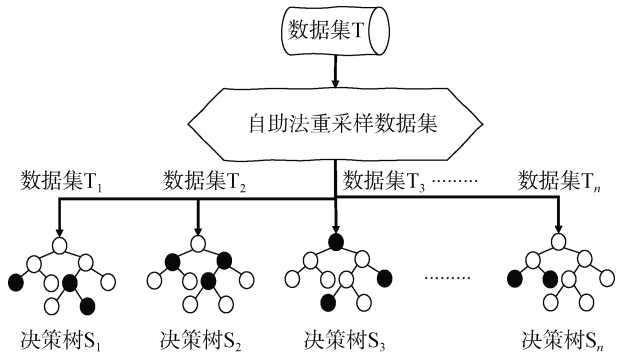


**图 3　随机森林结构**

**Figure 3　Random forest structure**

本文提出的基于随机森林的集成学习方式, 使用不同数量的决策树进行投票选取最优特征组合。同时, 这种自动发现有效特征、特征组合的方式也可以弥补对于原始数据特征提取的人工经验不足, 达到缩短实验周期的目的。本文使用的特征组合算法如算法 1 所示。

**算法 1.** 基于集成学习的特征组合算法.

**输入:** 样本的原始特征数据集 $X$, $X$ 包含 $m$ 行 $n$ 列, 其中 $m$ 为数据量, $n$ 为特征数;

**输出:** 组合后的新特征矩阵 $\boldsymbol{X}'$;

1) 使用训练集($X_{train}$,$Y_{train}$)训练好的随机森林 $R$, $R$ 包含 $t$ 棵决策树, 第 $i$ 颗决策树有 $l_i$ 个叶子节点;

2) FOR $i$=1 … $m$ do;

3) FOR $j$=1 … $t$ do;

4) 用第 $j$ 棵决策树预测第 $i$ 条数据, 得到数据对应的决策树的叶子节点编号为 $k_{i,j}$;

5) 使用 one_hot 编码将 $k_{i,j}$ 编码成向量 $\boldsymbol{v}_{i,j}$;

6) END FOR;

7) 将 $v_{i,1},v_{i,2},\cdots,v_{i,t}$ 拼接成一个新的 $\sum_{j=1}^{t} l_j$ 维的向量 $x_i'=(v_{i,1},v_{i,2},\cdots,v_{i,t})$;

8)END FOR

9)将 $x_1',x_2',\cdots,x_m'$ 拼接成一个新的 $\sum_{i=1}^{m} l_j$ 列的矩阵 $X'$。

## 3.3　基于神经网络的模型训练和分类识别

### 3.3.1　基于深度神经网络的模型训练方法

为了更好的提高模型检测和泛化效果, 本文提出的 FC-PSDS 利用神经网络对重新组合后的特征进行检测识别。目前在恶意代码检测领域中常用到的深度学习技术有卷积神经网络(Convolutional Neural Networks, CNN)、循环神经网络(Recurrent Neural Network, RNN)、长短期记忆网络(Long Short-Term Memory, LSTM)等, 但每种神经网络都有自己更适用的领域。本文提出的 FC-PSDS 在经随机森林训练得到组合的新特征 $X'$后, 采用深层神经网络(Deep Nerual Network, DNN)进行入侵检测。DNN 属于第三代人工神经网络, 在结构上拥有输入层、中间隐藏层、输出层, 其结构如图 4 所示。DNN 对输入数据没有要求, 相比选用 RNN、CNN 作为判别网络, 可以较好地从输入数据中检测恶意性。

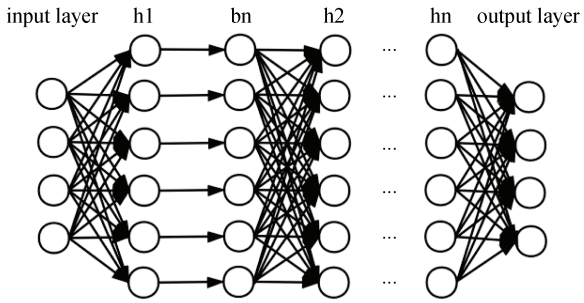


**图 4　采用的 DNN 网络结构**

**Figure 4　DNN structure adopted**

**FC-PSDS 采用方法。**FC-PSDS 在使用 DNN 对其进行特征学习和分类时, 使用 He_Normal 对神经网络权值进行初始化, 在隐藏层使用规范化层 BatchNormalization 和 Mini-batch, 使用 L2 正则避免过拟合, 在训练过程中使用学习率衰减和提前结束缩短训练时间。

FC-PSDS 使用 He_Normal 进行网络权值初始化。传统神经网络权值的初始化方式包括全 0 初始化、全 1 初始化、随机初始化等, 其中权值初始化为 0 或 1 会使神经网络学习能力有限, 随机初始化则会在随着神经网络的隐藏层增加后出现梯度消失问题。He_Normal 初始化的基本思想是在网络正向传播时, 状态值的方差保持不变; 网络反向传播时, 激活值梯度的方差不发生改变。He_Normal 初始化对于 ReLU 和 LeakyReLU 不同, 对于 ReLU 的方法如公式(2)所示:

$$W=N[0,\sqrt{\frac{2}{n_i}}],b=0 \tag{2}$$

对于 LeakyReLU 的方法如公式(3)所示:

$$W=N[0,\sqrt{\frac{2}{(1+a^2)n_i}}] \tag{3}$$

为避免梯度消失问题和增加训练速度, FC-PSDS 在 DNN 的隐藏层激活函数前添加批量规范化层 BatchNormalization。添加后的 DNN 网络在每次使用 Mini_batch 进行训练时, 对网络中隐藏层的规范化操作会使得网络每层输入的方差为 1, 均值为 0。

同时, FC-PSDS 模型使用了学习率衰减和提前结束, 训练过程中的学习率由初始学习率和学习率衰减系数决定, 在训练初始时, 为快速达到最优解附近, 设置较大的学习率。随后, 为避免因较大的学习率导致模型在训练的过程中震荡, 将学习率逐步降低。学习率指数衰减更新原理如公式(4)所示。其中, $l_r$ 为学习率, $l_{r0}$ 为初始化学习率, $d_r$ 学习率衰减系数, $g_s$ 为当前已迭代轮数, $d_s$ 为衰减步长、即多少轮衰减一次学习率。当模型的训练损失在一定轮数内不变时, 便及早停止训练, 减少训练损耗。

$$l_r = l_{r0}\, d_r^{\left\lfloor \frac{g_s}{d_s} \right\rfloor} \tag{4}$$

### 3.3.2 检测分类

为根据模型输出的数据进行预测分类合评估, FC-PSDS 方法采用了 Adam 优化器, 使用 Sigmoid 激活函数来对输出的结果做分类预测, Sigmoid 函数的计算公式如公式(5)所示, $t$ 为输入, $\delta(t)$为对应输出的概率值。当预测的概率小于 0.5 时为正常 Powershell 样本, 反之为恶意 Powershell 样本。

$$\delta(t)=\frac{1}{1+e^{-t}} \tag{5}$$

FC-PSDS 的损失值计算选用交叉熵 (cross-entropy loss), 原理如公式(6)所示, $x_i$ 为第 $i$ 个样本, $n$ 为分类的类别数目, $q(x_i)$代表第 $i$ 个样本的模型预测值, $p(x_i)$代表第 $i$ 个样本的真实值, $J(p,q)$代表交叉熵的计算结果值。

$$J(p,q)=-\sum_{i=1}^{n} p(x_i)\log(q(x_i)) \tag{6}$$

## 4 效果评估

为了验证本文所提出的 FC-PSDS 方法在 Powershell 恶意代码检测过程中的实际效果, 本文在收集的 Powershell 恶意样本和正常样本数据集上进行了实验, 分析了本文提出方法的效果, 并根据分类准确率、召回率、假阳率等多个指标评估了 FC-PSDS 在实际应用中的性能。

在实验设计上, 本文首先验证了基于随机森林的特征组合方法相比于使用原始手动提取的特征对于后续检测模型效果提升的有效性。同时, 本文使用 FC-PSDS 与常见机器学习方法、深度学习方法和当前研究中的代表性 Powershell 恶意代码检测模型的效果进行了横向的对比, 验证了基于随机森林的特征组合和深度学习的 Powershell 恶意代码检测方法的有效性。

### 4.1 实验环境和实验数据

本文方法的实验环境为 Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz, 内存 500G, 采用 Python 语言、Keras、Sklearn、imblearn 等对本文方法进行实现及检测测试。

为评估本文方法的普适性, 本文实验所用的数据集由 Powershell 正常代码、Powershell 恶意代码组成, 共计 12610 个样本。其中正常样本由 Github[36], Powershell Gallery[37], Daniel 等[19]的公开数据组成, 共计 8521 个样本; 恶意样本由 Github、Jeff White 等[38]的公开恶意样本组成, 样本包括包括 APT 攻击过程中常见的下载器类、嵌入型载荷类、本地持久化类等, 共计 4089 个样本。数据集中的样本分布如图 5 所示。

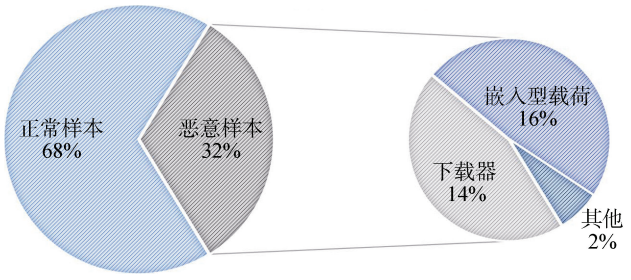


**图 5 数据中的样本分布示意**

**Figure 5 Sample distribution in dataset**

### 4.2 实验参数

本文首先使用随机森林算法对数据进行特征的重新组合, 为了能够使经随机森林计算得到的新特征可以最好的表征原始数据, 使神经网络模型可以更好对数据进行拟合和分类, 本文在测试随机森林、深度神经网络的结构时使用了网格搜索进行参数优化, 对不同的结构进行了测试实验, 以此获得一个最佳的检测模型。经实验后, 本文采用的最佳结构如下。随机森林的结构为决策树数量为 99, 随机森林最大深度为 16, 叶子节点最少数据数量为 10,

随机森林在该结构时对于原始数据的重表达效果最好。深度神经网络输入层大小为 128, 隐藏层大小为 64, batch_size 为 256。

### 4.3 评估指标

本文提出的 FC-PSDS 方法是分类模型, 为评估本文所采用检测方法的效果, 效果评估采用分类算法常用的用于度量有效性的指标, 包括如下。

1) 真阳性(True Positive, TP), 在分类结果中被预测为恶意样本, 实际为恶意样本的样本数量。

2) 假阳性(False Positive, FP), 在分类结果中被预测为恶意样本, 实际为正常样本的样本数量。

3) 真阴性(True Negative, TN), 在分类结果中被预测为正常样本, 实际为正常样本的样本数量。

4) 假阴性(False Negative, FN), 在分类结果中被预测为正常样本, 实际为恶意样本的样本数量。

在使用上述数据根据分类结果得到的混淆矩阵中, 采用准确率, 假阳率, 召回率三个指标来评价 FC-PSDS 方法, 同时对比其他方法的分类性能。具体如下。

1) 准确率(Accuracy, ACC): 分类结果中, 被正确分类的样本数占总样本数的百分比, 是用于评估恶意代码检测方法整体性能的评价指标。

$$ACC=\frac{TP+FN}{TP+TN+FP+FN} \tag{7}$$

2) 精确率(Precision, PRE): 分类结果中, 被正确分类为恶意样本的恶意样本数量占被分类为恶意样本总数的占比。

$$PRE=\frac{TP}{TP+FP} \tag{8}$$

3) 假阳率(False Positive Rate, FPR): 分类结果中, 被错误分类为恶意样本的正常样本占正常样本总数的比例。

$$FPR=\frac{FP}{FP+TN} \tag{9}$$

4) 召回率(True Positive Rate, TPR): 分类结果中, 被正确分类为恶意样本的恶意样本数占恶意样本总数的比例。

$$TPR=\frac{TP}{TP+FN} \tag{10}$$

### 4.4 实验结果

现阶段的 Powershell 恶意代码检测领域, 已有利用决策树、随机森林等一般机器学习方法和 CNN、LSTM 等深度神经网络技术的案例。本文提出的 FC-PSDS 方法没有采用单一检测方法, 而是利用随机森林和深度学习组合的方式进行分类识别。为了对 FC-PSDS 方法的效果进行评估, 以及证明随机森林和深度神经网络进行模型组合的效果, 本文首先利用实验数据对随机森林特征组合的效果进行了评估, 以此验证基于随机森林的特征组合方法对于原始数据可以进行更好的表达, 从而在传入神经网络后可以实现更优的分类效果。在对 FC-PSDS 整体效果的评估上, 选用了目前已有的 Powershell 恶意代码检测方法, 使用常见机器学习、深度学习技术对实验数据进行了多轮训练来观察评估指标的效果, 以此验证 FC-PSDS 方法的整体分类效果优于当前其他的方法。

#### 4.4.1 特征组合效果分析

本文首先评估了基于随机森林的特征组合对于算法检测准确率的提升效果。本文实验中首先进行了原始特征数据在不经随机森林做特征提取时, 直接使用朴素贝叶斯中的高斯朴素贝叶斯(Gaussian Naive Bayes, GNB)和深度学习中 DNN 的分类效果实验, 与使用随机森林做特征组合后传入相同的 GNB 和 DNN 模型的分类效果进行对比。

经实验表明, 随机森林的特征组合对于 FC-PSDS 方法检测效果的影响如图 6 和表 4 所示, 图表中的 ACC、TPR、FPR、PRE 分别代表 4.3 评估指标中的准确率、召回率、假阳率、精确率, 由于假阳率的值相比其他的值过小, 在图 6 中仅显示 ACC、TPR、PRE, 图表中的 RF 代表经过随机森林特征组合。实验结果显示, 基于随机森林的特征组合方法对于模型检测的效果有正向影响。与未使用随机森林做特征组合的对照实验的结果数据说明, 本文利用特征组合方法获得的新特征组合相比原始特征可以更好的还原对原始数据的表达。原因在于, 在原始数

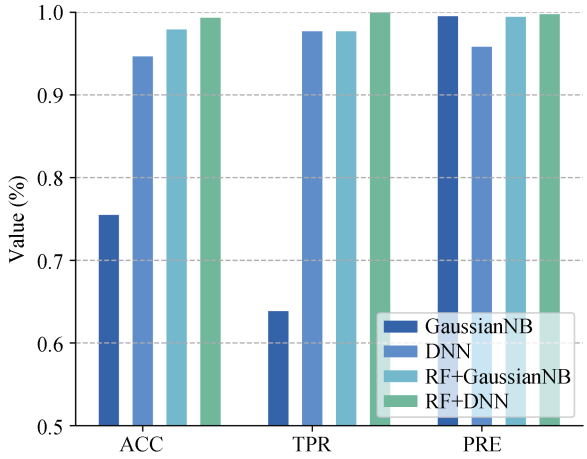


**图 6 特征组合对于分类效果的影响**

**Figure 6 Effect of feature combination on classification**

据集上使用简单机器学习算法难以捕捉到原始数据的有效信息，同时浅层的机器学习算法在数据规模和复杂度较大时，对数据的学习能力有限。因此，通过随机森林特征组合自动从原始数据中发现有效的特征组合，可以弥补原始数据的人工提取特征经验不足问题，同时随机森林的多棵树相比单决策树的表达能力更强，可以更好地生成表征原始数据的特征组合，再通过分类算法进行学习和分类，相较未经处理直接使用分类网络的方式可以更好地拟合原始数据并学习其内在规律。

**表 4　特征组合对于分类的效果影响**

**Table 4　Effect of feature combination on classification**

| IDS | ACC | TPR | FPR | PRE |
|---|---|---|---|---|
| GaussianNB | 0.7548 | 0.6384 | 0.0065 | 0.9951 |
| DNN | 0.9465 | 0.9769 | 0.0864 | 0.9581 |
| RF+GaussianNB | 0.9791 | 0.9769 | 0.0113 | 0.9944 |
| RF+DNN | 0.9934 | 0.9996 | 0.0136 | 0.9976 |

### 4.4.2　FC-PSDS 检测效果分析

**在训练轮次效果分析上，**深度学习模型训练时，一个轮次(epoch)意味着将训练数据集完整训练一次，模型分类效果受训练轮次的次数影响，FC-PSDS 的深度学习模型检测效果随训练轮次的增加效果如图所示。由图 7 所示结果可知，FC-PSDS 的分类准确率随训练的过程逐渐升高，并稳定在 99%。

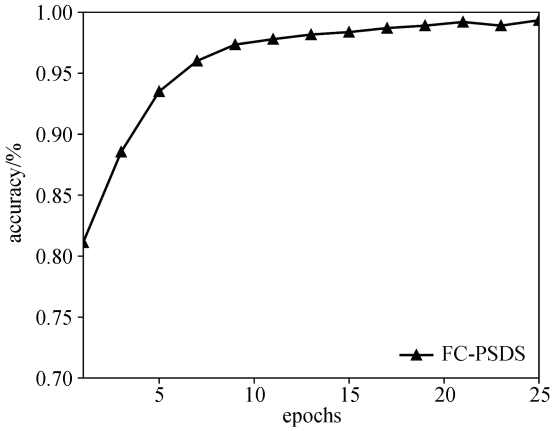


**图 7　训练轮次的效果的影响**

**Figure 7　Effect of train epoch**

**在对比实验效果上，**本文选取了 Rubin 研究方法[20]中提出的 CNN-LSTM 模型、Danny 研究方法[19]中的 CNN 模型以及决策树、朴素贝叶斯等常见机器学习算法和 DNN 神经网络与 FC-PSDS 在同一数据上进行二分类检测效果的对比比较。图 8 和表 5 显示了不同模型在相同训练环境下测试得到的 ACC、FPR、TPR 指标。

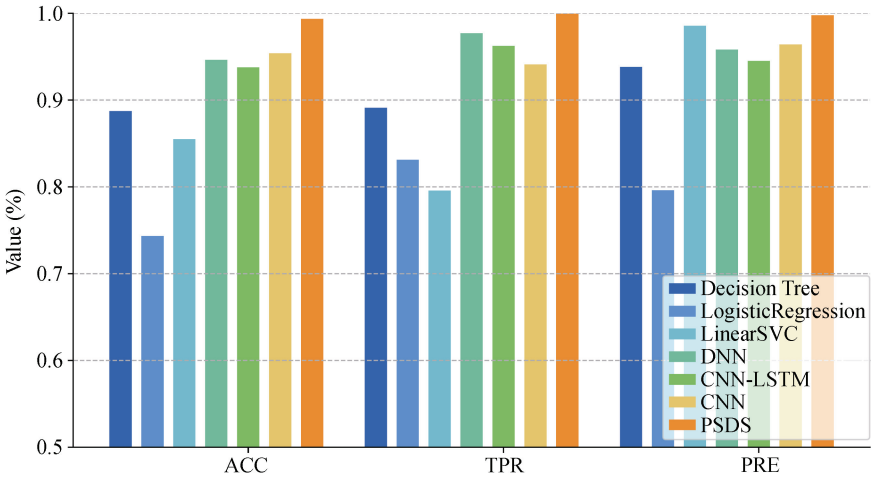


**图 8　FC-PSDS 与不同检测方法的效果对比**

**Figure 8　Comparison of FC-PSDS with different detection methods**

可以看出，FC-PSDS 方法的 ACC、FPR、TPR、PRE 分别达到了 0.9934, 0.0136, 0.9996, 0.9976。这说明本文所用到的批量规范化深度神经网络的方法比传统机器学习、深度学习技术可以取得更好的检测准确率和更低的假阳率，明显优于一般方法。效果更优的原因在于，在本文的实验结果中，由于深度神经网络可以更有效的拟合复杂的原始数据，因此相比 Decision Tree、LogisticRegression、LinearSVC 等机器学习方法，深度学习模型的检测效果更佳。相比 Rubin 等人提出的 CNN-LSTM 模型和 Danny 等人提出的 CNN 模型，本文提出的 FC-PSDS 通过随机森林集成组合的方式，生成了更好表征原始数据的特征

组合, 能够使后续的深度神经网络学习到更好拟合原始数据的信息, 从而检测效果更佳, 模型效果也可以达到实际使用要求

**表 5 FC-PSDS 与不同检测方法的效果对比**
**Table 5 Comparison of FC-PSDS with different detection methods**

| IDS | ACC | TPR | FPR | PRE |
|---|---|---|---|---|
| Decision Tree | 0.8875 | 0.8913 | 0.0684 | 0.9382 |
| LogisticRegression | 0.7434 | 0.8314 | 0.4370 | 0.7960 |
| LinearSVC | 0.8549 | 0.7956 | 0.0234 | 0.9858 |
| DNN | 0.9465 | 0.9769 | 0.0864 | 0.9581 |
| CNN-LSTM | 0.9377 | 0.9624 | 0.0221 | 0.9452 |
| CNN | 0.9538 | 0.9411 | 0.0132 | 0.9641 |
| **FC-PSDS** | **0.9934** | **0.9996** | **0.0136** | **0.9976** |

### 4.4.3 系统时间性能分析

图 9 展示了本文所采用的 FC-PSDS 方法与 Rubin 研究方法[20]中提出的 CNN-LSTM 模型、Danny 研究方法[19]中的 CNN 模型和决策树、随机森林算法构建的 Powershell 检测模型, 在同一实验环境训练时的时间性能对比, 图中 DT 和 RF 分别代表决策树和随机森林模型。从图中可以看出, FC-PSDS 在时间成本上, 由于采用多模型连接的结构, 导致模型训练时间高于一般机器学习模型, 但相比两种已有方法的时间性能损耗更小。实验表明, FC-PSDS 模型相对已有方法, 在提高召回和降低误报的同时, 可以降低模型训练的时间。

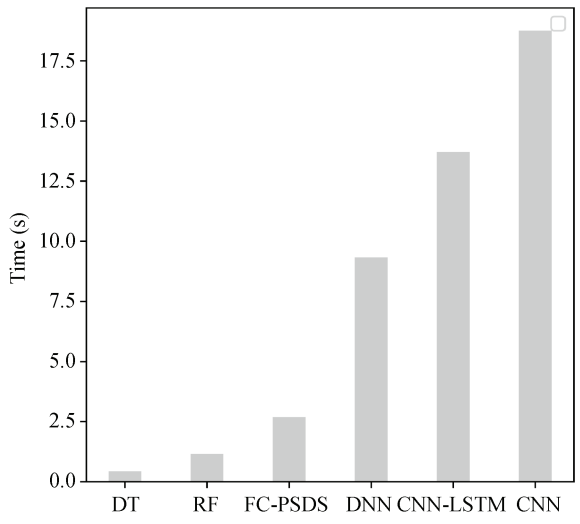


**图 9 FC-PSDS 与不同检测方法的时间性能对比**
**Figure 9 Comparison of time performance between fc-psds and different detection methods**

### 4.4.4 案例分析

为了评估 FC-PSDS 的实际效果, 本文以常见于 APT 攻击中的 Powershell 利用代码为例, 测试其在实际使用过程中的效果。图 10 所示为该段 Powershell 代码的利用流程。该 Powershell 攻击代码利用 Powershell 框架 Empire 下载远程服务器中的利用脚本, 使用 XOR 秘钥解密, 在获取待渗透系统的详细信息后提交到远程命令服务器, 下载对应的利用载荷。

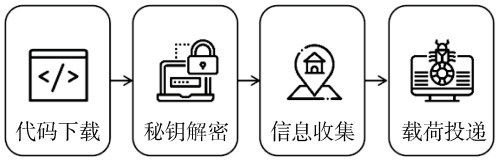


**图 10 Powershell Empire 恶意代码利用流程**
**Figure 10 Malware exploitation process of PowerShell Empire**

对于该段 Powershell 攻击代码的未经特征组合的数据, 本文在上述实验中对比使用的 GNB、DNN、CNN[19]等均无法检测到该段代码的恶意性。在原始数据经 FC-PSDS 特征组合后, 将数据传入 FC-PSDS 的 DNN 网络中则可以检测到该样本的恶意性。FC-PSDS 的效果优越性原因在于其基于随机森林特征组合的过程, 使用非线性的计算重表示了原始数据, 相比仅提取 1657 维特征后直接传入判别网络, FC-PSDS 的判别网络 DNN 可以比单一判别器获得更好表征原始样本的数据, 从而实现更好的检测效果。

## 5 讨论

本文提出的 FC-PSDS 方法相比传统的基于机器学习的恶意代码检测方法可以更好的表征原始数据和学习原有数据特征, FC-PSDS 的复杂性主要体现在: 对于原始特征组合需经过随机森林算法训练得到新特征, 该过程相比将特征直接输入机器学习算法需额外进行模型训练。

FC-PSDS 虽然在检测的准确率、召回率、误报率拥有较好的表现, 但由于双层模型连接以及特征组合的额外训练导致整体复杂性增加, 从而实际运行过程耗时较长。FC-PSDS 的时间消耗主要体现在新特征组合的生成上, 当输入数据维度更高后, 生成新特征组合的时间也会相应增加。特征组合计算和模型训练所需的时间受实验硬件环境影响, 使用 GPU 等高性能硬件计算会显著减少计算时间。考虑

到特征组合和深度学习模型的计算时间成本, 文本提出的 FC-PSDS 方法更适合离线场景。

除本文讨论方法外, 本文使用的特征提取方法和特征组合所用的集成学习方法可以使用其他深度学习算法尝试, 或可对不同特征进行特征融合, 从而实现对不同Powershell代码的泛化能力。除此之外, 对于更多混淆方式的 Powershell 代码的有效特征提取和表示也是接下来的工作重点。

## 6 结束语

由于网络攻击的多样性发展, 以 Powershell 恶意代码攻击为代表的无文件恶意代码攻击因其隐蔽性的特点层出不穷, 本文针对在 Powershell 恶意代码检测过程中存在的特征提取困难、检测泛化和效果差问题展开讨论, 提出了一种基于随机森林特征组合和深度神经网络检测识别的 Powershell 恶意代码检测方法 FC-PSDS。该方法使用随机森林和批量规范化的神经网络的组合方法, 先通过随机森林对原始 Powershell 代码生成更有效的特征组合, 随后利用批量规范化的深度神经网络对特征组合进行训练和学习。

通过随机森林特征组合的方式, FC-PSDS 无需参与不同特征的选取, 实验证明, 基于 FC-PSDS 方法的Powershell恶意代码检测系统性能良好, 相比当前仅使用机器学习或简单神经网络的方法, FC-PSDS 的准确率更高、假阳率更低, 可以满足使用需求。

**致 谢** 感谢中国科学院网络测评技术重点实验室、网络安全防护技术北京市重点实验室的各位老师和同学在实验过程中的帮助和支持, 感谢审稿专家和编辑老师的指导建议。

## 参考文献

[1] Potts M. The State of Information Security[J]. *Network Security*, 2012, 2012(7): 9-11.

[2] Internet Security Threat Report. Symantec, https://www.syman-tec.com/security-center/threat-report, Feb. 2019.

[3] Auty M. Anatomy of an Advanced Persistent Threat[J]. *Network Security*, 2015, 2015(4): 13-16.

[4] Hutchins E, Cloppert M, Amin R. Intelligence-driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains[C]. *In:Proc.of the ICIW*, 2011: 113-125.

[5] Jiang X J. Protection Against 'Fileless' Malware Attacks[J]. *Information Security and Communications Privacy*, 2017, 15(9): 40-47.
(蒋晓晶. "无文件" 恶意软件的攻击与防护[J]. *信息安全与通信保密*, 2017, 15(9): 40-47.)

[6] Sergey Golovanov. A unique files bot attacks news site visi-tors[EB/OL].https://web.archive.org/web/20121115032711/http://www.securelist.com/en/blog/687/A_unique_fileless_bot_attacks_news_site_visitors.

[7] Kaspersky Lab.The DUQU 2.0-Securelist.[EB/OL]. https://securelist.com/files/2015/06/The_Mystery_of_Duqu_2_0_a_sophisticated_cyberespionage_actor_returns.pdf.

[8] Matthew Dunwoody. Dissecting One of APT29′ s Fileless WMI and Powershell Backdoors (POSHSPY) [EB/OL].https://www.fireeye.com/blog/threat-research/2017/03/dissecting_one_ofap.html.

[9] BEEK C, DUNTON T, FOKKER J, et al. McAfee Labs COVID-19 威胁报告 2020 年 7 月 [EB/OL]. https://www.mcafee.com/enterprise/zh-cn/assets/reports/rp-quarterly-threats-july-2020.pdf.

[10] MICROSOFT. Antimalware Scan Interface (AMSI)[EB/OL]. https://docs.microsoft.com/en-us/windows/win32/amsi/antimalware-scan-interface-portal.

[11] Yuan Z L, Lu Y Q, Wang Z G, et al. Droid-Sec: Deep Learning in Android Malware Detection[C]. *The 2014 ACM conference on SIGCOMM - SIGCOMM '14*, 2014: 371-372.

[12] 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,* 2015: 11-20.

[13] Jozefowicz R, Vinyals O, Schuster M, et al. Ex-ploring the Limits of Language Modeling[EB/OL]. 2016: http://arxiv.org/abs/1602.02410.

[14] Zhang X, Zhao J B, LeCun Y. Character-level Convolutional Networks for Text Classification[EB/OL]. 2015: arXiv:1509.01626 [cs.LG]. https://arxiv.org/abs/1509.01626.

[15] Wang Y, Cai W D, Wei P C. A Deep Learning Approach for Detecting Malicious JavaScript Code[J]. *Security and Communication Networks*, 2016, 9(11): 1520-1534.

[16] Liu H X, Ai Z L. A Word-vector Based Malware Classification Model[J]. *Electronic Design Engineering*, 2020, 28(6): 10-16.
(刘恒讯, 艾中良. 一种基于词向量的恶意代码分类模型[J]. *电子设计工程*, 2020, 28(6): 10-16.)

[17] Chen H B, Wu Y, Zou F T. Malware Homology Identification Method Based on ASM2VEC[J]. *Communications Technology*, 2019, 52(12): 3010-3015.
(陈涵泊, 吴越, 邹福泰. 基于 Asm2Vec 的恶意代码同源判定方法[J]. *通信技术*, 2019, 52(12): 3010-3015.)

[18] FANG V. Malicious Powershell Detection via Machine Learn-ing[EB/OL].https://www.fireeye.com/blog/threat-research/2018/07/malicious-Powershell-detection-via-machine-learning.html.

[19] Hendler D, Kels S, Rubin A. Detecting Malicious PowerShell

Commands Using Deep Neural Networks[C]. *The 2018 on Asia Conference on Computer and Communications Security - ASIACCS '18*, 2018: 187-197.

[20] Rubin A, Kels S, Hendler D. AMSI-Based Detection of Malicious PowerShell Code Using Contextual Embeddings[EB/OL]. 2019: arXiv:1905.09538[cs.CR]. https://arxiv.org/abs/1905.09538

[21] Xing B, Wu Y. Analysis of detection and defense of PowerShell attack [J]. *Secrecy Science and Technology*, 2020(2): 39-44.
(邢彬, 吴越. 浅析 PowerShell 攻击的检测与防御[J]. *保密科学技术*, 2020(2): 39-44.)

[22] GAO X B, LI B S. A method and system for detecting fileless malicious code. CN106599684A [P/OL]. 2017-04-26.
(高喜宝, 李柏松. 一种无实体文件恶意代码的检测方法及系统, CN106599684A [P/OL]. 2017-04-26.)

[23] Gao X B, Zhang H Y, Li B S. A method and system for detecting malicious PowerShell. CN106803038A [P/OL]. 2017-06-06.
(高喜宝, 张慧云, 李柏松. 一种检测 Powershell 恶意代码的方法及系统, CN106803038A [P/OL]. 2017-06-06.)

[24] GAO X B, Li B S. A method and system for defending against execution of malicious PowerShell. CN106650447A [P/OL]. 2017-05-10.
(高喜宝, 李柏松. 一种防御 Powershell 恶意代码执行的方法及系统, CN106650447A [P/OL]. 2017-05-10.)

[25] Li J W. Design and implementation on Android malware detection system based on machine learning[D] Peking University,2020.
(李经纬. 基于机器学习的安卓恶意应用检测系统的设计与实现[D].北京大学,2020. )

[26] 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 Dis-tributed Systems* IEEE, 2016: 1175-1180.

[27] 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.

[28] Rusak G, Al-Dujaili A, O'Reilly U M. AST-Based Deep Learning for Detecting Malicious PowerShell[C]. *The 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018: 2276-2278.

[29] Jiang R X. Research on PowerShell Malware Detection Technique [D]. University of Chinese Academy of Sciences,2020.
(姜荣霞. Powershell 恶意软件检测技术研究[D].中国科学院大学,2020.)

[30] Cheng L, Zhongshuo Z. Powershell-based Malware Detection Method Using Command Execution Monitoring and Deep Learn-ing. 2018, 28(5):1197-1207.

[31] BOHANNON D, HOLMES L. Revoke-Obfuscation: Powershell Obfuscation Detection Using Science[EB/OL]. https://github.com/danielbohannon/Revoke-Obfuscation.

[32] Peng J H, Shen Y, Zhang L F. Search on Feature Selection of Data Mining and Its Algorithms[J]. *Computer Engineering and Design*, 2005, 26(5): 1176-1178.
(彭佳红, 沈岳, 张林峰. 数据挖掘中的特征选择及其算法研究[J]. *计算机工程与设计*, 2005, 26(5): 1176-1178.)

[33] He X, Bowers S, Candela J Q, et al. Practical Lessons from Predicting Clicks on Ads at Facebook[J]. *The ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* 2014:1-9.

[34] Dasarathy B V, Sheela B V. A Composite Classifier System Design: Concepts and Methodology[J]. *The IEEE*, 1979, 67(5): 708-713.

[35] Breiman L. Random forests[J]. *Machine Learning*, 2001, 45(1): 5-32.

[36] Github [OL]. https://github.com.

[37] Powershell Gallery [OL]. https://www.Powershellgallery.com.

[38] WHITE J. Pulling Back the Curtains on EncodedCommand Pow-erShell At-tacks[EB/OL].https://unit42.paloaltonetworks.com/unit42-pulling-back-the-curtains-on-encodedcommand-Powershell-attacks/.

**刘岳** 于 2018 年在重庆大学信息安全专业获得学士学位。现在中国科学院大学网络空间安全专业攻读硕士学位。研究领域为网络攻防技术。Email: liuyue0909@iie.ac.cn

**刘宝旭** 于 2002 年在中国科学院研究生院获得博士学位。现任中国科学院信息工程研究所研究员, 第六研究室主任。研究领域为网络安全攻防对抗、网络安全测评技术等。Email: liubaoxu@iie.ac.cn

**赵子豪** 于 2018 年在南昌大学信息安全专业获得学士学位。现在中国科学院大学网络空间安全专业攻读硕士学位。主要研究领域为 Web 安全和恶意代码。Email: zhaozihao@iie.ac.cn

**刘潮歌** 于 2019 年在中国科学院大学信息安全专业获得博士学位。现任中国科学院信息工程研究所副研究员。研究领域为网络攻击追踪溯源、Web 安全和恶意代码。Email: liuchaoge@iie.ac.cn

**王晓茜** 于 2017 年在北京工业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助理研究员, 同时于中国科学院大学网络空间安全学院攻读博士学位。研究领域为网络攻防、Web 安全。Email: wangxiaoxi@iie.ac.cn

**吴贤达** 于 2018 年在北京工业大学获得硕士学位。现任中国科学院信息工程研究所研究实习员。研究领域为 Web 安全、网络攻击追踪溯源、大数据关联取证等。Email: wuxiaoda@iie.ac.cn