

FLWD:基于联邦学习的Webshell检测方法

曾庆鹏¹, 柴江力¹, 吴水秀²

(1. 南昌大学数学与计算机学院, 江西 南昌 330031; 2. 江西师范大学计算机信息工程学院, 江西 南昌 330022)

摘要: Webshell攻击是网络攻击中常用的手段, 攻击者通过Webshell获得Web服务器的部分控制权实施恶意行为。Webshell运行具有隐蔽性, 且攻击者会不断创建新的Webshell变体来逃避安全检测, 同时服务器之间缺乏信息共享和协调, 导致应对Webshell攻击的检测能力参差不齐, 难以形成全面有效的防御体系。为应对以上挑战, 提出了一种基于联邦学习的Webshell检测方法, 该方法融合Webshell的抽象语法树节点值序列特征、代码结构特征、文本混淆度特征以及网络安全领域经验特征, 基于文本卷积神经网络(text convolutional neural network, TextCNN)设计网络模型以学习Webshell样本的恶意行为。同时, 基于联邦学习的FedAvg算法和差分隐私随机梯度下降(differentially private stochastic gradient descent, DP-SGD)算法实现数据不出域场景下多个参与方的协同训练, 保障了参与方在联邦学习过程中的数据隐私, 防止敏感信息泄露。实验结果显示, 在AMWD22数据集上, 所提方法模型的检测准确率为99.47%, $F1$ 值为99.67%, 相比其他算法模型的检测效果更好。在基于联邦学习的Webshell检测中, 所提模型可在数据不出域场景下学习到各参与方数据, 检测准确率由98.01%提升到99.01%。

关键词: Webshell检测; 联邦学习; 差分隐私; 抽象语法树; TextCNN

中图分类号: TP393

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2025033

FLWD: A Webshell detection method based on federated learning

ZENG Qingpeng¹, CHAI Jiangli¹, WU Shuixiu²

1. School of Mathematics and Computer Sciences, Nanchang University, Nanchang 330031, China

2. School of Computer Information Engineering, Jiangxi Normal University, Nanchang 330022, China

Abstract: Webshell attacks were a common technique where attackers gained partial control over the Web server through a Webshell to carry out malicious activities. Due to the covert nature of Webshell operations and the continuous creation of new Webshell variants by attackers to evade security detection, coupled with the lack of information sharing and coordination between servers, uneven detection capabilities in responding to Webshell attacks emerged, making it difficult to establish a comprehensive and effective defense system. To address these chal-

收稿日期: 2025-01-21; 修回日期: 2025-03-25

通信作者: 曾庆鹏, zengqingpeng@ncu.edu.cn

基金项目: 国家自然科学基金(62466028)

Foundation Item: The National Natural Science Foundation of China (62466028)

引用格式: 曾庆鹏, 柴江力, 吴水秀. FLWD:基于联邦学习的Webshell检测方法[J]. 网络与信息安全学报, 2025, 11(3): 109-119.

Citation Format: ZENG Q P, CHAI J L, WU S X. FLWD: A Webshell detection method based on federated learning[J]. Chinese Journal of Network and Information Security, 2025, 11(3): 109-119.

allenges, a Webshell detection method based on federated learning was proposed. The method integrated the abstract syntax tree node value sequence features, code structure features, text obfuscation features, and cybersecurity expertise and experience features of Webshells. A TextCNN - based network model was designed to learn the malicious behaviors of Webshell samples. Meanwhile, the FedAvg algorithm of federated learning and the DP-SGD algorithm were employed for collaborative training across multiple participants without data leaving their domains. This ensured data privacy and prevented sensitive information leakage. Experimental results on the AMWD'22 dataset show that the model's accuracy is 99.47%, with an F1 score of 99.67%, indicating better detection performance compared to basic algorithm models and existing research algorithm models. In the federated learning experiments, the proposed model could learn from the data of each participant without leaving the domain, increasing the detection accuracy from 98.01% to 99.01%.

Keywords: Webshell detection, federated learning, differential privacy, abstract syntax tree, TextCNN

0 引言

随着网络安全威胁的不断演变,网络犯罪分子利用日益普及的 Web 应用程序作为攻击媒介,通过恶意 Webshell 脚本远程控制受感染的服务器,在服务器上进行窃取数据、植入病毒、勒索核心数据、植入木马等恶意操作。微软安全团队的研究报告称^[1],全球网络攻击中 Webshell 的使用正稳步增加,且这种趋势还在加速。从 2020 年 8 月到 2021 年 1 月,微软每个月平均遭受 14 万次 Webshell 威胁,比上一年同期增加了 82%。CISCO TALOS 的研究报告称^[2],自 2023 年 1 月以来,网络攻击中 Webshell 威胁从上季度占所有威胁的 6% 增至接近 25%。由此可见,Webshell 攻防已成为安全行业持续对抗的焦点。防御方不断增强检测能力,而攻击者则不断开发新的绕过技术来规避现有检测机制。这个过程不仅对各组织自身安全能力的要求极高,而且,受害者之间缺乏信息共享和协调,导致应对 Webshell 攻击的检测能力参差不齐,难以形成全面的防御体系。

Webshell 通常包含可疑代码,如命令执行函数或文件操作函数,而正常服务文件则包含应用程序逻辑和业务规则。攻击者会对 Webshell 采用编码加密等手段,使其难以被基于规则的防护检测。然而,这些手段导致 Webshell 的文本与正常代码文本特征存在差异,使得机器学习模型更容易在文本特征上捕获 Webshell。近年来随着机器学习领域的发展及其在自然语言处理等任务上的突破,大量研究采用机器学习技术为 Webshell 检测提供解决方案^[3-4]。然而,现有的 Webshell 检

测方法存在以下问题:多数方法误报率高;对 Webshell 细粒度特征的处理不足;保护参与方数据隐私的相关研究较少。

联邦学习是一种新兴的人工智能基础技术,每个参与者都在本地训练数据集上独立训练模型,无须共享数据,这种技术在保护用户隐私的同时,为解决数据孤岛问题提供了新的途径,已被广泛应用于医疗图像分割、入侵检测、信用评估等领域^[5-7]。然而,尽管联邦学习在许多领域已取得显著进展,但在 Webshell 检测领域的相关研究仍较少。此外,即使联邦学习本身旨在保护隐私的情况下进行,但在实际应用中,依然面临数据隐私泄露、模型中毒攻击和通信安全等多方面的威胁。这些威胁不仅源于参与方和聚合器的潜在不可信性^[8],还在于它们在执行协议时可能并不完全诚实,存在推断其他方隐私信息的风险。本文假设攻击者遵循半诚实威胁模型,即攻击者会按照预定协议执行操作。在这种假设下,期望在协议执行过程中不会泄露额外的信息,从而保护数据隐私。

针对上述问题,本文提出的 FLWD (federated learning Webshell detection) 模型具有以下贡献。

1) 通过抽象语法树 (abstract syntax tree, AST) 获取 Webshell 细粒度特征,结合 AST 节点值序列、代码结构、文本混淆度以及网络安全领域经验特征,设计了基于 TextCNN^[9]的分类模型来学习这些特征,从而提升模型对 Webshell 的检测能力。

2) 提出了基于联邦学习的 Webshell 检测模

型, 该模型采用FedAvg^[10]算法, 能够在数据不出域的前提下, 协同训练多个参与方的Webshell检测模型。同时, 模型集成了差分隐私优化器, 可以有效保护参与方的数据隐私, 提升联邦学习过程的安全性。

1 相关工作

针对Webshell的检测, 研究者提出了多种不同的方法, 包括启发式方法、机器学习方法和深度学习方法。传统的启发式方法通过分析Webshell的特征和行为来进行检测, 包括检测特定的字符串、函数调用和文件名等。这种方法简单直接, 但容易受到Webshell的变种和加密等技术的绕过。Croix等^[11]结合了签名、模糊哈希、危险例程、混淆度和信息熵5种启发式特征, 通过遗传算法为每个特征设置权重来检测Webshell, 这种方法可以有效地检测现有的Webshell, 但缺乏泛化能力。

为了提高检测的准确性和鲁棒性, 研究者开始借鉴机器学习和深度学习的方法。机器学习方法利用已知的Webshell样本进行训练, 通过构建特征集合和使用各种分类算法来建立Webshell检测模型。这种方法可以自动化地检测Webshell, 但需要大量的样本和特征工程。Cui等^[12]提出了一种Webshell检测模型, 该模型使用词频-逆文件频率(term frequency-inverse document frequency, TF-IDF)提取特征并训练一个随机深度森林模型。然后, 结合统计特征构建一个特征矩阵, 使用GBDT分类器进行模型训练, 该方法的漏报率虽低, 但是误报率偏高。Yong等^[13]提出了一种基于集成学习的物联网Webshell攻击检测方法。他们使用TF-IDF提取特征, 并集成多种机器学习模型。尽管集成学习提高了准确率, 但由于特征提取不足, 该方法的整体性能仍然受到限制。林锋等^[14]提出了基于多视角特征融合的Webshell检测方法, 提取词法、句法和抽象特征。他们使用费舍尔评分对特征进行排序和筛选, 并通过SVM进行分类。然而, 该方法的缺点在于样本量不足和检测效果不佳。

近年来, 深度学习方法在Webshell检测中得到了广泛应用, 可以避免特征选择问题和模型学习能力受限的问题。通过使用卷积神经网络、循

环神经网络和长短时记忆网络等深度学习模型, 可以学习到更复杂的特征表示, 提高Webshell检测的准确性和鲁棒性。Zhang等^[15]提出了一种基于多模态特征融合的Webshell检测模型, 从视觉和语义模态提取RGB图像与文本矢量特征, 分别用基于卷积神经网络(convolutional neural network, CNN)的图像分类器和基于CodeBert^[16]的文本分类器处理, 再经决策机制检测。相比传统方法, 该模型检测准确率更高, 但仅在决策阶段进行分类, 未深度融合图像与文本特征, 难以充分挖掘多模态特征间潜在关系。郇洪奎等^[17]提出的ATWebshell模型利用对抗扰动来增强鲁棒性, 并通过双塔模型学习恶意行为, 提高检测准确性, 但依赖对抗扰动会增加一定的误报风险。Feng等^[18]使用Lchecker分析工具从PHP(hypertext preprocessor)脚本文件中提取过程间控制流图, 提出了一种新的基于图神经网络(graph neural network, GNN)的图学习Webshell检测框架, 但是该方法的检测准确率表现不佳。章嘉昊^[19]通过实验对比Bert模型、Word2Vec模型和GloVe模型提取的Webshell特征, 并在3种预训练模型下游均接入TextCNN模型, 实验结果表明, 使用Bert模型动态提取Webshell特征可以有效提升模型的分类效果, 但该方法受限于Bert模型的预训练, 效果不够理想。

针对上述问题, 本文提出了一种改进的Webshell检测方案。通过AST提取细粒度特征, 结合专家经验和统计特征, 实现了更全面的特征提取。本文设计了分类模型以识别Webshell的恶意特征, 从而提升了模型的性能和实用性, 并且引入联邦学习技术以应对实际场景中的挑战。

2 本文方法

2.1 模型总体架构

本文模型的总体架构如图1所示, 主要包括两个核心部分: Webshell检测模块和联邦学习模块。在Webshell检测模块中, 首先通过解析引擎对Webshell样本进行解析, 获得其抽象语法树。然后从AST中获取节点值序列、代码结构、文本混淆度、安全领域经验信息。再分别利用CodeBert和特征统计方法进行词嵌入和特征提取。最终, 通过FLWD方法的特征融合与分类网络实

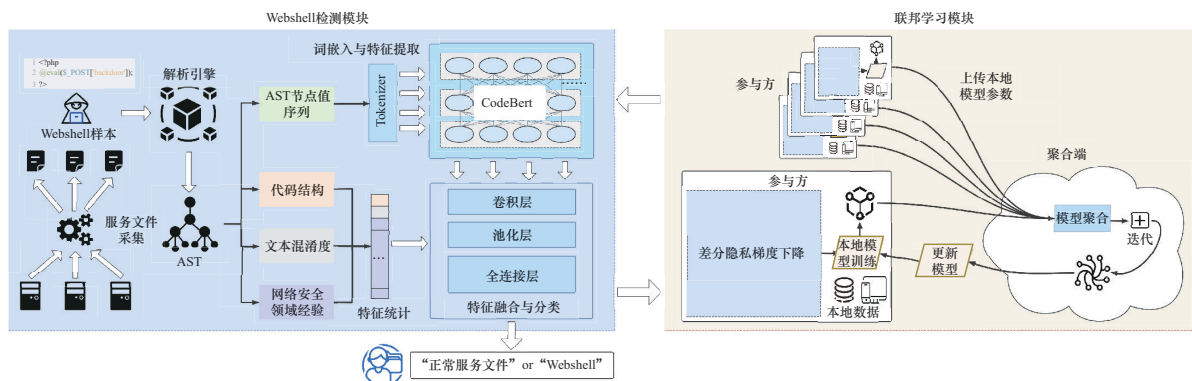


图 1 模型总体架构

Figure 1 Overall architecture of the model

现对恶意 Webshell 样本的有效识别。

在联邦学习迭代过程中,参与方首先接收聚合服务端的聚合模型,使用本地数据进行训练。在训练的过程中,使用差分隐私优化器保护本地数据的隐私。训练完成后,参与方将模型参数传输至聚合服务端,传输过程中通过加密技术确保模型参数的安全。聚合服务端评估选择参与联邦学习的客户端,通过 FedAvg 算法对接收的模型参数进行聚合。上述过程重复进行,直至模型收敛。

2.2 样本解析与特征提取

2.2.1 解析引擎

AST 是一种用于表示源代码语法结构的树状数据结构。它将代码中的每一个语法元素,如变量、操作符、函数调用等作为树节点,并以层次化的方式展示这些元素之间的关系。本文将通过 AST 来获取 Webshell 更细粒度的语法、词法和语义特征。

对于不同语言的程序,在编译之前会将源代码解析成 AST。如 PHP 的 ZendVM 体系架构如图 2 所示。在 ZendVM 的解释过程中,首先进行词法分析和语法分析。词法分析的任务是将要执行的 PHP 源代码去除空格和注释后,切分为一个个 Token,并处理程序的层次结构。接下来,语法分析接收这些 Token 序列,并根据定义的语法规则执行相应的操作。通过词法分析器和语法分析器生成 AST 后,编译器根据这些解析结果,依据 ZendVM 的规则生成中间代码 Op-code。最终,虚拟机执行这些操作码并输出结果。

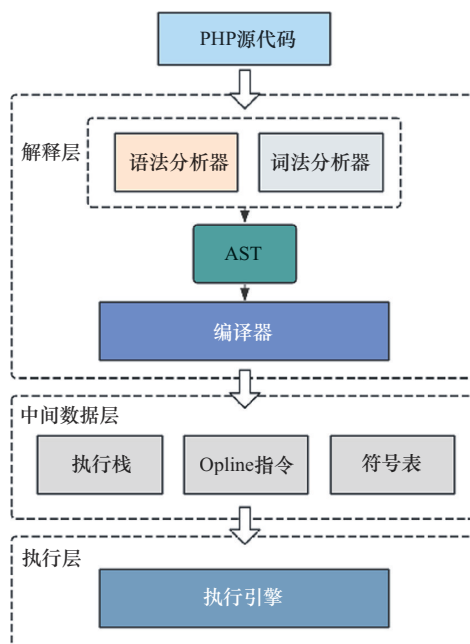


图 2 PHP 的 ZendVM 体系结构

Figure 2 ZendVM architecture of PHP

2.2.2 AST 节点值序列特征提取

为了充分提取样本的代码词法语义信息,利用 2.2.1 节中提取到的 AST,抽取 AST 中每个树节点的语法元素,并将其组合生成 AST 节点值序列。这种方法能够更关注实际操作、变量和函数调用等信息,从而忽略源码中冗余的元素。为了标准化输入,在 Tokenizer 进行词嵌入过程中,参照 AMWD'22^[20]数据集样本 Token 数的分布范围(如图 3 所示)来设置提取的 Token 数。对于 Token 数不足的样本,将通过 Tokenizer 定义的 max_length 进行填充;而对于超出 Token 数部分则进行舍弃,本文设置的最大 Token 数为 512。

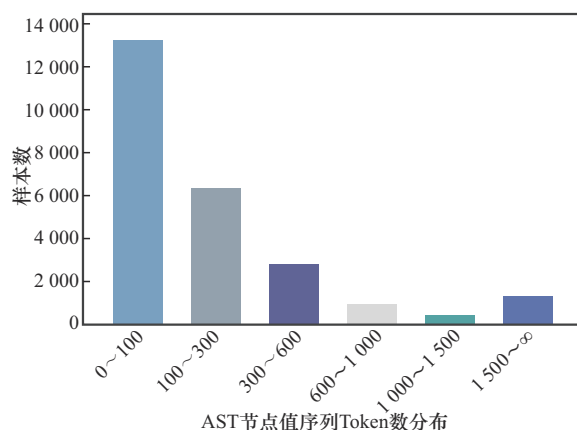


图3 AST节点值序列Token数范围统计

Figure 3 Statistics of token number range in AST node value sequences

在Tokenizer获取Token Embeddings、Segment Embeddings和Position Embeddings后,将这些嵌入向量输入到BERT模型中进行特征提取。本文使用的是微软预训练的CodeBERT模型,其BERT模型架构如图4所示。这3个嵌入向量将被输入到双向多层Transformer编码器中进行处理,利用Transformer编码器的多头注意力机制来编码并输出相应的特征词向量。

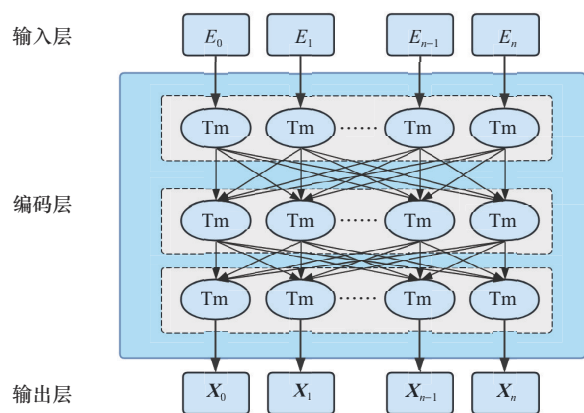


图4 BERT模型架构

Figure 4 BERT model architecture

2.2.3 统计特征提取

单靠提取代码的词法和语义特征,尚不足以使分类器达到理想的准确度。因此,本文进一步融合了文本混淆度特征、代码结构特征以及安全领域经验特征这3类统计特征,以提升分类器的性能。

文本混淆度特征:通常情况下,Webshell为

了绕过检测,会采用加密、编码等方式进行混淆,这会引入连续的随机字符串,从而导致其熵值较高。为了计算文本混淆度,本文将信息熵作为识别Webshell的重要特征之一。

对样本的所有节点值计算其信息熵^[21],节点值 s 的长度为 N ,字符集为 $c_1, c_2, \dots, c_k$,每个字符 c_i 出现的次数为 n_i ,则字符 c_i 的概率 p_i 可以表示为:

$$p_i = \frac{n_i}{N} \quad (1)$$

字符串的熵 H 可以表示为:

$$H(s) = - \sum_{i=1}^k p_i \lg p_i \quad (2)$$

得到样本所有节点值的熵 $H_1, H_2, \dots, H_n$ 后,对这些熵进行统计,计算其最大值、最小值、方差、标准差,如式(3)~式(7)所示。将所有节点值的熵以及这些统计结果作为文本混淆度特征。

$$H_{\max} = \max(H_1, H_2, \dots, H_n) \quad (3)$$

$$H_{\min} = \min(H_1, H_2, \dots, H_n) \quad (4)$$

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (H_i - \mu)^2 \quad (5)$$

$$\sigma = \sqrt{\sigma^2} = \sqrt{\frac{1}{n} \sum_{i=1}^n (H_i - \mu)^2} \quad (6)$$

$$\mu = \frac{1}{n} \sum_{i=1}^n H_i \quad (7)$$

代码结构特征:代码的结构特征可以通过多种属性进行描述,包括节点数、行号、特定节点、声明节点等。这些属性有助于理解代码的复杂性和结构特征。为了更全面地描述代码的结构特征,可以对AST进行以下统计分析:统计特定类型节点的数量及总节点数;计算每行节点数的方差和标准差;提取行号信息

安全领域经验特征:通常Webshell通过入口函数获取外部信息源,并通过各种函数绕过基于规则的检测,进入执行危险函数。基于安全领域的专业知识,分别统计入口函数、绕过检测常用函数及危险函数的信息,如POST、rot13、system等。将这些信息作为检测Webshell的重要经验特征,有助于有效识别潜在威胁。

2.3 Webshell 检测模型

本文提出的 Webshell 检测模型如图 5 所示, 模型主要利用 3 类统计特征序列: 文本混淆度统计序列、代码结构统计序列、安全领域经验统计序列。

对于 AST 节点值序列特征的词向量矩阵, 使用 TextCNN 模型进行特征学习。TextCNN 通过卷积、池化等操作提取局部特征。令卷积核为 $\omega \in \mathbb{R}^{h \times d}$, h 为卷积核高度, d 为词向量维度。模型使用 3 个卷积核, 其大小分别为 (2,768)、(3,768)、(4,768), 输入通道数为 1, 输出通道数为 256。 x_i 表示样本中第 i 个词的词向量, x_{ii+h-1} 是卷积核纵向滑动时根据卷积核大小获取的窗口, 表示样本中词语之间的关联, x_{ii+h-1} 通过卷积核的卷积运算得到特征图 c_i , 如式(8)所示。

$$c_i = f(x_{ii+h-1} \cdot \omega + b) \quad (8)$$

其中, $c \in \mathbb{R}^{n-h+1}$, f 为激活函数 ReLU, b 为偏置参数。在得到特征图后, 对特征图进行最大池化, 将每个特征图中最大值提取出来以代替整个特征图, 捕获其中最重要的特征, 以达到降维效果。然后, 将池化层所提取到的特征进行拼接融合, 并通过一个线性变换层 (Linear 层) 将特征映射到 128 维。接下来, 将这些特征与统计特征

进行融合 (统计特征包括文本混淆度特征、代码结构特征以及安全领域经验特征)。将这些统计特征拼接后, 通过 LSTM 网络, 获取局部特征之间的相互关系, 得到一个 128 维的高层次特征表示。最终, 将两者的特征进行拼接融合, 并通过一个全连接层将特征映射到二维, 得到最后的分类结果。

2.4 联邦学习模型

联邦学习模型如图 6 所示, 采用 Google 在 2017 年提出的 FedAvg 算法, 用户数据不出域即可参与共同的模型训练, 并且结合基于差分隐私的梯度下降 DP-SGD 算法^[22], 保障用户本地数据的隐私与安全。在联邦学习过程中, 每个参与方使用其私有数据 D_k 训练模型, 模型参数用 ω_k 表示。在联邦学习开始时, 首先由聚合服务器端预训练得到一个初始化模型, 令当前全局模型参数初始值为 ω_0 , 全局模型初始参数将通过网络下发到 M 个参与方, 每次迭代中, 参与方接收到模型参数 ω_l 并进行初始化, $\omega_{l+1}^k \leftarrow \omega_l$, 参与方本地数据 D_k 将批样本大小 B 分为若干批次进行训练。其中在反向传播的梯度下降过程中, 根据损失函数 L 计算轮次 t 时的损失, 求该轮次下各参数的梯度, 如式(9)所示。

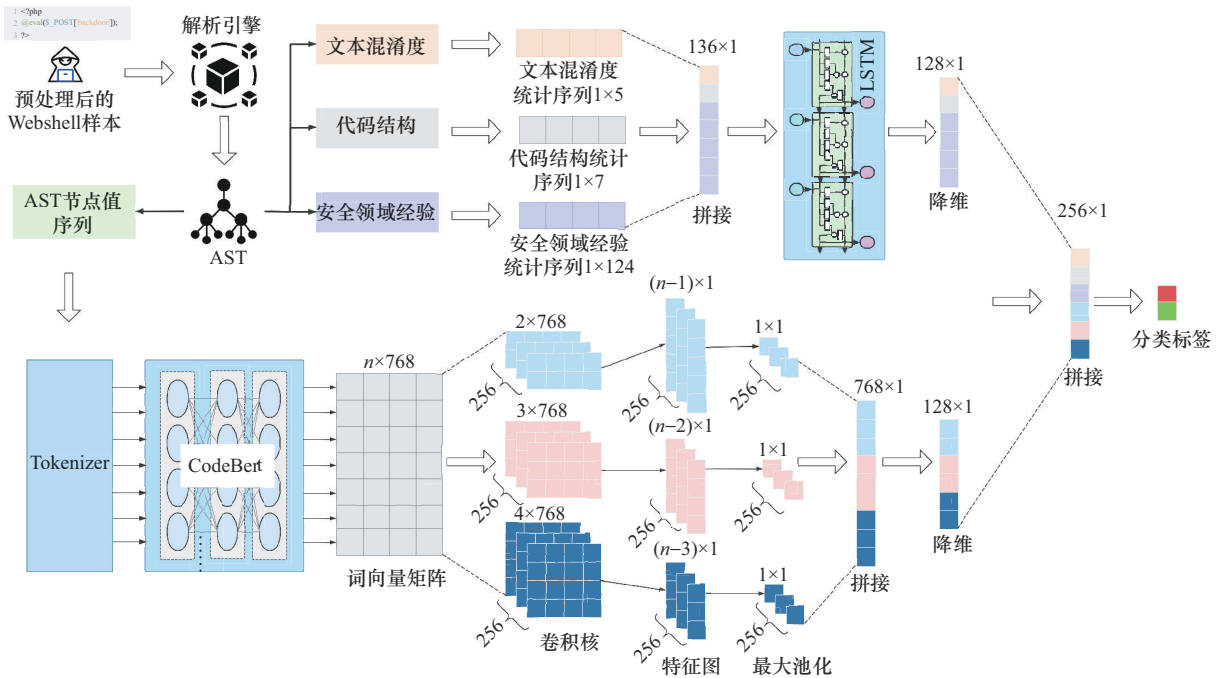


图 5 Webshell 检测模型

Figure 5 Webshell detection model

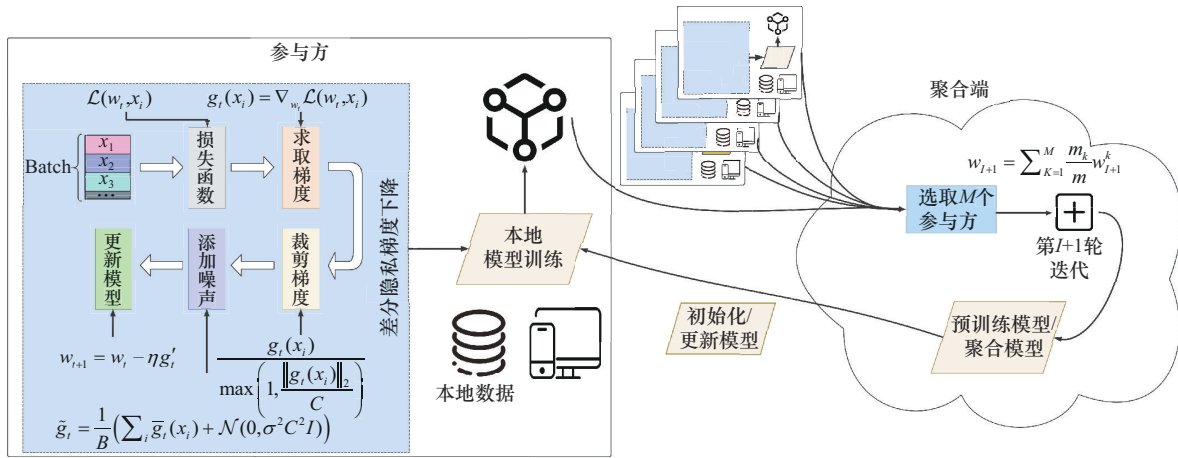


图6 联邦学习模型

Figure 6 Federated learning model

$$g_t(x_i) = \nabla_{w_t} L(W_t, x_i) \quad (9)$$

采用式(10)进行梯度裁剪, 通过梯度的最大值来防止梯度爆炸问题。

$$\bar{g}_t(x_i) \leftarrow g_t(x_i) / \max\left(1, \frac{\|g_t(x_i)\|_2}{C}\right) \quad (10)$$

通过梯度的 $L2$ 范数除以梯度范数界 C 来裁剪梯度。添加噪声得到加噪梯度 $\tilde{g}_t$, $\tilde{g}_t$ 的计算式为:

$$\tilde{g}_t = \frac{1}{B} \left(\sum_i \bar{g}_t(x_i) + N(0, \sigma^2 C^2 I) \right) \quad (11)$$

其中, N 是高斯分布, σ 是噪声尺度。使用 $\omega_t - \eta \tilde{g}_t$ 更新参数得到 ω_{t+1} , 其中 η 是学习率。各参与方训练完成后, 将该迭代下训练得到的本地模型参数 ω_{t+1} 上传至聚合端。聚合端对随机选取的 M 个参与方进行聚合, 全局模型参数 ω_{t+1} 计算式为:

$$\omega_{t+1} = \sum_{k=1}^M \frac{m_k}{m} \omega_{t+1}^k \quad (12)$$

其中, $\frac{m_k}{m}$ 为 $I+1$ 轮聚合中参与方 k 占参与训练的 M 个参与方总样本的比例。迭代以上步骤直到全局模型参数收敛。

3 实验与分析

3.1 数据集

本实验使用阿里云AMWD'22数据集, 该数据集来自Webshell文件经过沙箱程序静态模拟运行后的AST指令序列。数据集中包含PHP和JSP

两种语言的样本, 共计25 185个, 其中PHP和JSP样本的比例约为8:1, 正常文件与Webshell的比例约为4:1。

3.2 评价指标

为评估模型的性能, 本文实验Webshell检测效果采用的评价指标是准确率(accuracy)、精确率(precision)、召回率(recall)、 $F1$ 值、误报率(FNR)、漏报率(FPR)。计算式如式(13)~式(18)所示, P 表示正样本, N 表示负样本, TP 表示将正样本预测为正样本的数据量, FP 表示将负样本预测为正样本的数据量, TN 表示将负样本预测为负样本的数据量, FN 表示将正样本预测为负样本的数据量。

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (13)$$

$$\text{precision} = \frac{TP}{TP + FP} \quad (14)$$

$$\text{recall} = \frac{TP}{TP + FN} \quad (15)$$

$$F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (16)$$

$$\text{FNR} = \frac{FN}{TP + FN} \quad (17)$$

$$\text{FPR} = \frac{FP}{FP + TN} \quad (18)$$

3.3 实验环境及模型参数

本文在配置8核、30 GB内存搭载NVIDIA A100的Linux服务器上进行实验。采用Pytorch框架和Python语言开发。在Webshell检测实验中经CodeBert提取得到特征词向量维度为768维, 卷积层卷积核高度分别为2、3、4, 卷积核的输

出通道数为 256, 模型训练过程中采用 SGD 优化器, 学习率参数设置为 0.01。在联邦学习的 Webshell 检测实验中, 对数据集以 Dirichlet 分布^[23]进行划分, 模拟 10 个客户端参与联邦学习, Dirichlet 分布中参数 α 设置为 1, 采用 DP-SGD 和 DP-Adam 优化器进行实验对比, 差分隐私中的 σ 值设置为 0.2, 客户端选择中 k 设置为 0.5, drop-out 为 0.2, 训练 batch_size 为 64。

3.4 实验内容及结果分析

本文通过 2 组实验论证方法的有效性和合理性。

实验 1 为验证本文提出的 FLWD 模型在 Webshell 检测任务上的优势, 将 FLWD 模型分别与不同算法模型进行对比实验, 其结果见表 1 和表 2。实验中将数据集以 7:3 划分为训练集和测试集, 并保证不同实验训练集、测试集数据和共有的超参数一致。其中训练轮数均为 100 轮。

实验 2 为验证数据不出域场景下, 联邦学

习方法协同训练全局模型的有效性, 将数据集以 1:4:5 划分为预训练数据集、测试数据集、客户端数据集。本文模拟 10 个客户端, 采用 Non-IID 的方式进行数据划分。为评估差分隐私对模型的性能的影响, 本文在模型学习过程中加入差分隐私保护的方法, 使用 DP-SGD、DP-Adam 优化器进行实验对比。此外, 在服务端聚合各客户端模型参数时, 选择评估前 50% 的客户端模型进行聚合。实验具体设置如下: 预训练模型训练轮数: 100 轮; 联邦学习模拟过程: 30 次迭代, 每次迭代中每个客户端训练 30 轮。实验结果见表 3~表 5。

表 1 实验结果表明, FLWD 模型准确率达到 99.47%, 精准率达到了 99.54%, 误报率为 0.2%, 漏报率为 1.87%。表 2 实验结果表明, FLWD 模型在与现有研究中算法模型的对比中, 各项评估性能指标均有出色的表现。如图 7 和图 8 所示, FLWD 模型的准确率、精准率、召回率、

表 1 FLWD 模型与基本算法模型的对比实验

Table 1 Comparative experiments of FLWD model and basic algorithm models

模型	准确率	精确率	召回率	F1	FNR	FPR
SVM	0.832 6	0.869 7	0.930 7	0.899 2	0.069 3	0.565 6
MLP	0.930 0	0.936 5	0.979 0	0.957 3	0.021 0	0.268 5
GRU	0.939 8	0.984 4	0.939 7	0.961 5	0.060 3	0.060 2
LSTM	0.951 7	0.957 3	0.983 7	0.970 3	0.016 3	0.177 8
FLWD	0.994 7	0.995 4	0.998 0	0.996 7	0.002 0	0.018 7

表 2 FLWD 模型与现有研究中算法模型的对比实验

Table 2 Comparative experiments of FLWD model and existing algorithm models in research

模型	准确率	精确率	召回率	F1	FNR	FPR
XGBoost ^[24]	0.981 1	0.978 5	0.998 4	0.988 3	0.001 6	0.089 0
AST-DF ^[25]	0.993 0	0.994 4	0.996 9	0.995 6	0.003 1	0.022 8
GRU-Attention ^[26]	0.982 1	0.987 5	0.990 3	0.988 9	0.009 7	0.050 8
DNN ^[27]	0.973 8	0.980 2	0.987 3	0.983 7	0.012 7	0.080 9
FLWD	0.994 7	0.995 4	0.998 0	0.996 7	0.002 0	0.018 7

表 3 联邦学习对全局模型准确率影响的对比实验

Table 3 Comparative experiment on the impact of federated learning on the accuracy of global models

模型	准确率	精确率	召回率	F1	FNR	FPR
Pre_model	0.980 1	0.981 5	0.993 9	0.987 7	0.006 1	0.075 9
FLWD	0.991 1	0.993 6	0.995 4	0.994 5	0.004 6	0.026 1

表 4 联邦学习过程中使用不同优化器的对比实验

Table 4 Comparative experiment on using different optimizers in federated learning process

模型	优化器	准确率	精确率	召回率	<i>F1</i>	FNR	FPR
FLWD	SGD	0.991 1	0.993 6	0.995 4	0.994 5	0.004 6	0.026 1
	DP-SGD	0.990 1	0.992 9	0.994 8	0.993 8	0.005 2	0.028 6
	Adam	0.896 5	0.981 9	0.887 4	0.932 3	0.112 6	0.066 5
	DP-Adam	0.900 6	0.893 8	0.994 6	0.941 5	0.005 4	0.486 3

表 5 联邦学习过程中客户端评估选择对比实验

Table 5 Comparative experiment on client evaluation selection in federated learning process

模型	评估选择参数	准确率	精确率	召回率	<i>F1</i>	FNR	FPR
FLWD	$k=0.5$	0.991 1	0.993 6	0.995 4	0.994 5	0.004 6	0.026 1
	$k=1$	0.990 4	0.991 7	0.996 4	0.994 0	0.003 6	0.033 8

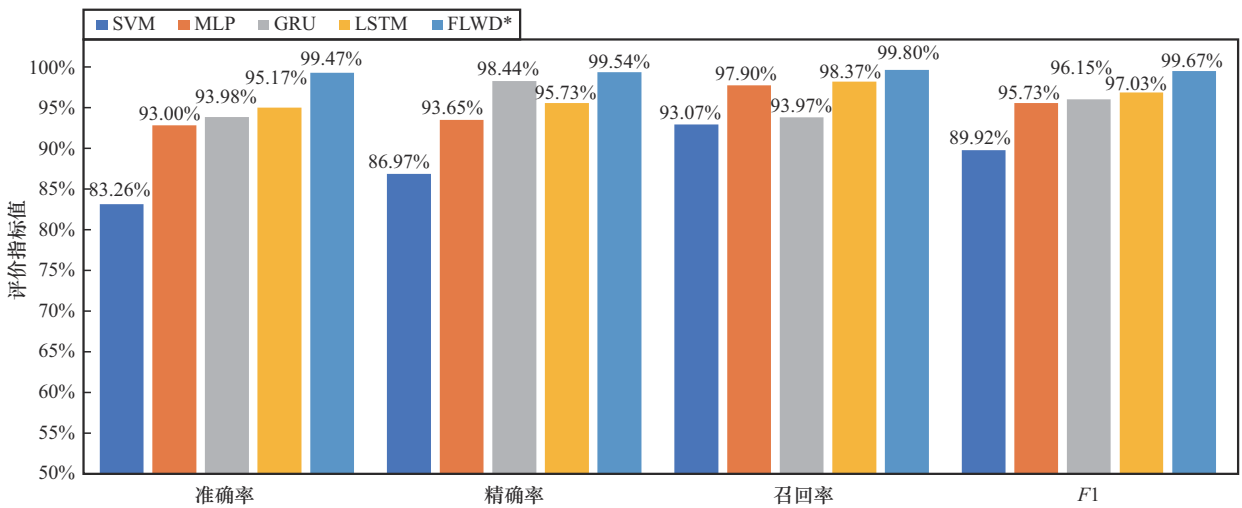


图 7 FLWD和基本算法模型的表现对比

Figure 7 Comparison of performance between FLWD and basic algorithm models

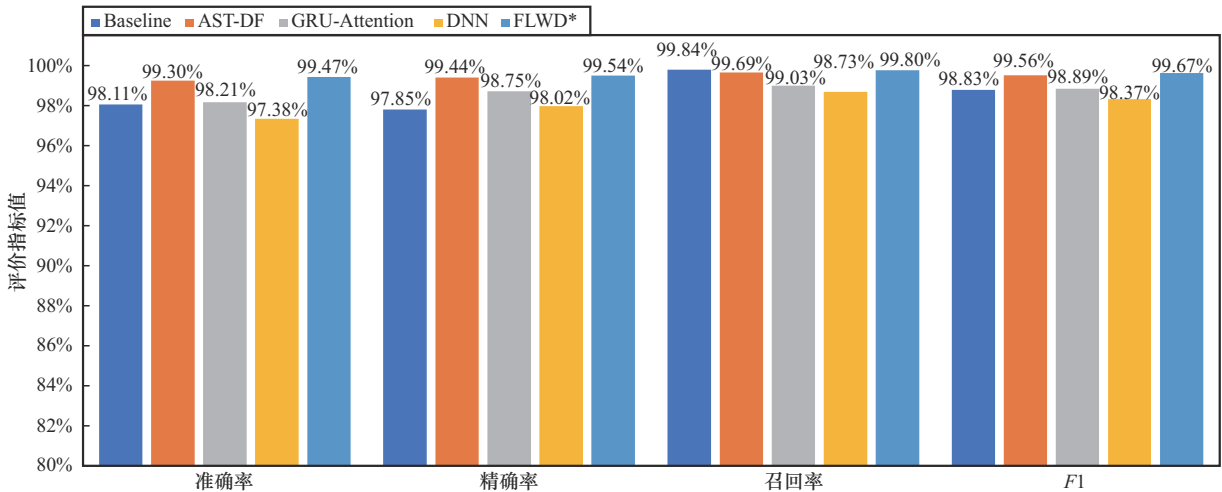


图 8 FLWD和相关研究算法模型的表现对比

Figure 8 Comparison of performance between FLWD and related research algorithm models

F1 值性能表现均为最优。在实际的网络环境中,高准确率和低误报率至关重要,本实验验证了 FLWD 模型能够有效避免错误分类,具有较高的可靠性。

表 3 的实验结果表明,仅使用本地数据进行训练时,通过联邦学习,在不考虑差分攻击情况下,随着更多的客户端参与,模型准确率由 98.01% 提升到 99.11%。漏报率由 7.59% 下降到 2.61%,且其他指标也都有一定的提升。这些数据表明,联邦学习模式对于网络空间中 Webshell 攻击的协同防护效果有极大的提升。在考虑差分攻击情况下,表 4 对比了采用 SGD、DP-SGD、Adam、DP-Adam 优化器的实验结果,证明了检测模型采用 DP-SGD 优化器具有更好的效果。准确率对比不采用联邦学习由 98.01% 提升到 99.01%,漏报率由 7.59% 下降到 2.86%。此外,选择合适的客户端评估也有利于联邦学习聚合模型的性能。表 5 实验对比了选择一部分表现更好的模型进行聚合,聚合模型准确率由 99.04% 提升到 99.11%,且各项性能表现都更好。由此可见,联邦学习在 Webshell 检测应用中具有可行性。

4 结束语

本文提出了一种基于联邦学习的 Webshell 检测模型,该模型利用联邦学习技术,在数据不出域的场景下进行联邦学习训练,并通过 DP-SGD 优化器在模型训练过程中添加扰动,以增强模型对差分攻击的鲁棒性。实验结果表明,本文提出的模型通过结合 AST 和统计特征进行 Webshell 检测,取得了良好的性能表现。此外,实验结果还证明了联邦学习在提升网络空间中多方协同检测 Webshell 能力方面的可行性。这为研究 Webshell 在实际应用中的探索提供了新的视角。

参考文献:

- [1] MICROSOFT. Web shell attacks continue to rise[EB]. 2021.
- [2] CISCO TALOS. Quarterly report: incident response trends in Q1 2023[EB]. 2023.
- [3] HANNOUSSE A, YAHIOUCHE S. Handling Webshell attacks: a systematic mapping and survey[J]. Computers & Security, 2021, 108 (102366): 1-26.
- [4] GIBERT D, MATEU C, PLANES J. The rise of machine learning for detection and classification of malware: research developments, trends and challenges[J]. Journal of Network and Computer Applications, 2020, 153(102526): 1-22.
- [5] SUN G, SHU H, SHAO F, et al. FKD-med: privacy-aware, communication-optimized medical image segmentation via federated learning and model lightweighting through knowledge distillation[J]. IEEE Access, 2024, 12(3): 33687-33704.
- [6] FRIHA O, FERRAG M A, BENBOUZID M, et al. 2DF-IDS: decentralized and differentially private federated learning-based intrusion detection system for industrial IOT[J]. Computers & Security, 2023, 127(103097): 1-18.
- [7] LIU Z, SUN H, SONG J, et al. Vertical federated learning architecture for power company and financial company and electricity pricing model considering user credit evaluation[C]//2023 3rd International Conference on Consumer Electronics and Computer Engineering (ICCECE). Guangzhou: IEEE Press, 2023: 820-826.
- [8] 杨帆, 孙奕, 陈性元, 等. 基于同态代理重认证的可验证联邦聚合方法[J]. 网络与信息安全学报, 2024, 10(3): 91-106.
- [9] YANG F, SUN Y, CHEN X Y, et al. Verifiable federated aggregation method based on homomorphic proxy re-authentication[J]. Chinese Journal of Network and Information Security, 2024, 10(3): 91-106.
- [9] KIM Y. Convolutional neural networks for sentence classification [EB]. arXiv preprint arXiv:1408.5882, 2014.
- [10] MCMAHAN B, MOORE E, RAMAGE D, et al. Communication-efficient learning of deep networks from decentralized data[C]//Artificial Intelligence and Statistics. Fort Lauderdale: PMLR, 2017: 1273-1282.
- [11] CROIX A, DEBATTY T, MEES W. Training a multicriteria decision system and application to the detection of PHP Webshells[C]//2019 International Conference on Military Communications and Information Systems (ICMCIS). Budva: IEEE Press, 2019: 1-8.
- [12] CUI H, HUANG D, FANG Y, et al. Webshell detection based on random forest - gradient boosting decision tree algorithm[C]//2018 IEEE Third International Conference on Data Science in Cyber-space (DSC). Guangzhou: IEEE Press, 2018: 153-160.
- [13] YONG B, WEI W, LI K C, et al. Ensemble machine learning approaches for Webshell detection in Internet of things environments[J]. Transactions on Emerging Telecommunications Technologies, 2022, 33(6): 1-13.
- [14] 林锋, 徐柳婧, 陈晓华, 等. 一种基于多视角特征融合的 Webshell 检测方法[J]. 电信科学, 2020, 36(6): 125-132.
- [14] LIN F, XU L J, CHEN X H, et al. Method of Webshell detection based on multiview feature fusion[J]. Telecommunications Science, 2020, 36(6): 125-132.
- [15] ZHANG Y F, KANG H Y, WANG Q. MMFDetect: Webshell evasion detect method based on multimodal feature fusion[J]. Electronics, 2025, 14(3): 1-19.
- [16] FENG Z, GUO D, TANG D, et al. Codebert: a pretrained model for programming and natural languages[EB]. arXiv preprint arXiv: 2002.08155, 2020.

- [17] 郇洪奎, 安通鉴, 税雪飞, 等. ATWebshell: 基于对抗学习和长短期语义感知的Webshell检测方法[J]. 数据与计算发展前沿, 2022, 4(5): 68-76.
- GAO H K, AN T J, SHUI X F, et al. ATWebshell: Webshell detection model based on adversarial learning and long-short semantic awareness[J]. Frontiers of Data & Computing, 2022, 4(5): 68-76.
- [18] FENG P, WEI D, LI Q, et al. GlareShell: graph learning-based PHP Webshell detection for web server of industrial internet[J]. Computer Networks, 2024, 245(110406): 1-10.
- [19] 章嘉昊. 基于深度学习的Webshell检测算法的研究[D]. 南昌: 南昌大学, 2023.
- ZHANG J H. Research of Webshell detection algorithm based on deep learning[D]. Nanchang: Nanchang University, 2023.
- [20] AMWD'22-阿里云安全Webshell文本检测数据集[EB]. 2023.
- AMWD'22-alibaba cloud security Webshell text detection dataset [EB]. 2023.
- [21] 潘桐, 陈伟, 吴礼发. 面向不平衡样本的物联网入侵检测方法[J]. 网络与信息安全学报, 2023, 9(1): 130-139.
- PAN T, CHEN W, WU L F. IoT intrusion detection method for unbalanced samples[J]. Chinese Journal of Network and Information Security, 2023, 9(1): 130-139.
- [22] ABADI M, CHU A, GOODFELLOW I, et al. Deep learning with differential privacy[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. Vienna: ACM Press, 2016: 308-318.
- [23] MACKAY D J C. Information theory, inference and learning algorithms[M]. United Kingdom: Cambridge University Press, 2003. 311-318.
- [24] ZHANG H, LIU M, YUE Z H, et al. A PHP and JSP Webshell detection system with text processing based on machine learning[C]//2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). Guangzhou: IEEE Press, 2020: 1584-1591.
- [25] DONG C, LI D. AST-DF: a new Webshell detection method based on abstract syntax tree and deep forest[J]. Electronics, 2024, 13(1482): 1-15.
- [26] LI T, REN C, FU Y, et al. Webshell detection based on the word attention mechanism[J]. IEEE Access, 2019, 7: 185140-185147.
- [27] HANNOUSSE A, NAIT-HAMOUD M C, YAHIOUCHE S. A deep learner model for multi-language Webshell detection[J]. International Journal of Information Security, 2023, 22(1): 47-61.

[作者简介]



曾庆鹏 (1974-), 男, 江西吉水人, 南昌大学副教授、硕士生导师, 主要研究方向为网络与信息安全、信息检索与数据挖掘。



柴江力 (1998-), 男, 浙江杭州人, 南昌大学硕士生, 主要研究方向为网络与信息安全、联邦学习。



吴水秀 (1975-), 女, 江西南丰人, 江西师范大学副教授、硕士生导师, 主要研究方向为自然语言处理, 机器学习和信息安全。