

基于 Stacking 融合模型的 Web 攻击检测方法

万巍^{1,2}, 石鑫^{1,2}, 魏金侠^{1,2}, 李畅³, 龙春^{1,2}

¹中国科学院计算机网络信息中心 北京 中国 100190

²中国科学院大学 北京 中国 101408

³中国信息通信研究院 北京 中国 100191

摘要 随着计算机技术与互联网技术的飞速发展, Web 应用在人们的生产与生活中扮演着越来越重要的角色。但是在人们的日常生活与工作中带来了更多便捷的同时, 却也带来了严重的安全隐患。在开发 Web 应用的过程中, 大量不规范的新技术应用引入了很多的网站漏洞。攻击者可以利用 Web 应用开发过程中的漏洞发起攻击, 当 Web 应用受到攻击时会造成严重的数据泄露和财产损失等安全问题, 因此 Web 安全问题一直受到学术界和工业界的关注。超文本传输协议(HTTP)是一种在 Web 应用中广泛使用的应用层协议。随着 HTTP 协议的大量使用, 在 HTTP 请求数据中包含了大量的实际入侵, 针对 HTTP 请求数据进行 Web 攻击检测的研究也开始逐渐被研究人员所重视。本文提出了一种基于 Stacking 融合模型的 Web 攻击检测方法, 针对每一条文本格式的 HTTP 请求数据, 首先进行格式化处理得到既定的格式, 结合使用 Word2Vec 方法和 TextCNN 模型将其转换成向量化表示形式; 然后利用 Stacking 模型融合方法, 将不同的子模型(使用配置不同尺寸过滤器的 Text-CNN 模型搭配不同的检测算法)进行融合搭建出 Web 攻击检测模型, 与融合之前单独的子模型相比在准确率、召回率、F1 值上都有所提升。本文所提出的 Web 攻击检测模型在公开数据集和真实环境数据上都取得了更加稳定的检测性能。

关键词 入侵检测; stacking; 融合模型; web 攻击

中图分类号 TP39 DOI 号 10.19363/J.cnki.cn10-1380/tn.2024.01.06

Web Attack Detection Method Based on Stacking Fusion Model

WAN Wei^{1,2}, SHI Xin^{1,2}, WEI Jinxia^{1,2}, LI Chang³, LONG Chun^{1,2}

¹ Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China

² University of Chinese Academy of Sciences, Beijing 101408, China

³ China Academy of Information and Communications Technology, Beijing 100191, China

Abstract With the rapid development of computer technology and Internet technology, Web applications play an increasingly important role in people's production and life. However, while it has brought more convenience to people's daily life and work, it has also brought serious safety risks. In the process of developing Web applications, a large number of irregular new technology applications have introduced many vulnerabilities. Attackers can exploit the vulnerabilities in the development of Web applications to launch Web attacks. When a Web application is attacked, it will cause serious security problems, such as data leakage and property damage. Therefore, Web security issues have always attracted the attention of academia and industry. Hypertext Transfer Protocol (HTTP) is an application-layer protocol that is widely used in Web applications. With the extensive use of the HTTP protocol, a large number of actual intrusions are included in the HTTP request data, and the research on Web attack detection based on the HTTP request data has also begun to be paid more and more attention by researchers. In this paper, we propose a Web attack detection method based on the Stacking fusion model. For each HTTP request data in text format, it is firstly formatted to obtain a predetermined format, and then the Word2Vec method and the TextCNN model are combined to convert it into a vectorized representation; we use the Stacking-based model fusion method to fuse different sub-models (using TextCNN models with filters of different sizes and different detection algorithms) to build a Web attack detection model. Compared with the individual sub-models before fusion, the precision, recall, and F1-score are improved. The proposed web attack detection model achieves more stable detection performance on both public dataset and real-world data.

Key words intrusion detection; stacking; fusion model; web attack

通讯作者: 龙春, 博士, 正高级工程师, Email: anquanip@cnic.cn。

本课题得到中国科学院战略性先导科技专项(C类)项目(No. XDC02030600), 中国科学院青年创新促进会(No. 2022170)资助。

收稿日期: 2022-04-29; 修改日期: 2022-06-28; 定稿日期: 2023-10-31

1 引言

现如今, 互联网技术与计算机技术在人们的工作与生活中扮演着越来越重要的角色, 无论是居家上网还是公司办公, 都越来越离不开 Web 应用。Web 应用具有操作简便、无需下载、随时可用以及无需主动升级的优点, 这些优点使得 Web 应用的用户量迅速上升, 出现了越来越多的 Web 应用, 随之而来的安全问题也变得越来越严重。Web 应用采用浏览器/服务器(B/S)架构, 用户通过客户端向服务器发送访问请求, 服务器收到用户发送的请求后向用户返回对应的数据。而攻击者可以利用服务器存在的漏洞向服务器发送携带恶意参数或恶意数据的请求来攻击服务器, 实现比如假冒身份绕过系统访问控制进入系统内核或通过注入恶意代码使系统瘫痪进而泄露敏感信息等攻击行为^[1]。常见的 Web 应用攻击主要包含 SQL 注入攻击、跨站脚本攻击、远程恶意文件包含^[2]等。

超文本传输协议(HTTP)作为一种分布式、可协作、超媒体信息系统的上层应用协议^[3], 目前已成为一种通用的传输协议。大量的数据通过 HTTP 协议进行传输, 它的安全问题已经引起研究人员的高度重视。针对 HTTP 请求数据进行检测已经成为入侵检测的一项重要手段。相关科研人员提出了大量针对 HTTP 请求数据的异常检测方法以保护网络安全。HTTP 请求数据是将一些字段信息以文本的方式进行存储, 要想将其输入到检测模型中进行 Web 攻击检测, 需将其做数值化处理。一些基于统计学的方法可以将原始的 HTTP 请求数据转换成数值形式的特征, 但这样的处理往往会忽略原始请求中上下文信息之间的关系, 同时在实际应用中也会产生大量的人力物力资源消耗。

当前很多 Web 攻击检测模型都是使用单一模型进行检测或是将多个模型进行检测性能对比以选择最优检测模型, 将多个检测模型进行融合可以得到更强的检测模型。模型融合的方法已经在很多领域得到了应用。模型融合策略有很多, 比如加权平均法, 这在集成学习中的 Boosting 方法^[4]中有所应用; 再比如投票法, 包括多数投票、相对多数投票、加权投票等方法, 在集成学习中的 Bagging^[5]方法中有所应用。与前述几种融合策略相比, 学习法则更胜一筹, 这种方法主要是利用一个元学习器将多个不同的学习器结合起来, Stacking 方法^[6]是一种广泛应用的学习法模型融合策略。

在本文中, 首先将原始的 HTTP 请求进行格式

化处理以生成所需格式, 然后经过 Word2Vec 方法^[7]和 TextCNN 模型^[8]处理进而得到向量化特征表示形式, 最后使用 Stacking 方法实现模型融合, 最终构建出 Web 攻击检测模型, 使用公开数据集和真实环境中的数据作为输入, 实现 Web 攻击检测。

本文的内容安排如下: 第 2 节给出 Web 攻击检测以及融合模型相关的国内外研究; 第 3 节详细介绍了本文的方法; 第 4 节通过实验对本文的方法进行了验证; 第 5 节总结全文并提出了未来工作。

2 相关工作

2.1 Web 攻击检测

Web 攻击检测研究工作在网络安全保障中起着很重要的作用, 一直以来都有许多 Web 攻击检测相关的研究。早期的研究是使用基于规则匹配和字段、会话过滤的 Web 应用程序防火墙, 针对 SQL 注入、跨站脚本攻击和应用层拒绝服务攻击等 Web 攻击的检测与阻断^[9]。这种方法很难发现未知攻击, 同时规则的制定也是一项很艰巨的工作。近几年, 随着机器学习、深度学习的蓬勃发展, 在网络安全领域得到了广泛的应用, 出现了很多使用机器学习、深度学习方法的 Web 攻击检测技术。

祝鹏程等人^[10]基于 TF-IDF 算法和随机森林搭建 Web 攻击检测模型, 利用 TF-IDF 算法构建词频矩阵, 将有效载荷转换成对应的特征表示形式, 以避免复杂的规则制定工作, 然后使用随机森林算法识别出正常流量与攻击流量。Mereani F A 等人^[11]使用了几种机器学习方法构建分类器对跨站脚本攻击进行检测, 在原始脚本文件的基础上提取了 59 个特征(包括结构特征和行为特征), 分别使用 SVM、KNN 和随机森林在公开数据集和真实环境下的数据进行对比实验, 以表明这些分类器的适用性。Yang W 等人^[12]利用有监督机器学习模型, 实现了对 HTTP 流量中 Webshell 攻击的检测, 使用 4-gram 方法处理 HTTP 请求, 并使用 SVM 分类器检查请求文件的合理性。Arumugam C 等人^[13]利用请求日志文件, 提取请求参数中的关键字作为特征, 通过逻辑回归分类算法, 实现对于在特定时间中从特定地点部署的 Web 应用程序上发生的 SQL 注入攻击进行预测。隐马尔可夫模型(Hidden Markov Models, HMM)是一种典型的机器学习模型, 由于其能够较好地捕获连续序列的依赖性, 对于字节序列(比如 HTTP 有效载荷)的分析是有效的^[14]。以上这些 Web 攻击检测方法都是使用的机器学习方法, 近几年深度学习得到了蓬勃的发展和广泛的应用, 越来越多的研究者倾向于使用深度

学习方法进行 Web 攻击检测的研究。

田俊峰等人^[15]提出了一种基于卷积神经网络的检测方法, 利用卷积神经网络将 Web 请求流量转换为灰度图, 使用空间金字塔池化的方法以处理不同大小的 Web 请求流量。刘新等人^[16]提出了面向物联网服务的 Web 攻击检测方案 IoTGuardEye, 该方案在对 HTTP 请求的文本序列进行特征抽取的基础上, 针对应用程序接口请求的报文格式相对固定的特点, 使用双向长短期记忆网络实现 Web 攻击检测。Zhang M 等人^[17]在处理 HTTP 请求的时候使用了卷积神经网络, 他们利用专门设计的卷积神经网络针对 HTTP 请求自动提取特征, 并通过 Softmax 将 HTTP 请求分类到正常或异常类。同样是使用卷积神经网络, Ito M 等人^[18]提出了 Character-level Convolutional Neural Network(CLCNN)的方法用于检测恶意请求, 这种方法首先将 HTTP 请求转换成包含 1000 个字符的字符串, 再经过嵌入层以及两次卷积和最大池化的处理, 最后通过全连接层得到一维输出, 区分正常与异常。Liu Z 等人^[19]提出了一种基于图卷积网络的跨站脚本负载检测模型, 将用户提交内容中的跨站脚本负载进行处理以得到一种图结构, 在训练样本数量较少的情况下依然有着很高的识别准确率。这些方法都是使用了一个模型或是几个模型进行对比选取最优模型, 可能会受限于所使用的模型, 模型融合策略则可以将多个子模型进行融合进而得到更强的检测模型。

2.2 模型融合

模型融合策略在入侵检测领域得到了广泛的应用。Aljawarneh S 等人^[20]提出了一种基于异常的入侵检测系统, 使用借助于信息增益的投票算法过滤数据, 进而结合每个基学习器的概率分布。M. Zaman 等人^[21]使用了 6 种常用的机器学习技术以进行网络流量异常检测, 并采取多数投票的方式将这 6 种方法进行集成, 虽然集成后的方法并没有取得最优的分类效果, 但作者肯定了集成技术的潜力, 值得进一步研究。N. T. Pham 等人^[22]提出了一种使用集成模型方法的改进的入侵检测系统, 其中的集成模型是基于 Bagging 和 Boosting 技术构建的, 使用基于树的算法作为基本分类器, 提高了准确率并降低了误报率。Tama B A 等人^[23]提出了一种基于异常的入侵检测方法, 该方法先后使用 Rotation Forest 方法和 Bagging 方法, 分阶段将这两种方法相结合, 在不同数据集上与其它方法进行比较, 取得了更优的分类性能。万子云等人^[24]在 MOOC 作弊行为的检测研究中提出了一种基于深度学习的混合模型, 通过将

卷积神经网络、双向门控循环单元和注意力机制进行融合, 与融合前的单一模型相比大大提升了检测性能。

Stacking 方法作为一种强大的模型融合方法, 在很多研究中都有所应用。Subudhi S 等人^[25]提出了一种检测数据库中的入侵活动的方法, 该方法分别使用三种集成学习方法(Bagging、Boosting、Stacking)将所使用的不同的分类器集成起来, 实验结果表明, 使用 Stacking 方法所得到的分类结果优于 Bagging 和 Boosting 两种方法。李勃等人^[26]提出一种基于密度聚类 and 集成学习的数据库异常检测方法, 该方法首先使用 OPTICS 聚类算法构建用户行为特征, 然后利用 Bagging、Boosting 和 Stacking 方法组合的集成学习模块对用户行为作进一步分析, 创建用户行为特征库, 进而实现数据库异常检测。Nkenyereye L 等人^[27]提出了一种基于 Stacking 的网络异常检测系统, 将不同参数配置的深度神经网络子模型进行融合, 通过将融合模型与不同的子模型进行检测性能对比, 进而强调基于 Stacking 集成模型的入侵检测系统的有效性。在标记分布学习中, 针对单个算法对预测模型性能产生影响的问题, 王一宾等人^[28]结合 Stacking 集成框架, 组合多个分类器对标记分布进行学习, 提出基于标记分布学习的异态集成学习算法(HELA-LDL), 在不同规模的标记分布数据集上均能产生良好效果。Zhang H 等人^[29]提出了一种多维特征融合与叠加集成机制用于搭建网络入侵检测系统, 综合考虑流量的时间、空间、负载等不同方面, 建立了多个基础特征数据集, 并对这些数据集进行叠加集成学习处理, 实现了一种有效的多维全局异常检测模型, 检测性能优于所采用的基本分类器和元分类器, 具有更强的鲁棒性。

总结相关工作, 可以发现多数 Web 攻击检测方法都是使用一个模型或是在几个模型中选取最优的模型, 这样的检测结果会受限于单个模型的性能, 而将多个模型进行融合, 则可以得到一个性能更优的检测模型, Stacking 方法则是一种有效且应用广泛的模型融合方法。

3 基于 Stacking 融合模型的 Web 攻击检测方法

为了解决单个模型性能受限的问题, 本文提出一种基于 Stacking 融合模型的 Web 攻击检测方法, 共包括两个部分: 第一部分在将原始的 Web 访问请求格式化处理后, 结合使用 Word2Vec 方法和 TextCNN 模型处理格式化后的 Web 访问请求, 并直

接使用原始 TextCNN 模型中的 Softmax 方法实现 Web 攻击检测, 以此作为 Baseline; 第二部分在原始 TextCNN 模型的基础上搭建不同子模型, 再基于 Stacking 方法设计模型融合方法, 将不同的子模型进行融合以得到更强的检测模型, 通过与 Baseline 之间检测性能对比体现出融合后的模型的稳定健壮性。整体方法的流程如图 1 所示。

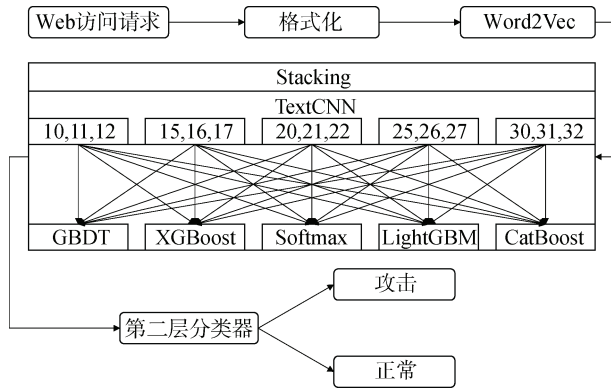


图 1 基于 Stacking 融合模型的 Web 攻击检测方法整体流程

Figure 1 The overall process of Web attack detection method based on Stacking fusion model

3.1 实现 Web 攻击检测

一条完整的基于 HTTP 的 Web 访问请求报文是由请求行、请求头部、有效载荷等部分组成, 在请求头部中包括请求方法、请求路径、协议版本等信息。为能够将其输入到 Word2Vec 方法和 TextCNN 模型中, 需对其进行格式化操作。首先将 HTTP 请求中的主机、路径以及具体的请求有效载荷提取出来。其中路径与请求数据之间用“?”进行连接; 然后对提取后的数据进行解码, 由于在 URL 中“+”表示空格, 故“+”都用空格替代; 最后利用“/”、“?”、“%”、“#”、“&”、“=”等特殊字符对解码后的数据进行分词。

针对所使用数据的格式化处理后数据格式对比如图 2 所示。

经过格式化处理后, 每一条 Web 访问请求变成了由不同词语组成的句子。使用 Word2Vec 方法对每个词语进行向量化表示, 将格式化后的请求转换成 n 行 k 列的矩阵 (n 为词语数量, k 为词向量维度), 并将其作为 TextCNN 模型的输入, 接下来使用 h 行 k 列矩阵形式的过滤器对其进行卷积操作 (h 为过滤器尺寸), 同样尺寸的过滤器执行 l 次卷积操作, 分别得到 l 个 $n-h+1$ 行 1 列的矩阵, 再使用最大池化方法得到 l 个 1 行 1 列的矩阵, 再将这些结果进行级联操作, 即拼接起来得到一维的向量化表示形式,

最后使用原始 TextCNN 模型中的 Softmax 方法分别得到将该请求判定为正常请求的概率值和将该请求判定为攻击请求的概率值, 进而实现了 Web 攻击检测, 并以此作为 Baseline, 与后面提出的融合模型进行对比。TextCNN 模型的执行流程如图 3 所示。

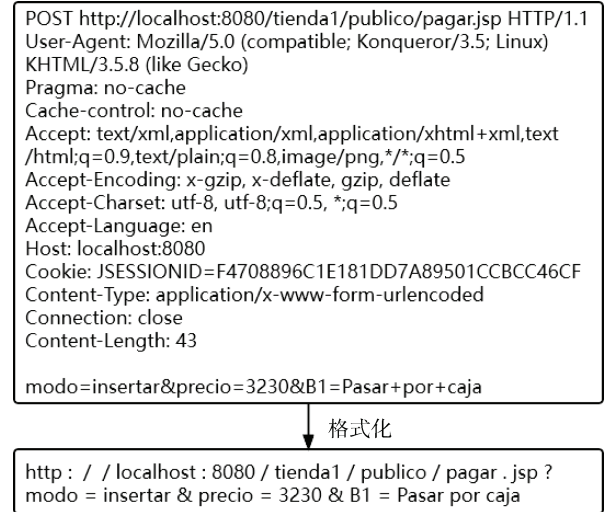


图 2 数据格式化前后对比

Figure 2 Comparison before and after data formatting

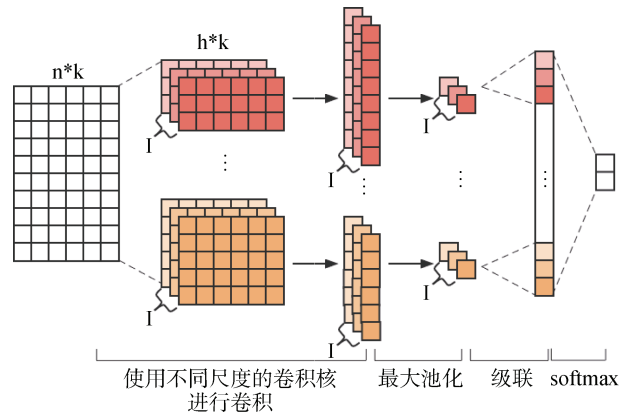


图 3 TextCNN 模型工作流程

Figure 3 TextCNN Model Workflow

3.2 利用 Stacking 方法实现多模型融合

Stacking 方法是一种广泛使用的模型融合策略, 包含两层模型, 利用第一层中的各子模型对训练集和测试集分别进行预测, 并将预测结果进行整合进而得到新的训练集和测试集, 再将新生成的训练集和测试集输入到第二层的检测器中进而得到最终的检测结果。

具体实现步骤如下:

第一步(处理原始的训练集和测试集): 将原始数据分成原始训练集 $Train_{raw}$ 和原始测试集 $Test_{raw}$, 使

用 k 折交叉方法处理原始训练集 $Train_{raw}$, 将原始训练集 $Train_{raw}$ 分成 k 等份。具体计算公式如下:

$$Train_{raw} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{\frac{n}{k}} \end{pmatrix} + \begin{pmatrix} x_{\frac{n}{k}+1} \\ x_{\frac{n}{k}+2} \\ \vdots \\ x_{2*\frac{n}{k}} \end{pmatrix} + \dots + \begin{pmatrix} x_{(i-1)*\frac{n}{k}+1} \\ x_{(i-1)*\frac{n}{k}+2} \\ \vdots \\ x_n \end{pmatrix}, i \in [1, k] \quad (1)$$

其中, x 为每个请求对应的特征数据, n 为请求总数, k 为 k 折交叉次数。

第二步(不同子模型训练预测): 取其中的一份数据作为测试集 $Test_i$, 其余为训练集 $Train_i$, 利用选取的模型 $Model$ 进行训练, 将训练之后的模型分别对测试集 $Test_i$ 以及原始测试集 $Test_{raw}$ 进行预测, 共执行 k 次。具体计算公式如下:

$$Train_i = \begin{pmatrix} x_1 \\ \vdots \\ x_{i*\frac{n}{k}} \\ x_{(i+1)*\frac{n}{k}+1} \\ \vdots \\ x_n \end{pmatrix}, i \in [2, k-1]; \quad (2)$$

$$Train_1 = \begin{pmatrix} x_{\frac{n}{k}+1} \\ x_{\frac{n}{k}+2} \\ \vdots \\ x_n \end{pmatrix}; Train_k = \begin{pmatrix} x_1 \\ \vdots \\ x_{(k-1)*\frac{n}{k}-1} \\ x_{(k-1)*\frac{n}{k}} \end{pmatrix}$$

$$Test_i = \begin{pmatrix} x_{i*\frac{n}{k}+1} \\ x_{i*\frac{n}{k}+2} \\ \vdots \\ x_{(i+1)*\frac{n}{k}} \end{pmatrix}, i \in [0, k-1] \quad (3)$$

$$Train_i \xrightarrow{Model_j \text{ 训练}} Test_i \xrightarrow{Model_j \text{ 预测}} P_Train_{ij} \quad (4)$$

$$Train_i \xrightarrow{Model_j \text{ 训练}} Test_{raw} \xrightarrow{Model_j \text{ 预测}} P_Test_{ij} \quad (5)$$

其中, $j \in [1, m]$, m 为第一层模型个数。

第三步(数据拼接得到新的训练集和测试集): 针对测试集 $Test_i$ 得到的 k 次预测结果按行拼接, 得到最终的预测结果 P_Train ; 再将针对 $Test_{raw}$ 的 k 次预测结果按列拼接取均值, 得到最终的预测结果 P_Test 。每个模型都如此处理, 将每个模型针对测试集 $Test_i$ 的最终预测结果按列拼接得到 n 行 m 列的数据作为新的训练集 $Train_{new}$; 再将每个模型针对原始测试集 $Test_{raw}$ 的最终预测结果按列拼接得到 n 行 m 列的数据作为新的测试集 $Test_{new}$ 。具体计算公式如下:

$$(P_Test_{1j} \dots P_Test_{kj}) \xrightarrow{\text{取均值}} P_Test_j \quad (6)$$

$$Train_{new} = \begin{pmatrix} P_Train_{11} & \dots & P_Train_{1m} \\ \vdots & \ddots & \vdots \\ P_Train_{n1} & \dots & P_Train_{nm} \end{pmatrix} \quad (7)$$

$$Test_{new} = (P_Test_1 \dots P_Test_m) \quad (8)$$

第四步(第二层模型训练预测), 将新的训练集输入到第二层模型中进行训练, 将训练之后的模型针对新的测试集进行预测, 所得到的最终预测结果即为最终检测结果。

上述实现过程如图 4 所示:

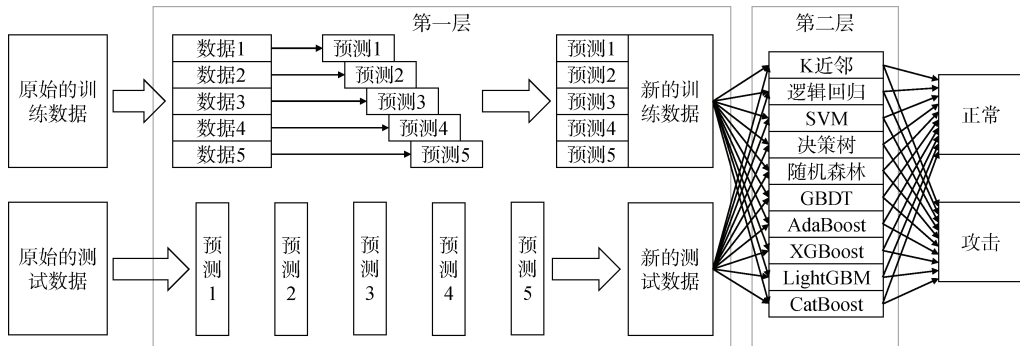


图 4 Stacking 方法实现过程

Figure 4 Stacking method implementation process

本文在原始的 TextCNN 模型基础上搭建不同的子模型, 再将不同的子模型基于 Stacking 方法进行融合, 进而构建出更强的检测模型, 具体方法如下:

在 k 折交叉方面, 根据同样使用 Stacking 方法的相关研究^[24-25], 选取 $k=5$ 。本文使用 $k=5$ 的 k 折交叉方法处理原始的训练集, k 折交叉的每轮迭代中分别使用不同尺寸的过滤器, 包括[10,11,12]、[15,16,17]、[20,21,22]、[25,26,27]、[30,31,32]; 原始的 TextCNN 模型使用的是 Softmax 方法进行分类, 本文分别将使用 Softmax、GBDT^[30]、XGBoost^[31]、LightGBM^[32]、CatBoost^[33-34]作为分类算法的 TextCNN 模型作为第一层子模型, 基于 Stacking 方法将第一层中的各子模型进行融合。针对融合之后的新的训练集和测试集, 分别使用 10 种分类算法进行 Web 攻击检测, 通过性能对比选取最优分类算法作为第二层模型分类器。所选的 10 种分类算法包括 K 近邻(KNN)、逻辑回归(LR)、支持向量机(SVM)、决策树(DT)、随机森林(RF)、GBDT、AdaBoost、XGBoost、LightGBM、CatBoost。其中, K 近邻、逻辑回归、支持向量机和决策树都是常见的机器学习分类算法^[25]; 随机森林是集成学习中基于 Bagging 方法的分类算法^[28-29]; GBDT、AdaBoost、XGBoost、LightGBM 和 CatBoost 是集成学习中基于 Boosting 方法的分类算法, 选取这几种分类算法与同样属于集成学习分类算法的随机森林进行对比。

基于 Stacking 方法的模型融合算法如下表所示。

表 1 基于 Stacking 方法的模型融合算法

Table 1 Model Fusion Algorithm Based on Stacking Method

算法: 模型融合算法
输入: 训练集 <i>Train</i> , 测试集 <i>Test</i> , k 折交叉次数 $k=5$, 不同过滤器大小的 TextCNN 模型包 <i>filters</i> , 第一层分类算法包 <i>al1</i> , 第二层分类算法包 <i>al2</i> 输出: 预测结果 <i>Predict</i>
<pre> 1 For i in $(0, k)$: 2 $test = Train[i]$ 3 $train = Train - Train[i]$ 4 $trainv = filters[i](train)$ 5 $testv = filters[i](test)$ 6 $Testv = filters[i](Test)$ 7 For j in $(0, len(al1))$: 8 $model = al1[j](trainv)$ 9 $pred[i][j] = model(testv)$ 10 $Pred[i][j] = model(Testv)$ 11 $pred[j] =$ 按行拼接 $pred[i][j]$ 12 $Pred[j] =$ 按列拼接 $Pred[i][j]$且每行取均值 13 End 14 End 15 $Train_new =$ 按列拼接 $pred[]$中各成员 16 $Test_new =$ 按列拼接 $Pred[]$中各成员 17 Return $Predict = al2(Train_new, Test_new)$ 18 End </pre>

4 实验

4.1 实现数据及实验环境

在实验数据方面, 本文首先使用公开数据集 HTTP DATASET CSIC 2010, 其中包含了上万条自动生成的 Web 访问请求(36000 条正常访问请求和超过 25000 条异常访问请求), 将这些正常访问请求与异常访问请求整合起来, 选择其中 50%作为训练集, 剩余 50%作为测试集。接下来使用来自中科院网站群系统 2021 年 6 月 23 日“cas.cn”这个站点的 Web 访问日志, 共计 2307126 条。其中包括已标记的正常 Web 访问日志以及异常 Web 访问日志, 即使用真实环境下的数据来验证模型的有效性与实用性。为得到更多的训练及测试数据且提高数据的复杂性, 将来自中科院网站群系统的 Web 访问日志与 CSIC 2010 数据集的 Web 访问请求融合起来, 将得到的全新数据集作为模型的输入。首先在原始的 TextCNN 模型(Baseline)中使用不同尺寸的过滤器以及不同的检测算法进行实验, 再使用融合后的模型进行实验, 并将其检测结果与 Baseline 对应的检测结果进行对比。

在实验环境方面, 本文的实验运行在搭载 2.3 GHz Intel Core i7、16GB 内存的 Win10 平台上, 采用 Python 语言作为编程语言。本文使用的模型性能评价指标包括准确率(accuracy, ACC)、精确率(precision, P)、召回率(recall, R)和 F1 值(f1-score)。具体公式如下:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (9)$$

$$precision = \frac{TP}{TP + FP} \quad (10)$$

$$recall = \frac{TP}{TP + FN} \quad (11)$$

$$f1-score = \frac{2 * precision * recall}{precision + recall} \quad (12)$$

其中, TP(True Positive)是预测正确的异常请求数量, TN(True Negative)是预测正确的正常请求数量, FP(False Positive)是预测错误的异常请求数量, 以及 FN(False Negative)是预测错误的正常请求数量。

4.2 实现结果

4.2.1 使用 TextCNN 模型实现 Web 攻击检测

本文参照前人的工作^[8], 搭建了一个原始 TextCNN 模型, 相关参数如表 2 所示。使用 gensim 库中的 skip-gram 方法进行词向量的生成, 为使得模型更加轻量化, 词向量维度为 8, 每一个请求的限定长度为 50。表 3 显示了使用所搭建的原始 TextCNN 模型针对 CSIC 2010 数据集进行 Web 攻击检测的检测性能,

表 2 Word2Vec 方法和 TextCNN 模型参数
Table 2 Word2Vec method and TextCNN model parameters

参数	值
词向量生成方法	gensim.models.Word2Vec.skip-gram()
词向量维度	8
Web 访问请求长度	50
相同尺寸过滤器数量	100
训练周期	10

表 3 不同尺寸过滤器的原始 TextCNN 模型检测性能 (%)

Table 3 Detection performance of the original TextCNN model with filters of different sizes (%)				
过滤器尺寸	accuracy	precision	recall	f1-score
[10,11,12]	94.70	99.75	79.76	88.64
[15,16,17]	95.43	99.78	82.55	90.35
[20,21,22]	95.54	99.83	82.96	90.61
[25,26,27]	95.52	99.86	82.83	90.55
[30,31,32]	95.66	99.76	83.45	90.88

其中各模型的过滤器尺寸分别为[10,11,12]、[15,16,17]、[20,21,22]、[25,26,27]、[30,31,32], 从表 3 中可以看出, 虽然各模型的精确率都很高, 达到 99%以上, 但召回率却只在 80%左右, 进而 F1 值也并不高, 只有 90%左右, 这样的检测性能并不稳定。接下来使用本文的方法实现模型融合, 进而构建 Web 攻击检测模型实现 Web 攻击检测, 并与所搭建的原始 TextCNN 模型(Baseline)进行对比, 以显示所提出方法的性能改善作用。

4.2.2 使用本文的方法

本文分别使用表 3 中的各个过滤器尺寸搭建原始 TextCNN 模型(使用 Softmax 方法)进行 Web 攻击检测, 并作为 Baseline。使用本文的方法将使用 5 种不同分类方法的子模型进行融合, 这 5 种分类方法分别是 Softmax、GBDT、XGBoost、LightGBM 和 CatBoost。表 4 显示了使用不同分类算法(k 近邻、逻辑回归、SVM、决策树、随机森林、GBDT、AdaBoost、XGBoost、LightGBM 和 CatBoost)作为第二层模型分类器对应的检测性能, 从表中可以看出, 当使用随机森林分类算法作为第二层分类器时取得了最优的检测性能, 因此将随机森林作为第二层分类器搭建 Web 攻击检测模型。

接下来分别使用 5 种配置不同尺寸过滤器的 TextCNN 模型(使用不同检测算法), 和使用本文的方法搭建的 Web 攻击检测模型, 针对 CSIC 2010 数据集进行 Web 攻击检测的各项检测性能指标如表 5 所

示。从表中可以看出, 在使用相同尺寸的过滤器的情 况下使用不同的检测算法得到了不同的检测性能,

表 4 使用不同分类算法作为第二层分类器的检测性能(%)

Table 4 Detection performance (%) of using different classification algorithms as second-layer classifiers				
第二层分类器	accuracy	precision	recall	f1-score
K 近邻	97.95	99.56	92.51	95.90
逻辑回归	97.71	99.75	91.41	95.40
SVM	97.93	99.79	92.21	95.85
决策树	97.25	96.57	92.70	94.59
随机森林	98.10	99.68	92.98	96.21
GBDT	98.07	99.64	92.91	96.16
AdaBoost	98.04	99.57	92.84	96.09
XGBoost	98.09	99.69	92.93	96.19
LightGBM	97.39	99.71	92.51	95.97
CatBoost	98.07	99.70	92.84	96.15

表 5 不同子模型和提出的融合模型的检测性能(%)
Table 5 Detection performance (%) of different sub-models and the proposed fusion model

检测模型(过滤器尺寸+检测算法)	accuracy	precision	recall	f1-score
[10,11,12]+Softmax	94.70	99.75	79.76	88.64
[10,11,12]+GBDT	96.06	98.36	86.26	91.92
[10,11,12]+XGBoost	97.76	99.74	91.59	95.49
[10,11,12]+LightGBM	97.83	99.83	91.78	95.63
[10,11,12]+CatBoost	97.72	99.78	91.42	95.42
[15,16,17]+Softmax	95.43	99.78	82.55	90.35
[15,16,17]+GBDT	96.54	98.70	87.83	92.95
[15,16,17]+XGBoost	97.78	99.79	91.62	95.53
[15,16,17]+LightGBM	97.72	99.84	91.36	95.42
[15,16,17]+CatBoost	97.61	99.80	90.98	95.19
[20,21,22]+Softmax	95.54	99.83	82.96	90.61
[20,21,22]+GBDT	96.30	99.46	86.18	90.61
[20,21,22]+XGBoost	97.59	99.81	90.88	95.13
[20,21,22]+LightGBM	97.69	99.85	91.22	95.34
[20,21,22]+CatBoost	97.57	99.81	90.81	95.10
[25,26,27]+Softmax	95.52	99.86	82.83	90.55
[25,26,27]+GBDT	96.31	99.12	86.55	92.41
[25,26,27]+XGBoost	97.54	99.85	90.65	95.03
[25,26,27]+LightGBM	97.53	99.89	90.58	95.00
[25,26,27]+CatBoost	97.31	99.74	89.85	94.54
[30,31,32]+Softmax	95.66	99.76	83.45	90.88
[30,31,32]+GBDT	96.05	99.37	85.31	91.81
[30,31,32]+XGBoost	97.51	99.78	90.60	94.97
[30,31,32]+LightGBM	97.59	99.81	90.87	95.13
[30,31,32]+CatBoost	97.32	99.84	89.82	94.57
本文的方法	98.10	99.68	92.98	96.21

在使用相同检测算法的情况下使用不同尺寸的过滤器也得到了不同的检测性能。其中,当过滤器尺寸相同时,使用 Softmax 方法(Baseline)对应的准确率、召回率和 F1 值均低于其他几种检测算法。在其余几种检测算法中,XGBoost 是在 GBDT 的基础上进行的修改,从检测性能对比中也可看出使用 XGBoost 对应的各项检测性能均优于使用 GBDT; LightGBM 和 CatBoost 是在 XGBoost 的基础上进行的修改,二者的修改更偏向于计算资源方面,从检测性能对比也可看出这三种算法的各项检测性能区别不大。当使用的检测算法相同时,不同尺寸的过滤器对应的检测性能相近。而使用本文的方法搭建的 Web 攻击检测模型与这些子模型相比在各项检测性能上有了明显的提升,精确率为 99.68%; 准确率为 98.10%,与 Baseline(使用 Softmax 方法)中的最高值(过滤器尺寸: [30,31,32], 95.66%)相比提升了 2.44%; 召回率为 92.98%,与 Baseline 中的最高值(过滤器尺寸: [30,31,32], 83.45%)相比提升了 9.53%; F1 值为 96.21,与 Baseline 中的最高值(过滤器尺寸: [30,31,32], 90.88%)相比提升了 5.33%, F1 值是精确率与召回率的调和平均数,由于本文提出的模型基于 Stacking 方法实现了模型融合,使得精确率和召回率更加均衡,进而 F1 值得到了提升。上述实验结果表明使用本文的方法进行 Web 攻击检测可以得到更加均衡稳定的检测结果。

图 5 显示了本文的方法与其它方法在 Web 攻击检测准确率上的对比,从图中可以看出,田俊峰等人^[15]提出的方法对应的检测准确率为 93.9%, Zhang M 等人^[17]提出的方法对应的检测准确率为 96.49%,而本文的方法对应的检测准确率为 98.1%,高于另两种方法的检测准确率。

4.2.3 使用真实环境下的数据

首先将来自中科院网站群系统的 Web 访问日志与 CSIC 2010 数据集中的 Web 访问请求进行融合进而得到更加复杂的数据,使用本文的方法搭建 Web 攻击检测模型进行 Web 攻击检测。所提出的格式化方法和特征提取方法对于两种数据都可以有效地提取向量化特征,也证明了所提出方法的适用性。表 6 显示了使用不同尺寸过滤器的原始 TextCNN 模型和使用本文的方法搭建的 Web 攻击检测模型实现 Web 攻击检测的各项检测性能指标。从表 6 中可以看出,使用不同尺寸过滤器的原始 TextCNN 模型(Baseline)的检测性能相近,本文的方法与 Baseline 相比却有了明显的提升。其中,精确率为 99.67%; 准确率为 96.53%,与 Baseline 中的最高值(过滤器尺寸:

[10,11,12], 94.28%)相比提升了 2.25%; 召回率为 93.38%,与 Baseline 中的最高值(过滤器尺寸: [10,11,12], 88.64%)相比提升了 4.74%; F1 值为 96.42%,与 Baseline 中的最高值(过滤器尺寸: [10,11,12], 93.95%)相比提升了 2.47%,上述实验结果表明使用本文的方法进行 Web 攻击检测可以得到更加均衡稳定的检测结果。

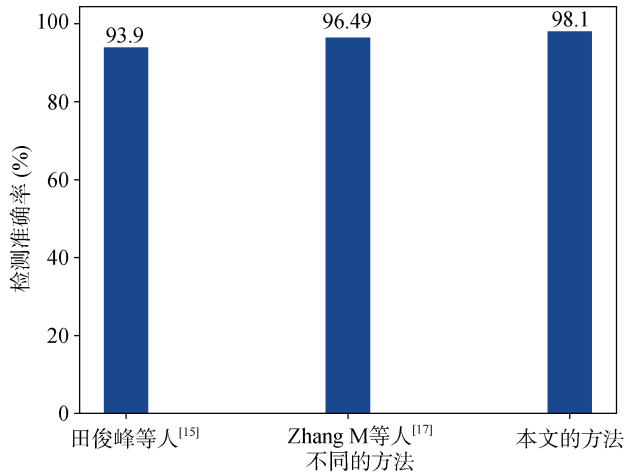


图 5 本文的方法与其他方法在 Web 攻击检测准确率上的对比

Figure 5 Comparison between the method proposed in this paper and other methods in the accuracy of Web attack detection

表 6 原始 TextCNN 模型与本文的方法搭建的 Web 攻击检测模型的检测性能(%)

Table 6 Detection performance (%) of the original TextCNN model and the web attack detection model built by the proposed method					
检测方法		accuracy	precision	recall	f1-score
配置不同尺寸过滤器	[10,11,12]	94.28	99.93	88.64	93.95
原始	[15,16,17]	94.08	99.96	88.22	93.72
TextCNN 模型(Baseline)	[20,21,22]	93.94	99.95	87.96	93.57
	[25,26,27]	93.70	99.95	87.48	93.30
	[30,31,32]	94.06	99.96	88.17	93.70
本文的方法		96.53	99.67	93.38	96.42

图 6 是配置不同尺寸过滤器的原始 Text-CNN 模型与使用本文的方法搭建的 Web 攻击检测模型对应的检测性能指标对比,从图中可以看出,准确率、召回率以及 F1 值对应的曲线都在“本文的方法”刻度位置有了明显的上升趋势,虽然精确率对应的曲线有所下降,但幅度却很小。这些检测结果表明了使用本文的方法搭建的 Web 攻击检测模型对于真实环境下的数据同样适用且具有更加稳定的检测性能。

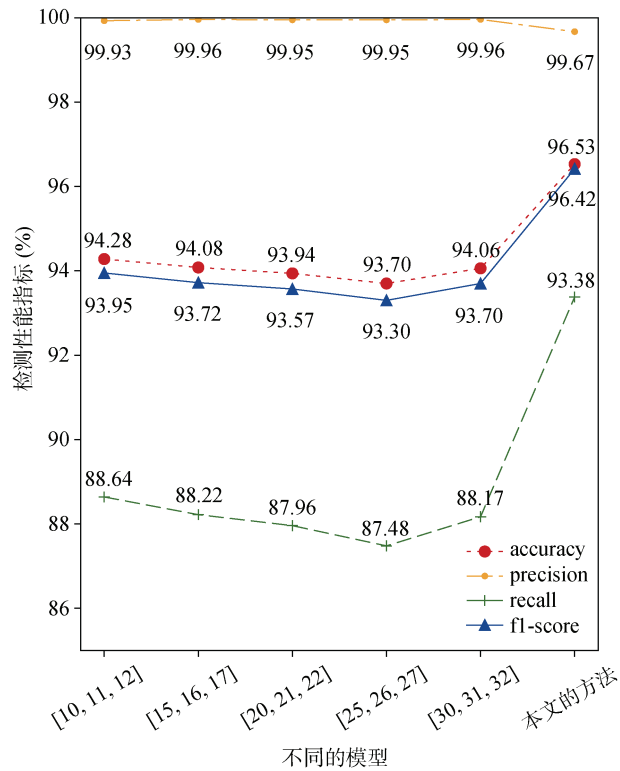


图 6 原始 TextCNN 模型与本文的方法搭建的 Web 攻击检测模型的检测性能指标对比

Figure 6 Comparison of detection performance indicators of the original TextCNN model and the web attack detection model built by the proposed method

最后将来自中科院网站群系统的 Web 访问日志作为输入, 使用本文的方法搭建 Web 攻击检测模型实现 Web 攻击检测, 表 7 显示了各项检测性能指标。从表中可以看出, 各项检测性能指标都达到了 99.9% 以上, 其中, 精确率达到了 100%, 证明检测结果中并没有出现将正常请求误报为攻击请求的情况, 出现这种情况可能是因为本文仅使用了一天的 Web 访问日志, 样本量较小, 在后续工作中需要采集更多数据进行验证。本次实验的检测结果表明了该模型对于真实环境下的数据的有效性。

表 7 本文的方法搭建的 Web 攻击检测模型在真实环境中的检测性能(%)

Table 7 Detection performance (%) of the web attack detection model built by the proposed method in the real environment

检测方法	accuracy	precision	recall	f1-score
本文的方法	99.97	100.0	99.91	99.96

4.3 讨论

本文中为检测 Web 攻击使用的是 Web 访问请求, 但在实际的 Web 访问过程中, 不仅仅存在 Web 访问

请求, 服务器在收到 Web 访问请求的同时, 也会发出应答报文, 有些 Web 攻击仅仅从 Web 访问请求中并不能发现任何存在异常的信息, 而是需要分析服务器给出的应答报文才能检测出异常。比如敏感信息泄露, 这种攻击的 Web 访问请求仅仅是一个 get 文件的操作, 与正常访问文件的请求是相同的, 单从 Web 访问请求中也看不出被访问的文件是否存在敏感信息, 进而不能检测出这种类型的攻击。这时应答报文就派上了用场, 在应答报文中会返回被访问文件中的内容, 这样就可以判断被访问的文件中是否存在敏感信息, 进而可以发现异常并检测出此类攻击。随着 Web 攻击越来越多样化, 结合考虑 Web 访问请求与应答报文将会精确地检测出更多种类的攻击。与此同时, 每一次 Web 访问都对应着具体的访问行文, 比如访问视频网站、参与秒杀活动等等, 将每一次 Web 访问的行为提取出来作为一项特征, 明确哪些行为是正常行为, 哪些行为是异常行为, 这样可以提升对于 Web 攻击的检测精度。

在使用 Stacking 方法实现模型融合搭建 Web 攻击检测模型的过程中, 我们发现其中有很多地方可以实现并行处理, 比如 k 折交叉的每一轮迭代过程、不同子模型的训练与测试过程等, 将处理之后的结果再以按行或按列的形式进行拼接, 最终得到新的训练集和测试集。这样的并行处理会大大提升整个 Web 攻击检测过程的运行效率。

5 总结与未来工作

在本文中, 我们提出了一种基于 Stacking 融合模型的检测方法, 将其用于搭建 Web 攻击检测模型以实现 Web 攻击检测。通过与使用原始 TextCNN 模型的 Web 攻击检测模型相对比, 所搭建的 Web 攻击检测模型在检测性能上有着更加稳定的表现。使用真实环境下的数据进行 Web 攻击检测依然取得了高精度且稳定的检测结果, 进而验证了所搭建的 Web 攻击检测模型的有效性与实用性。

正如在第四节中讨论的内容, 在未来工作中, 我们将结合使用 Web 访问请求与应答报文, 以检测出更多种类的 Web 攻击; 为考虑到更加全面的信息, 提取用户访问行为, 进而形成一个更加完善的体系; 同时将所提出的检测模型进行优化, 以提升检测过程中的运行效率。

参考文献

[1] Jing J W, Long C, Li C. New Trend of Cyberspace Security Technology[J]. *Frontiers of Data & Computing*, 2021, 3(3): 1-8.

- (荆继武, 龙春, 李畅. 网络安全技术的新趋势探讨[J]. *数据与计算发展前沿*, 2021, 3(3): 1-8.)
- [2] Watson D. Web Application Attacks[J]. *Network Security*, 2007, 2007(10): 10-14.
- [3] Danielsson H, Rönnerberg, Jerker, Levén, Anna, et al. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content[J]. *Faculty of Arts & Sciences*, 2014, 30(4):595-599.
- [4] Freund Y, Schapire R E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting[J]. *Journal of Computer and System Sciences*, 1997, 55(1): 119-139.
- [5] Breiman L. Bagging Predictors[J]. *Machine Learning*, 1996, 24(2): 123-140.
- [6] Wolpert D H. Stacked Generalization[J]. *Neural Networks*, 1992, 5(2): 241-259.
- [7] Mikolov T, Chen K, Corrado G, et al. Efficient Estimation of Word Representations in Vector Space[EB/OL]. 2013: arXiv: 1301.3781. <https://arxiv.org/abs/1301.3781.pdf>.
- [8] Kim Y. Convolutional Neural Networks for Sentence Classification[EB/OL]. 2014: arXiv: 1408.5882. <https://arxiv.org/abs/1408.5882.pdf>.
- [9] Wang Y, Lu S N. Design and Implementation of Web Application Firewall[J]. *Information Security and Communications Privacy*, 2011, 9(5): 104-106.
(王宇, 陆松年. Web 应用防火墙的设计与实现[J]. *信息安全与通信保密*, 2011, 9(5): 104-106.)
- [10] Zhu P C, Fang Y, Huang C, et al. Research on Web Attack Traffic Detection Based on TF-IDF and Random Forest Algorithm[J]. *Journal of Information Security Research*, 2018, 4(11): 1040-1045.
(祝鹏程, 方勇, 黄诚, 等. 基于 TF-IDF 和随机森林算法的 Web 攻击流量检测方法研究[J]. *信息安全研究*, 2018, 4(11): 1040-1045.)
- [11] Mereani F A, Howe J M. Detecting Cross-Site Scripting Attacks Using Machine Learning[C]. *International Conference on Advanced Machine Learning Technologies and Applications*, 2018: 200-210.
- [12] Yang W C, Sun B, Cui B J. A Webshell Detection Technology Based on HTTP Traffic Analysis[C]. *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2019: 336-342.
- [13] Arumugam C, Dwarakanathan V B, Gnanamary S, et al. Prediction of SQL Injection Attacks in Web Applications[C]. *International Conference on Computational Science and Its Applications*, 2019: 496-505.
- [14] Jian S J, Lu Z G, Du D, et al. Overview of Network Intrusion Detection Technology[J]. *Journal of Cyber Security*, 2020, 5(4): 96-122.
(蹇诗婕, 卢志刚, 牡丹, 等. 网络入侵检测技术综述[J]. *信息安全学报*, 2020, 5(4): 96-122.)
- [15] Tian J F, Shi W. Web Attack Detection Method Based on Convolutional Neural Network[J]. *Journal of Chinese Computer Systems*, 2019, 40(3): 584-588.
(田俊峰, 石伟. 一种基于卷积神经网络的 Web 攻击检测方法[J]. *小型微型计算机系统*, 2019, 40(3): 584-588.)
- [16] Liu X, Huang Y Y, Liu Z A, et al. IoTGuardEye: A Web Attack Detection Method for IoT Services[J]. *Computer Science*, 2021, 48(2): 324-329.
(刘新, 黄缘缘, 刘子昂, 等. IoTGuardEye: 一种面向物联网服务的 Web 攻击检测方法[J]. *计算机科学*, 2021, 48(2): 324-329.)
- [17] Zhang M, Xu B Y, Bai S, et al. A Deep Learning Method to Detect Web Attacks Using a Specially Designed CNN[C]. *International Conference on Neural Information Processing*, 2017: 828-836.
- [18] Ito M, Iyatomi H. Web Application Firewall Using Character-Level Convolutional Neural Network[C]. *2018 IEEE 14th International Colloquium on Signal Processing & Its Applications*, 2018: 103-106.
- [19] Liu Z L, Fang Y, Huang C, et al. GraphXSS: An Efficient XSS Payload Detection Approach Based on Graph Convolutional Network[J]. *Computers & Security*, 2022, 114: 102597.
- [20] Aljawarneh S, Aldwairi M, Yassein M B. Anomaly-Based Intrusion Detection System through Feature Selection Analysis and Building Hybrid Efficient Model[J]. *Journal of Computational Science*, 2018, 25: 152-160.
- [21] Zaman M, Lung C H. Evaluation of Machine Learning Techniques for Network Intrusion Detection[C]. *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, 2018: 1-5.
- [22] Pham N T, Foo E, Suriadi S, et al. Improving Performance of Intrusion Detection System Using Ensemble Methods and Feature Selection[C]. *The Australasian Computer Science Week Multiconference*, 2018: 1-6.
- [23] Tama B A, Comuzzi M, Rhee K H. TSE-IDS: A Two-Stage Classifier Ensemble for Intelligent Anomaly-Based Intrusion Detection System[J]. *IEEE Access*, 2019, 7: 94497-94507.
- [24] Wan Z Y, Chen S W, Qin B, et al. Research on MOOC Cheating Detection Based on Deep Learning[J]. *Journal of Cyber Security*, 2021, 6(1): 32-39.
(万子云, 陈世伟, 秦斌, 等. 基于深度学习的 MOOC 作弊行为检测研究[J]. *信息安全学报*, 2021, 6(1): 32-39.)
- [25] Subudhi S, Panigrahi S. Application of OPTICS and Ensemble Learning for Database Intrusion Detection[J]. *Journal of King Saud University - Computer and Information Sciences*, 2022, 34(3): 972-981.
- [26] Li B, Shou Z, Liu X Y, et al. Database Anomaly Detection Based on Density Clustering and Ensemble Learning[J]. *Journal of Chinese Computer Systems*, 2021, 42(3): 666-672.
(李勃, 寿增, 刘昕禹, 等. 融合密度聚类与集成学习的数据库异常检测[J]. *小型微型计算机系统*, 2021, 42(3): 666-672.)
- [27] Adhi Tama B, Nkenyereye L, Lim S. A Stacking-Based Deep Neural Network Approach for Effective Network Anomaly Detection[J]. *Computers, Materials & Continua*, 2021, 66(2): 2217-2227.
- [28] Wang Y B, Tian W Q, Cheng Y S. Heterogeneous Ensemble Learning Algorithm Based on Label Distribution Learning[J]. *Pattern Recognition and Artificial Intelligence*, 2019, 32(10): 945-954.
(王一宾, 田文泉, 程玉胜. 基于标记分布学习的异态集成学习算法[J]. *模式识别与人工智能*, 2019, 32(10): 945-954.)
- [29] Zhang H, Li J L, Liu X M, et al. Multi-Dimensional Feature Fusion and Stacking Ensemble Mechanism for Network Intrusion Detection[J]. *Future Generation Computer Systems*, 2021, 122: 130-143.
- [30] Friedman J, Hastie T, Tibshirani R. Additive Logistic Regression:

A Statistical View of Boosting (with Discussion and a Rejoinder by the Authors)[J]. *The Annals of Statistics*, 2000, 28(2): 337-374.

- [31] Chen T Q, Guestrin C. XGBoost: A Scalable Tree Boosting System[EB/OL]. 2016: arXiv: 1603.02754. <https://arxiv.org/abs/1603.02754.pdf>.
- [32] Ke G L, Meng Q, Finley T, et al. LightGBM: A Highly Efficient Gradient Boosting Decision Tree[C]. *The 31st International Con-*

ference on Neural Information Processing Systems, 2017: 3149-3157.

- [33] Prokhorenkova L, Gusev G, Vorobev A, et al. CatBoost: Unbiased Boosting with Categorical Features[EB/OL]. 2017: arXiv: 1706.09516. <https://arxiv.org/abs/1706.09516.pdf>.
- [34] Dorogush A V, Ershov V, Gulin A. CatBoost: Gradient Boosting with Categorical Features Support[EB/OL]. 2018: arXiv: 1810.11363. <https://arxiv.org/abs/1810.11363.pdf>.



万巍 于 2015 年在中国科学院计算机网络信息中心计算机软件与理论专业获得博士学位, CCF 会员。现任中国科学院计算机网络信息中心高级工程师。研究领域为网络空间安全。研究兴趣包括: 基于人工智能的网络安全异常检测、安全大数据分析等。Email: wanwei@cnic.cn



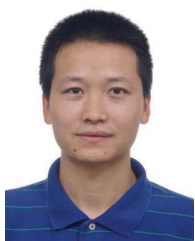
石鑫 于 2014 年在西南大学自动化专业获得学士学位。现在中国科学院大学计算机技术专业攻读硕士学位。研究领域为网络安全态势感知、安全大数据分析。研究兴趣包括: 基于人工智能的网络流量分析、入侵检测。Email: shixin@cnic.cn



魏金侠 于 2017 年在北京邮电大学密码学专业获得博士学位。现任中国科学院计算机网络信息中心安全部高级工程师。研究领域为网络空间安全、网络安全身份认证、同态加密多方安全计算等。研究兴趣包括: 基于人工智能的恶意流量检测、基于同态加密的隐私保护等等。Email: weijinxia@cnic.cn



李畅 于 2007 年在巴黎高等商业学院获得经济学硕士学位, 现就职中国信息通信研究院政务服务中心, 主要研究方向为国际互联网治理、互联网基础资源政策管理、互联网安全等。Email: lichang@caict.ac.cn



龙春 于 2015 年在中国科学院大学获得计算机软件与理论专业博士学位, CCF 会员。现任中国科学院计算机网络信息中心网络空间安全技术与应用发展部主任, 正高级工程师。研究领域为网络空间安全。研究兴趣包括: 智能动态网络安全保障、安全大数据挖掘与深度分析、云计算与移动互联网安全事件管控、多源安全数据融合与复合攻击预测等。Email: anquanip@cnic.cn