

基于 CBAM-GNN 高特征向量提取的 工业软件安全漏洞检测

刘勇辰[†]

(上海雷昶科技有限公司, 上海 200443)

摘要: 现有深度学习技术在漏洞检测任务中具有一定缺陷, 例如常常因未充分保留代码的完整信息而导致检测效果不佳。基于此, 提出了一种基于改进图神经网络的漏洞检测方法, 旨在全面提升漏洞检测的效率和精度。首先将源代码数据转换成文本信息, 然后在传统的图神经网络中加入卷积注意力机制提高特征向量的敏感度, 最终搭建了漏洞检测模型并对其性能进行了测试。研究结果表明, 所提出的漏洞检测模型在训练集和验证集中分别能取得 98.25% 和 98.69% 的平均检测精度, 其均方误差低至 1.15。在实际应用中, 该模型的误报率和漏报率分别低至 0.01% 和 0.03%。由此可见所搭建的漏洞检测模型具有较好的检测性能, 能够有效完成漏洞检测任务。

关键词: 漏洞数据; GNN; 图结构; 检测; 软件安全

中图分类号: TP391

文献标识码: A

Industrial Software Security Vulnerability Detection Based on CBAM-GNN High Feature Vector Extraction

LIU Yongchen[†]

(Shanghai Leateck Co., Ltd., Shanghai 200443, China)

Abstract: Existing deep learning techniques have certain shortcomings in the vulnerability detection task, such as often poor detection results due to insufficient retention of the complete information of the code. Based on this, the study proposes a vulnerability detection method based on improved graph neural networks, aiming to comprehensively improve the efficiency and accuracy of vulnerability detection. Firstly, the source code data is converted into textual information, and then a convolutional attention mechanism is added to the traditional graph neural network to improve the sensitivity of the feature vectors, and finally the vulnerability detection model is constructed and its performance is tested. The results show that the proposed vulnerability detection model can achieve an average detection accuracy of 98.25% and 98.69% in the training set and validation set, respectively, and its mean square error is as low as 1.15. In practical applications, the false alarm rate and the leakage rate of the model are as low as 0.01% and 0.03%, respectively. This shows that the constructed vulnerability detection model has good detection performance and can effectively accomplish the vulnerability detection task.

Key words: vulnerability data; GNN; graph structure; detection; software security

随着信息技术的飞速发展, 工业软件安全漏洞已成为威胁工业系统安全运行的重大隐患^[1,2]。

传统的安全漏洞检测方法多依赖于专家经验或基于静态和动态的分析技术, 但在面对日益增长的软

件复杂性和漏洞检测效率要求时,这些方法显得力不从心,检测效率和检测精度都处于较低水平^[3]。图神经网络(Graph Neural Network, GNN)作为一种能够有效处理图结构数据的深度学习技术,近年来已经在多个领域显示出了巨大的潜力^[4,5]。利用图神经网络所搭建的安全漏洞检测模型能够捕获软件系统的结构特征和行为模式,通过学习这些特征和模式,更准确地识别出潜在的安全漏洞,从而提高漏洞检测的准确性和效率^[6,7]。鉴于此,本研究旨在对传统的 GNN 进行优化,并利用优化后的 GNN 搭建一个全新的工业软件安全漏洞检测模型。优化后的 GNN 模型旨在利用图神经网络处理复杂的软件结构和数据流,并加入卷积注意力机制(Convolutional Block Attention Mechanisms, CBAM)以提高漏洞检测的准确性和效率,从而更好地保障工业软件系统的安全。

1 基于改进图神经网络的工业软件安全漏洞检测研究

1.1 融合卷积注意力机制的图神经网络算法优化设计

在复杂的工业软件系统中,直接应用 GNN 往往难以捕捉到细粒度的特征变化和节点间的动态依赖关系,从而影响漏洞检测的效果^[8,9]。为了解决这一问题,此次研究通过融合卷积层来提取节点及其邻域的局部特征,并引入 CBAM 来自适应地调整各个节点及其特征在漏洞检测任务中的重要性,从而实现对安全漏洞更为敏感和精确的识别。

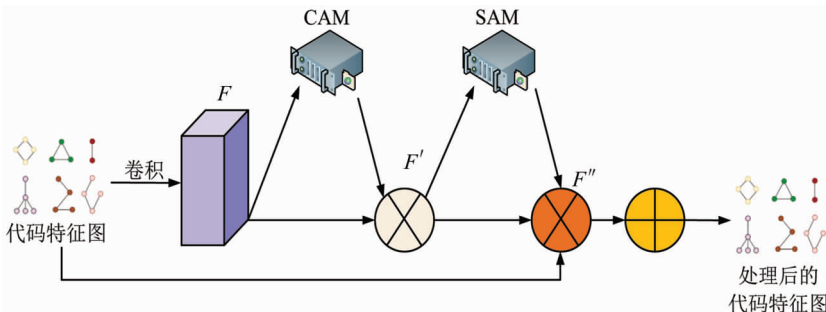


图 1 CBAM 结构图

图 1 中,CBAM 作为一个高效的注意力机制融合模块,主要包含两个部分,分别是通道注意力机制(Channel Attention Mechanism, CAM)和空间注意力机制(Spatial Attention Mechanism, SAM)^[11,12]。在 CAM 部分,CBAM 学习不同通道的特征并对特征进行加权,使得模型能够更关注信

假设图神经网络中的某个节点为 V ,由于图神经网络结构中的每个节点的特征是由自身特征和其他节点特征共同决定的,因此可以得到 V 的特征表达推导过程如式(1)~式(4)所示。首先得到 V 的状态向量表达式如式(1)所示^[10]。

$$h_v = f(\mathbf{X}_v, \mathbf{X}_{co[v]}, \mathbf{h}_{ne[v]}, \mathbf{X}_{ne[v]}) \quad (1)$$

式中, h_v 表示 V 的状态, f 表示图神经网络中的节点更新函数, $\mathbf{X}_v$ 、 $\mathbf{X}_{co[v]}$ 、 $\mathbf{h}_{ne[v]}$ 、 $\mathbf{X}_{ne[v]}$ 分别表示 V 的特征向量、与 V 相关的边特征向量、与 V 相邻的节点的状态向量、与 V 相邻的节点的特征向量。通过式(1)进一步得到 V 的输出向量表达式如式(2)所示。

$$\mathbf{O}_v = g(h_v, \mathbf{X}_v) \quad (2)$$

式中, $\mathbf{O}_v$ 表示 V 的输出向量, g 表示局部输出函数。令图神经网络中所有节点的特征向量为 $\mathbf{X}_N$,并且令所有节点的状态向量为 $\mathbf{H}$,得到 $\mathbf{H}$ 的表达式如式(3)所示。

$$\mathbf{H} = F(\mathbf{H}, \mathbf{X}) \quad (3)$$

式(3)中, $F(\cdot)$ 表示图神经网络的全局更新函数, $\mathbf{X}$ 表示图神经网络所有边的特征向量。图神经网络的全局输出函数如式(4)所示。

$$\mathbf{O} = G(\mathbf{H}, \mathbf{X}_N) \quad (4)$$

式中, $\mathbf{O}$ 表示图神经网络的全局输出函数, $G(\cdot)$ 表示所有节点输出函数和更新函数的堆叠。

为了避免图神经网络在增加网络深度的过程中出现节点过平滑现象,此次研究引入 CBAM 进行深度全局特征信息的有效提取,从而确保所有节点信息都能被检测到,并进一步提高模型最终的漏洞识别能力。CBAM 的结构图如图 1 所示。

息丰富的通道。而在 SAM 部分,CBAM 专注挑选出特征图像中的关键特征,进而指导网络聚焦于这些关键区域并提取相关特征信息。这种结合通道和空间的双重注意力机制,使 CBAM 能够在细化特征提取的同时,进一步提高网络处理图像数据的效率和准确性。CBAM 的运算公式如式(5)所

示^[13]。

$$F' = M_c(F) \otimes F \quad (5)$$

式中, F 为经过上一层卷积操作得到的特征图, $\otimes$ 为特征之间的乘法运算符号, M_c 表示 CAM, F' 表示经过通过 CAM 处理后的新特征图。经过 SAM 处理后的特征图的表达式如式(6)所示。

$$F'' = M_s(F') \otimes F' \quad (6)$$

式中, M_s 表示 SAM, F' 表示将 F' 经过 SAM 处理所得到的注意力权重值。将最终的注意力权重值与原有的特征图进行加权求和,再作为新的特征图输出。CAM 与 SAM 的处理流程如图 2 所示。

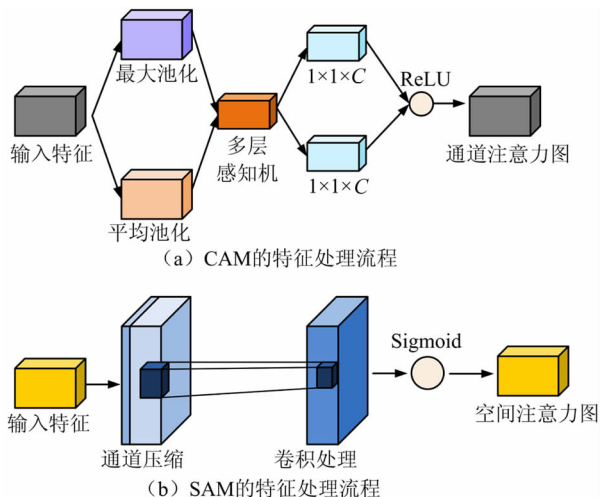


图 2 CAM 和 SAM 的特征处理流程图

图 2(a)为 CAM 的特征处理流程,主要分成三个步骤,分别是全局池化、多层感知以及激活应用。在 CAM 中,首先对输入特征图进行全局平均池化和全局最大池化,生成两个不同的通道描述信息。接下来这两个通道描述信息分别通过一个共享权重的多层感知器进行信息的加工处理。最后,两个多层感知器的输出信息会进行特征元素相加处理,并通过一个 Sigmoid 激活函数生成通道注意力图。图 2(b)为 SAM 的特征处理流程,主要分成了通道压缩、卷积处理和激活应用三步。首先在通道轴上对特征图进行压缩,然后通过平均池化和最大池化来产生两个二维特征图,接着将这两个特征图合并,并通过一个 3×3 的卷积层进行处理,以产生一个单通道的空间注意力图。最后,通过 Sigmoid 激活函数处理这个单通道的注意力图并生成最终的空间注意力图。通道注意力图的生成公式如式(7)所示。

$$MC(F) = \sigma \text{MLP}(\text{AvgPool}(F)) + \sigma \text{MLP}(\text{MaxPool}(F)) \quad (7)$$

式中, $MC(F)$ 为通道注意力图, σ 表示激活函数 Relu, MLP 表示多层感知机, $\text{AvgPool}(F)$ 表示平均池化, $\text{MaxPool}(F)$ 表示最大池化。空间注意力图的生成公式如式(8)所示。

$$MS(F) = \theta f^{3 \times 3} [\text{AvgPool}(F); \text{MaxPool}(F)] \quad (8)$$

式中, $MS(F)$ 表示空间注意力图。 θ 表示激活函数 Sigmoid。 $f^{3 \times 3}$ 表示滤波器大小为 3×3 的卷积运算。

1.2 基于改进图神经网络算法的漏洞检测模型构建

为了更好地检测工业软件中的安全漏洞,此次研究将各类漏洞代码数据看作文本分类问题,采用融合卷积注意力机制的图神经网络搭建最终的漏洞检测模型^[14,15]。将最终所搭建的漏洞检测模型记作卷积注意力机制改进下的图神经网络(Convolutional Block Attention Mechanisms-Graph Neural Network, CBAM-GNN), CBAM-GNN 的检测流程结构图如图 3 所示。

图 3 所示为 CBAM-GNN 模型的检测流程。整个检测流程可以分成七个步骤,分别是输入代码数据、源代码数据预处理、生成代码图结构数据集、特征提取、特征分类、漏洞检测网络搭建以及漏洞检测。首先输入源代码数据并对其进行一系列预处理,确保初始源代码数据中的所有冗余信息能够有效去除。接着对处理好的数据进行清洗和切片,生成一系列代码图结构数据集。将生成好的代码图结构数据集作为 CBAM-GNN 的输入,利用 CBAM 中的 CAM 和 SAM 生成输入图数据的特征向量,并将该特征向量作为最终检测模型中分类器的输入。为了确保特征向量中的所有维度特征都能得到有效的识别和检测,此次研究利用求和池化和平均池化进行降维处理,最后将高维度的特征向量全部转化成低维度向量并利用 Softmax 进行分类。利用 CBAM-GNN 网络完成特征向量的检测后,最终模型会输出两种不同的检测结果,分别记作 Y 和 N。其中, N 表示所检测的输入代码图结构数据集不包含安全漏洞,而 Y 则表示输入的代码图结构数据集包含安全漏洞。

在利用 CBAM-GNN 安全漏洞检测模型进行漏洞检测时,令 C_0 和 C_1 分别表示两个不同的常

量,令 v_0 和 v_1 分别表示有漏洞和没有漏洞的最终样本数量,得到所有数据样本的总数量的表达式如式(9)所示。

$$N = v_0 + v_1$$

(9)

式中, N 表示所有数据样本的总数量。将特征向量经过分类器分类之后的结果作为最终输出,其计算公式如式(10)所示。

$$G_0 = \frac{e^{v_i}}{\sum_j e^{v_j}}$$

(10)

式中, G_0 表示分类后的特征向量, e^{v_i} 和 $\sum_j e^{v_j}$ 分别表示当前的样本指数和所有样本的指数之和。当 e^{v_i} 满足漏洞模式,则能够说明当前的样本中含有漏洞代码,反之则无。当对代码图结构

数据集中的所有样本都进行检测之后,还需要更新 C_0 和 C_1 ,从而确保能够得到所有的评价指标。由于 CBAM-GNN 网络中的所有输入节点都是等价的,因此每个节点的更新并不会影响最终的漏洞检测结果,反而能够遍历全局,确保所有节点都得到检测。在实际的检测任务中,由于一些代码语句和漏洞的关系并不明显,因此为了使 CBAM-GNN 网络能够更好捕捉到一些敏感信息,研究加入了 CAM 和 SAM 对主网络进行优化。最后采用最小交叉熵函数来完成模型的训练,得到最终训练好的漏洞检测模型。利用训练好的漏洞检测模型来检测不同的实际漏洞代码数据集,并得到不同的漏洞检测结果。

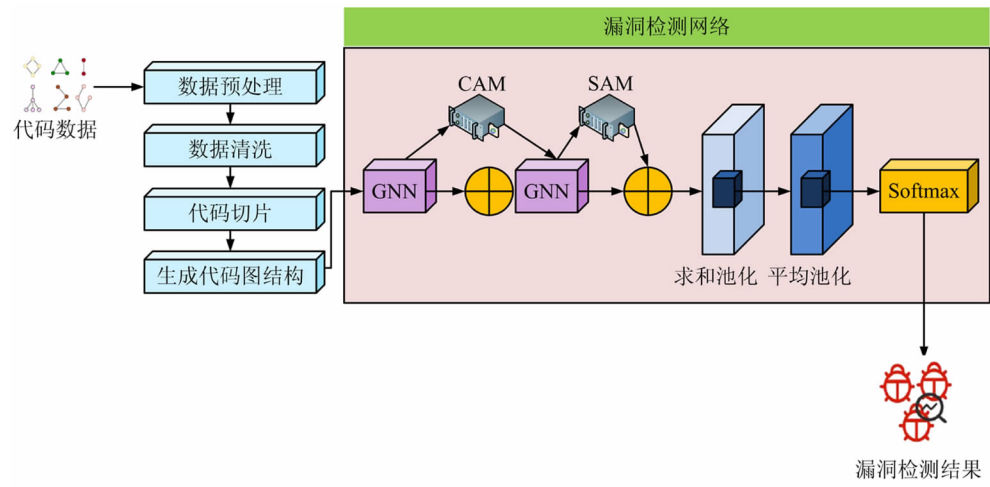


图 3 CBAM-GNN 的检测流程结构图

2 基于 CBAM-GNN 的工业软件安全漏洞检测结果分析

为了验证 CBAM-GNN 算法的有效性,研究采用了广泛认可的公开测试数据源,即 VulnCodeDB 和 SARD 数据集,进行工业软件安全漏洞检测的数据集构建。VulnCodeDB 集合了多种来源的漏洞代码,提供了丰富的注释信息。而 SARD 则被设计用于软件安全性测试,包含了大量带有和不带有漏洞的代码样例。通过对这些开源数据的深入分析和预处理,最终得到了一个综合性的代码图结构数据集。该数据集涵盖了 1200 种不同的代码类型,将数据集按照 9 : 1 的比例划分为训练数据集和验证数据集,以支持模型的训练和性能验证。为了避免实验误差,研究采用同一台设备展开各项实验,实验环境和基本网络参数设置表如表 1 所示。

表 1 实验环境设置表

配置	参数
CPU	Intel(R) Core i7-9700K @ 3.6GHz
GPU	NVIDIA GeForce GTX 2080Ti 11G * 2
内存	64GB
操作系统	Windows 10
开发程序	Python 3.2
开发框架	Pytorch 1.9

表 1 中给出了此次实验的环境配置和网络参数。另外选择三种常见的目标检测算法作为对比算法来验证 CBAM-GNN 的性能,分别为单次多框检测算法(Single Shot MultiBox Detector, SSD)、掩码区域卷积神经网络(Region-based Convolutional Neural Network, Mask R-CNN)、多尺度共享和独立特征网络(Multi-scale Shared and Inde-

pendent Feature Network, MSSIF-Net), 得到四种检测算法在训练数据集中的总损失值和回归损失值, 如图 4 所示。

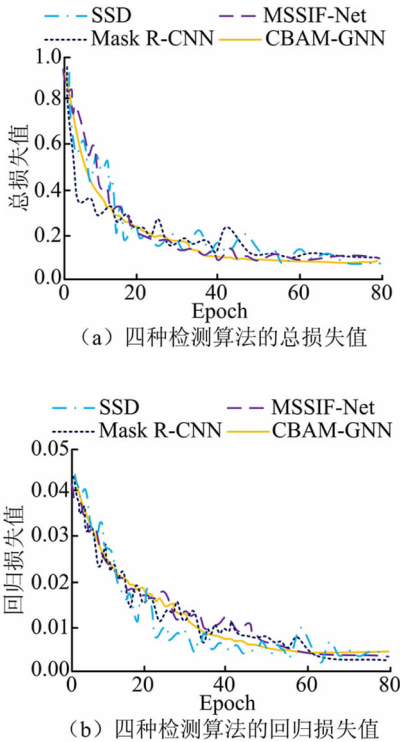


图 4 不同检测算法的总损失和回归损失

图 4(a)和图 4(b)所示分别为 SSD、Mask R-CNN、MSSIF-Net、CBAM-GNN 四种检测算法的总损失值曲线和回归损失值曲线。由图 4(a)可知,随着迭代周期的增加,四种算法的总损失值曲线都会逐渐趋于稳定。在四种算法中, CBAM-GNN 的总损失值曲线最先达到稳定状态,只需迭代 38 次即可达到稳定。由图 4(b)可知, CBAM-GNN 仅迭代 52 次就可以达到稳定的回归损失值, 其在迭代过程中的震荡幅度远小于其他三种对比算法, 由此可见该算法具有较好的稳定性和收敛性。

精度-召回率曲线作为评估分类任务的工具, 其曲线下方的面积通常可以展示模型的整体分类性能。选择精度-召回率曲线作为评价指标, 得到四种算法在不同数据集中的样本分类效果如图 5 所示。

图 5(a)和图 5(b)分别为四种检测算法在训练数据集和验证数据集中的精度-召回率曲线。结合图 5 可知, CBAM-GNN 在训练数据集和验证数据集中的曲线下面积 (Area Under the Curve, AUC), 即 AUC 值最大, 几乎接近 1, 由此可见该算法具有较好的分类能力, 能够准确区分数据集中

的漏洞样本和安全样本。此外, MSSIF-Net 的 AUC 面积大于 SSD 和 Mask R-CNN, 说明 MSSIF-Net 的分类效果优于另外两种。SSD 的 AUC 面积最小, 说明该算法的分类能力最差。

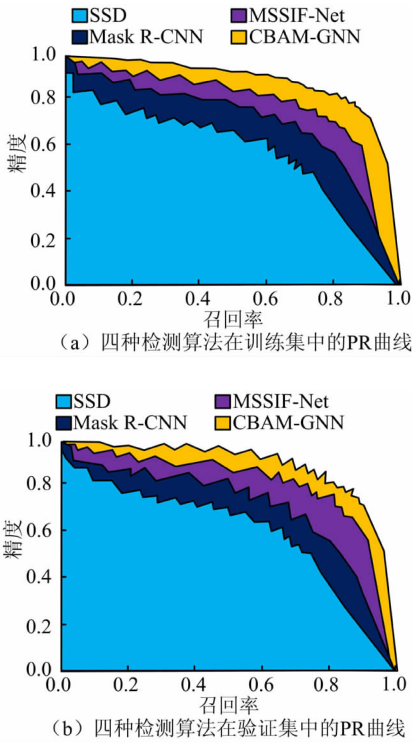


图 5 不同检测算法在训练集和验证集中的精度-召回率曲线

表 2 不同算法的平均检测精度和检测误差

数据集	算法模型	mAP/%	MSE
训练集	SSD	85.47	8.87
	Mask R-CNN	90.02	7.26
	MSSIF-Net	94.38	3.02
	CBAM-GNN	98.25	1.15
验证集	SSD	84.74	8.19
	Mask R-CNN	89.02	4.58
	MSSIF-Net	91.87	3.01
	CBAM-GNN	98.69	1.24

表 2 中给出了四种算法在两个数据集中的均方误差 (Mean Square Error, MSE) 和平均检测精度 (Mean Average Precision, mAP)。由表 2 可知, CBAM-GNN 在训练集和验证集中的 MSE 值均最小, mAP 值均最大。在训练集中, SSD、Mask R-CNN、MSSIF-Net、CBAM-GNN 的 mAP 值分别为 85.47%、90.02%、94.38%、98.25%, 在验证集中均为 84.74%、89.02%、91.87%、98.69%。在训练集中, SSD、Mask R-CNN、MSSIF-Net、

CBAM-GNN 的 MSE 值分别为 8.87、7.26、3.02、1.15,在验证集中分别为 8.19、4.58、3.01、1.24。

表 3 不同模型在实际安全漏洞检测中的性能表现				
指标	SSD	Mask R-CNN	MSSIF-Net	CBAM-GNN
检测时间/s	4.59	2.28	1.47	0.75
误报率/%	0.21	0.17	0.11	0.01
漏报率/%	0.35	0.28	0.12	0.03
成本效益/%	10.25	14.98	20.91	32.64
用户满意度/%	88.64	90.25	95.63	99.18

表 3 所示为四种模型检测 5000 个实际软件代码安全漏洞的性能情况。由表 3 可知,相较于 SSD、Mask R-CNN、MSSIF-Net 来说, CBAM-GNN 在实际的代码安全漏洞检测中有着更少的检测时间,仅需 0.75 s 就能完成所有检测任务,此外其误报率和漏报率都更低,分别为 0.01% 和 0.03%,最后该检测模型能够为软件企业增加 32.64% 的成本效益并获得 99.18% 的用户满意度。

3 结 论

为了帮助软件工作人员及时发现各类工业软件中的漏洞风险,本研究结合 CBAM 和 GNN 搭建了一个全新的漏洞检测模型。研究结果表明,相较于 SSD、Mask R-CNN、MSSIF-Net 三种对比算法来说, CBAM-GNN 有着更好的性能表现。首先 CBAM-GNN 仅需迭代 38 次和 52 次就能够达到稳定的总损失值和回归损失值,而其他三种模型在迭代过程中的损失波动则较大,达到稳定损失值的迭代次数也更多。其次, CBAM-GNN 在两个数据集集中的 AUC 最大,由此可见该算法具有较好的样本分类能力。最后, CBAM-GNN 在训练集和验证集中的 mAP 值分别高达 98.25% 和 98.69%, MSE 值分别低至 1.15 和 1.24。在实际应用中, CBAM-GNN 也有着更好的性能表现,其检测时间、误报率、漏报率、成本效益和用户满意度分别为 0.75 s、0.01%、0.03%、32.64%、99.18%。尽管

所提出的 CBAM-GNN 模型具有优异的性能,但研究仍存在一些不足之处。未来的研究可以探索更高效的算法优化策略,以减少模型的计算需求,同时提高模型对不同类型漏洞的适应性和泛化能力。

参考文献

[1] 余昊杨,武昕. 基于一维卷积神经网络的非侵入工业负荷事件检测方法[J]. 计算机应用, 2022, 42(2):277—284.

[2] 郑明明,刘胜全,马前. 基于可变形卷积融合双注意力机制的缺陷检测方法[J]. 东北师大学报(自然科学版), 2023, 55(2):52—61.

[3] 郑鹏,沙乐天. 基于混合分析的 Java 反序列化漏洞检测方法[J]. 计算机工程, 2023(12):136—145.

[4] 杨宏宇,杨海云,张良,et al. 基于特征依赖图的源代码漏洞检测方法[J]. 通信学报, 2023, 44(1):103—117.

[5] PRIYAWATI D, ROKHMAH S, UTOMO I C. Website vulnerability testing and analysis of website application using OWASP[J]. International Journal of Computer and Information System (IJCIS), 2022, 3(3): 142—147.

[6] 文敏,王荣存,姜淑娟. 基于关系图卷积网络的源代码漏洞检测[J]. 计算机应用, 2022, 42(6):1814—1821.

[7] 张俊,李山山,李磊,等. 基于残差门控图卷积网络的源代码漏洞检测[J]. 计算机工程与应用, 2023, 59(22):293—299.

[8] 潘超,吕翹楚,肖巍. 基于启发式遗传算法的即时通信网络漏洞检测[J]. 计算机仿真, 2023, 40(8):191—195.

[9] 宋健,张超. 基于规则过滤的机场网站漏洞自动化检测系统[J]. 自动化与仪器仪表, 2023, (4):134—137.

[10] 唐成华,高庆泽,杜征,等. 基于敏感路径的精确行为依赖图跟踪检测恶意代码[J]. 小型微型计算机系统, 2023, 44(7): 1593—1600.

[11] GOLMOHAMADI A, ZHANG M, ARCURI A. Testing restful apis: A survey[J]. ACM Transactions on Software Engineering and Methodology, 2023, 33(1): 1—41.

[12] 胡雨涛,王溯远,吴月明,等. 基于图神经网络的切片级漏洞检测及解释方法[J]. 软件学报, 2023, 34(6):2543—2561.

[13] 冯昭阳,武泽慧,吕卓,等. 基于能量优化的 Windows 模糊测试系统[J]. 计算机工程与设计, 2023, 44(10):2926—2934.

[14] 程靖云,王布宏,罗鹏. 基于图表示和 MHGAT 的代码漏洞静态检测方法[J]. 系统工程与电子技术, 2023, 45(5): 1535—1543.

[15] 胡飞,陈昊,王媛,等. 基于图网络的 Java 反序列化漏洞检测方法[J]. 计算机技术与发展, 2023, 33(5):122—129.