

大语言模型赋能的漏洞检测技术综述

刘春玲, 井 靖, 王奕森, 娄 睿

(信息工程大学, 河南 郑州 450001)

摘要: 软件漏洞是网络安全事件的主要根源, 大语言模型(LLMs)在漏洞检测中显示巨大的潜力。为提升LLMs漏洞检测技术的性能, 对2022—2025年该领域的研究技术进行系统综述。首先, 分析提示工程、指令微调、检索增强等关键技术; 其次, 梳理漏洞数据集的粒度、规模、支持的检测语言; 最后, 从模型规模、适应技术、数据集特性、模型融合和模型幻觉5个方面深入探讨影响漏洞检测性能的因素, 并讨论当前漏洞检测技术面临的挑战与未来研究方向。

关键词: 大语言模型; 漏洞检测; 提示工程; 软件安全

中图分类号: TP311.13

文献标识码: A

文章编号: 1671-0673(2026)03-0354-08

A Review of Vulnerability Detection Techniques Based on Large Language Models

LIU Chunling, JING Jing, WANG Yisen, LOU Rui

(Information Engineering University, Zhengzhou 450001, China)

Abstract: Software vulnerabilities remain a leading cause of cybersecurity incidents, and large language models (LLMs) have shown considerable promise in vulnerability detection. In order to facilitate the performance of LLMs in vulnerability detection, the research techniques in this field from 2022 to 2025 are systematically reviewed. Firstly, the key technologies such as prompt engineering, instruction fine-tuning, and retrieval-augmented generation are analyzed. Secondly, the granularity, scale, and supported detection languages of vulnerability datasets are sorted out. Finally, the factors that influence the performance of vulnerability detection are deeply discussed from 5 aspects, which include model scale, adaptation technology, dataset characteristics, model ensemble strategies, and model hallucination. The challenges faced by current vulnerability detection technologies and future research directions are discussed.

Key words: large language models; vulnerability detection; prompt engineering; software security

软件漏洞是网络安全事件的主要根源。文献[1]研究表明, 约70%的安全漏洞源于软件开发过程中的缺陷, 随着软件复杂度的不断提升和漏洞造成的损失日益扩大, 源代码漏洞检测面临新的挑战。例如, 2021年, Log4Shell漏洞(CVE-2021-44228)全球性爆发, 迅速演变为一场波及整个数字生态系统的链式安全危机。权威机构Gartner的研究数据表明, 该漏洞涉及全球超过3.5亿台服务器设备以及83%的云服务集群, 其冲击波影响至金融、政务、医疗及能源等关键基础设施领域。为

在开发早期识别潜在的漏洞并降低安全风险, 研究者提出了多种优秀的软件漏洞检测方法, 但根据常见漏洞与暴露数据库的统计, 漏洞数量仍呈现上升的趋势, 这对智能化漏洞检测工作提出了新的技术要求。

源代码漏洞检测是保障软件安全的首要任务。静态源代码漏洞检测方法主要包括4类: 基于规则、基于传统的代码相似性、基于机器学习/深度学习(Deep Learning, DL)以及基于大语言模型(Large Language Models, LLMs)的方法。

基于规则的漏洞检测工具能够报告漏洞的具体位置和类型,为进一步漏洞定位及修复提供了详细的信息指导。然而,其检测效率和准确性在很大程度上依赖于专家的经验 and 知识,专家预定义的规则往往难以全面覆盖真实程序中复杂多变的漏洞类型,无法有效识别所有潜在的漏洞。文献[2]实验验证,尽管基于规则的检测工具在软件保障参考数据集上表现出理想的性能,但真实世界的漏洞检测表现却不理想。

基于传统的代码相似性检测技术通过分析程序,将代码抽象为中间表征检测漏洞。常见中间表征形式有文本(VUDDY^[3])、抽象语法树(POSTER^[4])、代码属性图(VGRAPH^[5])等。这些技术通过分析源代码的词法、语法和语义,利用程序的结构信息、语义信息检测漏洞,其局限性体现在3个方面:定义漏洞模式高度依赖专家经验,不仅增加人力成本,也限制了检测的灵活性;易受攻击的代码和非易受攻击的代码的细微差别难以区分,误报率高;在处理复杂代码结构时,难以捕捉代码中的上下文语义依赖关系,进一步降低了检测的准确性。

基于深度学习的方法通过词法、语法和语义分析将程序抽象为中间形式,利用深度学习架构学习漏洞特征用来检测漏洞,如利用循环神经网络(SySeVR^[6])、卷积神经网络(LATNN^[7])、图神经网络(LineVD^[8])以及预训练Transformer架构(LineVul^[9])等。其核心在于利用神经网络训练漏洞检测模型,根据学习到的漏洞特征模式来检测潜在漏洞。这种自动学习漏洞特征的技术,发掘了漏洞挖掘新潜力,然而同样存在一定局限。深度学习模型在捕获代码中的长距离语义依赖方面存在困难,可能导致某些复杂漏洞的漏检;模型的性能受检测任务的影响,模型泛化能力差,文献[10]研究表明不同类型的漏洞往往对应不同的数据分布,单一漏洞类型的检测性能通常优于多类型漏洞检测;检测性能受到模型架构和数据集质量的限制,文献[11]显示真实数据集检测性能急剧下降。

LLMs已在漏洞检测任务中展现出巨大的应用潜力,其发展可以概括为3个阶段。早期探索期,该阶段直接利用通用LLMs(如GPT),验证其零样本/少样本漏洞检测的可行性;专业适应期,以CodeBERT、CodeLlama等代码预训练模型为基础,通过微调(Fine-Tuning)和高级提示工程技术,使其专门化;LLMs与传统分析工具在系统协同工作,构建混合智能框架。本文主要聚焦2022—2025年来利

用LLMs进行源代码漏洞检测的最新研究成果,系统性分析LLMs适配漏洞检测的技术,并展望未来研究方向。

1 相关知识

LLMs是以多层Transformer架构为核心的深度学习模型,通过海量参数与复杂网络结构实现对多模态数据(文本、源代码等)的自监督学习,最终获得高效执行多样化自然语言任务的能力。LLMs在自然语言处理领域的成功探索,激发了研究者将其应用于网络安全等下游任务的研究热情,如漏洞检测、恶意代码分析和漏洞修复等。为使LLMs更好地适应特定任务,研究者使用多种提升性能的适配技术,主要包括如下3种。

1)微调。微调是指对预训练LLMs的参数进行调整,使其适应特定任务。虽然预训练模型已经掌握广泛的通用语言知识,但这些知识在特定应用场景中可能并不完全适用。因此,通过在特定任务的数据集上对模型进行进一步训练,能够使其更好地契合目标任务的需求。全参数微调更新模型中的所有参数,这种技术适用于任务数据充足且与预训练数据差异较大的场景。参数高效微调指仅调整模型的一部分参数,降低计算和存储成本。低秩适应(Low-Rank Adaptation, LoRA)是其中常用的微调技术,通过在Transformer层中插入低秩可训练矩阵来调整模型,同时保留了原始权重。

2)提示工程。提示工程根据任务需求,通过精心设计输入提示,优化LLMs输出。常用的技术包括零样本提示、少样本提示和思维链提示。零样本提示不需要额外的训练数据,直接利用LLMs的预训练能力,通过设计合适的提示来引导模型输出。少样本提示通过少量标注数据优化模型的提示设计,提升模型对任务的理解。思维链提示(Chain-of-Thought Prompting, CoT)通过引导模型生成中间推理步骤来增强其推理能力。与传统上下文学习(In-Context Learning, ICL)不同,CoT不仅提供问题描述,还包含一系列清晰的推理步骤。这有助于模型更精准地把握任务目标,并逐步推导出最终答案。鉴于漏洞检测需要深入理解复杂的代码逻辑(如函数调用、控制流和数据流),思维链提示通过引导模型生成中间推理步骤,能更有效地揭示潜在漏洞。同时,由于该过程生成了可追溯的中间步骤,模型的决策逻辑变得更为透明,从

而提升了其可解释性。

3)检索增强(Retrieval Augmentation Generation, RAG)。RAG是一种需要借助外部知识增强模型在特定领域生成能力的技术。在漏洞检测中,通过引入外部知识库,RAG技术能够提供更丰富的上下文信息,提高模型的判断准确性,减少误报率。

2 LLMs漏洞检测技术文献分类

从国际权威文献数据库筛选出 33 篇 2022—2025 年与漏洞检测密切相关的研究文献,并划分为 5 类。1)综述类研究:主要探讨漏洞检测领域的研究现状与发展趋势;2)LLMs 与传统检测方法的对比研究:重点对比分析 LLMs 相较于传统方法的优势与局限;3)LLMs 检测性能评估研究:深入探讨 LLMs 在不同场景下的检测效能;4)数据集相关研究:主要研究实践中形成的基准数据集;5)模型幻觉研究:分析模型幻觉对检测性能的影响。

文献[12-16]全面探讨了 LLMs 在网络安全领域的多元化应用,其研究范围涵盖漏洞检测与修复、恶意软件分析等传统安全任务。基于现有研究基础,本文聚焦 LLMs 在漏洞检测这一核心领域的应

用,从模型架构选择、性能评估、影响因素分析等多个维度展开系统性研究。通过系统性地梳理和分析这些最新研究成果,本文旨在全面把握 LLMs 在漏洞检测领域的研究现状、技术进展及未来发展方向,为后续研究提供理论参考和实践指导。

2.1 LLMs漏洞检测常用模型

重点分析文献中使用的大型语言模型及数据集的使用情况。在研究文献中,GPT 系列模型在漏洞检测研究中占比最高,达到 28%,这主要得益于其在自然语言理解和代码语义分析方面的卓越表现;LLaMA 和 CodeLlama 模型分别占比 15% 和 12%,这两种模型因其在代码表征学习方面的优势而受到研究者青睐;CodeBERT 作为专门针对代码理解任务设计的预训练模型,也在研究中得到了广泛应用,这反映出领域特定模型在漏洞检测任务中的重要性。

2.2 LLMs漏洞检测模型常用数据集

基于 LLMs 的常用数据集如表 1 所示。表 1 详细列出了这些数据集的语言类型、检测粒度、规模、数据来源等信息。这些公开数据集的有效利用不仅推动了漏洞检测技术的发展,也为评估不同检测方法的性能提供了标准化基准。

表 1 常用数据集

数据集	语言	粒度	漏洞数	非漏洞数	链接
DiverseVul ^[17]	C/C++	函数	18 945	330 492	https://github.com/wagner-group/diversevul
Devign ^[18]	C/Java	函数	12 458	14 855	https://github.com/HuantWang/FUNDED_NISL
Reveal ^[19]	C/C++	函数	2 238	20 487	https://bit.ly/3bX30ai
Big-Vul ^[20]	C/C++	函数	11 834	253 096	https://github.com/ZeoVan/MSR_20_Code_Vulnerability_CSV_Dataset
CVEfixes ^[21]	C/C++/JAVA 等	函数	50 322	138 974	https://github.com/rmusab/vul-llm-finetune
VulDeePecker ^[22]	C/C++	切片	17 725	43 913	https://github.com/CGCL-codes/VulDeePecker

表 1 显示,基于 LLMs 的漏洞检测技术使用的数据库聚焦 C/C++ 和 Java 编程语言。这一趋势可能源于这些语言的广泛应用及其在安全关键领域的重要性。尤其是 C/C++ 语言,因其手动内存管理机制与指针运算特性,更易引发内存安全漏洞,故针对此类语言的漏洞检测需求具有显著紧迫性。

在检测粒度方面,选择函数作为漏洞检测粒度有理论和实验依据,因为函数是实现特定功能的完整单元。文献[23]研究表明了函数粒度能在漏洞检测中达到可扩展性和精确性的平衡。代码切片粒度能够捕捉跨函数或跨模块的复杂漏洞模式,提供更精细的漏洞定位能力,随着漏洞模式的多样化和复杂化升级,切片级别的漏洞检测方法可能是值得探索的研究方向。

2.3 LLMs漏洞检测评估指标

评估 LLMs 漏洞检测性能的指标主要有精确率、召回率、F1 值、准确率和一致性分数等。

精确率(Precision)是评估分类模型性能的关键指标,用于衡量模型预测为正类的样本(即存在漏洞)中实际为正类(真实存在漏洞)的比例。例如,精确率为 90% 表示模型每报告 10 个漏洞,其中大约 9 个是真实存在的漏洞。该指标直接反映了模型预测结果的准确性,特别是在正类预测中的可靠性。在漏洞检测任务中,高精确率具有至关重要的意义,它意味着模型报告的大部分“漏洞警报”是真实存在的,而非误报。

召回率(Recall)又称查全率,它是针对真实情况而言的,表示模型正确预测为正类的样本数占

所有实际正类(真实存在的漏洞)样本数的比例。在漏洞检测中,高召回率意味着模型能检测到更多的真实漏洞,降低漏报率。例如,召回率为 90% 表示模型能够找出所有真实漏洞中的 90%,仍有 10% 的漏洞未被发现。

在漏洞检测中,高精确率和召回率通常意味着更好的检测性能,但事实上这两个指标在实践中通常会出现矛盾。例如,目标系统中存在 10 个正样本,但模型只检测到 1 个正样本且是真实存在的漏洞,即没有误报,此时精确率为 100%,但召回率却很低,只有 10%。所以,通常综合考虑这两个指标,也就是 $F1$ 值,即精确率与召回率的调和平均值。 $F1$ 值越高,说明漏洞检测模型的性能越好。

准确率(Accuracy)指模型预测正确的样本数占总样本数的比例,包括正确识别出的漏洞和正确识别的安全代码,是分类模型性能评估中最直观的指标之一。

为评估模型在多样化场景中的逻辑推理稳定性,文献[24]引入了一致性分数(Consistency Score)评价指标。该指标通过度量模型输出随任务情境变化的一致性,反映其推理过程的可靠性。一致性分数越高,说明模型在面对相关但具有差异性的输入时,越能保持推理逻辑的连贯性,从而避免出现随机或不一致的响应。

3 LLMs 漏洞检测技术性能分析

LLMs 在漏洞检测中展现出广阔的应用前景,然而其检测性能受到多种因素的综合影响。为深入探究这一问题,围绕表 2 中的实验结果,深入探讨不同 LLMs 架构和适配策略在漏洞检测任务中的表现差异,并评估其与传统检测方法的协同效果。表 2 中的性能指标均源自对应文献,为了便于性能对比,部分指标求取其均值,对于筛选文献中未提及的性能指标,采用空白表示。

表 2 LLMs 漏洞检测技术性能							单位:%
文献	数据集	模型	适配技术	召回率	准确率	精确率	$F1$ 值
文献[25]	VulDeePecker	CodeBERT-125M	全参数微调			94.76	94.99
	VulDeePecker	GPT-2XL-1.5B	全参数微调			96.27	96.04
文献[26]	GitHub	CodeBERT	全参数微调	53.3	60.3	62.3	57.3
	GitHub	GPT-3.5	提示工程	47.2	62.7	80.0	54.0
文献[27]	GitHub	GPT-3.5-Turbo	提示工程	80		51	62
文献[28]	Chrome、Linux、Android、Qemu	DL 模型+LLMs+静态扫描工具	ICL+CoT	72.8		54.1	58.4
		Vicuna-13B	LoRA	43		88.7	52.8
文献[29]	DiverseVul	Llama2-7B	LoRA	17	65	36	25
	DiverseVul	CodeGemma-7B	LoRA	87	44	43	58
	DiverseVul	CodeLlama-7B	LoRA	14	60	32	20
文献[30]	Draper VDISC	GraphCodeBERT+UniXcoder+CodeBERT	全参数微调	82.36	82.36	82.85	82
文献[31]	DVWA、DVNA、OWASP Juice Shop、WebGoat	SAST+Llama3-70B	提示工程		62	86.11	76.54
文献[32]	FFmpeg、QEMU	UniXcoder+Conformer	无微调和提示				70.59
文献[33]	CVEfixes	GPT3.5-Turbo	直接提示	18.8		14.98	
	CVEfixes	Davinci	全参数微调	71.17		50.32	
	VulDeePecker	Davinci	全参数微调	94		60	73.23
	Juliet	GPT-4	CoT(8 步提示)	60.2	70.6	76	67.2
文献[34]	CVEfixes、Manually-Curated Dataset、VCMatch	WizardCoder	LoRA				71(平衡) 27(非平衡)
文献[35]		CodeLlama-13B					66.54
	DiverseVul、Devign、BigVul、CVEfixes、ReVeal、Juliet	CodeBERT-125M	LoRA+指令微调+				49.22
		GraphCodeBERT-125M	提示工程				55.50
		UniXcoder-126M					61.61

表 2 LLMs漏洞检测技术性能(续)							单位:%
文献	数据集	模型	适配技术	召回率	准确率	精确率	F1 值
文献[36]	VulBench	Llama2-7B	2 Shot	92.85		31.7	43.95
	VulBench	Vicuna-7B	2 Shot	63.2		33.3	36.15
	VulBench	CodeLlama-34B	2 Shot	38.55		35.55	30.8
	VulBench	GPT-4	2 Shot	49.05		41.0	32.65
文献[37]	ReVeal、Deign、 Draper、BigVul、 DiverseVul	Llama2-7B	LoRA	27.18	89.14	48.66	32.54
		CodeLlama-7B	LoRA	31.07	90.50	47.04	35.76
文献[38]	securityEval、PyT	GPT-3.5+SAST	提示工程	27.81		78.07	41.01
文献[39]	Linux 内核等	Llama3-8B	零样本+少样本	64.78	79.15	93.16	60.15
文献[40]	CVE 中提取的 多维度知识库	GPT-3.5-Turbo+GPT-4	提示工程+RAG	61	61	61	61

3.1 模型规模对检测性能的影响

不同规模模型下检测方法的评估指标如表 3 所示。混合模型在平均召回率和平均精确率方面表现最佳;小规模模型取得最好的平均 $F1$ 值;中等规模模型表现最差。统计结果表明:面向代码领域专门预训练的小规模模型(如 CodeBERT),因其在训练目标、数据与任务上的高度对齐,实现了 $F1$ 值的最优表现;中等规模模型在当前评估体系下受限于其通用能力与专业需求间的适配瓶颈,性能表现相对有限;而超大规模模型虽展现出较高的漏洞发现潜力(召回率),却因模型幻觉等问题在判断精确率上存在显著挑战。

表 3 不同规模模型下检测方法的评估指标				单位:%
模型	平均 召回率	平均 精确率	平均 F1 值	
超大规模模型 (参数量级 $>100\times10^9$)	60.06	53.33	57.82	
中等规模模型 ($1\times10^9\leq$ 参数量级 $\leq100\times10^9$)	47.86	53.22	46.67	
小规模模型 (参数量级约 100×10^6)			77.03	
混合模型(多个模型协同)	60.99	72.43	64.92	

3.2 数据集特性对模型选择的倾向性

对现有文献中使用的数据集进行分类统计,如表 4 所示。基于单一数据集训练的模型在平均召回率上显著高于基于混合数据集的模型,表明单一数据集在发现潜在漏洞方面覆盖更广;而使用混合数据集的模型则在平均精确率上表现更优,显著高于单一数据集模型,反映出其判断结果更为可靠。在综合性能指标上,两者平均 $F1$ 值接近。统计结果表明:数据集的设计需与检测任务的目标相协调。若优先追求漏洞的高召回率,可采用目标明确、标注质量高的单一或同质数据集;

若更重视高精确率,则应构建覆盖更广、类型丰富的混合数据集。未来的研究可进一步探索动态或层次化的数据采样与训练策略,使模型既能从混合数据中学习泛化能力,又能通过特定数据的强化训练保持对关键漏洞类型的高敏感度,从而在召回与精确之间实现更优的平衡。

表 4 不同数据集下检测方法的评估指标				单位:%
数据集	平均 召回率	平均 精确率	平均 F1 值	
单一数据集	57.00	55.13	54.53	
混合数据集	47.37	69.36	55.75	

3.3 LLMs 针对特定漏洞检测任务适应技术的遵循原则

基于适配技术性能的统计分析如表 5 所示。参数微调在平均召回率、平均精确率与平均 $F1$ 值上表现最优;提示工程虽能快速建立性能基准,但其效能上限受限于模型固有知识,难以实现深度优化;技术融合策略的平均 $F1$ 值已逼近最优的单一参数微调方法,展现出通过技术集成突破性能瓶颈的潜力。表 5 的统计结果表明:适配技术的选择构成性能差异的基础逻辑,追求高性能与专业化需依赖参数微调,注重灵活性与低成本可选用提示工程。摒弃单一技术路径,构建技术融合框架,可实现程序深度逻辑与 LLMs 语义生成能力的有机统一。

表 5 不同适配技术的评估指标				单位:%
适配技术	平均 召回率	平均 精确率	平均 F1 值	
提示工程	55.93	56.25	52.79	
参数微调	57.73	64.62	59.76	
技术融合			58.77	

3.4 模型融合对提升漏洞检测性能的影响

表3结果显示,混合模型在召回率和精确率表现最佳。文献[30,32]的实验结果表明,混合模型在准确率、F1值和AUC分数等指标上显著优于单个模型,并通过消融研究了不同模型组合对检测性能的影响。多模型融合可以显著提高漏洞检测的性能,并有发现零日漏洞潜力。

3.5 LLMs幻觉对漏洞检测性能影响

幻觉是LLMs在漏洞检测中常见的缺陷之一,由于模型本身的复杂性以及模型缺乏对上下文的理解而产生的幻觉会降低检测性能。文献[41]通过对LLMs响应的分析,发现由幻觉产生的错误占11%。这种幻觉错误会导致LLMs在解释漏洞时提供不准确的信息,导致误报;文献[42]研究显示,幻觉问题可能会显著影响LLMs漏洞检测的性能和可靠性,并提出结合RAG减少模型生成幻觉的可能性。

4 研究挑战与展望

尽管基于大语言模型的软件漏洞检测技术近年来取得了显著进展,但在理论研究与实际应用中仍面临一些挑战。

1)漏洞样本数据集规模有限且存在滞后性。当前广泛使用的漏洞数据集,其规模与真实世界漏洞库(如NVD)所收录的漏洞总量相比存在显著差距。而且数据集的构建与更新往往滞后于漏洞的公开与修复周期。这种规模不足且时效性滞后的特点,导致基于此类数据集训练的模型在面对真实、复杂且持续演化的代码环境时,其泛化能力与检测有效性面临严峻考验。

2)代码标准化对漏洞检测性能的影响研究较弱。代码传统漏洞检测技术常利用数据预处理技术,剔除噪音干扰,提高检测性能,但是数据处理对基于LLMs的漏洞检测性能的影响研究较少,当前还没有形成统一标准。例如:文献[43]在代码规范化前后的数据集上实验对比,结果显示尽管代码规范化能显著减少token数量,但并未显著提升模型性能;文献[34,37]使用WizardCoder模型研究数据的平衡性对检测性能的影响,实验结果显示在平衡数据集上的F1值比非平衡数据集有显著提高。

3)检测泛化能力面临严峻考验。当前方法受限于训练数据多来自于单一数据来源,致使模型易于过拟合特定漏洞模式,泛化至真实复杂项目

的能力不足;同时,检测仍主要依赖于对已知漏洞变体的表面模式匹配,对检测零日漏洞能力不足。

4)跨语言漏洞检测研究仍处于初级阶段。尽管已有文献[41-45]等研究LLMs漏洞检测的跨语言迁移能力,但当前工作主要集中于C/C++等少数语言,缺乏对Python、JavaScript等语言的广泛研究。不同语言在内存管理、类型系统与并发模型上有根本差异,使得跨语言漏洞表征迁移困难,并且多语言基准数据集的缺失,进一步阻碍了跨语言漏洞检测技术的研究。

在基于LLMs的漏洞检测研究中,“多模型融合”与“适配技术融合”是提升检测性能的两条重要技术路径。

1)多模型融合。“LLMs+传统工具+知识库”的融合框架,是构建高可靠、可解释、自动化漏洞检测系统的研究方向。通过横向大模型融合及纵向集成传统静态分析工具,减少LLMs产生的幻觉,提升漏洞检测的性能。

2)适配技术融合。结合适配技术的特点,通过有机组合高效微调、思维链提示与检索增强生成等技术,形成优势互补的完整适配方案,以更低的成本与更高的灵活性,实现模型在复杂真实场景下的鲁棒且可靠的检测性能。

5 结束语

为系统梳理基于LLMs的软件漏洞检测领域的研究进展,从模型规模、适应技术、数据集特性、模型融合和模型幻觉5个关键维度出发,深入分析其对漏洞检测性能的影响。研究发现:领域专用的小规模预训练模型在综合性能上表现优异,而超大规模模型虽然召回率较高,但是受幻觉问题影响精确率受限;“多模型融合+多技术协同”的混合智能框架在提升漏洞检测性能方面展现出显著优势。当前研究仍面临跨语言检测能力不足、泛化能力薄弱等挑战。未来可深入研究构建“多模型融合+多技术协同”的混合智能框架,探索幻觉抑制与可解释性机制,推动LLMs赋能漏洞检测技术向高精度、低误报、强泛化方向持续发展。

参考文献:

- [1] ASLAN Ö, AKTUĞ S S, OZKAN-OKAY M, et al. A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions[J]. Electronics, 2023,12(6):No.1333.
- [2] LI W X, DOU S H, WU Y M, et al. COCL: an intelligent

- framework for enhancing deep learning-based vulnerability detection[J]. *IEEE Transactions on Industrial Informatics*, 2024, 20(3):4953-4961.
- [3] KIM S, WOO S, LEE H, et al. VUDDY: a scalable approach for vulnerable code clone discovery[C]//*Proceedings of the 2017 IEEE Symposium on Security and Privacy*. Piscataway, USA: IEEE, 2017:595-614.
- [4] LIN G J, ZHANG J, LUO W, et al. POSTER: vulnerability discovery with function representation learning from unlabeled projects[C]//*Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. New York, USA: ACM, 2017:2539-2541.
- [5] BOWMAN B, HUANG H H. VGRAPH: a robust vulnerable code clone detection system using code property trip-lets[C]//*Proceedings of the 2020 IEEE European Symposium on Security and Privacy*. Piscataway, USA: IEEE, 2020:53-69.
- [6] LI Z, ZOU D Q, XU S H, et al. SySeVR: a framework for using deep learning to detect software vulnerabilities[J]. *IEEE Transactions on Dependable and Secure Computing*, 2022, 19(4):2244-2258.
- [7] 楚嘉玮,董卫宇,刘春玲.LATNN:基于自注意力机制的抽象语法树表示方法[J]. *信息工程大学学报*, 2023, 24(5):586-592.
- [8] HIN D, KAN A, CHEN H, et al. LineVD: statement-level vulnerability detection using graph neural networks[C]//*Proceedings of the 19th International Conference on Mining Software Repositories*. New York, USA: ACM, 2022:596-607.
- [9] FU M, TANTITHAMTHAVORN C. LineVul: a transformer-based line-level vulnerability prediction[C]//*Proceedings of the 19th International Conference on Mining Software Repositories*. New York, USA: ACM, 2022:608-620.
- [10] STEENHOEK B, RAHMAN M M, JILES R, et al. An empirical study of deep learning models for vulnerability detection[C]//*Proceedings of the 2023 IEEE/ACM 45th International Conference on Software Engineering*. Piscataway, USA: IEEE, 2023:2237-2248.
- [11] CHAKRABORTY S, KRISHNA R, DING Y, et al. Deep learning based vulnerability detection: are we there yet[J]. *IEEE Transactions on Software Engineering*, 2022, 48(9):3280-3296.
- [12] ZHANG J, BU H Y, WEN H, et al. When LLMs meet cybersecurity: a systematic literature review[J]. *Cybersecurity*, 2025, 8:No.55.
- [13] XU H X, WANG S N, LI N K, et al. Large language models for cyber security: a systematic literature review[J]. *ACM Transactions on Software Engineering and Methodology*, 2025. DOI: 10.1145/3769676.
- [14] SHENG Z, CHEN Z C, GU S N, et al. LLMs in software security: a survey of vulnerability detection techniques and insights[J]. *ACM Computing Surveys*, 2026, 58(5): No.134.
- [15] ZHOU X, CAO S C, SUN X B, et al. Large language model for vulnerability detection and repair: literature review and the road ahead[J]. *ACM Transactions on Software Engineering and Methodology*, 2025, 34(5):No.145.
- [16] SUN Y Q, WU D Y, XUE Y, et al. LLM4Vuln: a unified evaluation framework for decoupling and enhancing LLMs' vulnerability reasoning [DB/OL]. (2025-06-07) [2026-02-25]. <https://arxiv.org/abs/2401.16185>.
- [17] CHEN Y Z, DING Z J, ALOWAIN L, et al. DiverseVul: a new vulnerable source code dataset for deep learning based vulnerability detection [C]//*Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*. New York, USA: ACM, 2023: 654-668.
- [18] WANG H T, YE G X, TANG Z Y, et al. Combining graph-based learning with automated data collection for code vulnerability detection[J]. *IEEE Transactions on Information Forensics and Security*, 2021, 16:1943-1958.
- [19] FERRAG M A, ALWAHEDI F, BATTAH A, et al. Generative AI in cybersecurity: a comprehensive review of LLM applications and vulnerabilities[J]. *Internet of Things and Cyber-Physical Systems*, 2025, 5:1-46.
- [20] FAN J H, LI Y, WANG S H, et al. A C/C++ code vulnerability dataset with code changes and CVE summaries [C]//*Proceedings of the 17th International Conference on Mining Software Repositories*. New York, USA: ACM, 2020:508-512.
- [21] BHANDARI G, NASEER A, MOONEN L. CVEfixes: automated collection of vulnerabilities and their fixes from open-source software[C]//*Proceedings of the 17th International Conference on Predictive Models and Data Analytics in Software Engineering*. New York, USA: ACM, 2021:30-39.
- [22] LI Z, ZOU D Q, XU S H, et al. VulDeePecker: a deep learning-based system for vulnerability detection [C]//*Proceedings of the 2018 Network and Distributed System Security Symposium*. Rosten, USA: Internet Society, 2018. DOI: 10.14722/ndss.2018.23158.
- [23] 刘春玲,戚旭衍,唐永鹤,等.基于词汇的源代码克隆检测技术综述[J]. *计算机科学*, 2024, 51(6):12-22.
- [24] LI Y S, BRANCO P, HOOLE A M, et al. SV-TrustEval-C: evaluating structure and semantic reasoning in large language models for source code vulnerability analysis [C]//*Proceedings of the 2025 IEEE Symposium on Security and Privacy*. Piscataway, USA: IEEE, 2025: 3014-3032.
- [25] THAPA C, JANG S I, AHMED M E, et al. Transformer-

- based language models for software vulnerability detection [C]//Proceedings of the 38th Annual Computer Security Applications Conference. New York, USA: ACM, 2022: 481-496.
- [26] ZHOU X, ZHANG T, LO D. Large language model for vulnerability detection: emerging results and future directions [C]//Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results. New York, USA: ACM, 2024: 47-51.
- [27] CHESHKOV A, ZADOROZHNY P, LEVICHEV R. Evaluation of ChatGPT model for vulnerability detection [DB/OL]. (2023-04-12) [2026-02-25]. <https://arxiv.org/abs/2304.07232>.
- [28] YANG Y J, ZHOU X, MAO R F, et al. DLAP: a deep learning augmented large language model prompting framework for software vulnerability detection [J]. Journal of Systems and Software, 2025,219:No.112234.
- [29] SULTANA S, AFREEN S, EISTY N U. Code vulnerability detection: a comparative analysis of emerging large language models [DB/OL]. (2024-09-16) [2026-02-25]. <https://arxiv.org/abs/2409.10490>.
- [30] RIDOYSZ, SHAZZADHOSSAINSHAONM, CUZZOCREA A, et al. EnStack: an ensemble stacking framework of large language models for enhanced vulnerability detection in source code [C]//Proceedings of the 2024 IEEE International Conference on Big Data. Piscataway, USA: IEEE, 2024:6356-6364.
- [31] KELTEK M, HU R, SANIM F, et al. LSAST: enhancing cybersecurity through LLM-supported static application security testing [M]//NEMEC ZLATOLAS L, RANNENBERG K, WELZER T, et al. ICT systems security and privacy protection. Cham, Switzerland: Springer, 2025:166-179.
- [32] WANG J, HUANG Z S, LIU H L, et al. DefectHunter: a novel LLM-driven boosted-conformer-based code vulnerability detection mechanism [DB/OL]. (2023-09-27) [2026-02-25]. <https://arxiv.org/abs/2309.15324>.
- [33] DAS PURBA M, GHOSH A, RADFORD B J, et al. Software vulnerability detection using large language models [C]//Proceedings of the 2023 IEEE 34th International Symposium on Software Reliability Engineering Workshops. Piscataway, USA: IEEE, 2023:112-119.
- [34] SHESTOV A, LEVICHEV R, MUSSABAYEV R, et al. Finetuning large language models for vulnerability detection [J]. IEEE Access, 2025,13:38889-38900.
- [35] DU X H, WEN M, ZHU J H, et al. Generalization-enhanced code vulnerability detection via multi-task instruction fine-tuning [C]//Findings of the Association for Computational Linguistics ACL 2024. Stroudsburg, PA, USA: ACL, 2024:10507-10521.
- [36] TAMBERG K, BAHSI H. Harnessing large language models for software vulnerability detection: a comprehensive benchmarking study [J]. IEEE Access, 2025,13:29698-29717.
- [37] JIANG X F, WU L H, SUN S, et al. Investigating large language models for code vulnerability detection: an experimental study [DB/OL]. (2025-01-05) [2026-02-25]. <https://arxiv.org/abs/2412.18260>.
- [38] BAKHSHANDEH A, KERAMATFAR A, NOROUZI A, et al. Using ChatGPT as a static application security testing tool [DB/OL]. (2023-08-28) [2026-02-25]. <https://arxiv.org/abs/2308.14434>.
- [39] ZIBAEIRAD A, VIEIRA M. VulnLLMEval: a framework for evaluating large language models in software vulnerability detection and patching [DB/OL]. (2024-09-16) [2026-02-25]. <https://arxiv.org/abs/2409.10756>.
- [40] DU X Y, ZHENG G, WANG K X, et al. Vul-RAG: enhancing LLM-based vulnerability detection via knowledge-level RAG [J]. ACM Transactions on Software Engineering and Methodology, 2026. DOI: 10.1145/3797277.
- [41] STEENHOEK B, RAHMAN M M, ROY M K, et al. A comprehensive study of the capabilities of large language models for vulnerability detection [DB/OL]. (2025-01-07) [2026-02-25]. <https://arxiv.org/abs/2403.17218v1>.
- [42] SOOD A K, ZEADALLY S, HONG E. The paradigm of hallucinations in AI-driven cybersecurity systems: understanding taxonomy, classification outcomes, and mitigations [J]. Computers and Electrical Engineering, 2025, 124:No.110307.
- [43] GONÇALVES J, DIAS T, MAIA E, et al. SCoPE: evaluating LLMs for software vulnerability detection [DB/OL]. (2024-07-19) [2026-02-25]. <https://arxiv.org/abs/2407.14372>.
- [44] YU J J, SHU H L, FU M, et al. A preliminary study of large language models for multilingual vulnerability detection [C]//Proceedings of the 34th ACM SIGSOFT International Symposium on Software Testing and Analysis. New York, USA: ACM, 2025:161-168.
- [45] ZHANG C Y, LIU H, ZENG J T, et al. Prompt-enhanced software vulnerability detection using ChatGPT [C]//Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering-Companion. New York, USA: ACM, 2024:276-277.

(编辑:李志豪)