软件学报 ISSN 1000-9825, CODEN RUXUEW
2026,37(1):344−377 [doi: 10.13328/j.cnki.jos.007413] [CSTR: 32375.14.jos.007413]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

智能合约与 DeFi 协议漏洞检测技术综述*

揭晚晴[1,2,3], 邱望洁[1,2,3], 黄鑫鹏[1,2,3], 杨浩甫[1,2,6], 赵冠球[1,2], 张沁楠[1,2], 夏 清[4], 郑宏威[1,5], 郑志明[1,2,3]

[1](未来区块链与隐私计算高精尖创新中心 (北京航空航天大学), 北京 100191)

[2](北京航空航天大学 人工智能学院, 北京 100191)

[3](中关村实验室, 北京 100191)

[4](中国科学院 软件研究所 并行软件与计算科学实验室, 北京 100190)

[5](北京微芯区块链与边缘计算研究院, 北京 100190)

[6](南开大学 计算机学院, 天津 300071)

通信作者: 邱望洁, E-mail: wangjieqiu@buaa.edu.cn; 张沁楠, E-mail: zhangqn@buaa.edu.cn

摘 要: 智能合约作为区块链核心的可编程组件, 承担了资产管理和复杂业务逻辑处理的功能, 它们共同构成了去中心化金融 (decentralized finance, DeFi) 协议. 然而, 随着区块链的快速发展, 智能合约和 DeFi 协议的安全问题日益凸显, 吸引了大量攻击者利用其漏洞牟取利益. 近年来, 多起涉及智能合约和 DeFi 协议的重大安全事件强调了漏洞检测技术研究的必要性, 已成为安全防护的重中之重. 系统性地总结了现有工作, 提出了智能合约与 DeFi 协议漏洞检测技术研究框架, 分别从智能合约和 DeFi 协议两个层面对漏洞类型和检测技术进行梳理. 在智能合约方面, 重点分析了大语言模型 (large language model, LLM) 作为主要检测引擎和与传统方法结合的漏洞检测技术应用情况; 在 DeFi 协议方面, 系统性地分类并整理了 DeFi 协议层的漏洞及其检测方法, 并探讨了攻击发生前后检测方法的优势与局限性, 弥补了现有综述在 DeFi 协议漏洞检测方面的不足. 最后, 对现有检测方法面临的挑战进行总结, 并展望了未来的研究方向, 旨在为智能合约与 DeFi 协议的安全检测提供新的思路和理论支持.

关键词: 智能合约; DeFi 协议; 漏洞检测; 大语言模型

中图法分类号: TP309

中文引用格式: 揭晚晴, 邱望洁, 黄鑫鹏, 杨浩甫, 赵冠球, 张沁楠, 夏清, 郑宏威, 郑志明. 智能合约与DeFi协议漏洞检测技术综述. 软件学报, 2026, 37(1): 344–377. http://www.jos.org.cn/1000-9825/7413.htm

英文引用格式: Jie WQ, Qiu WJ, Huang XP, Yang HF, Zhao GQ, Zhang QN, Xia Q, Zheng HW, Zheng ZM. Survey on Vulnerability Detection Techniques for Smart Contract and DeFi Protocol. Ruan Jian Xue Bao/Journal of Software, 2026, 37(1): 344–377 (in Chinese). http://www.jos.org.cn/1000-9825/7413.htm

Survey on Vulnerability Detection Techniques for Smart Contract and DeFi Protocol

JIE Wan-Qing[1,2,3], QIU Wang-Jie[1,2,3], HUANG Xin-Peng[1,2,3], YANG Hao-Fu[1,2,6], ZHAO Guan-Qiu[1,2], ZHANG Qin-Nan[1,2], XIA Qing[4], ZHENG Hong-Wei[1,5], ZHENG Zhi-Ming[1,2,3]

[1](Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing (Beihang University), Beijing 100191, China)

[2](School of Artificial Intelligence, Beihang University, Beijing 100191, China)

[3](Zhongguancun Laboratory, Beijing 100191, China)

[4](Joint Laboratory of Blockchain Technology and Application, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

[5](Beijing Academy of Blockchain and Edge Computing, Beijing 100190, China)

[6](College of Computer Science, Nankai University, Tianjin 300071, China)

* 基金项目: 国家重点研发计划 (2022ZD0116800); 国家自然科学基金 (62141605, 62372493); 中国博士后科学基金 (373500); 北京市自然科学基金 (Z230001); 未来区块链与隐私计算高精尖创新中心建设项目 (GJJ-23-001, GJJ-23-002); 北航敢为行动计划 (KG16336101)
收稿时间: 2024-11-13; 修改时间: 2024-12-26; 采用时间: 2025-02-12; jos 在线出版时间: 2025-09-24
CNKI 网络首发时间: 2025-09-25

Abstract: As core programmable components of blockchain, smart contracts are responsible for asset management and the execution of complex business logic, forming the foundation of decentralized finance (DeFi) protocols. However, with the rapid advancement of blockchain technology, security issues related to smart contracts and DeFi protocols have become increasingly prominent, attracting numerous attackers seeking to exploit vulnerabilities for illicit gains. In recent years, several major security incidents involving smart contracts and DeFi protocols have highlighted the importance of vulnerability detection research, making it a critical area for security defense. This study systematically reviews existing literature and proposes a comprehensive framework for research on vulnerability detection in smart contracts and DeFi protocols. Specifically, vulnerabilities and detection techniques are categorized and analyzed for both domains. For smart contracts, the study focuses on the application of large language models (LLM) as primary detection engines and their integration with traditional methods. For DeFi protocols, it categorizes and details various protocol-level vulnerabilities and their detection methods, analyzing the strengths and limitations of detection strategies before and after attacks, addressing gaps in existing reviews on DeFi vulnerability detection. Finally, this study summarizes the challenges faced by current detection approaches and outlines future research directions, aiming to provide new insights and theoretical support for the security detection of smart contracts and DeFi protocols.

Key words: smart contract; DeFi protocol; vulnerability detection; large language model (LLM)

区块链[1]作为一种分布式账本技术, 具备去中心化、不可篡改、透明等特性. 智能合约[2]作为区块链的核心组成部分, 则赋予了区块链可编程的特性, 使其可以利用自动化脚本代码来编辑和操作业务及数据, 大幅推动区块链应用迅速渗透到金融、供应链、医疗等经济社会多个领域[3]. 在此基础上, 去中心化金融 (decentralized finance, DeFi)[4]得以迅速发展. DeFi 协议通过智能合约提供了无须信任中介的金融服务, 例如借贷、资产交易、流动性挖矿和衍生品合约等. DeFi 协议的业务特性包括去中心化治理、无中介信任、资产流动性优化、可组合性和开放性, 允许用户直接在区块链上进行点对点的金融交易, 促进了更为公平和高效的金融市场发展. 然而, 这些特性在为用户提供便利和创新的同时, 也为攻击者创造了潜在的漏洞利用机会.

随着区块链技术的广泛应用, 智能合约和 DeFi 协议的安全性问题也愈发凸显[5,6], 是制约区块链生态系统稳定运行的关键. 智能合约的不可篡改性虽然确保了交易的透明性和信任度, 但也使得一旦代码中存在漏洞, 修复这些漏洞将变得极为困难. 随着 DeFi 生态系统的发展, 智能合约的安全问题逐步延伸至 DeFi 协议, 导致了更为复杂和严重的安全风险, 下面结合图 1 典型的安全事件按时间顺序进行分析.

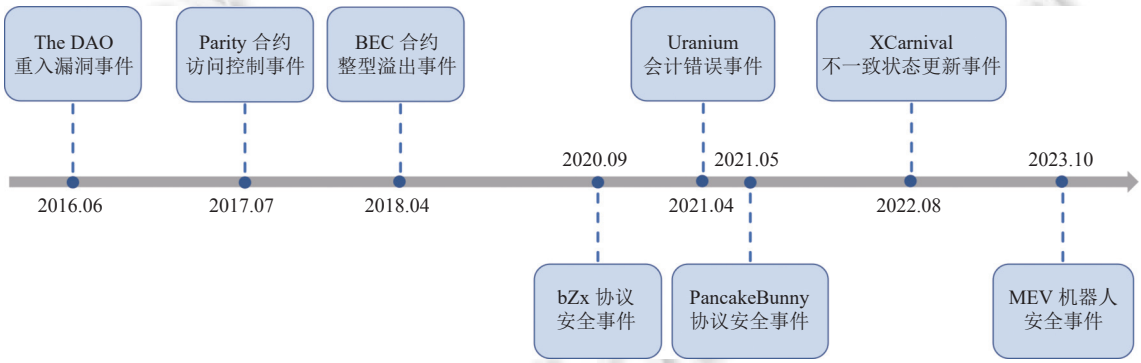

图 1 典型安全事件时间线

在智能合约层面, 重入攻击[7]、访问控制[8]和整数溢出[9]等传统漏洞仍然具有重要影响, 近年来涉及复杂合约逻辑和业务流程的漏洞在实际安全事件中的比例也在显著上升[10]. 例如, 2021 年, Uranium Finance 因会计错误导致利息计算多了两个 0, 被攻击者利用, 造成 5 700 万美元的损失[11]; 2022 年, XCarnival 由于状态更新不一致, 非同质化代币 (non-fungible token, NFT) 借贷池在交易过程中遭遇智能合约攻击, 攻击者通过反复借入贷款而不偿还, 耗尽了整个资金池, 造成 380 万美元被盗[12].

DeFi 协议通常由一组智能合约组成, 服务于去 DeFi 应用, 旨在实现无中介的、自动化的金融服务, 满足用户在不同场景中的需求, 如去中心化借贷、去中心化交易所等. 由于 DeFi 协议的执行逻辑涉及多个合约之间的交互, 其安全漏洞往往源于智能合约间的复杂互动, 从而引发许多新的漏洞. 这些漏洞可能导致整个协议的安全性下

降, 增加了检测和防范的难度, 因此本文将其划分为 DeFi 协议层漏洞.

事实上, DeFi 协议漏洞引发的安全事件在整个区块链安全事件中占比极高, 且其复杂性和破坏性日益加剧. 据统计, 截至 2024 年 9 月, 10 大最常见的攻击事件中, 针对 DeFi 协议漏洞的攻击占比高达 39.01%[13]. 例如, 2020 年 bZx 协议因依赖的预言机被操纵, 攻击者通过一系列复杂交易操纵市场价格, 最终窃取了约 800 万美元的资产[14]; 2021 年, PancakeBunny 协议遭遇严重攻击, 攻击者利用协议中未充分保护的合约逻辑, 采用闪电贷在短时间内大幅抬高代币价格并迅速抛售, 导致损失约 2 亿美元[15]; 2023 年, 以太坊基金会在使用 Uniswap V3 进行交易时, 遭遇了矿工可提取价值 (miner extractable value, MEV) 机器人实施的三明治攻击, 攻击者通过监测以太坊网络上的未确认交易, 利用高优先级交易包夹受害者的交易, 导致以太坊基金会损失了 9 101 美元[16].

综上所述, 智能合约和 DeFi 协议中的漏洞不仅对个别用户构成威胁, 更对整个区块链金融生态系统的稳定性和可持续性构成重大挑战. 随着攻击手段的复杂化和攻击频率的增加, 如何有效地检测智能合约和 DeFi 协议中的安全漏洞, 已成为保障区块链系统安全运行和推动产业应用的关键. 随着通用人工智能 (artificial general intelligence, AGI) 的发展, 大语言模型 (large language model, LLM) 为智能合约与 DeFi 协议漏洞检测技术带来新的契机[17−19]. LLM 通过结合自然语言语义理解与代码结构分析能力, 既能够深入解析智能合约中隐含的复杂逻辑与潜在漏洞, 又能够持续学习和适应区块链技术及网络威胁演变过程中出现的新漏洞模式, 从而推动智能合约安全性的进一步提升.

1 引 言

目前已有多项综述工作针对智能合约与 DeFi 协议漏洞检测技术进行了梳理, 但这些研究主要集中在智能合约检测方法上, 对 DeFi 协议的漏洞检测方法缺乏深入讨论. 此外, LLM 驱动的智能合约漏洞检测方法正在迅速发展, 但相关的系统性分析和总结仍显不足.

在智能合约漏洞检测方法的研究中, 钱鹏等人[20]从 Solidity 代码层、EVM 执行层和区块链系统层这 3 个层面介绍智能合约常见的漏洞类型和典型案例, 并从指标上比较检测方法对于不同漏洞的有效性, 总结分析了智能合约漏洞检测方法的研究进展; 董伟良等人[21]从核心方法、漏洞类型、实验数据等方面展开分析, 并系统性地对比了国内外智能合约漏洞检测技术在研究内容、方法选择和发展重点上的差异, 揭示出国内外在该领域研究中的优势与不足; 崔展齐等人[22]根据智能合约漏洞的检测流程, 提出了信息收集 (漏洞发现与识别)、检测方法 (漏洞分析与检测)、评估标准 (数据集与评价指标) 三阶段研究框架, 并系统化地梳理和分类现有的漏洞类型, 进一步扩展了智能合约安全漏洞的分类体系; 刘哲旭等人[23]则从漏洞数据集、机器学习方法、漏洞修复技术和补丁部署策略这 4 个方面分析总结了现有智能合约漏洞检测与修复的研究进展, 致力于从机器学习视角归纳智能合约漏洞检测方法, 将传统的漏洞检测方法 (符号执行、形式化验证、污点分析、模糊测试等) 作为一种数据集标注技术, 然后将这些数据用于训练机器学习模型, 从而使机器学习方法成为智能合约漏洞检测的核心; Zhao 等人[24]从实际攻击事件出发, 梳理了传统的编码漏洞及新兴的复杂攻击模式, 分析了当前检测策略在应对复杂漏洞方面的局限性, 然而该研究未对漏洞和检测方法进行层次化划分, 且对 DeFi 协议漏洞描述有限. 然而, 这些工作仅围绕智能合约漏洞进行研究, 尚缺乏对 DeFi 协议的漏洞检测方法进行系统性梳理与探讨, 且未跟进 LLM 在智能合约检测中的最新应用和研究成果.

在 DeFi 协议漏洞检测方法方面, 相关研究尚未成熟, 现有综述在方法分类、数据覆盖和系统性理论梳理等方面存在不足. Qian 等人[25]面向 DeFi 安全评估了漏洞检测工具在分析复杂 DeFi 协议时的有效性和现有自动修复工具的优缺点, 提出了 DeFi 攻击的 6 种常见分类, 然而该分类对实际漏洞数据的覆盖范围有限, 未能充分反映 DeFi 生态中多样化的攻击模式和漏洞类型, 同时该研究对 DeFi 协议漏洞的检测方法缺乏系统化分类; Chaliasos 等人[26]针对开发者和审计人员的需求, 评估了当前自动化安全工具在智能合约和 DeFi 协议中的应用效果, 尽管其研究侧重于实际应用, 但缺乏对漏洞检测方法的系统性理论梳理, 未能形成统一的方法分类框架; Zhou 等人[27]从网络、共识、智能合约、DeFi 协议以及外部辅助服务这 5 个层面构建了用于分析 DeFi 系统和威胁模型的框架, 对真实世界的 DeFi 事件进行分类, 揭示学术界与实践界之间的差距, 并探讨可能的防御机制, 但并未对漏洞检

测方法进行具体的划分和描述, 未能提供针对不同 DeFi 漏洞类型的检测技术对比分析.

综上所述, 现有的综述工作虽然在智能合约漏洞检测方法上取得了长足进展, 但在 DeFi 协议层面和 LLM 驱动的漏洞检测方法上的研究仍需进一步完善. 为了帮助后续研究者更全面深入地理解智能合约安全, 本文系统性地梳理和分析了智能合约与 DeFi 协议漏洞检测技术, 从智能合约层和 DeFi 协议层两个维度对漏洞类型进行了分类, 并对相应的检测方法进行了详尽的综述. 此外, 本文着重总结和分析了 LLM 在智能合约漏洞检测中的应用, 揭示其在应对漏洞的潜力, 进而为未来的研究提供新的视角和理论支撑.

为了能够了解智能合约与 DeFi 协议漏洞检测领域的科研进展, 本文采用系统文献综述方法[28], 通过系统性地收集、筛选、评估和分析包含“smart contract”“DeFi protocol”“vulnerability detection”和“Large Language Model”等关键词的相关研究成果. 为制定详细的检索策略, 选择了 Google Scholar、Web of Science、Sci-Hub、IEEE Xplore、CNKI 等主要论文搜索引擎和在线数据库, 以全面准确地掌握最新研究进展. 关键词组合使用布尔逻辑运算符, 构建更精确的搜索查询. 最终, 本文重点查阅的文献聚焦以下 3 个方面.

(1) 主题相关性: 确保文献与研究主题直接或间接相关, 同时了解文献的研究背景, 判断其与本文研究问题的关联性.

(2) 文献质量: 评估文献的学术质量, 重点考虑论文作者的学术背景及其在相关领域的影响力, 优先选择具有较高学术权威性的期刊和会议论文, 例如 CCF-C 类或中国科学院《期刊分区表》二区及以上的文献. 同时, 优先选取近 5 年内发表的高质量论文, 特殊情况下可适当放宽文献选择标准.

(3) 文献覆盖度: 确保文献检索充分涵盖了与研究主题相关的各个领域, 通过分析文献的引用和被引用情况, 挖掘潜在的相关研究, 进一步扩展文献库.

最终, 本文共筛选出 78 篇高质量文献, 数量统计如图 2 所示. 统计时间跨度为 2010–2024 年, 其中 2010–2015 年间的文献为 1 篇, 2016–2020 年间为 28 篇, 2021–2024 年间增加至 49 篇, 反映出漏洞检测技术在近年来的研究热度持续上升. 从图中可以看出, CCF-A 级别的文献有 48 篇 (61.54%), 其中软件工程和安全领域的文献数量最多, 分别为 24 篇和 14 篇, 说明漏洞检测技术的研究在这两个领域具有重要影响. 发表在 CCF-B 和 CCF-C 级别的会议/期刊上的论文共计 30 篇 (38.46%), 这些文献提供了多样化的研究视角和方法, 进一步支撑了本文的研究框架.

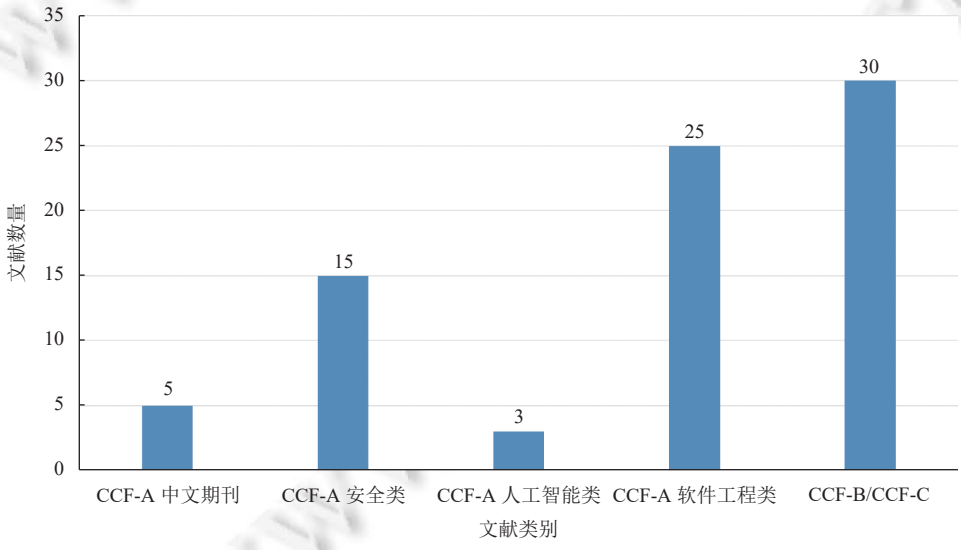

图 2　本文研究的智能合约与 DeFi 协议漏洞检测相关文献数量统计 (2010–2024)

本文的主要贡献总结如下.

(1) 系统性地总结了智能合约与 DeFi 协议漏洞检测的现有工作, 构建了智能合约与 DeFi 协议漏洞检测研究

框架, 从智能合约层与 DeFi 协议层两个维度, 对漏洞类型及检测方法进行了全面的梳理与总结.

(2) 在智能合约方面, 深入总结了 LLM 在智能合约漏洞检测中的应用现状, 并分析了其作为主要检测引擎和与传统方法相结合两方面的技术应用情况.

(3) 在 DeFi 协议方面, 系统性地分类并整理了 DeFi 协议层的漏洞及其检测方法, 弥补了现有综述在 DeFi 协议漏洞检测方面的不足.

(4) 全面梳理了当前智能合约与 DeFi 协议漏洞检测技术面临的挑战, 并展望了未来的研究方向, 以推动智能合约安全领域的进一步发展.

本文第 2 节介绍智能合约与 DeFi 协议的漏洞类型, 涵盖智能合约层和 DeFi 协议层的不同漏洞类别. 第 3 节详细探讨智能合约与 DeFi 协议的漏洞检测方法, 重点分析传统检测方法与 LLM 方法在智能合约层的优势与不足, 并探讨 LLM 在 DeFi 协议中的应用潜力. 第 4 节针对当前领域存在的挑战进行深入分析, 并展望未来的研究方向. 第 5 节对全文内容进行总结.

2 智能合约与 DeFi 协议漏洞类型

智能合约与 DeFi 协议在推动去中心化应用 (decentralized application, DApp) 发展过程中, 暴露出了一系列漏洞, 这些漏洞已被大量攻击者发现并利用. 智能合约作为 DeFi 协议的基础, 通过编写特定代码自动执行交易和管理资产的逻辑. DeFi 协议则由多个智能合约组成, 提供借贷、交易和流动性管理等金融服务和功能. 根据组成关系, DeFi 协议可分为两个层次: 关键组件与模块, 以及协议与衍生品. 关键组件与模块包括自动做市商 (automated market making, AMM)、收益耕作机制、治理机制、清算机制、流动性挖矿、稳定币及 NFT 与金融资产的标记化, 而协议与衍生品则涵盖去中心化交易所、借贷协议、收益聚集器、跨链桥和衍生品协议.

为了系统性地分析智能合约和 DeFi 协议的漏洞类型, 本文从智能合约层和 DeFi 协议层两个层面对漏洞类型进行分类和归纳. 我们使用 Zhou 等人[27]提出的 5 层漏洞分析框架作为分析基础, 包括网络、共识、智能合约、DeFi 协议和辅助服务. 本文的研究重点聚焦在智能合约层和 DeFi 协议层, 因为这两个层次的漏洞直接关系到智能合约的安全性与漏洞利用的实际风险, 具体研究框架如图 3 所示.

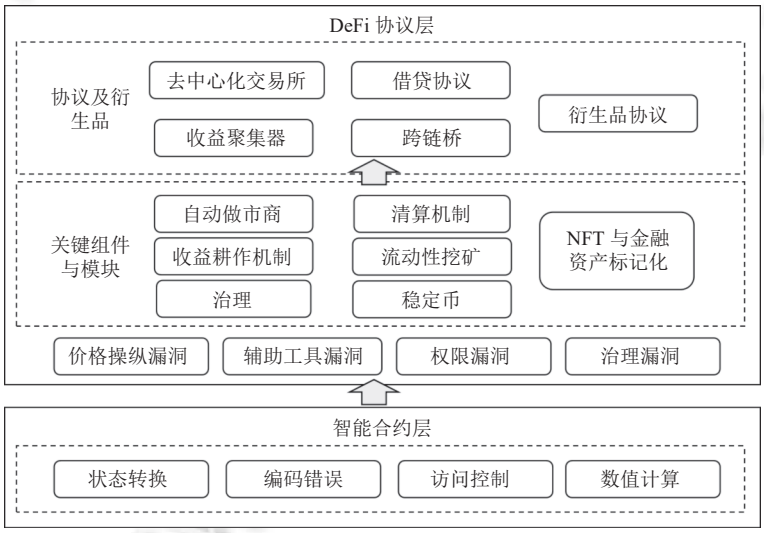

图 3 智能合约与 DeFi 协议漏洞检测研究框架

需要说明的是, 本文的研究对象是智能合约引发的漏洞, 不包括社会工程学、私钥泄漏等相关风险, 具体定义如下.

智能合约层漏洞: 智能合约是构成 DeFi 协议的基础, 其漏洞是指合约代码本身存在的缺陷. 智能合约层漏洞的具体类型包括状态转换类漏洞、编码错误类漏洞、访问控制类漏洞和数值计算类漏洞.

DeFi 协议层漏洞: DeFi 协议由一组合约组成, 其漏洞与 DeFi 应用及其场景特性密切相关. DeFi 协议层漏洞的具体类型包括价格操纵漏洞、辅助工具漏洞、权限控制漏洞和治理漏洞.

2.1 智能合约层漏洞类型

根据智能合约执行过程中不同的安全需求和常见攻击向量, 本文将智能合约层的漏洞特征归纳为 25 种主要的漏洞类型, 并将这些漏洞类型按其表现形式分为 4 类, 包括状态转换与调用安全、编码错误、访问控制和业务逻辑. 其中, 状态转换与调用安全、编码错误以及访问控制均为智能合约传统安全漏洞, 而业务逻辑则为智能合约业务逻辑漏洞. 我们将漏洞分类、漏洞类型、漏洞描述和相应的检测工具整理在表 1 中.

表 1 智能合约层漏洞与检测工具

漏洞分类	漏洞名称	漏洞描述	检测工具来源于以下文献
状态转换	定价不当的操作码	操作码定价不当, 导致智能合约在执行时消耗的Gas过低, 增加了潜在的攻击面	暂无检测工具, 定义见[29]
	编译器错误	由于编译器版本过时、本身的缺陷或误操作, 导致生成字节码时引入缺陷	[30–32]
	时间戳依赖漏洞	当函数将区块时间戳作为执行关键操作的条件时, 攻击者操控时间戳来影响合约行为	[5,30,33–43]
	调用栈溢出漏洞	合约执行过程中递归调用或深度链式调用超过了EVM允许的调用栈深度限制	[44,45]
	拒绝服务漏洞	通过消耗智能合约的资源 (如计算、存储或Gas), 使合约无法正常处理其他用户的请求	[36,46–49]
	直接调用不可信的合约	直接调用未经验证或不可信的合约, 导致意外的行为或被攻击	[50]
	重入漏洞	对外部合约函数存在递归调用, 导致合约的资金被多次转出或状态被不正确地多次更改	[33,34,37–43,49,51–55]
	委托调用漏洞	合约A可以在其当前执行环境中执行合约B的代码, 导致合约A的状态和权限等被恶意篡改	[30,33,37,38,40,56–58]
	Tx.Origin验证漏洞	使用Tx.Origin作为智能合约身份验证依据, 导致授权不当	[32,36,39,40,42,47,59,60]
	不一致状态更新	由于状态变量之间的相互关联性, 开发者在更新一个变量时可能会忘记同步更新关联变量或更新错误, 导致统计错误或资金损失	[54,61,62]
	原子性违规	业务流程未确保必要的业务级原子性, 导致并发操作对相同状态变量产生冲突	[54,62]
编码错误	异常处理不当	未能正确处理执行过程中出现的异常情况, 导致合约逻辑中断、状态不一致或资金损失	[30,37,39,40,46,52,63–67]
	资产锁定或冻结	由于编码错误, 合约中的资产被意外锁定或冻结, 无法正常使用	[35,38,40,57,66,68,69]
	缺乏编码逻辑或一致性检查	缺乏必要的逻辑验证或一致性检查, 导致不合理的操作或被攻击	[59,64,65,68–70]
	短地址漏洞	发送少于20字节的地址, 而合约代码没有进行长度检查, 导致交易参数错误地解析和拼接	暂无检测工具, 定义见[71]
	未知函数调用漏洞	没有正确实现fallback函数或没有妥善处理未定义的调用, 导致资金丢失或非预期调用	[47,72]
	无界或Gas费用高的操作	存在未限制执行次数或消耗大量Gas的循环或递归操作, 导致交易因超出Gas限制而失败	[5,42,45,54,57,69,73,74]
	类型混乱漏洞	由于错误地使用或处理数据类型, 导致程序行为不符合预期的漏洞	[47,75]
	类型转换错误	将一种数据类型错误地转换为另一种不兼容的类型, 从而导致程序行为异常	[49,54,65]

表 1 智能合约层漏洞与检测工具 (续)

漏洞分类	漏洞名称	漏洞描述	检测工具来源于以下文献
访问控制	不一致的访问控制	合约中不同部分对访问权限的控制不一致, 导致未经授权的用户能够访问或操作受限资源	[31,55,64,65,68,69,76]
	可见性错误和不受限的操作	由于可见性设置不当或对关键操作缺少限制, 导致未授权的操作被执行	[49,53,66,68,70,77,78]
	ID唯一性违规	未能确保ID字段唯一性, 从而允许攻击者冒充实体或创建虚假/重复资产, 进而绕过访问控制检查进行非法操作	[10,54]
数值计算	整数溢出或下溢	在算术运算时结果超出整数的表示范围, 导致数值环绕回起始值 (溢出) 或最大值 (下溢)	[30,31,38–40,49,52,54,56,58,63,65,66,69,73,79]
	会计错误	由于底层业务模型公式的错误实现, 导致数值计算中产生细微误差, 在多次执行后累积成严重的经济损失	[62,80,81]
	其他算术错误	其他在执行数学运算时出现的错误, 导致计算结果不正确, 进而影响合约的逻辑或状态	[31,45,63,65,76,82]

2.1.1 状态转换

状态转换类漏洞主要涉及智能合约在执行过程中的状态管理与函数调用的安全性. 状态转换是智能合约核心功能之一, 直接决定了合约内部状态的更新方式[83]. 任何状态转换设计上的错误都可能对合约的整体行为产生影响, 进而妨碍其预期功能的正确执行. 这不仅威胁到合约的完整性和安全性, 还可能导致不可预见的经济损失. 此外, 智能合约在执行过程中需要与外部合约或函数进行交互. 这些外部调用如果缺乏适当的安全保障或未经过严格的验证, 可能为攻击者提供可乘之机, 从而对合约实施恶意攻击.

具体而言, 定价不当的操作码[29]和拒绝服务漏洞[36]主要影响合约执行时的资源消耗, 可能导致状态转换无法正常进行, 进而阻止合约正常处理请求. 编译器错误[30]、时间戳依赖漏洞[33]以及调用栈溢出漏洞[44]则可能导致合约状态发生意外更改, 进而中断合约的正常服务. 直接调用不可信的合约[50]、重入漏洞[51]、委托调用漏洞[58]以及 Tx.Origin 验证漏洞[59], 则是由于函数调用中的安全隐患引发的, 可能导致合约的逻辑被篡改或权限被不当提升.

此外, 不一致状态更新[61]由于开发者在多个模块之间状态同步遗漏, 导致在一个模块中更新的状态未能及时传播到其他相关模块, 这种状态不一致性可能引发连锁反应, 导致合约无法正确执行其预期功能. 例如在抵押借贷合约中, 若更新用户抵押金额后未重新计算抵押率 (仅执行抵押金额却遗漏抵押率参数更新), 将造成抵押品充足性误判, 引发错误清算或超额借贷等资金风险. 而原子性违规[62]则是指业务流程未确保必要的原子性, 导致并发操作对相同状态变量产生冲突. 具体而言, 当多个合约之间的操作依赖于同一组数据时, 若未妥善处理可能出现的失败或竞争条件 (如在部分交易完成后, 另一个交易失败), 将导致整体交易处于不一致状态, 进而引发严重后果. 这类问题通常发生在跨合约事务或并发操作的典型场景中, 例如: 源链上的资产销毁成功, 但目标链上铸造失败, 导致资产永久丢失; 或在 NFT 拍卖合约中, 分步更新最高出价时, 由于交易打包顺序的差异, 导致最终状态被错误覆盖, 从而错误确定获胜者. 这类漏洞通常源于未对多合约交互或并发操作设置原子化事务 (例如哈希时间锁或状态互斥锁), 从而无法消除中间态暴露所带来的风险.

2.1.2 编码错误

编码错误类漏洞指在智能合约编写过程中, 由于开发者的疏忽或对合约逻辑的误解而引发的代码逻辑错误或实现缺陷. 这些错误涉及异常处理、类型转换、边界检查以及 Gas 费用管理等多个方面, 直接影响合约的计算与逻辑执行, 进而可能导致数据损坏、操作错误或资金损失等严重后果.

在实际应用中, 编码错误类漏洞可能引发多种安全问题. 例如, 异常处理不当[30]往往会导致合约在遇到意外情况时未能正确处理, 进而导致服务中断或交易失败. 类似地, 短地址漏洞[71]则由于地址解析不当, 可能使攻击者能够利用地址长度不足的输入数据来操纵合约逻辑, 导致安全隐患. 更为复杂的编码错误还包括类型转换错误[65]和未知函数调用漏洞[72]. 这些错误可能导致合约在处理不同数据类型时产生不可预见的行为, 或者在调用未定义的函数时触发异常, 最终影响合约的正常功能. 类型混乱[47]和无界循环操作[74]等问题则会使合约在执行过程中消耗过多的资源 (如 Gas), 导致交易失败或合约崩溃.

2.1.3 访问控制

访问控制类漏洞主要涉及对智能合约中资源、函数或数据的访问权限进行管理. 如果权限管理不当, 可能导致未授权用户执行应受限制的操作, 从而危及合约的安全性. 具体而言, 访问控制的漏洞包括访问控制不一致[31]、可见性错误和不受限[77]的操作, 导致未经授权的用户或合约可以访问受限资源或执行敏感操作. 此外, ID 唯一性违规[10]则指未能确保 ID 字段的唯一性, 使攻击者能够冒充实体或创建虚假资产, 从而绕过访问控制进行非法操作. 访问控制类漏洞不仅影响合约的安全性, 还可能导致资产非法转移和其他严重后果.

2.1.4 数值计算

数值计算类漏洞主要涉及智能合约在执行数学运算时的准确性和可靠性. 常见的漏洞类型包括整数溢出或下溢和会计错误. 整数溢出或下溢[56]发生在算术运算的结果超出整数的表示范围, 导致数值环绕回起始值 (溢出) 或最大值 (下溢). 这种情况可能导致合约状态出现不可预测的变化, 从而影响合约的逻辑执行. 会计错误[80]则是由于底层业务模型公式的错误实现, 导致数值计算中产生细微的误差. 这些小误差在多次执行后可能累积成严重的经济损失, 严重影响合约的财务稳定性. 此外, 其他算术错误指在执行数学运算时出现的各种错误, 这些错误会导致计算结果不正确, 进一步影响合约的逻辑或状态.

2.2 DeFi 协议层漏洞类型

随着 DeFi 协议在区块链生态系统中的广泛应用, 其安全性问题逐渐成为关注的焦点. DeFi 协议的复杂性、多合约交互的特性以及对外部数据源的依赖, 使得它们比传统的智能合约面临更多、更复杂的安全挑战. 表 2 展示了 DeFi 协议层漏洞的分类、漏洞利用方式和对应的安全事件描述, 本节将详细介绍其中所列的 DeFi 协议层漏洞.

表 2 DeFi 协议层漏洞类型分类

漏洞分类	细分类别	漏洞利用方式	描述
价格操纵漏洞	交易顺序依赖	抢先交易	攻击者通过提前执行交易, 影响市场价格或获取超额利润. https://www.coindesk.com/sparkpool-taichimining-network-front-running-defi
		三明治攻击	通过在目标交易前后分别进行交易, 操纵价格波动. 攻击者在受害者的交易前后分别执行买入和卖出操作, 利用交易间的价格波动从中获利. 这种攻击主要通过操纵流动性池中的资产价格来实现. https://www.coindesk.com/markets/2022/10/12/how-market-manipulation-led-to-a-100m-exploit-on-solana-defi-exchange-mango
		清算攻击	攻击者利用市场价格的异常波动, 迫使某些抵押品的价值快速下降, 导致大量资产被强制清算. 攻击者在清算过程中通常会以低于市场价格的价格购买这些被清算的资产, 从而获利. https://medium.com/perpdex/liquidation-related-attacks-to-perpetual-future-protocols-6457bb6693de
	基于AMM价格滑点的价格操纵	哄抬价格与套利	攻击者通过操控交易量、制造虚假需求或供应等手段操纵市场价格制造价格波动, 并利用这些波动进行套利. https://etherscan.io/tx/0xb5c8bd9430b6cc87a0e2fe110ece6bf527fa4f170a4bc8cd032f768fc5219838
		价格预言机操纵	攻击者通过操控链上价格数据源来影响智能合约依赖的价格预言机, 这种攻击通常发生在链上操作, 通过实时的价格变化对链上依赖价格的协议产生直接影响. https://etherscan.io/tx/0x35f8d2f572fceaac9288e5d462117850ef2694786992a8c3f6d02612277b0877
	市场供需、代币量相关	通缩代币操纵	攻击者通过大量购买通缩代币, 从而推高价格, 再在价格高位抛售以获取利润. 由于代币的供应量减少, 价格波动可能会更加剧烈. https://medium.com/balancer-protocol/incident-with-non-standard-erc20-deflationary-tokens-95a0f6d46dea
		通胀代币操纵	攻击者通过协议漏洞或故意制造代币的通货膨胀, 从而使代币供应量迅速增加, 导致代币价值急剧下跌, 最终对协议和持币者造成损害. https://decrypt.co/71585/pancakebunny-defi-exploit
辅助工具漏洞	跨链组件相关	跨链桥攻击	由于跨链桥的资产锁定、释放或验证机制中的漏洞, 导致资产丢失或被盗. https://medium.com/thorchain/certik-completes-thorchain-audit-c6d88fad3613
	预言机相关	链外预言机操控	攻击者在链外的数据源层面干预或欺骗预言机, 使其传回链上的数据被操控. 例如, 攻击者可能干扰预言机的数据来源 (如外汇市场、商品价格数据等), 使预言机传递的链上数据与实际市场情况不符. https://beosin.com/resources/beosin-detailed-analysis-of-fortresss-oracle-attack?lang=zh-HK

表 2 DeFi 协议层漏洞类型分类 (续)

漏洞分类	细分类别	漏洞利用方式	描述
权限漏洞	权限滥用	拉地毯攻击	攻击者通过吸引投资者投入资金后, 突然将流动性池中的所有资金抽走, 导致投资者资金损失. https://bscscan.com/tx/0xc346adf14e5082e6df5aeae650f3d7f606d7e08247c2b856510766b4dfcdc57f
	设计缺陷	权限提升	攻击者可以创建一个新的智能合约并诱使用户与之交互, 从而获得不当权限, 例如控制用户的资产或执行未经授权的操作. https://docs.cdfi.ai/unpacking-the-defi-security-breach-a-comprehensive-incident-report
		代币批准的无限权限	代币批准是指用户允许某个合约支配其代币的操作. 无限权限指用户无意中授予某个合约无限量的代币支配权. 当该合约被攻击或利用后, 用户的代币被完全提走. https://metamask.zendesk.com/hc/en-us/articles/6174898326683-What-is-a-token-approval-#h_01G6X0J818RMX8E35CCPE0KQEH
治理漏洞	N/A	治理攻击	攻击者通过购买大量治理代币, 获得足够的投票权, 从而影响治理决策. 这可能导致协议的参数被不当调整, 或者资金被用于不利于社区的目的. https://cointelegraph.com/news/key-takeaways-golden-boys-attack-compound-dao

2.2.1 价格操纵漏洞

攻击者可以利用价格操纵漏洞进行巨额获利, 如 2024 年 10 月 25 日的 Base 链上检测到一个针对未验证借贷合约的价格操纵攻击, 攻击者利用过度借贷获利约 100 万美元的代币[84]. 本文依据是否依赖交易顺序、是否依赖 AMM 机制的价格滑点等特征将价格操纵漏洞分为交易顺序依赖、基于 AMM 价格滑点的价格操纵与市场供需、代币量相关这 3 种类别. 下面将分别介绍这 3 种类别对应的漏洞利用方式.

交易顺序依赖类型是价格操纵漏洞中常见且危害较大的漏洞类别之一, 漏洞的利用过程和交易顺序紧密相关, 主要的漏洞利用方式包括抢先交易、三明治攻击和清算攻击.

抢先交易[85]: 在区块链上, 交易在被矿工打包进区块前是公开可见的. 攻击者可以通过监控这些待确认交易, 预先提交一笔带有更高 Gas 费的交易, 以确保其交易优先执行. 这种攻击方式特别适用于竞争激烈的市场活动中, 如代币抢购、拍卖或套利, 攻击者通过抢先交易锁定优势, 获取超额利润. 例如, 在去中心化交易所 (decentralized exchange, DEX) 中, 攻击者可以在用户提交一笔大额买单前抢先购买代币, 然后在价格上涨后迅速卖出, 从中获利.

三明治攻击[86]是抢先交易的复杂变体, 攻击者通过在目标交易前后分别提交一笔买入和卖出交易来操控市场价格. 例如, 当用户提交一笔大额买单时, 攻击者先行提交一笔买单推高价格, 待用户的交易被执行后, 再卖出获取利润. 这种攻击不仅会导致用户遭受经济损失, 还可能影响市场价格的稳定性.

清算机制[6]是指在 DeFi 借贷平台中, 当抵押品的价值下降到一定阈值以下时, 抵押品会被强制清算卖出. 攻击者利用市场价格的异常波动, 迫使某些抵押品的价值快速下降引发清算机制, 导致大量资产被强制清算, 然后攻击者以低于市场价格的价格购买这些被清算的资产从而获利.

基于 AMM 价格滑点的价格操纵类型是在 AMM 机制的前提下, 通过大额交易引发价格滑点进行价格操纵, 主要的漏洞利用方式包括价格哄抬与套利和价格预言机操纵等. 同时, 闪电贷[87]是一种允许用户在单个区块内借入大量资金的机制, 这意味着借款人无需提供抵押品, 只要在同一交易中归还贷款即可. 由于具备在一笔交易中可以执行多个动作并且可以借贷大量代币的特性, 闪电贷成为实现基于 AMM 价格滑点的价格操纵的有力工具.

价格哄抬与套利[88]: 攻击者通过操控交易量、制造虚假需求或供应等手段来人为推高或压低市场价格, 从而在价格波动中获利. 例如, 攻击者可能会通过大规模的买入或卖出操作操纵价格, 在市场中造成波动, 然后利用这一波动在不同的交易对中进行套利. 这样的攻击在流动性较低的市场或交易对中更为有效, 因为这些市场对大额交易的价格敏感性更高.

价格预言机操纵[89]: 预言机是 DeFi 协议从外部世界获取数据的关键组件, 攻击者可以通过操控预言机的数据输入 (例如提供虚假价格信息), 误导协议做出错误的决策. 价格预言机操纵通常发生在链上数据系统中, 攻击者通过操控链上价格数据源来影响智能合约依赖的价格预言机. 这种攻击通常利用流动性较低的交易池或特定的去中心化交易所来制造价格波动.

市场供需、代币量相关的价格操纵漏洞是指 DeFi 协议机制方面存在缺陷, 使得攻击者通过改变市场供需和代币量来影响价格、扰乱市场. 具体的漏洞利用方式包括通缩代币操纵和通胀代币操纵.

通缩代币操纵[90]: 某些 DeFi 协议引入了通缩代币 (deflationary tokens), 即在每次交易时销毁部分代币以减少总供应量. 攻击者可能会利用这一机制, 通过控制代币的流通量来制造市场稀缺性, 从而推高价格, 并在高位时抛售获利. 这类操纵通常依赖于协议设计中的激励机制不平衡, 容易被攻击者利用.

通胀代币操纵[91]: 攻击者通过协议漏洞或故意制造代币的通货膨胀, 从而使代币供应量迅速增加, 导致代币价值急剧下跌, 最终对协议和持币者造成损害. 具体的攻击方式包括利用协议中代币铸造功能的漏洞不受限制地生成新代币、操纵奖励机制使代币超额分发等.

2.2.2 辅助工具漏洞

辅助工具漏洞是指预言机、跨链桥等辅助层的组件存在的漏洞, 这些组件可以被攻击或者操纵从而形成跨链桥攻击和链外预言机操控等攻击方式, 帮助攻击者获利. 例如 2022 年 8 月 1 日, Nomad 跨链桥遭到黑客攻击, 1.9 亿美元的锁定资金被盗. 在一名攻击者率先利用该漏洞并大获成功后其他黑客也纷纷效仿[92].

跨链桥[93]是连接不同区块链平台之间的桥梁, 允许资产和信息在不同链之间进行转移. 由于跨链桥的复杂性, 其安全性成为 DeFi 生态系统中的一个重要问题. 跨链桥漏洞主要存在于资产锁定和释放机制、验证机制以及多链交互的协调机制这 3 个方面. 在资产锁定和释放机制方面, 涉及将资产从源链锁定, 并在目标链上释放相应的资产代币. 如果锁定或释放机制存在漏洞, 攻击者可能通过伪造交易或重复请求资产释放, 导致资产丢失或被盗. 从技术上, 锁定机制需保证资产在源链的不可篡改性, 而释放机制必须确保仅在合法条件下才释放资产. 这对智能合约的执行和状态同步提出了挑战, 特别是在跨链交易涉及多方和多合约时, 可能出现合约间状态不一致或错误释放的风险. 在验证机制方面, 跨链桥的安全性高度依赖于验证机制的完善性. 若验证节点或机制受到攻击, 攻击者可能通过提交虚假跨链交易或篡改验证结果, 导致资产被非法转移或丢失. 跨链验证过程中, 存在节点间信息延迟和同步的挑战, 攻击者可利用时间窗口进行中间人攻击或延迟攻击. 在多链交互的协调机制方面, 跨链桥需要协调多个链之间的数据交互和资产转移, 如何保证跨链交互的原子性、可靠性和一致性是面临的核心挑战. 此外, 不同链之间的安全性和隐私保护机制存在差异, 这要求跨链桥具备灵活且统一的安全策略来处理多链之间潜在的安全漏洞.

链外预言机操控[94]是指攻击者在链外的数据源层面干预或欺骗预言机, 使其传回链上的数据被操控. 例如, 攻击者可能干扰预言机的数据来源 (如外汇市场、商品价格数据等), 使预言机传递的链上数据与实际市场情况不符.

2.2.3 权限漏洞

权限漏洞是另一类存在于 DeFi 协议中的漏洞, 主要分为权限滥用和设计缺陷两大类. 权限滥用指的是攻击者通过操控某些具有权限的操作来窃取资产或执行未授权操作, 经常发生于具有高级权限的内部人员之中. 设计缺陷指的是智能合约在设计中存在的逻辑问题, 导致特定权限没有得到合理的限制或保护, 从而被攻击者利用. 2024 年 2 月 9 日, 区块链游戏平台 PlayDapp 遭攻击, 黑客入侵了 PlayDapp (PLA) 代币智能合约. 黑客非法获取了私钥, 从而改变了智能合约的所有权和铸币权限, 将其转移到自己的账户上. 黑客移除了现有管理员的授权, 并非法铸造了 2 亿个 PLA 代币[95].

拉地毯攻击[96]是一种常见于 DeFi 生态系统中的恶意行为, 通常发生在 DEX 或流动性池中. 攻击者通常是项目的创建者或开发团队, 在吸引了大量投资者和资金进入他们的项目后, 突然撤走项目的资金或流动性, 导致投资者的资产迅速贬值或完全丧失. 拉地毯攻击具备快速性、匿名性和高收益的特点, 攻击通常发生得非常快, 往往在数小时内完成, 使得投资者没有时间反应.

权限提升[97]是指攻击者利用协议的设计缺陷提升自身权限, 例如攻击者可以创建一个新的智能合约, 并诱导用户与之交互, 然后在交互过程中获得不当权限, 进而实现可以未经授权控制用户的资产或执行其他未授权的操作.

代币批准的无限权限[98]: 无限批准是一项功能, 使用户能够授予平台和智能合约代表您无限制地使用代币/硬币的权限. 通常, 在特定 AMM (例如 Uniswap) 上进行交换时, 用户需要批准智能合约/平台代表他们转移这些代币. 在这种情况下, 代币批准的无限权限允许用户将某合约设置为具有无限量的代币支配权. 这可能会被攻击者利

用, 一旦该合约被恶意利用, 用户的代币可能会被完全提走.

2.2.4 治理漏洞

在去中心化治理中, 项目通常会发行治理代币, 持有这些代币的用户可以参与投票, 决定项目的参数更改、资金分配以及重要协议升级. 一般而言, 治理提案会依据代币持有者的投票结果来决定是否通过. 治理攻击正是指利用该机制的漏洞来影响投票过程来达成攻击目的. 例如 2022 年 4 月 17 日, DeFi 项目 Beanstalk Farms 被盗取 1.82 亿美元, 原因是一名攻击者以闪电般的速度进行了恶意收购, 购买了代币的控股权, 并立即投票决定将所有资金都发给自己[99].

治理攻击[100]的手段包括代币囤积、恶意提案和治理篡改等. 代币囤积是指攻击者在短时间内购买大量的治理代币, 以增加自己的投票权, 确保在关键提案上占据主导地位. 恶意提案是指攻击者通过提交对自己有利的提案, 例如将项目资金转移到自己的地址、修改协议中的关键参数, 并且可以利用囤积的治理代币确保提案通过. 治理篡改是指攻击者试图改变治理规则本身.

3 智能合约与 DeFi 协议漏洞检测方法

本节针对智能合约层和 DeFi 协议层的漏洞分别进行检测方法的介绍. 在智能合约层, 由于该领域的漏洞类型相对明确且检测技术较为成熟, 现有方法已能有效覆盖多种漏洞类型. 因此, 本文重点关注基于深度学习和大型语言模型等新兴检测方法的研究进展. 在 DeFi 协议层, 由于漏洞类型之间差异较大, 大多数检测方法只能针对一种漏洞类型. 值得注意的是, 部分重要 DeFi 协议漏洞类型 (如治理漏洞) 仍缺乏有效的检测手段, 亟待进一步研究突破.

3.1 智能合约层漏洞检测方法

本节在前期智能合约漏洞检测技术及相关综述的基础上[18,25], 将智能合约层漏洞检测方法归纳为 4 类: 静态分析方法、模糊测试方法、基于深度学习的方法和基于 LLM 的方法. 随后, 分别对传统智能合约漏洞检测方法和基于 LLM 的智能合约漏洞检测方法进行总结, 并深入分析 LLM 在智能合约漏洞检测领域的发展潜力.

3.1.1 静态分析方法

由于形式化验证、符号执行、污点分析以及基于中间表示的核心检测功能均在程序未执行的状态下进行, 本文将其统一归类为静态分析方法. 本节将重点阐述这些技术的定义与原理, 并直观展示其在漏洞检测中的具体应用. 智能合约漏洞检测的静态分析方法工作流程如图 4 所示.

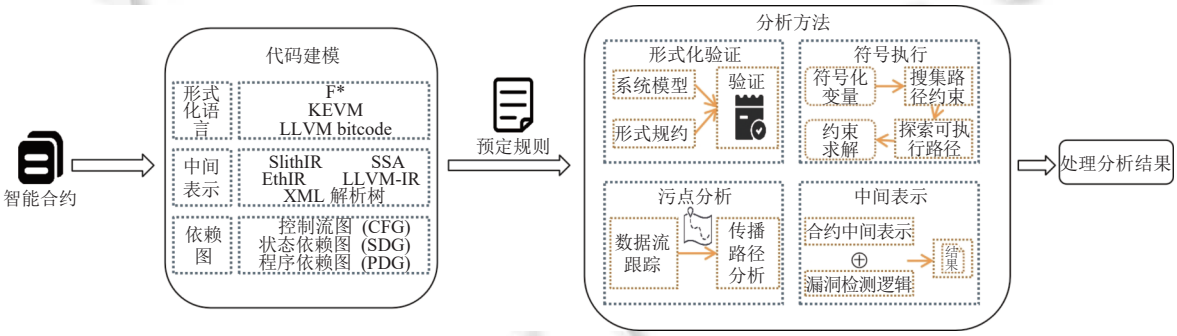

图 4 智能合约静态分析方法工作流程

3.1.1.1 形式化验证

形式化验证是一种通过数学证明确保系统行为符合预期的技术, 近年来已广泛应用于智能合约的安全性和功能正确性验证. 智能合约的形式化验证旨在通过严谨的数学方法排查潜在漏洞, 确保合约在各类场景中的安全性和可靠性[101].

Bhargavan 等人[102]于 2016 年在 PLAS 会议上提出基于形式化验证的框架, 专门用于验证智能合约的安全性和功能正确性. 该框架通过将智能合约翻译为 F 语言 (F*), 在源代码和字节码级别进行形式化分析. 研究团队开发

了 Solidity 到 F*的翻译工具和 EVM 字节码的反编译器, 确保在不同级别上对合约进行形式化验证. 此研究为智能合约形式化验证提供了一套完整的工具链, 奠定了该领域进一步发展的基础.

Kalra 等人[103]提出了 ZEUS, 在延续文献 [102] 工作的基础上, 进一步丰富了形式化验证的方法. ZEUS 结合了抽象解释和符号模型检测, 用于智能合约的验证. 该工具通过将智能合约转换为 LLVM bitcode 的中间表示, 结合抽象解释和符号模型检测技术进行验证. 抽象解释用于计算数据域上的循环和函数摘要, 而符号模型检测则在缩减的状态空间中进行验证, 以确保智能合约的安全性. ZEUS 通过精确的分析和有效的状态空间缩减, 显著提升了智能合约验证的准确性和扩展性, 特别是在处理复杂合约时表现优异.

文献 [104] 提出的 KEVM 是基于 K 框架构建的 EVM 形式化语义模型. K 框架是一种用于定义可执行语义规范的重写系统, 研究团队通过该框架定义了 EVM 的语法、状态配置和执行规则, 构建了完整的 EVM 形式化语义. KEVM 的创新之处在于其不仅通过了所有官方测试套件, 成为首个完整的 EVM 可执行规范, 还扩展了多种形式化分析工具, 包括调试器、符号执行引擎和演绎验证器, 这为 EVM 的形式化分析提供了强有力的支持.

后续还有基于其他特征进行形式化验证的研究, 如基于静态程序切片框架的 HoRStify[41]. HoRStify 通过分析程序依赖图, 捕捉智能合约中变量和节点之间的依赖关系, 并通过逻辑编码和约束霍恩子句定义安全性模式, 以验证合约的安全属性.

3.1.1.2 符号执行

符号执行是一种在智能合约漏洞检测中广泛应用的技术, 其核心是将程序变量抽象为符号输入, 并通过模拟程序的执行路径, 分析可能导致漏洞的输入条件[105].

Luu 等人[5]于 2016 年提出了 Oyente, 这是首个使用符号执行技术检测智能合约漏洞的工具. Oyente 直接处理 EVM 字节码, 通过构建控制流图和执行符号状态, 利用 Z3 求解器分析路径约束, 从而检测交易顺序依赖、时间戳依赖、异常处理不当和重入漏洞等问题. 在此基础上, Tsankov 等人[68]提出的 Securify 通过将 EVM 字节码反编译为静态单赋值形式的无栈表示, 推导出语义事实, 以验证合约的安全性. 尽管 Securify 在处理数值属性推理上有局限, 但它有效减少了用户在分类警告时的工作量, 成为智能合约安全分析中的重要工具.

后续的研究继续丰富了符号执行检测方法, 如 TEETHER[77]和 SAILFISH[55]系统. Krupp 等人[77]开发了 TEETHER, 这是首个能够自动识别智能合约漏洞并生成攻击载体的工具. TEETHER 通过逆向工程字节码, 构建控制流图, 使用符号执行分析关键指令, 并生成可利用的攻击路径. Bose 等人[55]提出了 SAILFISH 系统, 专注于检测智能合约中的状态不一致性漏洞. SAILFISH 采用混合分析方法, 通过轻量级探索阶段识别潜在漏洞, 并在精炼阶段通过符号求解和值摘要分析验证漏洞的真实性.

3.1.1.3 污点分析

污点分析是一种程序分析技术, 旨在追踪数据从源头到达目标之间的流动情况. 在智能合约漏洞检测中的应用, 通过标记和跟踪“污点 (可能受外部输入影响的数据)”, 污点分析可以帮助识别检测可能导致安全问题的路径[106].

Torres 等人[107]于 2018 年提出了 Osiris 工具, 专门用于检测以太坊智能合约中的整数溢出漏洞. 该工具基于智能合约的字节码构建控制流图, 利用符号执行模拟合约的执行路径, 检测可能的整数溢出操作. 为了减少误报, Osiris 使用污点分析技术追踪未经验证的输入 (污点) 在合约中的传播路径, 如果这些污点影响到了溢出操作, 该操作便被标记为潜在漏洞.

Ethainter 工具[108]用于检测以太坊智能合约中的复合信息流漏洞. Ethainter 在信息流分析的基础上, 结合了污点分析技术, 详细建模合约中的数据清理机制, 追踪未清理的数据在合约中的传播路径, 从而识别出可能绕过守卫机制的漏洞. 同样是基于信息流分析, SmartACE 框架[109]用于验证复杂智能合约, 尤其是那些涉及大量用户的合约. 在污点分析阶段, SmartACE 利用局部抽象技术, 有效地捕捉潜在的漏洞路径, 从而提高了合约验证的效率和精度.

后续也出现了针对特定漏洞的污点分析检测方法, 如 Ghaleb 等人[78]提出的 AChecker 工具, 专注于检测智能合约中的访问控制漏洞, 尤其是那些缺乏明确访问控制策略说明的合约. AChecker 首先通过静态数据流分析识别出合约中的访问控制检查点和相关状态变量, 然后利用跨过程的上下文敏感污点分析, 追踪用户输入对这些关键指令或状态变量的影响.

3.1.1.4 中间表示

利用中间表示法检测智能合约漏洞是一种通过将合约代码转换为中间表示 (intermediate representation, IR) 的形式, 以便进行更高效、更精确的分析和验证. 这种方法能够抽象出智能合约代码的核心逻辑和操作, 简化复杂性, 从而更容易检测潜在的安全漏洞. 其核心思想是将智能合约的源代码或字节码转换为一种标准化的、较低级别的表示形式. 这种表示形式通常无关于具体编程语言, 更接近于机器代码, 但仍保留了程序的逻辑结构和数据流信息.

SmartCheck[47]使用 XML 解析树作为中间表示, 它是由 Tikbomirov 等人在 2018 年提出的一种基于静态分析的智能合约漏洞检测工具. SmartCheck 对 Solidity 源代码进行词法和语法分析, 生成 XML 解析树作为中间表示, 然后利用 XPath 查询在中间表示上检测特定的漏洞模式, 从而实现对合约的全面代码覆盖和漏洞检测.

Slither[82]将 Solidity 合约转换为 SlithIR 作为中间表示来实现静态分析. 该工具专门用于检测以太坊智能合约中的常见安全漏洞. SlithIR 采用静态单一赋值形式, 并配备简化的指令集, 以便于分析合约的语义信息. Slither 能够应用常见的程序分析技术, 如数据流分析和污点追踪, 同时支持代码优化与漏洞检测. 通过使用 SlithIR, Slither 提供了灵活且高效的分析平台, 显著提高了智能合约的检测精度并减少了误报率.

SmartState [48]和 EthIR[110]分别使用状态依赖图和规则基础表示作为中间表示. SmartState 框架由 Liao 等人[48]提出, 主要用于检测智能合约中的状态回滚漏洞. 该框架通过细粒度的状态依赖关系分析, 从智能合约的字节码和历史交易中提取状态依赖关系, 生成状态依赖图. 通过分析依赖图中的关键模式, SmartState 能够识别与状态回滚漏洞相关的潜在安全问题, 如缺乏适当访问控制的关键状态. EthIR 是由 Alber 等人[110]提出的以太坊字节码高级语义分析框架. 该方法通过逆向工程生成以太坊字节码的控制流图, 然后将其转化为 RBR. RBR 显式表示数据流, 并通过规则表示控制流, 从而将低级字节码信息转化为易于分析的高级表示. 通过这种中间表示, EthIR 能够利用现有的高级分析工具, 对智能合约的循环复杂度、资源消耗等性质进行自动化推理, 显著增强了字节码分析的能力.

3.1.2 模糊测试方法

模糊测试 (fuzz testing)[111]是一种自动化的软件测试技术, 用于发现程序中的错误和漏洞, 它也被广泛应用于智能合约的测试中. 总体来说, 模糊测试是一种通过向系统输入异常、随机或意外的数据来尝试揭示错误和漏洞的测试方法. 在智能合约的上下文中, 模糊测试涉及生成大量的随机输入数据, 并将这些数据作为交易发送到智能合约中, 以观察合约的响应并检测潜在的安全问题. 相关工作可以分为两种: 基于生成引导的模糊测试和基于突变引导的模糊测试. 智能合约漏洞检测的模糊测试方法工作流程如图 5 所示.

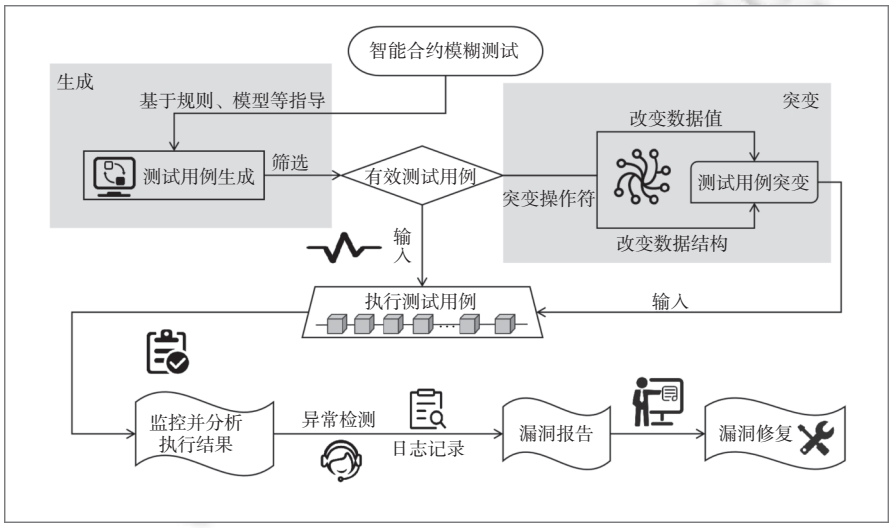

图 5 智能合约模糊测试方法工作流程

3.1.2.1 基于生成的方法

基于生成的模糊测试 (generation-based fuzzing) 是一种自动化测试方法, 它通过生成大量的测试用例来测试智能合约的功能. 这些输入数据通常是根据智能合约的输入格式和预期行为生成的, 目的是探索合约的所有可能执行路径, 通过生成大量的测试用例, 可以更全面地探索智能合约的输入空间, 以发现潜在的安全漏洞.

现有研究中, Wang 等人[52]介绍了 ContraMaster, 一个 Oracle 支持的智能合约动态漏洞生成框架. 由于现有的方法只改变单个事务, 而该方法通过改变事务序列来超越这些限制. 因此此框架使用数据流、控制流和动态契约状态来指导其变化. 然后监视目标合约程序的执行, 并根据通用语义测试 Oracle 验证结果, 以发现漏洞. 作为一种动态技术, 它保证每个发现的漏洞都违反了测试 Oracle, 并且能够生成攻击脚本来利用此漏洞.

同时 Zhou 等人[37]提出了一种方法 SmartGift, 它通过从现实世界智能合约的交易记录中学习, 为测试智能合约生成实际输入. 该方法从现实世界智能合约的交易记录中学习, 以便为被测新智能合约生成实际测试输入. 该研究进一步表明, SmartGift 能够为 77% 的智能合约函数生成相关的测试输入, 且生成的输入与通过模糊测试获得的输入是互补的.

在此基础上, Olsthoorn 等人[112]针对 Solidity 智能合约提出了 SynTest-Solidity, 这是一种用于 Solidity 的自动测试用例生成和模糊测试框架. 该框架通过应用先进的元启发式搜索算法, 自动化测试用例的生成过程, 帮助开发人员更加有效且高效地测试 Solidity 智能合约. 文章中还使用了基于链接学习的进化算法、MOSA 和 sFuzz 扩展框架, 并结合额外算法进行更广泛的评估.

考虑到当前智能合约模糊测试方法倾向于从合约初始状态开始进行测试, 导致在初始状态附近耗费过多能量, 且难以发现其他状态下的潜在错误, Liu 等人[38]提出了 3 种应对方案: (1) 在生成函数调用序列时, 明确考虑函数之间的数据依赖关系, 以促进对更丰富状态的探索. 通过附加新的序列 S2 来延长原有序列 S1, 使得附加序列 S2 能够从不同于初始状态的状态开始模糊测试. (2) 结合基于分支距离的度量, 迭代向目标分支进化测试用例. (3) 采用分支搜索算法发现稀有或脆弱分支, 并设计了一种能量分配机制以有效处理这些关键分支.

3.1.2.2 基于突变的方法

基于突变的模糊测试 (mutation-based fuzzing) 通过对已有的有效输入数据进行变异操作, 生成新的测试用例, 以探索智能合约在不同输入条件下的行为, 并揭示潜在的安全漏洞. 其核心思想是在一组已知的有效输入数据基础上进行小幅修改 (变异), 以创建新的测试用例. 变异操作可以涉及数据值或数据结构的改变. 通过这种方式, 测试能够覆盖更广的输入空间, 从而发现正常使用情况下可能未被触发的错误.

在突变过程中, 改变状态变量的复杂性和突变过程中对外部参数的处理给当前的智能合约模糊器带来了关键的技术挑战, 阻碍了它们在复杂约束下覆盖分支的能力, 并为攻击者留下了潜在的漏洞. 为了解决这些问题, Ji 等人[39]设计了一种结合两种新技术的引导突变策略 SeqFuzz: 动态依赖学习和动态变量分析. 动态依赖学习序列的依赖性, 为处理复杂约束中的状态变量提供指导性事务序列生成, 而动态变量分析则利用变量级动态污点分析来处理外部参数并引导突变.

Zhao 等人[45]提出了一种新的自适应突变调度框架 AMSFuzz. 该框架针对突变算子的随机选择, 能够自适应调整突变算子的概率分布, 以优化突变算子的选择过程. 对于突变位置的顺序选择, AMSFuzz 在模糊测试过程中动态地对种子进行不同大小的切片, 赋予更多种子优先突变的机会, 从而提高了模糊测试的效率. Sun 等人[79]的研究将突变测试应用于智能合约中的整数溢出漏洞测试, 提出了 5 种专门用于检测整数溢出漏洞的突变操作符.

在上述研究的基础上, Qian 等人[40]通过引入序列感知突变和种子掩码引导策略, 进一步阐明了智能合约模糊测试方法. 首先, 利用基于数据流的反馈机制, 以有意义的方式确定事务的执行顺序, 并引入序列感知突变技术, 以探索更深层次的状态. 然后设计了一种掩码引导的种子突变策略, 使生成的事务输入更加倾向于命中目标分支. 此外, 该文开发了一种动态自适应能量调整范式, 以在模糊测试期间平衡资源的分配.

为了将准确的分析扩展到真实世界智能合约, Rodler 等人[53]引入了 EFbroCF, 这是一种用于以太坊智能合约的高性能模糊器. 与以往的研究相比, EFbroCF 能够高效且准确地模拟复杂的智能合约交互场景, 并且具备极高的

模糊测试吞吐量. 为实现这一点, EFbroCF 将智能合约字节码编译为原生 C++代码, 借助现有的优化模糊测试工具链进行测试. 此外, EFbroCF 通过结构感知突变引擎生成智能合约事务序列, 并利用合约的 ABI 生成有效的事务输入, 从而显著提高了模糊测试的效率.

3.1.3 基于深度学习的方法

早期的智能合约安全分析工具大多依赖符号执行、形式化验证和模糊测试等, 这些工具虽然能够检测预定义的漏洞类型, 但随着智能合约领域发展迅速, 新型攻击方法和漏洞不断涌现, 规则库的更新往往滞后于实际的攻击方法变化, 导致检测方法的有效性降低. 鉴于深度学习具有较强的泛化能力, 近年来深度学习逐渐被广泛应用于智能合约漏洞检测领域, 并取得了很好的效果.

在利用深度学习模型进行智能合约漏洞检测时, 特征提取是关键步骤. 主要特征包括源代码、字节码、操作码、控制流图、抽象语法树以及注释等信息. 字节码和操作码代表智能合约的底层逻辑, 由 EVM 执行. 控制流图描述代码块之间的控制流关系, 帮助识别不安全的跳转路径和逻辑漏洞. 抽象语法树提供代码的语法结构, 有助于静态分析. 通过特征增强生成的注释可以为代码提供背景信息, 结合自然语言处理技术, 可提取与安全相关的语义内容. 根据深度学习模型所使用的特征, 本文将基于深度学习的智能合约漏洞检测方法分为两大类: 单模态检测方法和多模态检测方法. 单模态检测方法专注于对单一特征的深入分析, 而多模态检测方法则结合了上述多种特征进行分析. 基于深度学习的智能合约漏洞检测流程如图 6 所示.

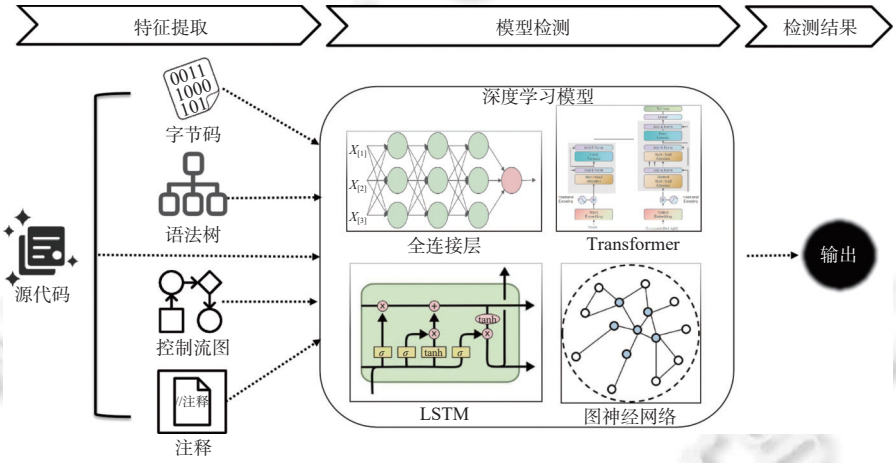

图 6 基于深度学习的智能合约漏洞检测流程

3.1.3.1 单模态检测

针对源代码分析层面, Liu 等人[113]于 2018 年提出了基于源代码的 S-gram 技术, 旨在解决现有安全审计工具在应对未知攻击和高计算成本方面的不足. S-gram 技术通过结合语言模型和语义分析, 提出了一种能够快速、准确检测智能合约漏洞的工具. 该方法通过轻量级静态分析生成语义元数据, 并结合类型词法分析, 实现了对智能合约的语义感知分析, 从而有效捕捉合约代码中的潜在漏洞.

在此基础上, Zhuang 等人[114]提出了一种利用图神经网络 (graph neural network, GNN) 进行源代码漏洞检测的方法, 主要针对重入、时间戳依赖和无限循环等漏洞. 该研究引入了 DR-GCN 和 TMP 两种模型, 通过将智能合约源代码转换为图结构并进行归一化处理, 使图神经网络能够更有效地捕捉代码中的语法和语义信息, 从而提高漏洞检测的准确性. 这项研究展示了图神经网络在智能合约安全检测中的潜力. 后续研究中, 涌现了许多基于图神经网络的源代码特征学习方法, 例如 Luo 等人[42]提出的 SCVHunter. SCVHunter 是一个基于异构图注意力网络的智能合约漏洞检测框架, 该框架通过将智能合约源代码转化为异构合约语义图, 利用 GNN 处理智能合约的结构化信息, 并通过节点级别和路径级别的双层注意力机制, 突出与漏洞相关的非良性节点, 从而提高了检测的准确

性. 另一方面, Chen 等人[34]提出的 Clear 方法采用对比学习来捕捉智能合约之间的细粒度关联信息, 并生成关联标签来指导模型训练, 从而提升了漏洞检测的准确性.

除了在源代码层面进行分析, 许多研究也基于操作码展开. Tann 等人[115]提出了一种基于 LSTM 模型的智能合约安全威胁检测方法, 主要针对智能合约的操作码进行分析, 利用序列学习技术快速检测新的攻击模式, 以提高智能合约的安全性. 这项研究首次将 LSTM 应用于智能合约的安全威胁检测, 展示了序列学习在该领域的潜力. 此外, Yu 等人[43]针对提高智能合约漏洞检测效率的问题, 提出了一种基于操作码的机器学习模型. 该研究通过提取智能合约操作码中的二元语法特征来构建特征空间, 结合多种机器学习算法 (如 XGBoost、AdaBoost、随机森林等) 对智能合约中的漏洞进行多标签分类, 从而有效提升了漏洞检测的效率和准确性.

3.1.3.2 多模态检测

在智能合约漏洞检测领域, 多模态检测方法通过结合不同模态的数据特征, 旨在提升检测的准确性. Qian 等人[33]提出了一种跨模态互学习框架, 进一步探索在源代码缺失的情况下, 通过字节码检测智能合约漏洞的能力. 该框架设计了一个教师-学生网络结构, 其中教师网络同时处理源代码和字节码的联合表示, 而学生网络仅处理字节码表示. 通过跨模态互学习策略, 学生网络从教师网络中学习源代码的语义信息, 从而在仅有字节码输入的情况下也能准确地进行漏洞检测.

在此基础上, Jie 等人[116]也提出了一种创新的多模态深度学习框架, 用于智能合约漏洞检测. 该方法结合了源代码层、构建基础层和 EVM 字节码层的特征, 采用 Word2Vec 和 BERT 模型生成代码嵌入, 使用图卷积网络 (GCN) 生成图嵌入, 并通过水平和垂直特征连接实现特征融合, 最终使用 Bi-LSTM 和随机森林模型进行决策. 这种多模态特征融合策略有效克服了现有方法在特征选择和融合方面的不足, 显著提高了智能合约漏洞检测的效果.

Cai 等人[49]基于合约代码的语法树和控制流图提出了一种细粒度智能合约漏洞检测方法, 通过自动化数据集构建和异构代码特征学习来提升检测精度. 该研究首先开发了一种基于安全最佳实践的自动化数据集构建方法, 有效生成带有漏洞标注的语句级数据集, 减少了人工标注工作量. 其次, 设计了一种语法敏感的图神经网络 (SS-GNN), 结合语法树和控制流图, 解决了语法与语义特征异构性的问题.

3.1.4 基于 LLM 的方法

通过利用 LLM 的语义理解和生成能力, 对智能合约代码及相关信息进行分析, 或者辅助漏洞检测流程, 从而实现对智能合约中潜在漏洞的高效识别与检测.

本文系统调研了当前基于 LLM 的智能合约漏洞检测方法, 并根据 LLM 在这些方法中的应用技术, 围绕其在漏洞检测过程中的不同用途, 将其归纳为两大类: 一类是将 LLM 作为主要检测引擎, 另一类是将 LLM 与传统方法相结合.

3.1.4.1 LLM 作为主要检测引擎

LLM 作为主要检测引擎的方法, 利用其强大的语义理解与生成能力, 对智能合约中的潜在漏洞进行分析与识别, 具体如图 7 所示. 尽管 LLM 在智能合约漏洞检测中展现出语义理解和逻辑推理的优势, 但其应用仍存在局限性. 首先, 生成误差问题限制了检测结果的可靠性. 例如, Chen 等人[17]通过在公开智能合约数据集 SmartBugs[117]上评估 ChatGPT 对 9 种漏洞的检测性能, 并设计了相应的提示语以辅助漏洞检测. 研究结果表明, ChatGPT 在实现较高召回率的同时, 精确定位智能合约漏洞的能力仍然有限, 并分析了 ChatGPT 产生误报的根本原因. 为缓解生成随机性与漏洞误报之间的矛盾, Hu 等人[118]提出了一种名为 GPTLENS 的框架, 将传统的单阶段检测划分为两个既对抗又协同的阶段: 生成阶段和判别阶段. 在生成阶段, LLM 作为审计者生成潜在漏洞和推理; 在判别阶段, LLM 则作为批判者, 对生成的漏洞进行评价与打分, 从而筛选出最可信的结果. 为了解决预训练模型在预训练任务和智能合约漏洞检测任务之间不匹配的问题, Yu 等人[119]提出了一种结合代码切片和提示调优的混合方法 PSCVFinder. 该方法首先通过代码切片识别包含漏洞匹配模式的代码行, 分析其周围的控制流和数据流关系, 然后生成仅包含关键漏洞相关代码的切片, 并对用户定义的变量和参数进行归一化处理. 在此基础上, 利用 CodeT5 预训练模型对代码片段进行编码, 将漏洞检测任务转化为填空问题, 通过设计提示模板来引导模型进行预测, 并设计特定的标签映射词汇, 显著提升了检测的准确性和效率. Du 等人[18]针对 GPT-4 设计了思维链 (chain of thought,

CoT) 提示, 以模拟专业审计员的审计流程, 并在 SolidiFI-benchmark 数据集[120]上评估了 7 种常见漏洞类型的检测效果. 实验结果表明, 虽然该方法的精确率高达 96.6%, 但召回率仅为 37.8%, 显示其在检测过程中存在遗漏漏洞的风险. 这种误差源于 LLM 的生成机制可能过度依赖训练数据的统计模式, 而非对代码逻辑的深层理解. 其次, 对训练数据的依赖性导致 LLM 在面对新型或罕见漏洞时表现不佳. 若训练数据缺乏某些漏洞类型, 模型的泛化能力将显著下降.

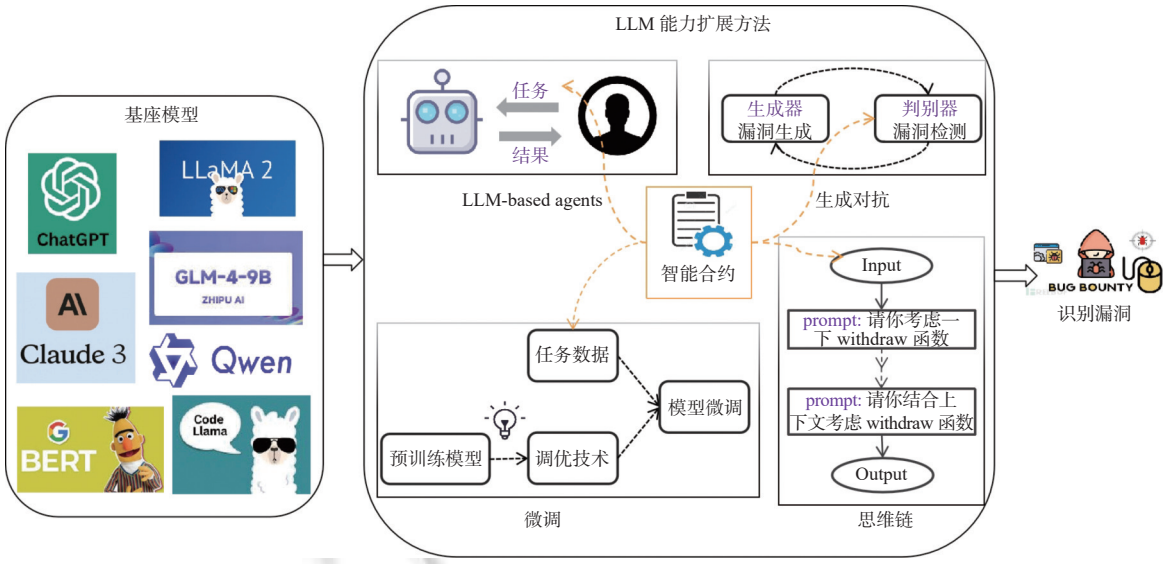

图 7 LLM 作为主要检测引擎的方法示意图

为检测逻辑漏洞, Ma 等人[121]提出了一种结合微调和基于 LLM 的代理的通用框架 TrustLLM. 该方法采用两阶段微调策略, 首先调整检测器模型以做出决策, 然后调整推理器模型以生成漏洞原因. 随后引入两个基于 LLM 的代理——Ranker 和 Critic, 根据微调后的 Reasoner 模型输出, 反复选择和辩论最合适的漏洞原因. Soud 等人[122]通过对代码进行切片、令牌化和语法分析, 将其转换为适合模型处理的格式, 并基于 RoBERTa[123]的 CodeBERTa[124]模型进行分析. CodeBERTa 是一种专为处理编程语言语义关系而设计的变体 Transformer 模型, 具有 6 层结构和 84M 参数. 该模型能够通过深度上下文理解有效捕捉代码中的漏洞特征, 并在包含 9 种逻辑漏洞的大型数据集上进行训练, 实验结果表明 Sóley 在漏洞检测方面表现优异.

3.1.4.2 LLM 与传统方法相结合

LLM 与传统方法相结合的应用, 通过将 LLM 的上下文理解、生成和推理能力融入传统漏洞检测技术, 增强了在处理复杂语义、逻辑推断和测试生成方面的效率与准确性, 从而优化了现有的检测流程.

在静态分析方面, 传统的形式化验证通常需要专家手动编写不变式、前置条件、后置条件和规则等属性, 这一过程耗时且易出错. Liu 等人[125]提出了一种新的方法 PropertyGPT, 旨在自动生成形式化属性, 以简化和加速形式化验证过程. 该方法利用如 GPT-4 的上下文学习能力, 将现有由人类编写的属性 (如来自 Certora 审计报告的属性) 转换并自动生成适用于未知代码的自定义属性. 整体流程具体如图 8 所示, 通过将现有属性嵌入向量数据库, 并检索参考属性辅助上下文学习, 生成新的属性. 在生成属性后, 方法通过编译和静态分析反馈对这些属性进行迭代修正, 确保其可编译、适用且可在运行时验证. Wang 等人[54]提出了一种精确且快速的智能合约不变性推断框架 SMARTINV, 设计了新的微调和提示策略思维树 (tree of thought, ToT) 以应用于基础模型, 能够跨越智能合约的多个模态进行推理, 并生成不变性属性, 然后通过检测生成的不变性属性是否被违反来定位潜在漏洞. 此外, Sun 等人[81]结合了 GPT 与静态分析, 提出了 GPTScan 以检测智能合约中的逻辑漏洞. 该方法首先对智能合约代码进行静态分析, 以确定可能涉及逻辑漏洞的函数, 然后通过多维过滤技术, 如调用路径、变量依赖等多个维度,

筛选出可能包含逻辑漏洞的候选函数对, 接着基于 GPT 分析候选函数对以及函数的输入输出、状态变量变化等属性, 识别出可能存在漏洞的函数对及与漏洞相关的关键变量和语句, 最终通过静态分析工具确认 GPT 识别出的漏洞类型、关键变量和语句, 确保漏洞真实存在.

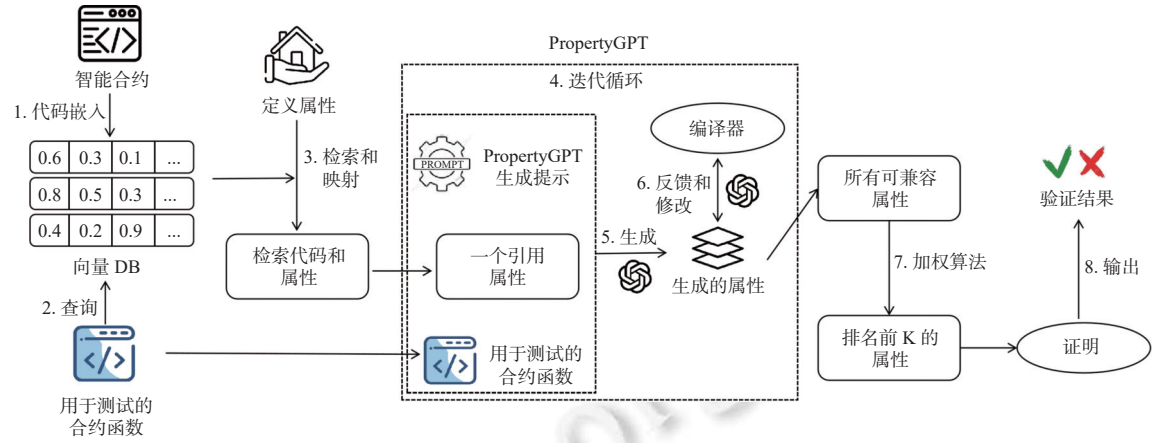

图 8 结合 LLM 生成形式化验证属性的 PropertyGPT 流程图

在模糊测试方面, 结合 LLM 进行优化的整体流程如图 9 所示, 针对传统模糊分析在探索广阔状态空间时效率较低的问题, Shou 等人[126]引入了 LLM4Fuzz, 通过利用 LLM 智能指导模糊测试活动并确定其优先级, 优化了智能合约模糊测试技术. 该方法首先利用静态分析提取智能合约的复杂度、状态变量依赖性、外部依赖性和控制流信息等属性, 为 LLM 提供必要的上下文信息, 然后引入复杂性、顺序可能性、漏洞可能性和不变性依赖性这 4 个模块, 利用 LLM 对代码片段进行评分, 并根据评分为每个测试用例分配“能量”, 通过调整这些“能量”优化模糊测试的优先级调度, 引导模糊测试聚焦于更可能发现漏洞的代码区域. 针对传统模糊测试通常依赖于随机生成初始种子, 导致种子覆盖率不足的问题, Sun 等人[127]提出了基于链式提示的生成方法. 该方法首先深入分析任务需求, 设计出覆盖关键信息的基础提示, 然后结合用户反馈和 LLM 生成的结果, 不断补充和调整提示内容, 以提高生成初始种子的逻辑性和连贯性. 同时, 该方案引入了基于生成器和评估器的对抗生成机制, 生成器基于链式提示生成方法生成智能合约初始测试用例, 评估器则负责检查并优化这些生成的测试用例. 通过生成器与评估器之间的交互和优化, 进一步提高了生成测试用例的质量, 增强了模糊测试的效率和覆盖率.

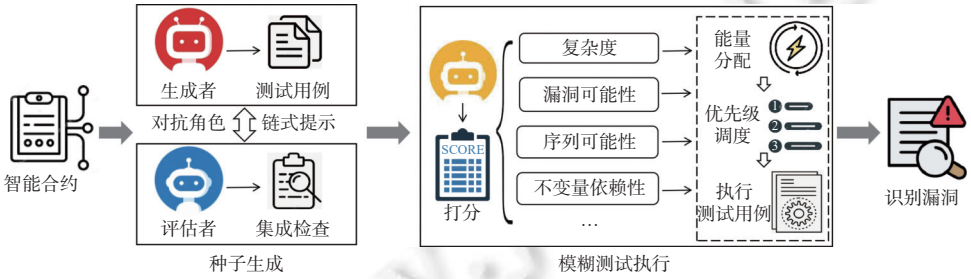

图 9 结合 LLM 优化的模糊测试流程示意图

3.1.5 传统智能合约层漏洞检测方法小结

传统智能合约层漏洞检测包括静态分析、模糊测试和基于深度学习的漏洞检测方法, 表 3 对这些方法根据类别、方法概述、发表会议/期刊以及发表年份进行了总结. 其中需要注意, 本文梳理传统智能合约层检测方法不包含涉及 LLM 的方法. 静态分析在不实际执行程序的情况下分析智能合约, 包括形式化验证、符号执行、污点分析和基于中间表示这 4 种类型的方法. 模糊测试通过向被测智能合约输入大量测试输入 (即测试用例), 并通过监测

报告的异常结果或异常情况来识别漏洞, 包括基于生成和基于突变的方法. 基于深度学习的方法通常是将智能合约进行特征提取, 然后构建深度神经网络作为检测模型, 包括单模态和多模态检测两类.

表 3 传统智能合约层漏洞检测方法

方法类别	方法名称	相关文献	方法概述	会议/期刊	发表年份
静态分析	形式化验证	[41]	基于静态程序切片框架, 通过逻辑编码和约束霍恩子句定义安全性模式, 以验证合约的安全属性	CSF	2023
		[103]	将智能合约转换为LLVM bitcode的中间表示, 结合抽象解释和符号模型检测技术进行验证	NDSS	2018
		[104]	通过所有官方测试套件的EVM可执行规范, 构建了完整的EVM形式化语义	CSF	2018
		[102]	开发了Solidity到F*的翻译工具和 EVM 字节码的反编译器, 以确保合约在不同级别的形式化验证	PLAS	2016
	符号执行	[55]	利用混合分析方法, 在探索阶段识别漏洞, 并在精炼阶段通过符号求解和值摘要分析验证漏洞真实性	SP	2022
		[68]	基于符号抽象的轻量级、可扩展智能合约分析工具	CCS	2018
		[77]	通过逆向工程智能合约的字节码, 构建其控制流图, 使用符号执行技术分析合约的执行路径	USENIX Security	2018
		[5]	构造控制流图, 执行符号状态, 通过约束进行可达性检验以降低误报率	CCS	2016
	污点分析	[78]	利用识别出的访问控制状态变量, 执行跨过程的上下文敏感污点分析	ICSE	2023
		[109]	利用局部抽象技术, 有效地捕捉潜在的漏洞路径, 提高合约验证的效率和精度	VMCAI	2022
		[108]	建模合约中的数据清理机制, 追踪未清理的数据在合约中的传播路径, 识别绕过守卫机制的漏洞	PLDI	2020
		[107]	使用污点分析追踪未经验证输入如何在合约中传播	ACSAC	2018
	基于中间表示	[48]	从字节码和历史交易中提取状态依赖关系, 基于状态依赖图进行分析	ISSTA	2023
		[82]	将合约转换为中间表示SlithIR, 以便于分析合约的语义信息, 识别合约漏洞	WETSEB	2019
		[47]	通过对Solidity源代码进行词法和语法分析, 生成XML解析树作为中间表示	WETSEB	2018
		[110]	通过将低级字节码表示转换为高层次规则化表示, 极大地扩展了字节码分析的能力	ATVA	2018
模糊测试	基于生成	[38]	在生成函数调用序列时, 明确考虑函数之间的数据依赖关系以促进探索更丰富的状态	TIFS	2023
		[52]	通过改变事务序列, 并使用数据流和控制流引导输入生成, 以发现安全漏洞	TDSE	2022
		[112]	基于元启发式搜索算法, 自动生成测试用例	ICSE	2022
		[37]	从现实世界智能合约的交易记录中学习, 为测试智能合约生成实际输入	ICSME	2021
	基于突变	[40]	设计了序列感知突变和种子掩码引导策略, 动态自适应能量调整范式, 以平衡模糊活动的资源分配	ICDE	2024
		[39]	一种结合两种新技术的引导突变策略, 结合动态依赖学习和动态变量分析	ICSME	2023
		[53]	通过将智能合约字节码编译为原生C++代码, 使用结构感知突变引擎, 生成有效的事务输入	EuroS&P	2023
		[45]	具有自适应调整突变算子的概率分布以选择突变算子的能力, 动态地对种子进行不同的切片	ESWA	2022
		[79]	将突变应用于ESCs中整数溢出漏洞测试, 提出了5个专门用于整数溢出漏洞的突变操作符	Tsinghua Sci Technol	2022

表 3 传统智能合约层漏洞检测方法 (续)

方法类别	方法名称	相关文献	方法概述	会议/期刊	发表年份
深度学习	单模态	[42]	将源代码转化为异构合约语义图, 利用图神经网络和两层注意力机制突出与漏洞相关的节点	ICSE	2024
		[34]	利用对比学习模型捕捉智能合约代码之间的细粒度关联信息, 生成关联标签指导模型的训练过程	ICSE	2024
		[43]	提取操作码中的二元语法特征来构建特征空间, 并使用多种机器学习算法进行多标签分类	IJCNN	2021
		[114]	将源代码转换为图结构, 并进行归一化处理, 使得图神经网络能够学习和检测漏洞	IJCAI	2020
		[113]	基于源代码进行语言模型和语义分析, 从而有效捕捉合约代码中的潜在漏洞	ASC	2018
		[115]	将操作码序列输入LSTM模型, 通过序列学习来检测潜在的安全威胁	arXiv	2018
	多模态	[49]	基于合约代码的语法树和控制流图, 通过异构代码特征学习来提升检测精度	JSS	2024
		[33]	智能合约的源代码和字节码转换为图结构, 并使用图注意力网络提取图的特征嵌入	WWW	2023
		[116]	结合了源代码层、构建层和EVM字节码层的特征, 提出了基于多模态学习的漏洞检测框架	Information Sciences	2023

在静态分析方面, 形式化验证通过数学方法严格验证智能合约的安全性和功能正确性, 通常将合约代码转化为形式化模型, 通过自动化求解器进行验证. 该方法优势在于验证结果高度准确, 但自动化程度低, 验证过程复杂且耗时, 特别是在处理复杂合约时难以扩展. 符号执行通过将程序变量抽象为符号输入, 模拟合约所有可能的执行路径, 通常结合符号求解器与控制流图, 生成路径约束来检测漏洞. 但其面临路径爆炸问题, 尤其在复杂合约中易导致分析效率低下. 污点分析基于数据流分析和控制流图构建以标记和追踪外部输入 (污点) 在程序中的传播路径, 能够有效捕捉外部输入引发的漏洞. 基于中间表示的方法将合约的源代码或字节码转换为易于分析的中间表示 (如静态单赋值、XML 解析树等), 然后在此基础上进行数据流和控制流分析, 使分析更加简便和高效. 但转换过程中需注意语义一致性问题, 可能导致误报或漏报.

在模糊测试方面, 基于生成的模糊测试方法结合控制流、数据流分析以及预定义的输入规则来生成测试用例, 通过覆盖尽可能多的输入空间来探索智能合约的执行路径, 从而发现潜在的安全漏洞. 此类方法的优势在于其能系统性探索合约输入的所有可能性, 但生成测试用例的复杂性和高计算开销可能成为瓶颈. 基于突变的模糊测试方法通过对已有的有效输入数据进行插入、删除、替换和重新排列等突变操作, 生成新的测试用例, 以检测智能合约在不同输入条件下的响应. 此类方法的优势在于不需要预定义输入格式, 且可以迅速生成大规模的测试用例, 其挑战在于如何有效引导突变方向, 使得测试覆盖尽可能多的关键分支.

在深度学习方面, 单模态检测方法聚焦于单一类型特征的输入, 基于源代码、字节码及其构建的图、抽象语法树和反编译后的操作码等特征进行漏洞检测. 该方法具备计算效率高、实现简便的优势, 适用于某些特定场景下的快速检测. 多模态检测方法则通过结合多种类型的数据特征 (如源代码、字节码、控制流图、抽象语法树等), 通过多模态信息的融合来进行漏洞检测. 多模态特征融合技术虽然能够有效提升漏洞检测性能, 但复杂的模型结构及高效特征融合策略的设计仍是当前研究的主要挑战. 此外, 现有的多模态与单模态检测方法大多集中于智能合约传统基础漏洞的检测, 缺乏对业务逻辑相关复杂合约漏洞的深入检测和研究.

在不同的应用场景中, 这些方法的优势和局限性需要根据合约的复杂性和特定需求来选择. 静态分析在检查较为简单或已知的漏洞时较为有效, 而模糊测试适合于在广泛的输入空间中寻找潜在漏洞. 深度学习方法则在大规模合约分析和自动化检测方面有潜力, 尤其是多模态检测方法, 通过综合不同特征的融合, 能够提高检测的准确性. 在实际应用中, 针对各方法的优缺点, 可以采取组合使用的策略, 以弥补单一方法的局限性. 例如, 静态分析可

作为初步筛选工具, 帮助验证并过滤掉明显无关的漏洞, 为模糊测试提供输入种子; 静态分析中所提取的规则可与深度学习模型结合, 以增强模型的可解释性, 提升整体检测系统的性能.

3.1.6 基于 LLM 的智能合约层漏洞检测方法小结

表 4 对基于 LLM 的智能合约层漏洞检测方法进行了总结, 涵盖了 LLM 的用途、应用技术、相关文献、方法概述、发表会议/期刊以及发表年份. LLM 的引入为智能合约漏洞检测开辟了新的方向, 当前主要在两个方面得到了应用: 作为主要检测引擎以及与传统方法相结合.

表 4 基于大语言模型的智能合约层漏洞检测方法

用途	应用技术	相关文献	方法概述	会议/期刊	发表年份
主要检测引擎	基础模型应用	[17]	评估ChatGPT在智能合约漏洞检测中的表现, 比较其与现有漏洞检测工具的效果	arXiv	2023
	对抗性生成	[118]	将漏洞检测任务分为生成和辨别两个阶段, 提高漏洞识别能力并降低误报率	TPS-ISA	2023
	提示调优	[119]	结合代码切片和提示调优的混合方法, 以充分利用预训练模型在智能合约领域的潜力	ISSRE	2023
	连贯推理	[18]	使用思维链提示模拟专业审计员的审计流程, 评估了GPT-4在智能合约漏洞检测方面的性能	arXiv	2024
	微调与agent结合	[121]	结合微调和基于LLM的4个代理 (agent), 实现智能合约漏洞检测并提供解释	ICSE	2024
	模型微调	[122]	对LLM进行微调, 检测智能合约中逻辑漏洞	arXiv	2024
与传统方法结合	形式化属性生成	[125]	通过LLM从现有的属性中提取信息, 并生成适用于新的智能合约代码的属性, 进而检测合约漏洞	NDSS	2024
	不变量生成	[54]	通过微调和提示策略, 利用多模态信息进行推理并生成不变量, 然后通过检查不变量来定位漏洞	SP	2024
	结合静态分析	[81]	结合了GPT和静态分析以检测智能合约中的逻辑漏洞	ICSE	2024
	指导模糊测试	[126]	通过利用LLM智能指导和优先排序模糊活动, 优化智能合约自动化检测	arXiv	2024
	模糊测试种子生成	[127]	提出基于Chain-based LLM的智能合约模糊测试种子对抗生成方法, 利用LLM的深度语义理解能力辅助种子集生成	ASE	2024

在 LLM 作为主要检测引擎的应用中, LLM 被作为基础模型用于漏洞检测, 并在此基础上进行思维链推理, 以及通过微调 LLM 以适配下游漏洞检测任务. 此外, 还设计了多个基于 LLM 的 agent, 通过让 LLM 扮演不同角色来执行漏洞检测任务. 此类方法充分利用了 LLM 在文本理解、推理和生成方面的强大能力, 展现了其在智能合约漏洞检测中的潜力. 然而, 当前不同 LLM 以及提示设计对检测效果的影响较大, 且在不同漏洞类型上的检测表现差异显著, 模型的稳定性和检测效果是目前存在的问题.

在 LLM 与传统方法结合的应用中, LLM 有效地融入了静态分析和模糊测试等传统漏洞检测方法中. 在静态分析方面, LLM 被引入到形式化验证方法中自动生成形式化属性, 以及利用 LLM 根据场景和属性匹漏洞函数, 并结合静态分析检查漏洞是否存在. 在模糊测试方面, LLM 主要用于指导模糊测试活动以及生成测试种子, 从而提高了漏洞检测的覆盖率, 并生成质量更高的测试样例. 尽管 LLM 与传统方法的结合优化了检测流程, 但仍面临挑战. 例如, LLM 生成的测试种子或形式化属性可能存在语义偏差, 需依赖静态分析工具二次验证, 增加了流程复杂度. 同时, LLM 的解释性不足使得其推理过程难以追溯, 导致安全审计人员对检测结果的信任度降低. 尽管如此, LLM 与传统方法结合的应用已初见成效, 未来研究可进一步探索 LLM 在检测流程中的更广泛应用, 例如同时利用 LLM 生成种子并指导模糊测试活动, 以及探索 LLM 与符号执行、污点分析和基于中间表示的方法相结合的应用.

基于 LLM 的检测方法在检测特性和适用场景上存在差异. 从方法特性来看, LLM 作为主要检测引擎在自然语言推理、代码意图理解和复杂逻辑漏洞发现方面更具优势, 适用于需结合合约语义的漏洞类型; 而 LLM 与传统方法结合则更擅长利用结构化分析 (如控制流追踪、数据依赖检测) 提升检测覆盖率, 适用于需精准模式匹配的

漏洞类型. 在场景适配性上, LLM 作为主要检测引擎更适用于开发阶段的轻量级实时检测, 而与传统方法结合则更适合安全审计阶段的高精度分析.

基于 LLM 的智能合约漏洞检测方法与传统方法相比, 具有以下核心优势. 首先, LLM 在代码语义理解、逻辑推理和生成方面的能力使其能够处理复杂的漏洞检测任务, 尤其在检测合约中的高阶逻辑漏洞时表现出优势和潜力. 传统的静态分析和模糊测试方法往往局限于基础的代码分析和输入覆盖, 而 LLM 能够通过推理和上下文理解, 识别出更加复杂和隐蔽的漏洞类型. 其次, LLM 具有自适应微调能力, 使得它能够根据特定的漏洞场景进行快速调优, 从而快速适配不同的漏洞检测任务, 减少对领域特定规则的依赖, 提升检测的精度和适用性. 此外, LLM 在与传统方法结合时, 能够通过生成测试用例、指导模糊测试活动以及自动生成形式化属性, 增强传统方法的检测覆盖度和效率. 然而, LLM 的方法也面临着稳定性和漏洞类型适应性的问题, 当前不同 LLM 及其提示设计对检测效果的影响较大, 需要进一步优化.

LLM 在智能合约层漏洞检测中的应用具有重要潜力, 尤其是在处理新兴合约漏洞类型和提高检测性能方面. 作为一种新的技术范式, LLM 在语义理解、推理能力和任务适配性方面显著优于传统方法. 然而, LLM 的稳定性和泛化能力仍需进一步提升, 未来研究应聚焦于优化提示设计、增强模型鲁棒性, 并探索 LLM 与传统方法的深度融合, 以提升其在智能合约安全领域的实际应用效果.

3.2 DeFi 协议层漏洞检测方法

DeFi 协议由多个智能合约组成, 但是协议的复杂的业务逻辑和多合约交互特性显著增加了漏洞利用和攻击检测的难度, 相比传统智能合约安全漏洞检测更具挑战性. 目前, DeFi 协议层的漏洞检测方法主要集中在价格操纵漏洞的检测, 以下将分别介绍价格操纵漏洞检测方法及其他检测技术.

3.2.1 价格操纵漏洞检测方法

交易顺序依赖类型漏洞在 DeFi 协议中普遍存在, 攻击者可以利用抢先交易和三明治攻击等方式进行攻击. 为了应对这些威胁, 研究者们开发了多种检测和防御方法.

首先, 对于抢先交易漏洞, Torres 等人[85]通过基于交易和攻击模式的分析方法, 详细揭示了以太坊区块链上发生的抢先交易行为. 研究表明, 这些攻击者能够通过提前获取交易信息并优先执行, 从中获利, 这一现象对 DeFi 生态系统构成了重大威胁. 该研究的方法深入分析交易数据流, 并构建了一个框架, 用于识别和量化这些攻击的影响. 另一个应对抢先交易漏洞的方法是静态分析技术. Nyx 系统[128]通过在智能合约部署前的静态分析, 能够有效识别出潜在的抢先交易漏洞. 该方法通过分析合约代码的执行路径和逻辑, 提前发现可能被利用的安全漏洞, 从而在合约部署前进行修复, 降低攻击发生的可能性.

针对三明治攻击, Zhou 等人[129]通过数学形式化的方式对去中心化交易所上的三明治攻击进行了深入研究. 研究表明, 攻击者通过操控交易顺序, 使得他们能够在人们购买代币之前先买入, 然后在人们买入后立即卖出, 获利丰厚. 同时, 该研究根据交易在区块链区块内的相对位置来量化敌对交易者能够进行攻击的概率. Heimbach 等人[130]还提出了通过博弈论来消除三明治攻击的可能性, 该研究提出了一个“三明治博弈”, 从攻击者和受害者的角度分析三明治攻击, 并且提供了一种简单而高效的算法, 交易者可以使用它来设置滑点容忍度, 保障 DeFi 协议避免遭受三明治攻击.

为实现实时检测针对 DeFi 协议的三明治攻击, Li 等人[131]还开发了一种基于 Geth 客户端的实时三明治攻击检测系统. 该系统集成到 go-Ethereum 客户端 Geth 之中, 通过实时分析交易数据流, 能够在攻击发生时立即检测并响应, 防止攻击者从中获利, 并且保持全节点的正常运行速度. 这种实时检测方法可以为保护 DeFi 用户的私人资金提供重要的安全保障.

闪电贷和预言机是实现价格操纵攻击中常用的工具, 其中闪电贷利用大额、快速借贷影响市场价格, 预言机则可能被操控以篡改价格数据. 针对这些漏洞, 研究者提出了多种检测和预防方法.

闪电贷攻击的检测方面, LeiShen[132]通过基于规则的方法进行检测闪电贷攻击, 研究者分析了以太坊区块链上发生的实际攻击事件, 并定义了 3 种闪电贷价格操纵的攻击模式, 基于这些攻击模式对交易数据进行分析从而

检测闪电贷价格操纵攻击, 实验表明该工具能够准确识别出正在进行的攻击行为并及时响应. 另一种检测方法是利用对抗攻击和插值技术[133]. 该方法绕过 DeFi 协议的复杂性, 使用多项式近似和最近邻插值的数值方法来近似 DeFi 协议的功能, 然后通过生成对抗性攻击来检测包括闪电贷价格操纵攻击在内的闪电贷攻击. 文章展示了使用该技术的工具 FlashSyn 在预防闪电贷攻击中的有效性, 通过这种方法, 系统能够在攻击完成之前识别并阻止恶意行为, 减少攻击带来的经济损失. DeFiTail[134]采用符号执行和深度学习的方法针对闪电贷利用和访问控制漏洞进行检测. 该框架通过符号执行堆栈来验证控制流图中数据路径的可行性, 然后将经过验证的可行数据路径输入模型, 并使用 BERT 和图卷积网络 (graph convolutional network, GCN) 来学习这些路径的特征, 从而实现对 DeFi 协议的漏洞检测.

预言机操纵漏洞的检测同样可以使用符号执行的方法. BLOCKEYE[135]是一种通过符号分析和交易分析来实时检测预言机操纵漏洞的方法. 该系统首先对给定 DeFi 协议的智能合约进行符号分析, 一旦发现依赖于 Oracle 的状态更新, BLOCKEYE 就会将 DeFi 协议识别为潜在的漏洞. 然后 BLOCKEYE 为易受攻击的 DeFi 协议安装运行时监视器, 以检测外部攻击, 并生成攻击验证的分析报告. ProMutator[136]通过模拟攻击的方式来检测 DeFi 协议的预言机价格操纵漏洞, 该方法改变预言机价格计算所需的数据在本地模拟价格预言机攻击, 可以不需要任何已检查合约的高级源代码就能实现漏洞的检测. OVer [137]同样采取符号分析的方法检测预言机操纵漏洞, OVer 首先对给定的合约进行符号分析, 并构建一个约束模型. 然后, 该框架利用 SMT 求解器来识别允许其安全运行的参数. 此外, 可以为可能使用预言机值的智能合约生成 guard 语句, 从而有效防止预言机操纵攻击. SecPLF 系统[138]通过基于价格观测的攻击预防方法来抵御预言机操纵攻击. SecPLF 通过形式化可贷资金 (protocols for loanable fund, PLF) 协议的标准操作和对手模型来剖析预言机操纵攻击的攻击机制, 然后跟踪每种加密资产的价格状态并对价格预言机的使用施加价格限制, 以此达到预防攻击的目的.

除了针对特定的借助闪电贷进行价格操纵以及预言机价格操纵的研究, 目前也有一些针对价格操纵漏洞进行检测的通用检测工具. DeFiRanger[139]基于交易的高级语义来检测价格操纵攻击, 该方法首先从原始交易构建现金流树, 然后将低级语义提升为高级语义, 包括 5 种高级 DeFi 操作. 最后, 使用恢复后的 DeFi 语义所表达的模式来检测价格操纵攻击. DeFiTainter[140]是一个用于检测价格操纵漏洞的合约间污点分析框架, 该框架通过恢复调用信息来构建合约间污点分析的调用图, 然后专门为检测价格操纵漏洞而定制的高级语义归纳, 最终实现准确识别污点源和跟踪污点数据, 完成对 DeFi 协议的价格操纵漏洞检测. 林炼升等人[141]基于价格清洗和代理防护实现了一种针对价格操纵攻击的防御机制. 该机制利用链下价格来指导链上价格操纵行为的识别, 并通过合同代理拦截价格操纵交易.

3.2.2 其他方法

除了上述针对价格操纵漏洞的检测方法外, 研究者们还开发了一些其他方法, 用于检测 DeFi 协议中的各类漏洞, 例如跨链桥漏洞等.

在跨链桥漏洞的检测方面, Xscope[142]提出了 3 类跨链桥漏洞, 并基于攻击模式匹配进行跨链桥攻击检测. 基于这些模式, Xscope 可以自动化查找跨链桥中的安全漏洞并检测现实世界的攻击. SmartAxe[143]通过构建跨链控制流图和数据流图, 并结合污点分析技术, 成功检测出跨链环境中的漏洞. SmartAxe 不仅适用于单链环境, 还能够在复杂的多链生态系统中发挥作用, 体现出其强大的适应性和检测能力.

此外, 还有研究针对 DeFi 攻击开发了多种检测方法. Liao 等人[144]提出了一个通过字节码级静态分析检测跨合约漏洞的新框架 SmartDagger. SmartDagger 从智能合约字节码中恢复合约属性信息, 同时有选择地分析函数子集并重用数据流结果以提高静态分析的效率. DeFiScanner[145]是一种基于深度学习进行表征和检测利用逻辑漏洞的 DeFi 攻击的系统, 其中运用了一个新颖的神经网络, 包括全局模型、局部模型和融合模型来表征 DeFi 攻击. Parhizkari 等人[146]基于字节码分析的方法来检测 DeFi 攻击, 该方法通过分析恶意智能合约的部署字节码来及时识别 DeFi 攻击中受害者地址, 包括提取潜在受害者地址、分析部署者地址、确定实际受害者, 并通过实验验证其有效性. 这一方法可以在攻击发生前识别受害者, 从而采取预防措施, 减少攻击带来的损失.

3.2.3 小 结

表 5 总结了上述 DeFi 协议层漏洞检测方法的检测对象、相关文献、方法描述、发表会议/期刊和年份, 并且按照发表时间进行了排序. 通过对 DeFi 协议层漏洞检测方法的分析, 可以看出不同的漏洞检测方法各具特点和适用场景. 针对交易顺序依赖、价格操纵等常见攻击, 研究者们提出了多种基于静态分析、符号执行以及深度学习的检测工具. 这些方法不仅有效应对了 DeFi 协议的复杂性, 还通过分析交易数据流、智能合约字节码和合约间调用关系等方式提升了检测精度和效率.

表 5 DeFi 协议层漏洞检测方法分类

检测对象	相关文献	方法描述	会议/期刊	发表年份
抢先交易	[85]	将抢先交易攻击分成3类, 分析交易数据流, 并构建框架用于识别和量化抢险交易攻击的影响	USENIX Security	2021
	[128]	结合符号执行和具体交易生成, 分析智能合约的交易流程, 以检测潜在的可被利用的抢跑机会	S&P	2024
三明治攻击	[129]	形式化并量化了三明治攻击, 模拟了单个和多个攻击者的情境, 并对这种攻击在AMM模型中的盈利性进行了评估	S&P	2021
	[130]	通过将交易者与潜在攻击者之间的互动建模为一个策略博弈, 交易者可以通过使用其设置滑点容忍度来减少三明治攻击的风险	ASIA CCS	2022
	[131]	部署在Geth中, 通过实时分析交易数据流, 能够在攻击发生时立即检测并响应	Discover Computing	2024
闪电贷攻击	[132]	定义了3种闪电贷价格操纵的攻击模式, 基于攻击模式对交易数据进行分析从而检测闪电贷价格操纵攻击	ICDCS	2023
	[133]	使用多项式近似和最近邻插值的数值方法来近似 DeFi 协议的功能, 然后通过生成对抗性攻击来检测闪电贷攻击风险	ICSE	2024
	[134]	通过符号执行堆栈验证控制流图中数据路径的可行性, 将可行数据路径输入BERT和GCN进行访问控制和闪电贷漏洞利用检测	WWW	2024
价格预言机操纵	[135]	对DeFi协议的智能合约进行符号分析识别潜在漏洞, 为存在漏洞的协议安装监视器识别外在攻击	ICSE	2021
	[136]	改变预言机价格计算所需的数据在本地模拟价格预言机攻击, 对DeFi协议进行漏洞检测	EuroS&PW	2021
	[137]	对合约进行符号分析并构建一个约束模型, 利用SMT求解器识别安全运行的参数, 为可能使用预言机值的智能合约生成guard语句	ICSE	2024
	[138]	形式化可贷资金 (PLF) 协议的标准操作和对手模型来剖析攻击机制, 跟踪加密资产的价格状态并对价格预言机的使用施加价格限制以实现攻击预防	ASIA CCS	2024
通用价格操纵漏洞	[139]	从原始交易构建现金流树, 并提取高级语义, 如5种高级DeFi操作. 最后使用DeFi 语义所表达的模式来检测价格操纵攻击	TDSC	2023
	[140]	通过恢复调用信息来构建合约间污点分析的调用图, 并基于高级语义归纳识别污点源和跟踪污点数据, 实现价格操纵漏洞检测	ISSTA	2023
	[141]	提出改进的预言机喂价框架, 通过链下价格引导识别链上价格操控行为, 并通过代理合约实现对价格操纵交易的拦截	计算机研究与发展	2024
跨链桥攻击	[142]	提出了3类跨链桥漏洞, 并基于攻击模式匹配进行跨链桥攻击检测	ASE	2022
	[143]	构建跨链控制流图和数据流图, 并结合污点分析技术, 成功检测出跨链环境中的漏洞	FSE	2024
其他	[144]	从智能合约字节码中恢复合约属性信息, 同时有选择地分析函数子集并重用数据流结果以提高静态分析的效率	ISSTA	2022
	[145]	通过处理由内部交易生成的非结构化事件, 并融合多层次特征 (全局特征和局部特征), 使用包含全局模型、局部模型和融合模型的神经网络, 来高效地检测DeFi攻击	TCSS	2022
	[146]	通过分析恶意智能合约的部署字节码来及时识别DeFi攻击中受害者地址, 先后进行提取潜在受害者地址、分析部署者地址、确定实际受害者等操作	ESORICS	2023

根据攻击发生前和发生后的不同阶段, 可以将 DeFi 协议层的漏洞检测方法分为两类, 如图 10 所示. 这些方法各有优缺点, 在攻击发生前, 静态分析和符号执行等方法能够在智能合约部署前识别潜在漏洞, 具有提前预防的优势. 这类方法通过对字节码或源代码进行详细分析, 能够有效检测已知漏洞, 并减少部署后遭受攻击的风险. 然而, 静态分析和符号执行对新型或复杂攻击的防御能力较弱, 且可能产生较高的误报率. 在攻击发生后, 基于神经网络的方法则更加适用. 这类方法通过自动提取交易和合约执行过程中的特征, 在无需明确漏洞模式的情况下识别攻击行为, 具有较高的检测准确性, 尤其在处理复杂或未知攻击时表现优异. 然而, 这类方法依赖大量高质量的训练数据, 且难以在攻击发生前进行预测性防御. 因此, 针对 DeFi 漏洞的检测应结合攻击前后的检测方法, 以实现更全面的防护.

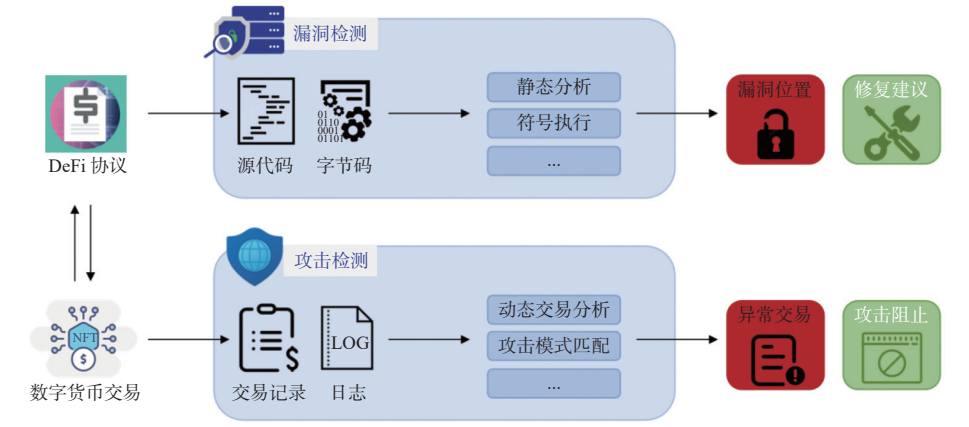

图 10 DeFi 协议层漏洞检测方法框架图

4 现存挑战与未来研究方向

针对当前智能合约与 DeFi 协议漏洞检测中的问题, 本文深入讨论了研究挑战, 并总结了未来研究方向, 对应关系如图 11 所示.

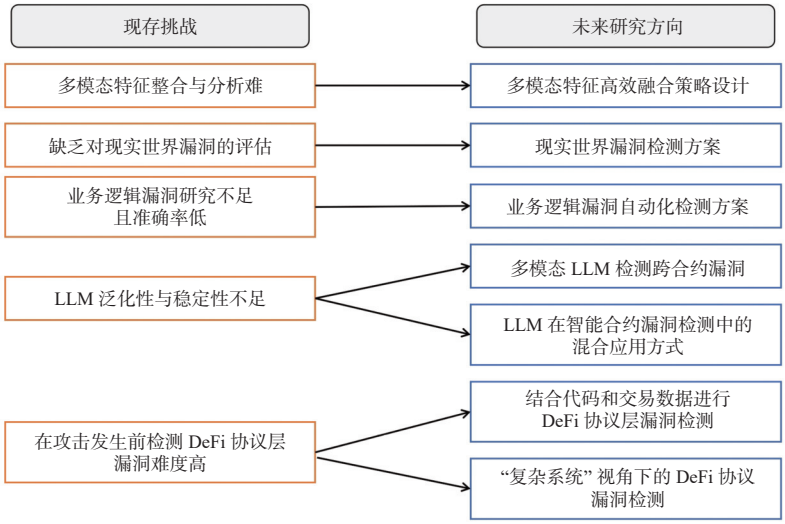

图 11 现存挑战与未来研究方向对应关系

4.1 现存挑战

(1) 多模态特征整合与分析难

智能合约的多模态特征主要包括源代码、字节码、操作码、控制流图、抽象语法树、注释以及交易产生的数据. 这些特征不仅在结构和语义上存在显著差异, 其相互作用也复杂且多样化. 由于特征间的高度异质性, 传统的单模态检测方法难以充分捕捉智能合约的全貌, 导致表征能力的局限性. 目前, 多模态特征的整合与分析[33,49,116]仍处于初期探索阶段, 研究在深度与广度上均存在局限性, 这在很大程度上限制了对智能合约潜在漏洞的全面分析与检测.

(2) 缺乏对现实世界漏洞的评估

现实世界智能合约是指实际部署在区块链网络中的智能合约. 相较于实验环境中的合约, 现实世界合约具有数据分布不平衡、结构复杂、漏洞少见等特点. 由于实际智能合约中漏洞较为稀少、数据样本的不均衡性, 现有检测工具[33,34]在训练过程中往往依赖于较为简单的合约来提高其性能表现, 这可能导致在面对复杂的现实世界合约时, 出现检测能力不足的情况, 从而难以有效应对实际场景中的潜在漏洞.

(3) 业务逻辑漏洞研究不足且准确率低

业务逻辑漏洞自 2023 年被正式提出以来引起了学术界的关注[10], 通常源于合约实现过程中复杂的业务规则不当设置, 导致攻击者能够通过规避或滥用业务逻辑来获取不正当利益. 由于其隐蔽性强、形式多样, 目前针对该类漏洞的研究相对较少, 亟需更多深入的探索. 此外, 已有的检测工具[62]在面对业务逻辑漏洞时, 表现出较低的检测准确率, 无法有效应对复杂的业务流程和条件逻辑.

(4) LLM 泛化性与稳定性不足

目前 LLM 作为主要检测引擎和与传统方法结合的方式用于智能合约漏洞检测的方法中, 已展现出一定的潜力. 然而, 现有研究[118,119,121]在不同类型漏洞检测上的泛化性仍显不足, 基础模型在处理大规模和复杂智能合约样本时推理能力有限[17,18], 亟需提升模型的性能和检测精度. 此外, 不同的 LLM 模型以及提示设计对漏洞检测效果的影响较大[121], 导致在面对多样化智能合约场景时, 其检测结果表现出较大的波动性和不稳定性.

(5) 在攻击发生前检测 DeFi 协议层漏洞难度高

由于 DeFi 协议层漏洞通常涉及多个合约和 DeFi 平台, 其交互逻辑复杂, 且伴随 DeFi 协议独有的设计特性, 传统智能合约漏洞检测方法难以应对这类漏洞的复杂性[25]. 尤其在漏洞利用过程中, 攻击者的行为往往隐藏在庞大的交易数据中, 通过闪电贷等手段迅速实施攻击、获取利润并转移资产, 进一步增加了检测难度. 因此, 在检测此类攻击时, 不仅需要具备较快的反应速度, 还需具备高效的检测能力, 以应对瞬息万变的攻击态势.

4.2 未来研究方向

(1) 多模态特征高效融合策略设计

针对智能合约多模态特征的日益复杂化与异构化的挑战, 如何高效融合这些特征以提升漏洞检测的性能至关重要. 未来研究应着重探索更高效的特征融合方法, 主要从多模态特征整合、多角度特征融合、自适应模型训练和高精度决策分析这 4 个方面入手. 利用多模态学习、集成学习、多尺度融合和自适应权重等技术, 从不同层次挖掘特征间的语义和结构关联性, 提高多模态特征的整合与协同分析能力, 从而更全面地捕捉合约中的潜在漏洞特征.

(2) 现实世界漏洞检测方案

当前针对现实世界智能合约漏洞的自动化检测仍然面临诸多挑战, 现有检测工具在处理现实世界漏洞时表现不足. 未来的研究应着力于开发能够应对复杂合约结构和数据分布不均衡问题的高效检测模型, 使用数据增强技术、改进特征提取方法或引入生成对抗网络来生成更多模拟现实世界场景的漏洞样本, 从而提高检测工具在实际场景中的适用性和检测准确率.

(3) 业务逻辑漏洞自动化检测方案

针对智能合约业务逻辑漏洞的检测目前仍处于初步阶段, 现有的自动化检测工具较为匮乏. 未来研究应重点开发抽象的漏洞模型, 这些模型可以作为检测业务逻辑漏洞的自动化“测试准则”, 从而提高检测效率与准确性. 同时, 自动化检测工具的开发应加强与区块链生态系统的兼容性, 确保其在多种区块链平台上的适用性, 并通过持续

优化检测算法和测试准则实现更高的智能化和自适应性.

(4) 多模态 LLM 检测跨合约漏洞

通过多模态 LLM 结合源码、文档、交易记录等多种信息, 深入理解并分析合约间的调用链, 识别其交互逻辑是否符合预期. 合约间的调用关系不仅需要在技术层面进行静态分析, 还需要结合交易执行场景和自然语言文档, 推断出潜在的交互冲突或违背原子性的情况. 这种检测方法有望显著提高跨合约场景下复杂漏洞的发现率, 尤其是在涉及多个合约组合的情况下.

(5) LLM 在智能合约漏洞检测中的混合应用方式

未来研究方向可以重点放在如何让 LLM 在漏洞检测过程中主动参与漏洞推理和上下文分析, 不仅限于简单的信息抽取, 同时通过与其他检测引擎协同工作, 生成动态的、情境化的检测路径. 此外, 未来可以进一步优化 LLM 的推理能力, 使其在合约的不同层次上自动推导漏洞的潜在成因, 形成一个从信息提取到推理分析的闭环流程, 从而实现更精准的智能合约安全检测.

(6) 结合代码和交易数据进行 DeFi 协议层漏洞检测

为了提升 DeFi 协议层漏洞的检测精度和速度, 可以将静态代码分析与实时交易数据分析相结合. 静态分析能够发现潜在的漏洞点, 而交易数据则提供了动态行为信息, 两者的结合有助于构建更全面的漏洞检测框架. 此外, 基于大规模数据的机器学习模型也可以帮助识别复杂的交互模式, 从而提升检测的准确性. 结合区块链的透明特性, 未来的研究方向应侧重于建立能够快速捕捉攻击行为的多层次、多维度检测体系, 充分利用跨平台、跨合约的交互数据, 提升对复杂攻击场景的防御能力.

(7) “复杂系统”视角下的 DeFi 协议漏洞检测

从“复杂系统”视角进行 DeFi 协议漏洞检测是未来的重要研究方向. DeFi 协议中智能合约的复杂交互关系带来了系统性风险, 传统单合约检测方法难以应对. 未来研究应构建全局化、多层次的检测框架, 系统分析合约间的交互路径, 识别跨合约依赖导致的潜在漏洞. 重点在于开发能够动态监控合约状态变化的工具, 及时发现复杂交互中的安全隐患, 提高整体系统的安全性.

5 总 结

智能合约作为区块链技术中最具潜力的创新组件之一, 为去中心化应用场景提供了安全、可信的解决方案. 然而, 频发的安全事件严重阻碍了其广泛应用, 因此智能合约与 DeFi 协议的漏洞检测技术成为研究重点. 本文系统梳理了智能合约与 DeFi 协议中的常见漏洞类型, 并分析了静态分析、模糊测试、深度学习以及 LLM 驱动等多种检测方法的现状与挑战. 研究表明, 当前研究在业务逻辑漏洞、跨合约交互漏洞及 DeFi 协议层漏洞的检测中存在显著不足, 且传统方法难以应对高动态性攻击场景. 尽管 LLM 在提升漏洞检测效率、处理多模态数据、应对复杂漏洞方面展现了潜力, 但它仍面临泛化能力和误报控制的挑战. 未来研究方向应关注如何整合现有检测技术与 LLM 的优势, 开发更智能高效的检测工具, 并针对智能合约和 DeFi 协议的复杂性及快速迭代特性, 提出更具适应性的解决方案.

References

[1] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. 2008. https://bitcoin.org/bitcoin.pdf

[2] Buterin V. A next-generation smart contract and decentralized application platform. White Paper, 2014.

[3] Zheng ZB, Xie SA, Dai HN, Chen XP, Wang HM. Blockchain challenges and opportunities: A survey. Int'l Journal of Web and Grid Services, 2018, 14(4): 352–375. [doi: 10.1504/IJWGS.2018.095647]

[4] Schär F. Decentralized finance: On blockchain- and smart contract-based financial markets. Federal Reserve Bank of St. Louis Review, 2021. [doi: 10.20955/r.103.153-74]

[5] Luu L, Chu DH, Olickel H, Saxena P, Hobor A. Making smart contracts smarter. In: Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security. Vienna: ACM, 2016. 254–269. [doi: 10.1145/2976749.2978309]

[6] Qin KH, Zhou LY, Gamito P, Jovanovic P, Gervais A. An empirical study of DeFi liquidations: Incentives, risks, and instabilities. In: Proc. of the 21st ACM Internet Measurement Conf. ACM, 2021. 336–350. [doi: 10.1145/3487552.3487811]

[7] CoinDesk. Understanding the DAO attack. 2016. https://www.coindesk.com/learn/understanding-the-dao-attack
[8] Hacking Distributed. Deep dive: The parity bug. 2017. http://hackingdistributed.com/2017/07/22/deep-dive-parity-bug/
[9] PeckShield. BatchOverflow: Understanding the vulnerability. 2018. https://blog.peckshield.com/2018/04/22/batchOverflow/
[10] Zhang Z, Zhang B, Xu W, Lin ZQ. Demystifying exploitable bugs in smart contracts. In: Proc. of the 45th Int'l Conf. on Software Engineering (ICSE). Melbourne: IEEE, 2023. 615–627. [doi: 10.1109/ICSE48619.2023.00061]
[11] CoinDesk. Binance chain DeFi exchange uranium finance loses $50M in exploit. 2021. https://www.coindesk.com/markets/2021/04/28/binance-chain-defi-exchange-uranium-finance-loses-50m-in-exploit/
[12] Kiran S. Did this hacker get away with a $3.8 million NFT hack? 2022. https://watcher.guru/news/did-this-hacker-get-away-with-a-3-8-million-nft-hack
[13] SlowMist. Blockchain Security Statistics. 2024. https://hacked.slowmist.io/statistics/?c=all&d=all
[14] Coindesk. DeFi lender bZx loses $8M in third attack this year. 2020. https://www.coindesk.com/markets/2020/09/14/defi-lender-bzx-loses-8m-in-third-attack-this-year/
[15] Yahoo Finance. PancakeBunny attacked with massive $200M flash loan exploit. 2021. https://finance.yahoo.com/news/pancakebunny-attacked-massive-200m-flash-050619340.html
[16] Blockchain News. Ethereum foundation suffered from MEV Bot attack in ETH selling. 2023. https://blockchain.news/news/Ethereum-foundation-suffered-from-mev-bot-attack-in-eth-selling
[17] Chen C, Su JZ, Chen JC, Wang YL, Bi TT, Yu JX, Wang YL, Lin XW, Chen T, Zheng ZB. When ChatGPT meets smart contract vulnerability detection: How far are we? arXiv:2309.05520, 2024.
[18] Du YY, Tang XY. Evaluation of ChatGPT's smart contract auditing capabilities based on chain of thought. arXiv:2402.12023, 2024.
[19] He ZY, Li ZH, Yang S, Ye H, Qiao A, Zhang XS, Luo XP, Chen T. Large language models for blockchain security: A systematic literature review. arXiv:2403.14280, 2025.
[20] Qian P, Liu ZG, He QM, Huang BT, Tian DZ, Wang X. Smart contract vulnerability detection technique: A survey. Ruan Jian Xue Bao/Journal of Software, 2022, 33(8): 3059–3085 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6375.htm [doi: 10.13328/j.cnki.jos.006375]
[21] Dong WL, Liu Z, Liu K, Li L, Ge CP, Huang ZQ. Survey on vulnerability detection technology of smart contracts. Ruan Jian Xue Bao/Journal of Software, 2024, 35(1): 38–62 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6810.htm [doi: 10.13328/j.cnki.jos.006810]
[22] Cui ZQ, Yang HW, Chen X, Wang LZ. Research progress of security vulnerability detection of smart contracts. Ruan Jian Xue Bao/Journal of Software, 2024, 35(5): 2235–2267 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/7046.htm [doi: 10.13328/j.cnki.jos.007046]
[23] Liu ZX, Li LX, Liu DJ, Du JZ, Lin H, Shi JP. Review of smart contract vulnerability detection and repair research. Journal of Frontiers of Computer Science and Technology, 2025, 19(4): 854–876 (in Chinese with English abstract). [doi: 10.3778/j.issn.1673-9418.2405019]
[24] Zhao WX, Mi W, Zhang XD. The security paradox of smart contracts: Blind spots and prospects of current detection strategies. In: Proc. of the 27th Int'l Conf. on Computer Supported Cooperative Work in Design (CSCWD). Tianjin: IEEE, 2024. 1546–1551. [doi: 10.1109/CSCWD61410.2024.10580546]
[25] Qian P, Cao R, Liu ZG, Li WQ, Li M, Zhang L, Xu YF, Chen JH, He QM. Empirical review of smart contract and DeFi security: Vulnerability detection and automated repair. arXiv:2309.02391, 2023.
[26] Chaliasos S, Charalambous MA, Zhou LY, Galanopoulou R, Gervais A, Mitropoulos D, Livshits B. Smart contract and DeFi security tools: Do they meet the needs of practitioners? In: Proc. of the 46th IEEE/ACM Int'l Conf. on Software Engineering. Lisbon: ACM, 2024. 60. [doi: 10.1145/3597503.3623302]
[27] Zhou LY, Xiong XH, Ernstberger J, Chaliasos S, Wang ZP, Wang Y, Qin KH, Wattenhofer R, Song D, Gervais A. SoK: Decentralized finance (DeFi) attacks. In: Proc. of the 2023 IEEE Symp. on Security and Privacy (SP). IEEE, 2023. 2444–2461. [doi: 10.1109/SP46215.2023.10179435]
[28] Turner M, Kitchenham B, Brereton P, Charters S, Budgen D. Does the technology acceptance model predict actual use? A systematic literature review. Information and Software Technology, 2010, 52(5): 463–479. [doi: 10.1016/j.infsof.2009.11.005]
[29] Hwang S, Ryu S. Gap between theory and practice: An empirical study of security patches in Solidity. In: Proc. of the 42nd ACM/IEEE Int'l Conf. on Software Engineering. Seoul: ACM, 2020. 542–553. [doi: 10.1145/3377811.3380424]
[30] Nguyen TD, Pham LH, Sun J, Lin Y, Minh QT. sFuzz: An efficient adaptive fuzzer for Solidity smart contracts. In: Proc. of the 42nd Int'l Conf. on Software Engineering. Seoul: ACM, 2020. 778–788. [doi: 10.1145/3377811.3380334]

[31] So S, Lee M, Park J, Lee H, Oh H. VERISMART: A highly precise safety verifier for Ethereum smart contracts. In: Proc. of the 2020 IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2020. 1678–1694. [doi: 10.1109/SP40000.2020.00032]

[32] Zhou EC, Hua S, Pi BF, Sun J, Nomura Y, Yamashita K, Kurihara H. Security assurance for smart contract. In: Proc. of the 9th IFIP Int'l Conf. on New Technologies, Mobility and Security. Paris: IEEE, 2018. 1–5. [doi: 10.1109/NTMS.2018.8328743]

[33] Qian P, Liu ZG, Yin YF, He QM. Cross-modality mutual learning for enhancing smart contract vulnerability detection on bytecode. In: Proc. of the 2023 ACM Web Conf. Austin: ACM, 2023. 2220–2229. [doi: 10.1145/3543507.3583367]

[34] Chen YZ, Sun ZY, Gong ZH, Hao D. Improving smart contract security with contrastive learning-based vulnerability detection. In: Proc. of the 46th Int'l Conf. on Software Engineering. Lisbon: ACM, 2024. 156. [doi: 10.1145/3597503.3639173]

[35] Jiang B, Liu Y, Chan WK. ContractFuzzer: Fuzzing smart contracts for vulnerability detection. In: Proc. of the 33rd Int'l Conf. on Automated Software Engineering (ASE). Montpellier: ACM, 2018. 259–269. [doi: 10.1145/3238147.3238177]

[36] Chen JC, Xia X, Lo D, Grundy J, Luo XP, Chen T. DefectChecker: Automated smart contract defect detection by analyzing EVM bytecode. IEEE Trans. on Software Engineering, 2022, 48(7): 2189–2207. [doi: 10.1109/TSE.2021.3054928]

[37] Zhou T, Liu K, Li L, Liu Z, Klein J, Bissyande TF. SmartGift: Learning to generate practical inputs for testing smart contracts. In: Proc. of the 2021 IEEE Int'l Conf. on Software Maintenance and Evolution (ICSME). Luxembourg: IEEE, 2021. 23–34. [doi: 10.1109/ICSME52107.2021.00009]

[38] Liu ZG, Qian P, Yang JX, Liu LF, Xu XJ, He QM, Zhang XS. Rethinking smart contract fuzzing: Fuzzing with invocation ordering and important branch revisiting. IEEE Trans. on Information Forensics and Security, 2023, 18: 1237–1251. [doi: 10.1109/TIFS.2023.3237370]

[39] Ji SY, Dong J, Wu J, Lu LS. A guided mutation strategy for smart contract fuzzing. In: Proc. of the 2023 IEEE Int'l Conf. on Software Maintenance and Evolution (ICSME). Bogotá: IEEE, 2023. 282–292. [doi: 10.1109/ICSME58846.2023.00036]

[40] Qian P, Wu HJ, Du ZR, Vural T, Rong DZ, Cao Z, Zhang L, Wang YB, Chen JH, He QM. MuFuzz: Sequence-aware mutation and seed mask guidance for blockchain smart contract fuzzing. In: Proc. of the 40th Int'l Conf. on Data Engineering (ICDE). Utrecht: IEEE, 2024. 1972–1985. [doi: 10.1109/ICDE60146.2024.00158]

[41] Holler S, Biewer S, Schneidewind C. HoRStify: Sound security analysis of smart contracts. In: Proc. of the 36th Computer Security Foundations Symp. (CSF). Dubrovnik: IEEE, 2023. 245–260. [doi: 10.1109/CSF57540.2023.00023]

[42] Luo F, Luo RJ, Chen T, Qiao A, He ZY, Song SW, Jiang Y, Li SX. SCVHunter: Smart contract vulnerability detection based on heterogeneous graph attention network. In: Proc. of the 46th Int'l Conf. on Software Engineering. Lisbon: ACM, 2024. 170. [doi: 10.1145/3597503.3639213]

[43] Yu XX, Zhao HY, Hou BT, Ying ZH, Wu B. DeeSCVHunter: A deep learning-based framework for smart contract vulnerability detection. In: Proc. of the 2021 Int'l Joint Conf. on Neural Networks (IJCNN). Shenzhen: IEEE, 2021. 1–8. [doi: 10.1109/IJCNN52387.2021.9534324]

[44] Wang W, Song JJ, Xu GQ, Li YD, Wang H, Su CH. ContractWard: Automated vulnerability detection models for Ethereum smart contracts. IEEE Trans. on Network Science and Engineering, 2021, 8(2): 1133–1144. [doi: 10.1109/TNSE.2020.2968505]

[45] Zhao XQ, Qu HP, Xu JL, Li S, Wang GG. AMSFuzz: An adaptive mutation schedule for fuzzing. Expert Systems with Applications, 2022, 208: 118162. [doi: 10.1016/j.eswa.2022.118162]

[46] Wüstholz V, Christakis M. Harvey: A greybox fuzzer for smart contracts. In: Proc. of the 28th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. ACM, 2020. 1398–1409. [doi: 10.1145/3368089.3417064]

[47] Tikhomirov S, Voskresenskaya E, Ivanitskiy I, Takhaviev R, Marchenko E, Alexandrov Y. SmartCheck: Static analysis of Ethereum smart contracts. In: Proc. of the 1st Int'l Workshop on Emerging Trends in Software Engineering for Blockchain. Gothenburg: ACM, 2018. 9–16. [doi: 10.1145/3194113.3194115]

[48] Liao ZQ, Hao SC, Nan YH, Zheng ZB. SmartState: Detecting state-reverting vulnerabilities in smart contracts via fine-grained state-dependency analysis. In: Proc. of the 32nd ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. Seattle: ACM, 2023. 980–991. [doi: 10.1145/3597926.3598111]

[49] Cai J, Li B, Zhang T, Zhang JL, Sun XB. Fine-grained smart contract vulnerability detection by heterogeneous code feature learning and automated dataset construction. Journal of Systems and Software, 2024, 209: 111919. [doi: 10.1016/j.jss.2023.111919]

[50] Schneidewind C, Grishchenko I, Scherer M, Maffei M. eThor: Practical and provably sound static analysis of Ethereum smart contracts. In: Proc. of the 2020 ACM SIGSAC Conf. on Computer and Communications Security. ACM, 2020. 621–640. [doi: 10.1145/3372297.3417250]

[51] Wang ZX, Chen JC, Wang YL, Zhang Y, Zhang WZ, Zheng ZB. Efficiently detecting reentrancy vulnerabilities in complex smart

contracts. In: Proc. of the 2024 ACM Software Engineering. New York: ACM, 2024. 161–181. [doi: 10.1145/3643734]

[52] Wang HJ, Liu Y, Li Y, Lin SW, Artho C, Ma L, Liu Y. Oracle-supported dynamic exploit generation for smart contracts. IEEE Trans. on Dependable and Secure Computing, 2022, 19(3): 1795–1809. [doi: 10.1109/TDSC.2020.3037332]

[53] Rodler M, Paaßen D, Li WT, Bernhard L, Holz T, Karame G, Davi L. EF↯CF: High performance smart contract fuzzing for exploit generation. In: Proc. of the 8th European Symp. on Security and Privacy (EuroS&P). Delft: IEEE, 2023. 449–471. [doi: 10.1109/EuroSP57164.2023.00034]

[54] Wang SJ, Pei KX, Yang JF. SmartInv: Multimodal learning for smart contract invariant inference. In: Proc. of the 2024 IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2024. 2217–2235. [doi: 10.1109/SP54263.2024.00126]

[55] Bose P, Das D, Chen YJ, Feng Y, Kruegel C, Vigna G. SAILFISH: Vetting smart contract state-inconsistency bugs in seconds. In: Proc. of the 2022 IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2022. 161–178. [doi: 10.1109/SP46214.2022.9833721]

[56] Zhou SF, Yang ZM, Xiang J, Cao YZ, Yang M, Zhang Y. An ever-evolving game: Evaluation of real-world attacks and defenses in Ethereum ecosystem. In: Proc. of the 29th USENIX Security Symp. USENIX Association, 2020. 2793–2809.

[57] He JX, Balunović M, Ambroladze N, Tsankov P, Vechev M. Learning to fuzz from symbolic execution with application to smart contracts. In: Proc. of the 2019 ACM SIGSAC Conf. on Computer and Communications Security. London: ACM, 2019. 531–548. [doi: 10.1145/3319535.3363230]

[58] Feng Y, Torlak E, Bodik R. Summary-based symbolic evaluation for smart contracts. In: Proc. of the 35th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE). ACM, 2020. 1141–1152. [doi: 10.1145/3324884.3416646]

[59] Kalra S, Goel S, Dhawan M, Sharma S. ZEUS: Analyzing safety of smart contracts. In: Proc. of the 2018 Network and Distributed System Security Symp. San Diego, 2018. 1–12. [doi: 10.14722/ndss.2018.23082]

[60] Brent L, Jurisevic A, Kong M, Liu E, Gauthier F, Gramoli V, Holz R, Scholz B. Vandal: A scalable security analysis framework for smart contracts. arXiv:1809.03981, 2018.

[61] Zhang XL, Sun WX, Xu ZC, Cheng HB, Cai CJ, Cui HL, Li Q. EVM-Shield: In-contract state access control for fast vulnerability detection and prevention. IEEE Trans. on Information Forensics and Security, 2024, 19: 2517–2532. [doi: 10.1109/TIFS.2024.3349852]

[62] Gao B, Wei QS, Liu Y, Goh RSM. Unveiling the potential of ChatGPT in detecting machine unauditable bugs in smart contracts: A preliminary evaluation and categorization. In: Proc. of the 2024 IEEE Conf. on Artificial Intelligence (CAI). Singapore: IEEE, 2024. 1481–1486. [doi: 10.1109/CAI59869.2024.00266]

[63] So S, Hong S, Oh H. SmarTest: Effectively hunting vulnerable transaction sequences in smart contracts through language model-guided symbolic execution. In: Proc. of the 30th USENIX Security Symp. USENIX Association, 2021. 1361–1378.

[64] Zhang MY, Zhang XK, Zhang YQ, Lin ZQ. TXSPECTOR: Uncovering attacks in Ethereum from transactions. In: Proc. of the 29th USENIX Security Symp. USENIX Association, 2020. 2775–2792.

[65] Permenev A, Dimitrov D, Tsankov P, Drachsler-Cohen D, Vechev M. VerX: Safety verification of smart contracts. In: Proc. of the 2020 IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2020. 1661–1677. [doi: 10.1109/SP40000.2020.00024]

[66] Choi J, Kim D, Kim S, Grieco G, Groce A, Cha SK. SMARTIAN: Enhancing smart contract fuzzing with static and dynamic data-flow analyses. In: Proc. of the 36th Int'l Conf. on Automated Software Engineering (ASE). Melbourne: IEEE, 2021. 227–239. [doi: 10.1109/ASE51524.2021.9678888]

[67] Torres CF, Iannillo AK, Gervais A, State R. ConFuzzius: A data dependency-aware hybrid fuzzer for smart contracts. In: Proc. of the 2021 IEEE European Symp. on Security and Privacy. Vienna: IEEE, 2021. 103–119. [doi: 10.1109/EuroSP51992.2021.00018]

[68] Tsankov P, Dan A, Drachsler-Cohen D, Gervais A, Bünzli F, Vechev M. Securify: Practical security analysis of smart contracts. In: Proc. of the 2018 ACM SIGSAC Conf. on Computer and Communications Security. Toronto: ACM, 2018. 67–82. [doi: 10.1145/3243734.3243780]

[69] Stephens J, Ferles K, Mariano B, Lahiri S, Dillig I. SmartPulse: Automated checking of temporal properties in smart contracts. In: Proc. of the 2021 IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2021. 555–571. [doi: 10.1109/SP40001.2021.00085]

[70] Chen T, Cao R, Li T, Luo XP, Gu GF, Zhang YF, Liao Z, Zhu H, Chen G, He ZY, Tang YX, Lin XD, Zhang XS. SODA: A generic online detection framework for smart contracts. In: Proc. of the NDSS Symp. 2020. San Diego, 2020. 1–17. [doi: 10.14722/ndss.2020.24449]

[71] Durieux T, Ferreira JF, Abreu R, Cruz P. Empirical review of automated analysis tools on 47, 587 Ethereum smart contracts. In: Proc. of the 42nd Int'l Conf. on Software Engineering. Seoul: ACM, 2020. 530–541. [doi: 10.1145/3377811.3380364]

[72] Grishchenko I, Maffei M, Schneidewind C. A semantic framework for the security analysis of Ethereum smart contracts. In: Proc. of the 7th Int'l Conf. on Principles of Security and Trust. Thessaloniki: Springer, 2018. 243–269. [doi: 10.1007/978-3-319-89722-6_10]

[73] Grech N, Brent L, Scholz B, Smaragdakis Y. Gigahorse: Thorough, declarative decompilation of smart contracts. In: Proc. of the 41st

Int'l Conf. on Software Engineering (ICSE). Montreal: IEEE, 2019. 1176–1186. [doi: 10.1109/ICSE.2019.00120]

[74] Chen T, Li XQ, Luo XP, Zhang XS. Under-optimized smart contracts devour your money. In: Proc. of the 24th Int'l Conf. on Software Analysis, Evolution and Reengineering (SANER). Klagenfurt: IEEE, 2017. 442–446. [doi: 10.1109/SANER.2017.7884650]

[75] Grech N, Kong M, Jurisevic A, Brent L, Scholz B, Smaragdakis Y. MadMax: Surviving out-of-gas conditions in Ethereum smart contracts. In: Proc. of the 2018 ACM on Programming Languages. New York: ACM, 2018. 116. [doi: 10.1145/3276486]

[76] Nguyen TD, Pham LH, Sun J. SGUARD: Towards fixing vulnerable smart contracts automatically. In: Proc. of the 2021 IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2021. 1215–1229. [doi: 10.1109/SP40001.2021.00057]

[77] Krupp J, Rossow C. TEETHER: Gnawing at Ethereum to automatically exploit smart contracts. In: Proc. of the 27th USENIX Security Symp. Baltimore: USENIX Association, 2018. 1317–1333.

[78] Ghaleb A, Rubin J, Pattabiraman K. AChecker: Statically detecting smart contract access control vulnerabilities. In: Proc. of the 45th Int'l Conf. on Software Engineering (ICSE). Melbourne: IEEE, 2023. 945–956. [doi: 10.1109/ICSE48619.2023.00087]

[79] Sun JL, Huang S, Zheng CY, Wang TY, Zong C, Hui ZW. Mutation testing for integer overflow in Ethereum smart contracts. Tsinghua Science and Technology, 2022, 27(1): 27–40. [doi: 10.26599/TST.2020.9010036]

[80] Zhang B. Towards finding accounting errors in smart contracts. In: Proc. of the 46th Int'l Conf. on Software Engineering. Lisbon: Association for Computing Machinery, 2024. 138. [doi: 10.1145/3597503.3639128]

[81] Sun YQ, Wu DY, Xue Y, Liu H, Wang HJ, Xu ZZ, Xie XF, Liu Y. GPTScan: Detecting logic vulnerabilities in smart contracts by combining GPT with program analysis. In: Proc. of the 46th Int'l Conf. on Software Engineering. Lisbon: ACM, 2024. 166. [doi: 10.1145/3597503.3639117]

[82] Feist J, Grieco G, Groce A. Slither: A static analysis framework for smart contracts. In: Proc. of the 2nd Int'l Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB). Montreal: IEEE, 2019. 8–15. [doi: 10.1109/WETSEB.2019.00008]

[83] Atzei N, Bartoletti M, Cimoli T. A survey of attacks on Ethereum smart contracts (SoK). In: Proc. of the 6th Int'l Conf. on Principles of Security and Trust. Uppsala: Springer, 2017. 164–186. [doi: 10.1007/978-3-662-54455-6_8]

[84] Etherscan. Price Manipulation Attack. 2024. https://basescan.org/tx/0x6ab5b7b51f780e8c6c5ddaf65e9badb868811a95c1fd64e86435283074d3149e

[85] Torres CF, Camino R, State R. Frontrunner jones and the raiders of the dark forest: An empirical study of frontrunning on the Ethereum blockchain. In: Proc. of the 30th USENIX Security Symp. USENIX Association, 2021. 1343–1359.

[86] Wang Y, Zuest P, Yao YX, Lu ZC, Wattenhofer R. Impact and user perception of sandwich attacks in the DeFi ecosystem. In: Proc. of the 2022 CHI Conf. on Human Factors in Computing Systems. New Orleans: ACM, 2022. 591. [doi: 10.1145/3491102.3517585]

[87] Qin KH, Zhou LY, Livshits B, Gervais A. Attacking the DeFi ecosystem with flash loans for fun and profit. In: Proc. of the 25th Int'l Conf. on Financial Cryptography and Data Security. Springer, 2021. 3–32. [doi: 10.1007/978-3-662-64322-8_1]

[88] Vakhmyanin I, Volkovich Y. Price arbitrage for DeFi derivatives. In: Proc. of the 2023 IEEE Int'l Conf. on Blockchain and Cryptocurrency (ICBC). Dubai: IEEE, 2023. 1–4. [doi: 10.1109/ICBC56567.2023.10174884]

[89] Tjiam K, Wang R, Chen HH, Liang KT. Your smart contracts are not secure: Investigating arbitrageurs and oracle manipulators in Ethereum. In: Proc. of the 3rd Workshop on Cyber-security Arms Race. ACM, 2021. 25–35. [doi: 10.1145/3474374.3486916]

[90] CoinEx Help Center. What's Deflationary Token. 2024. https://support.coinex.com/hc/en-us/articles/4415202344345-What-s-Deflationary-Token

[91] Gottsegen W. DeFi exploit siphons $45 million from PancakeBunny, crashes token price. Decrypt. 2021. https://decrypt.co/71585/pancakebunny-defi-exploit

[92] Sam Kessler BB. Crypto bridge nomad drained of nearly $200m in exploit. 2022. https://www.coindesk.com/tech/2022/08/02/nomad-bridge-drained-of-nearly-200-million-in-exploit/

[93] Zhang MY, Zhang XK, Zhang YQ, Lin ZQ. Security of cross-chain bridges: Attack surfaces, defenses, and open problems. In: Proc. of the 27th Int'l Symp. on Research in Attacks, Intrusions and Defenses. Padua: ACM, 2024. 298–316. [doi: 10.1145/3678890.3678894]

[94] Len J, Grubbs P, Ristenpart T. Partitioning oracle attacks. In: Proc. of the 30th USENIX Security Symp. USENIX Association, 2021. 195–212.

[95] SlowMist. Blockchain Security and AML Annual Report, 2024. https://www.slowmist.com/report/2024-Blockchain-Security-and-AML-Annual-Report(CN).pdf

[96] Lin ZW, Chen JC, Wu JJ, Zhang WZ, Wang YJ, Zheng ZB. CRPWarner: Warning the risk of contract-related rug pull in DeFi smart contracts. IEEE Trans. on Software Engineering, 2024, 50(6): 1534–1547. [doi: 10.1109/TSE.2024.3392451]

[97] Li WK, Bu JY, Li XQ, Peng HL, Niu YZ, Zhang YQ. A survey of DeFi security: Challenges and opportunities. Journal of King Saud University-computer and Information Sciences, 2022, 34(10): 10378–10404. [doi: 10.1016/j.jksuci.2022.10.028]

[98] Wang DB, Feng H, Wu SW, Zhou YJ, Wu L, Yuan XL. Penny wise and pound foolish: Quantifying the risk of unlimited approval of ERC20 tokens on Ethereum. In: Proc. of the 25th Int'l Symp. on Research in Attacks, Intrusions and Defenses. Limassol: ACM, 2022. 99–114. [doi: 10.1145/3545948.3545963]

[99] Lardinois F. Beanstalk lost $182 million in governance attack. The Verge. 2022. https://www.theverge.com/2022/4/22/23037325/beanstalk-dismissed-governance-attacks-lost-182-million

[100] Dotan M, Yaish A, Yin HC, Tsytkin E, Zohar A. The vulnerable nature of decentralized governance in DeFi. In: Proc. of the 2023 Workshop on Decentralized Finance and Security. Copenhagen: ACM, 2023. 25–31. [doi: 10.1145/3605768.3623539]

[101] Zhu J, Hu K, Zhang BJ. Review on formal verification of smart contract. Acta Electronica Sinica, 2021, 49(4): 792–804 (in Chinese with English abstract). [doi: 10.12263/DZXB.20200723]

[102] Bhargavan K, Delignat-Lavaud A, Fournet C, Gollamudi A, Gonthier G, Kobeissi N, Kulatova N, Rastogi A, Sibut-Pinote T, Swamy N, Zanella-Béguelin S. Formal verification of smart contracts: Short paper. In: Proc. of the 2016 ACM Workshop on Programming Languages and Analysis for Security. Vienna: ACM, 2016. 91–96. [doi: 10.1145/2993600.2993611]

[103] Kalra S, Goel S, Dhawan M, Sharma S. ZEUS: Analyzing safety of smart contracts. In: Proc. of the NDSS Symp. 2018. San Diego, 2018. 1–12. [doi: 10.14722/ndss.2018.23082]

[104] Hildenbrandt E, Saxena M, Rodrigues N, Zhu XR, Daian P, Guth D, Moore B, Park D, Zhang Y, Stefanescu A, Rosu G. KEVM: A complete formal semantics of the Ethereum virtual machine. In: Proc. of the 31st Computer Security Foundations Symp. (CSF). Oxford: IEEE, 2018. 204–217. [doi: 10.1109/CSF.2018.00022]

[105] Baldoni R, Coppa E, D'elia DC, Demetrescu C, Finocchi I. A survey of symbolic execution techniques. ACM Computing Surveys (CSUR), 2018, 51(3): 50. [doi: 10.1145/3182657]

[106] Wang L, Li F, Li L, Feng XB. Principle and practice of taint analysis. Ruan Jian Xue Bao/Journal of Software, 2017, 28(4): 860–882 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5190.htm [doi: 10.13328/j.cnki.jos.005190]

[107] Torres CF, Schütte J, State R. Osiris: Hunting for integer bugs in Ethereum smart contracts. In: Proc. of the 34th Annual Computer Security Applications Conf. San Juan: ACM, 2018. 664–676. [doi: 10.1145/3274694.3274737]

[108] Brent L, Grech N, Lagouvardos S, Scholz B, Smaragdakis Y. Ethainter: A smart contract security analyzer for composite vulnerabilities. In: Proc. of the 41st ACM SIGPLAN Conf. on Programming Language Design and Implementation. London: ACM, 2020. 454–469. [doi: 10.1145/3385412.3385990]

[109] Wesley S, Christakis M, Navas JA, Trefler R, Wüstholz V, Gurfinkel A. Verifying Solidity smart contracts via communication abstraction in SmartACE. In: Proc. of the 23rd Int'l Conf. on Verification, Model Checking, and Abstract Interpretation. Philadelphia: Springer, 2022. 425–449. [doi: 10.1007/978-3-030-94583-1_21]

[110] Albert E, Gordillo P, Livshits B, Rubio A, Sergey I. EthIR: A framework for high-level analysis of Ethereum bytecode. In: Proc. of the 16th Int'l Symp. on Automated Technology for Verification and Analysis. Los Angeles: Springer Int'l Publishing, 2018. 513–520. [doi: 10.1007/978-3-030-01090-4_30]

[111] Manès VJM, Han HS, Han C, Cha SK, Egele M, Schwartz EJ, Woo M. The art, science, and engineering of fuzzing: A survey. IEEE Trans. on Software Engineering, 2021, 47(11): 2312–2331. [doi: 10.1109/TSE.2019.2946563]

[112] Olsthoorn M, Stallenberg D, Van Deursen A, Panichella A. SynTest-Solidity: Automated test case generation and fuzzing for smart contracts. In: Proc. of the 44th Int'l Conf. on Software Engineering: Companion Proc. Pittsburgh: IEEE, 2022. 202–206. [doi: 10.1145/3510454.3516869]

[113] Liu H, Liu C, Zhao WQ, Jiang Y, Sun JG. S-gram: Towards semantic-aware security auditing for Ethereum smart contracts. In: Proc. of the 33rd ACM/IEEE Int'l Conf. on Automated Software Engineering. Montpellier: ACM, 2018. 814–819. [doi: 10.1145/3238147.3240728]

[114] Zhuang Y, Liu ZG, Qian P, Liu Q, Wang X, He QM. Smart contract vulnerability detection using graph neural network. In: Proc. of the 29th Int'l Joint Conf. on Artificial Intelligence. Yokohama, 2021. 3283–3290. [doi: 10.24963/ijcai.2020/454]

[115] Tann WJW, Han XJ, Gupta SS, Ong YS. Towards safer smart contracts: A sequence learning approach to detecting security threats. arXiv:1811.06632, 2019.

[116] Jie WQ, Chen Q, Wang JQ, Voundi Koe AS, Li J, Huang PF, Wu YQ, Wang Y. A novel extended multimodal AI framework towards vulnerability detection in smart contracts. Information Sciences, 2023, 636: 118907. [doi: 10.1016/j.ins.2023.03.132]

[117] Ferreira JF, Cruz P, Durieux T, Abreu R. SmartBugs: A framework to analyze Solidity smart contracts. In: Proc. of the 35th IEEE/ACM Int'l Conf. on Automated Software Engineering. ACM, 2020. 1349–1352. [doi: 10.1145/3324884.3415298]

[118] Hu SH, Huang TS, İlhan F, Tekin SF, Liu L. Large language model-powered smart contract vulnerability detection: New perspectives. In: Proc. of the 5th IEEE Int'l Conf. on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA). Atlanta: IEEE,

2023. 297–306. [doi: 10.1109/TPS-ISA58951.2023.00044]

[119] Yu L, Lu JY, Liu XL, Yang L, Zhang FJ, Ma JJ. PSCVFinder: A prompt-tuning based framework for smart contract vulnerability detection. In: Proc. of the 34th Int'l Symp. on Software Reliability Engineering (ISSRE). Florence: IEEE, 2023. 556–567. [doi: 10.1109/ISSRE59848.2023.00030]

[120] Ghaleb A, Pattabiraman K. How effective are smart contract analysis tools? Evaluating smart contract static analysis tools using bug injection. In: Proc. of the 29th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. ACM, 2020. 415–427. [doi: 10.1145/3395363.3397385]

[121] Ma W, Wu DY, Sun YQ, Wang TW, Liu SQ, Zhang J, Xue Y, Liu Y. Combining fine-tuning and LLM-based agents for intuitive smart contract auditing with justifications. arXiv:2403.16073, 2024.

[122] Soud M, Nuutinen W, Liebel G. Soley: Identification and automated detection of logic vulnerabilities in ethereum smart contracts using large language models. arXiv:2406.16244, 2024.

[123] Liu YH, Ott M, Goyal N, Du JF, Joshi M, Chen DQ, Levy O, Lewis M, Zettlemoyer L, Stoyanov V. RoBERTa: A robustly optimized BERT pretraining approach. arXiv:1907.11692, 2019.

[124] Feng ZY, Guo DY, Tang DY, Duan N, Feng XC, Gong M, Shou LJ, Qin B, Liu T, Jiang DX, Zhou M. CodeBERT: A pre-trained model for programming and natural languages. arXiv:2002.08155, 2020.

[125] Liu Y, Xue Y, Wu DY, Sun YQ, Li Y, Shi ML, Liu Y. PropertyGPT: LLM-driven formal verification of smart contracts through retrieval-augmented property generation. arXiv:2405.02580, 2024.

[126] Shou CF, Liu J, Lu DD, Sen K. LLM4Fuzz: Guided fuzzing of smart contracts with large language models. arXiv:2401.11108, 2024.

[127] Sun JZ, Yin ZQ, Zhang HS, Chen X, Zheng W. Adversarial generation method for smart contract fuzz testing seeds guided by chain-based LLM. Automated Software Engineering, 2025, 32(1): 12. [doi: 10.1007/s10515-024-00483-4]

[128] Zhang WQ, Zhang Z, Shi QK, Liu L, Wei LL, Liu YP, Zhang XY, Cheung SC. Nyx: Detecting exploitable front-running vulnerabilities in smart contracts. In: Proc. of the 2024 IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2024. 2198–2216. [doi: 10.1109/SP54263.2024.00146]

[129] Zhou LY, Qin KH, Torres CF, Le DV, Gervais A. High-frequency trading on decentralized on-chain exchanges. In: Proc. of the 2021 IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2021. 428–445. [doi: 10.1109/SP40001.2021.00027]

[130] Heimbach L, Wattenhofer R. Eliminating sandwich attacks with the help of game theory. In: Proc. of the 2022 ACM on Asia Conf. on Computer and Communications Security. Nagasaki: ACM, 2022. 153–167. [doi: 10.1145/3488932.3517390]

[131] Li DZ, Zhang KJ, Wang L, Du G. A Geth-based real-time detection system for sandwich attacks in Ethereum. Discover Computing, 2024, 27(1): 11. [doi: 10.1007/s10791-024-09445-6]

[132] Xia Q, Huang ZR, Dou WS, Zhang YF, Zhang FJ, Liang G, Zuo C. Detecting flash loan based attacks in Ethereum. In: Proc. of the 43rd Int'l Conf. on Distributed Computing Systems (ICDCS). Hong Kong: IEEE, 2023. 154–165. [doi: 10.1109/ICDCS57875.2023.00078]

[133] Chen ZY, Beillahi SM, Long F. FlashSyn: Flash loan attack synthesis via counter example driven approximation. In: Proc. of the 46th Int'l Conf. on Software Engineering. Lisbon: ACM, 2024. 142. [doi: 10.1145/3597503.3639190]

[134] Li WK, Li XQ, Zhang YQ, Li ZW. DeFiTail: DeFi protocol inspection through cross-contract execution analysis. In: Proc. of the 2024 ACM on Web Conf. Singapore: ACM, 2024. 786–789. [doi: 10.1145/3589335.3651488]

[135] Wang B, Liu H, Liu C, Yang ZQ, Ren Q, Zheng HX, Lei H. BLOCKEYE: Hunting for DeFi attacks on blockchain. In: Proc. of the 43rd Int'l Conf. on Software Engineering: Companion Proc. (ICSE-companion). Madrid: IEEE, 2021. 17–20. [doi: 10.1109/ICSE-Companion52605.2021.00025]

[136] Wang SH, Wu CC, Liang YC, Hsieh LH, Hsiao HC. ProMutator: Detecting vulnerable price oracles in DeFi by mutated transactions. In: Proc. of the 2021 IEEE European Symp. on Security and Privacy Workshops (EuroS&PW). Vienna: IEEE, 2021. 380–385. [doi: 10.1109/EuroSPW54576.2021.00047]

[137] Deng X, Beillahi SM, Minwalla C, Du H, Veneris A, Long F. Safeguarding DeFi smart contracts against oracle deviations. In: Proc. of the 46th Int'l Conf. on Software Engineering. Lisbon: ACM, 2024. 171. [doi: 10.1145/3597503.3639225]

[138] Arora S, Li YJ, Feng YB, Xu JH. SecPLF: Secure protocols for loanable funds against oracle manipulation attacks. In: Proc. of the 19th ACM Asia Conf. on Computer and Communications Security. Singapore: ACM, 2024. 1394–1405. [doi: 10.1145/3634737.3637681]

[139] Wu SW, Yu Z, Wang DB, Zhou YJ, Wu L, Wang HY, Yuan XL. DeFiRanger: Detecting DeFi price manipulation attacks. IEEE Trans. on Dependable and Secure Computing, 2024, 21(4): 4147–4161. [doi: 10.1109/TDSC.2023.3346888]

[140] Kong QP, Chen JC, Wang YL, Jiang ZG, Zheng ZB. Defitainter: Detecting price manipulation vulnerabilities in DeFi protocols. In: Proc. of the 32nd ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. Seattle: ACM, 2023. 1144–1156. [doi: 10.1145/3597926.3598124]

[141] Lin LS, Zheng HQ, Su S, Lei K, Chen XF, Tian ZH. An online defense mechanism for DeFi price manipulation attacks. Journal of Computer Research and Development, 2025, 62(2): 443–457 (in Chinese with English abstract).

[142] Zhang JS, Gao JB, Li Y, Chen ZM, Guan Z, Chen Z. Xscope: Hunting for cross-chain bridge attacks. In: Proc. of the 37th Int'l Conf. on Automated Software Engineering. Rochester: ACM, 2022. 171. [doi: 10.1145/3551349.3559520]

[143] Liao ZQ, Nan YH, Liang HL, Hao SC, Zhai J, Wu JJ, Zheng ZB. SmartAxe: Detecting cross-chain vulnerabilities in bridge smart contracts via fine-grained static analysis. In: Proc. of the 2024 ACM on Software Engineering. New York: ACM, 2024. 249–270. [doi: 10.1145/3643738]

[144] Liao ZQ, Zheng ZB, Chen X, Nan YH. SmartDagger: A bytecode-based static analysis approach for detecting cross-contract vulnerability. In: Proc. of the 31st ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. ACM, 2022. 752–764. [doi: 10.1145/3533767.3534222]

[145] Wang B, Yuan XH, Duan L, Ma HL, Wang B, Su CH, Wang W. DeFiScanner: Spotting DeFi attacks exploiting logic vulnerabilities on blockchain. IEEE Trans. on Computational Social Systems, 2024, 11(2): 1577–1588. [doi: 10.1109/TCSS.2022.3228122]

[146] Parhizkari B, Iannillo AK, Torres CF, Banescu S, Xu J, State R. Timely identification of victim addresses in DeFi attacks. In: Proc. of the 2024 European Symp. on Research in Computer Security. The Hague: Springer, 2024. 394–410. [doi: 10.1007/978-3-031-54204-6_24]

附中文参考文献

[20] 钱鹏, 刘振广, 何钦铭, 黄步添, 田端正, 王勋. 智能合约安全漏洞检测技术研究综述. 软件学报, 2022, 33(8): 3059–3085. http://www.jos.org.cn/1000-9825/6375.htm [doi: 10.13328/j.cnki.jos.006375]

[21] 董伟良, 刘哲, 刘逵, 黎立, 葛春鹏, 黄志球. 智能合约漏洞检测技术综述. 软件学报, 2024, 35(1): 38–62. http://www.jos.org.cn/1000-9825/6810.htm [doi: 10.13328/j.cnki.jos.006810]

[22] 崔展齐, 杨慧文, 陈翔, 王林章. 智能合约安全漏洞检测研究进展. 软件学报, 2024, 35(5): 2235–2267. http://www.jos.org.cn/1000-9825/7046.htm [doi: 10.13328/j.cnki.jos.007046]

[23] 刘哲旭, 李雷孝, 刘东江, 杜金泽, 林浩, 史建平. 智能合约漏洞检测与修复研究综述. 计算机科学与探索, 2025, 19(4): 854–876. [doi: 10.3778/j.issn.1673-9418.2405019]

[101] 朱健, 胡凯, 张伯钧. 智能合约的形式化验证方法研究综述. 电子学报, 2021, 49(4): 792–804. [doi: 10.12263/DZXB.20200723]

[106] 王蕾, 李丰, 李炼, 冯晓兵. 污点分析技术的原理和实践应用. 软件学报, 2017, 28(4): 860–882. http://www.jos.org.cn/1000-9825/5190.htm [doi: 10.13328/j.cnki.jos.005190]

[141] 林炼升, 郑焕钦, 苏申, 雷凯, 陈晓丰, 田志宏. 一种 DeFi 价格操纵攻击在线防御机制. 计算机研究与发展, 2025, 62(2): 443–457.

作者简介

揭晚晴, 博士生, CCF 学生会员, 主要研究领域为 Web3.0 安全, 智能合约安全.

邱望洁, 博士, 副研究员, 博士生导师, CCF 专业会员, 主要研究领域为区块链, 隐私计算.

黄鑫鹏, 博士生, 主要研究领域为区块链, DeFi, 机器学习.

杨浩甫, 本科生, 主要研究领域为区块链, 智能合约安全.

赵冠球, 硕士生, 主要研究领域为区块链, 智能合约安全.

张沁楠, 博士, CCF 专业会员, 主要研究领域为区块链安全, 联邦学习激励机制, 隐私增强技术.

夏清, 博士, 副研究员, CCF 专业会员, 主要研究领域为区块链攻击检测, 大模型应用.

郑宏威, 博士, 主任研究员, 博士生导师, 主要研究领域为区块链, 联邦学习, 隐私计算.

郑志明, 博士, 教授, 博士生导师, CCF 会士, 主要研究领域为区块链, 网络安全, 人工智能.