基于大语言模型的多智能体自动渗透测试框架构建与评估

江 颉 [1] 王 豪 [1] 李明达 [1] 朱添田 [1, 2]

摘 要 渗透测试作为一种主动的安全评估手段, 在保障网络安全中发挥着至关重要的作用. 传统的渗透测试通常高度依赖专家经验和人工操作, 测试过程复杂且耗时. 基于大语言模型的渗透测试智能体能够在测试环境中生成和调整策略, 相较于传统的方式, 具备更强的创新性和适应性. 在大语言模型辅助渗透测试的过程中, 存在因测试路径偏移、大语言模型"幻觉"问题而导致渗透测试任务中断或失败的情况. 基于此, 提出一个基于大语言模型的多智能体渗透测试框架 LangPentest, 旨在通过自然语言处理技术提高攻击策略的自动生成和执行能力, 框架采用了大语言模型驱动的程序框架 (LangChain) 和检索增强生成技术, 提高 LangPentest 性能并降低大语言模型在应用渗透测试方面的"幻觉"问题. 框架由任务生成、任务执行、经验管理和任务调整四部分模块组成, 对基准目标测试后, 基于 Claude 3.5 Sonnet 模型的框架任务成功率最高; 且与 AutoGPT 和 PentestGPT 相比, 本框架在任务成功率方面具有明显优势, 在任务完成和整体性能方面证明了 LangPentest 的可行性和有效性.

关键词 网络安全; 渗透测试; 大语言模型; 多智能体; 检索增强生成; 人工智能

引用格式 江颉, 王豪, 李明达, 朱添田. 基于大语言模型的多智能体自动渗透测试框架构建与评估. 自动化学报, 2026, **52**(4): 821−832

DOI 10.16383/j.aas.c250293 **CSTR** 32138.14.j.aas.c250293

Construction and Evaluation of Multi-agent Automated Penetration Testing Framework Based on Large Language Models

JIANG Jie[1] WANG Hao[1] LI Ming-Da[1] ZHU Tian-Tian[1, 2]

Abstract Penetration testing, as an active security assessment approach, plays a vital role in guaranteeing network security. Conventional penetration testing typically relies heavily on expert experience and manual operations, resulting in a complex and time-consuming testing process. The penetration testing agent based on large language models is capable of generating and adjusting strategies within the testing environment. Compared with traditional methods, it demonstrates stronger innovativeness and adaptability. During the process of large language model assisted penetration testing, there exist situations where the penetration testing tasks are interrupted or fail due to test path deviations and the "hallucination" issue of large language models. Based on this, a multi-agent penetration testing framework based on large language models, namely LangPentest, is proposed, aiming to enhance the automatic generation and execution of attack strategies through natural language processing techniques. The framework employs a large language model driven program framework (LangChain) and retrieval-augmented generation technology to improve the performance of LangPentest and mitigate the "hallucination" problem of large language models in the application of penetration testing. The framework is composed of four modules: Task generation, task execution, experience management, and task adjustment. After benchmark target testing, the framework based on the Claude 3.5 Sonnet model achieves the highest task success rate. In comparison with AutoGPT and PentestGPT, this framework exhibits a distinct advantage in terms of task success rate, and proves the feasibility and effectiveness of LangPentest in task completion and overall performance.

Keywords cybersecurity; penetration testing; large language model; multi-agent; retrieval-augmented generation; artificial intelligence

Citation Jiang Jie, Wang Hao, Li Ming-Da, Zhu Tian-Tian. Construction and evaluation of multi-agent automated penetration testing framework based on large language models. *Acta Automatica Sinica*, 2026, **52**(4): 821−832

收稿日期 2025-07-03 录用日期 2025-12-19
Manuscript received July 3, 2025; accepted December 19, 2025
浙江省属高校基本科研业务费专项资金 (RF-A2023009), 国家自然科学基金青年项目 (62002324), 浙江省高等教育 2025 年研究生教学改革项目 (JGCG2025539) 资助
Supported by the Special Funds for Basic Scientific Research Operation Expenses of Zhejiang Provincial Universities (RF-A2023009), Youth Program of National Natural Science Foundation of China (62002324), and Zhejiang Province Higher Education 2025 Postgraduate Teaching Reform Project (JGCG2025539)
本文责任编委 张家俊
Recommended by Associate Editor ZHANG Jia-Jun

1. 浙江工业大学计算机科学与技术学院 杭州 310023 2. 浙江工业大学台州研究院 台州 318001
1. College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023 2. Taizhou Research Institute, Zhejiang University of Technology, Taizhou 318001

随着信息技术的飞速发展, 网络系统已深度融入社会各个领域, 从个人通信到关键基础设施的运行, 几乎所有行业都依赖于安全可靠的网络环境. 近年来, 网络攻击的数量与复杂程度持续增加[1], 包括数据泄露、勒索软件等在内的安全事件频发, 对个人隐私和社会稳定构成了严重威胁. 因此, 如何对网络系统的安全性进行全面评估, 并及时发现和修复潜在的漏洞, 已成为确保信息资产安全的关键任务.

作为主动防御技术[2], 渗透测试是一种通过执行可能的黑客攻击, 识别潜在的安全漏洞来评估计算机系统、网络或应用程序安全性的重要方法. 通过模拟实际的攻击行为, 渗透测试帮助网络安全工作者了解系统的脆弱点并为修复提供指导, 帮助其降低实际的网络安全风险.

传统的渗透测试方法主要依赖于安全人员进行漏洞识别和攻击实施[3], 根据预定的测试目标和攻击场景, 通过人工或半自动化工具执行渗透测试. 该过程的核心包括识别和扫描目标系统的漏洞、利用已发现的漏洞进行攻击、验证漏洞的可利用性, 并最终评估渗透测试的结果. 该方式具有对特定系统进行定制化攻击测试的优点, 但也存在人工投入大、效率低、测试的深度和全面性高度依赖专家经验的缺点[4]. 为应对不断增加的网络安全威胁, 安全领域亟须一种更加高效、自动化的解决方案.

同时, 随着人工智能领域的快速发展, 特别是大语言模型 (large language model, LLM) 的兴起[5], 为渗透测试智能化带来全新的解决思路. 大语言模型具有卓越的自然语言理解和生成能力, 能够高效处理复杂信息并推导多种可能性. 随着大语言模型技术的进步及任务需求的推动, 基于大语言模型的智能体 (agent)[6] 快速发展, 智能体不仅限于传统的文本生成任务, 还涵盖自动化决策、智能体的设计以及复杂系统的优化等多方面[7]. 基于大语言模型的智能体也逐渐在多种场景中得以广泛应用[8−9].

在此背景下, 基于大语言模型的智能体应用于渗透测试[10−12], 能够在不同的网络环境下生成攻击策略并动态调整测试路径, 有效地减少对人工干预的依赖. 然而, 将智能体直接应用于渗透测试中仍然存在以下挑战:

偏移问题. LLM 在应用于渗透测试的多步决策过程中, 容易出现理解与执行偏差, 此问题会通过级联传播机制产生非线性路径偏移, 从而导致任务的计划和执行与预期目标产生偏差.

幻觉问题. LLM 在应用于渗透测试过程中, 会出现与现实不符的幻觉[13−14] 问题, LLM 可能生成看似合理但实际无效的 payload、构造违反目标系统架构规范的 shellcode 指令、虚构不存在的漏洞组合攻击路径 (如跨通用漏洞披露 (common vulnerabilities & exposures, CVE)[15] 编号的虚假依赖关系). 此类问题往往是由于大语言模型在渗透测试领域知识学习有限, 导致生成的渗透测试任务缺乏足够的可执行性或正确性.

针对上述局限, 本文提出一个新的渗透测试框架 LangPentest, 其是由大语言模型驱动的多智能体系统. 本文的主要贡献如下:

任务调整. 提出任务调整驱动的多智能体动态协同机制, 通过反馈感知与任务链重构, 使该框架具备基于执行后状态的自适应任务重规划能力, 缓解渗透测试过程中任务的计划和执行与目标之间的偏移现象, 为复杂多步决策问题提供新的多智能体协调范式.

经验优化. 借鉴增强检索思考 (retrieval augmented thoughts, RAT)[16], 以渗透知识库和历史任务轨迹为检索源, 将零提示思维链 (chain-of-thought, CoT) 和 RAG (retrieval-augmented generation)[17] 相结合. 具体地, LangPentest 在生成每一步思维链后, 会利用该步及其上下文信息从经验库中检索相关渗透知识与历史案例, 并基于检索结果对思维链进行因果式修订. 据此, 模型的推理过程不再依赖内部联想, 每一步推理与外部经验知识绑定, 使中间思维链具备证据支撑. 这种“逐步思考–检索–修订”的机制, 能在推理层面缓解大语言模型由于专业知识不足而产生的幻觉问题, 改善由此问题导致的执行失败情况.

LangPentest 将上述任务调整、经验优化与任务生成、任务执行结合起来, 聚焦于全局任务规划与动态调整, 旨在缓解 LLM 在渗透测试领域的幻觉问题, 提高任务执行能力.

本文后续内容的结构如下: 第 1 节回顾渗透测试工具、基于强化学习和大语言模型的渗透测试相关工作; 第 2 节介绍多智能体渗透测试框架的设计和 RAG 经验检索; 第 3 节概述本研究中实验设置; 第 4 节详细介绍实验并对结果进行评估; 第 5 节对研究进行总结, 并探讨局限性和未来研究方向.

1 相关工作

1.1 渗透测试相关工具

自动化渗透测试工具[18] 是近年来发展迅速的研究领域, 旨在通过使用自动化工具来执行渗透测试的某些步骤, 从而减少人工参与的程度, 提高渗

透测试效率和覆盖范围. 目前市面上已有许多自动化渗透测试工具, 通常侧重于渗透测试流程中的特定单项任务. 例如, Nmap[19] 是一个专注于信息收集的工具, 它通过对网络中的设备、端口和服务等进行直接扫描, 从目标中收集相应数据信息. OpenVAS[20] 主要用于检测网络和系统中的安全漏洞, 通过集成的服务和工具提供全面的漏洞扫描功能. Metasploit Framework[21] 的重点在于漏洞的利用, 一旦识别出漏洞, 它能够提供多种可定制的 payload 攻击. 虽然这些工具在特定任务中表现出色, 但是掌握这些工具的使用并将其集成到一个连贯的攻击计划中, 需要大量的渗透测试专业知识和安全工作者的手动操作.

1.2　基于强化学习的渗透测试

随着机器学习技术的兴起, 自动化渗透测试领域也逐渐开始采用这些先进技术来提高渗透测试的准确性和灵活性. 强化学习 (reinforcement learning, RL)[22−23] 作为一种智能决策技术, 为自动化渗透测试提供了一种新的解决方案. 其通过模拟智能体与环境间的交互, 不断学习并优化测试策略, 实现了对攻击路径的自动规划和动态调整. Takaesu[24] 提出 DeepExploit, DeepExploit 是一个使用强化学习进行漏洞利用和攻击路径生成的自动化渗透测试框架. 它采用了深度强化学习 (deep reinforcement learning, DRL) 来模拟渗透测试人员的决策过程, 能够自动选择最优的攻击路径, 并针对目标系统的漏洞进行利用, 在多种网络环境中展现了较高的灵活性和攻击成功率. Moreno 等[25] 提出一种结合上下文感知 Transformer 与强化学习估计器的自主渗透测试方法, 通过对渗透环境特征进行编码并生成候选攻击步骤, 再利用 RL 模型评估各操作的可行性与影响, 从而实现对后续测试动作的推荐与调整, 为自动化渗透流程提供更具针对性的决策支持. 高文龙等[26] 提出的深度强化学习算法在 DDQN 算法的基础上增加了路径启发信息和深度优先渗透的动作选择策略, 路径启发信息充分挖掘历史经验, 修剪了深度优先渗透的动作选择策略空间, 有助于更快地收敛至最优策略. 然而, 这些方法仍面临一些局限性, 如实际网络环境中状态空间维度过高、真实数据稀缺、模型对抗性较弱以及训练时间较长等问题, 使得其在真实场景中的部署和应用尚需进一步探索和优化.

1.3　基于大语言模型的渗透测试

基于 LLM 的渗透测试技术, 近年来受到了研究者的广泛关注. 如 GPT-3.5[27] 和 GPT-4[28] 等大语言模型凭借其强大的自然语言理解和生成能力, 可以在渗透测试过程中发挥重要作用. 基于大语言模型的渗透测试框架能够通过自然语言输入自动生成攻击策略, 与外部安全工具协作完成更复杂的攻击任务. Deng 等[29] 提出利用大语言模型进行自动渗透测试的框架 PentestGPT, 其由三个自交互模块构成, 分别是推理模块、生成模块以及解析模块, 这三个模块分别处理渗透测试阶段的子任务, 通过在设计的基准上进行实验, 证明了 PentestGPT 在应对渗透测试目标和夺旗赛 (capture the flag, CTF) 任务中的有效性. Happe 等[12] 提出低层次攻击执行的闭环反馈系统 HackingBuddyGPT, 用于连接 GPT-3.5 和漏洞虚拟机, 通过提交命令、执行命令、收集输出并分析输出, 以生成新命令的反馈循环方式来运行, 实验结果表明其能在多个场景下成功获取 root 权限. Xu 等[30] 提出大语言模型驱动的自动网络攻击系统 Autoattacker, 该系统通过总结器、计划器、导航器和经验管理器四个组件与大语言模型进行交互, 在设计好的基准攻击任务上进行实验并评估, 该系统在无需人工干预情况下的自动攻击方面展现了出色的能力. Muzsai 等[31] 提出一种基于大语言模型的自动化渗透测试智能体 HackSynth, 其由计划器和总结器两个模块组成, 并与大语言模型交互, 迭代地生成命令和处理反馈, 通过在其引入的两个标准化基准数据集上进行实验, 结果表明能够自主解决大量 CTF 挑战.

2　多智能体渗透测试框架

2.1　框架概述

如图 1 所示, 本文提出的 LangPentest 是一种多智能体协作的自动化渗透测试框架, 通过分工明确的智能体模块并结合外部经验知识库, 利用大语言模型对任务逻辑的理解和生成能力, 完成渗透测试的规划、执行和优化.

LangPentest 框架包括四个主要模块: 任务生成、任务执行、经验管理和任务调整模块. 任务生成模块负责将用户输入的任务目标解析为可执行的基本任务链; 任务执行模块生成具体的基本操作; 通过经验管理模块结合外部经验库优化基本操作; 任务调整模块通过接收环境反馈动态调整任务链以适应实际情况. 智能体间的协作形成了任务规划、操作执行、反馈调整的闭环, 确保任务能够适应复杂环境并有效执行.

2.2　任务生成模块

任务生成模块的核心功能是根据用户输入的渗

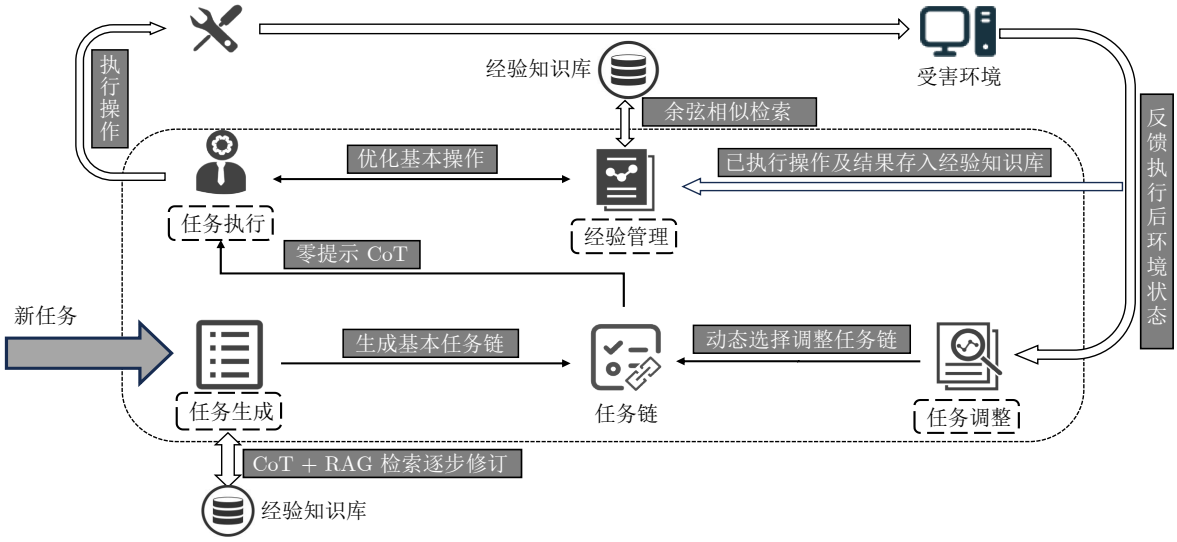


图 1 多智能体框架

Fig. 1 Multi-agent framework

透测试目标或任务描述, 生成一组细化的基本任务规划, 从而为后续的操作提供指引, 确保整个攻击流程具有清晰的逻辑结构和可操作性.

在框架初始阶段, 任务生成模块首先接收用户输入的渗透测试目标或任务描述, 通过零提示 CoT 生成初始任务链, 基于当前任务提示和推理结果, 检索渗透测试经验知识向量库, 结合前一步修订结果和新检索到的信息, 递归修订当前任务, 最终生成合理的渗透测试任务序列.

先前观察到, 由 LLM 直接生成的任务链中某些任务由于自身的幻觉问题, 针对目标系统是不可行的. 然而, 这种借助外部检索的信息进行逐步修订的形式可以提高任务链准确性并减少幻觉生成. 该模块将任务链中第一个基本任务按顺序提交给任务执行模块处理, 同时记录下剩余的未完成任务, 用于后续流程中反馈执行后的环境状态并触发任务调整模块的操作, 推动 LangPentest 框架完成渗透测试任务链中的基本任务. 图 2 展示了任务生成模块所使用的提示词模板.

2.3 任务执行模块

任务执行模块在整个框架中负责将任务生成模块生成的基本任务转化为具体操作步骤并执行, 同时通过与经验管理模块的协同工作进一步优化操作步骤, 以确保其在目标环境中的适用性和执行效果.

首先, 任务执行模块接收来自任务生成模块的基本任务, 并将其与设计好的提示词模板组合作为输入, 利用大语言模型生成能够具体化基本任务的初步基本操作. 然而, 在大语言模型直接生成的基

```
任务生成初步提示词模板
You are a professional cybersecurity researcher currently helping a company detect potential security risks in its systems. To complete your task, please create a step-by-step task list based on the following task description, ensuring that each task is a basic and actionable unit of work. Please think step by step.
Task objective: {Task objective or description}
The basic task list is output in the following format:
1. Basic Task 1
2. Basic Task 2
...
你是一位专业的网络安全研究员, 当前正在帮助一家企业检测其系统的潜在安全风险. 为了完成你的任务, 请根据以下任务描述, 创建分步任务列表, 确保每个任务都是基本、可操作的工作单元. 请一步一步地思考.
任务目标: {任务目标或描述}
基本任务链按以下格式输出:
1. 基本任务 1
2. 基本任务 2
...
```

图 2 任务生成提示词模板

Fig. 2 Task generation prompt template

本操作中发现错误的操作命令, 与目标操作系统命令规则不匹配, 这类错误的操作将会导致任务执行失败. 为解决该问题, 本文引入了经验管理模块来管理外部经验知识库. 任务执行模块将初步生成的基本操作提交至经验管理模块, 与外部经验知识库进行交互, 进而选择出最佳的基本操作后, 再由任务执行模块执行. 任务执行模块的这一流程设计实现了任务从规划阶段转化为具体执行阶段, 同时利用历史经验和大语言模型的推理能力对基本操作进行优化. 图 3 展示了任务执行模块所使用的提示词模板.

2.4　经验管理模块

经验管理模块的功能是管理外部经验知识库并基于 RAG 技术优化操作步骤. 该模块通过与任务执行模块协作, 从外部经验知识库中提取相应信息, 使基本操作具有实践经验的支持. 如图 4 所示, 任务执行模块通过 CoT 生成当前任务步骤, 并调用经验管理模块进行 RAG 检索与证据回填. 生成经验证据约束的最优操作 op*, 执行前实现经验优化, 降低模型幻觉风险. 其中, Ti 为基本操作, Qi 为当前基本操作对应的检索查询.

当经验管理模块接收到任务执行模块提供的初步基本操作后, 它会基于向量化检索机制计算初步基本操作与经验库中所存储内容的相似度. 为提高检索效率与精确度, 该模块采用了基于余弦相似度的检索算法, 从经验库中召回与初步操作步骤最相关的前三条经验记录. 这些经验包括任务的执行步骤和结果, 能够为当前基本操作的优化提供可靠的参考. 随后, 经验管理模块将召回的三条经验与初步基本操作一起拼接为提示词交由大语言模型, 大语言模型基于这些信息进行分析, 选择出最优的基本操作步骤, 并将其返回给任务执行模块作为最终基本操作. RAG 检索流程如图 5 所示.

在经验管理模块的经验检索过程中, 任务执行模块的初步基本操作被向量化为高维空间中的向量. 通过计算输入向量与经验向量之间的余弦相似度, 可以量化它们在语义空间中的相似程度. 对于向量 $A=[a_1,\ a_2,\ \cdots,\ a_n]$ 和 $B=[b_1,\ b_2,\ \cdots,\ b_n]$,它们的余弦相似度的计算公式如下:

$$\cos(A,\ B)=\frac{A\cdot B}{\parallel A\parallel\times\parallel B\parallel}=\frac{\sum\limits_{i=1}^{n}a_ib_i}{\sqrt{\sum\limits_{i=1}^{n}a_i^2}\sqrt{\sum\limits_{i=1}^{n}b_i^2}}$$

余弦相似度关注的是向量的方向, 而非长度, 因此特别适合衡量文本内容在语义层面的相似性, 而不受文本长度或频率的影响. 图 6 展示了经验管理模块所使用的提示词模板.

2.5　知识库构建

RAG 机制的性能在很大程度上取决于外部知识库的内容质量与组织结构. 若知识覆盖不足或语料分布不均, 可能导致检索召回的相关性下降, 进而影响模型在复杂任务中的推理稳定性. 为确保 LangPentest 在任务推理和操作生成阶段能够获得准确且高相关性的知识支撑, 本文通过多源整合与质量控制构建一个面向渗透测试领域的知识库. 该知识库的构建过程包括原始数据收集、数据清洗与分块、向量化及向量数据库存储, 具体如下:

为保证数据的代表性与质量, 在经验知识库的构建中整合了渗透测试内部报告、公开的安全社区以及从安全博客收集的经验分享. 总体上, 知识库共整理了 820 篇文档, 经清洗与分块后生成约 4 万条语义片段. 其中, 内部报告主要包含以往渗透测试项目中形成的漏洞分析与执行日志; 公开社区与博客数据则来源于常见安全平台 NVD[32]、OSV[33] 以及渗透测试技术博客中公开的漏洞复现案例与脚本分析内容.

针对收集的博客数据, 使用 BeautifulSoup 去除 HTML 标签、广告及导航栏等非正文内容, 仅保留主要文本信息, 并采用基于正则表达式与关键词

任务执行提示词模板
You are a professional cybersecurity researcher currently helping a company detect potential security risks in its systems. To complete your task, please generate basic operation instructions that can be executed on Kali-Linux-2024 based on the {Task ID} in the following basic task list.
Basic task list: {Task Chain}
Corresponding Task ID is required before basic operation instructions
你是一位专业的网络安全研究员, 当前正在帮助一家企业检测其系统的潜在安全风险. 为了完成你的任务, 请根据以下基本任务链中 {Task ID}, 生成可在 Kali-Linux-2024 上执行的基本操作指令.
基本任务链: {Task Chain}
基本操作指令以对应的 Task ID 开始

图 3　任务执行提示词模板

Fig. 3　Task execution prompt template

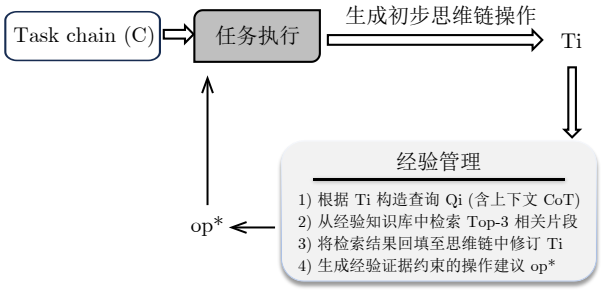


图 4 LangPentest 任务执行与经验优化流程

Fig. 4 Task execution and experience optimization process in LangPentest

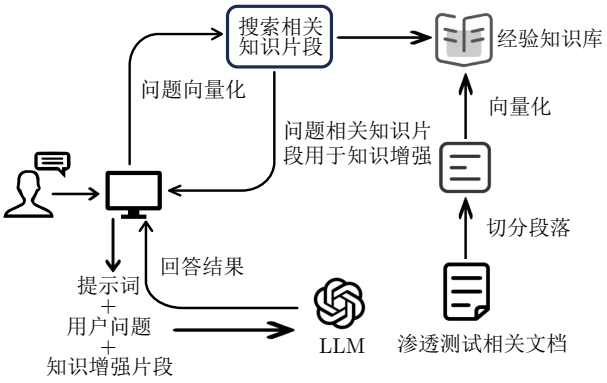


图 5 RAG 检索流程图

Fig. 5 RAG retrieval flow chart

过滤的规则, 对与渗透任务无关的语句 (如免责声明、更新日志) 进行去除, 确保数据唯一性和内容一致性. 为适应向量化与后续检索, 借助 LangChain 提供的文本分割工具, 将清洗后的正文按语义边界划分为 500 字符的文本块, 为保持语义连续性、减少语义断裂, 设置 20% 的片段重叠.

在向量化阶段, 采用 OpenAI Embeddings 模型将文本块转换为高维向量表示, 该模型在技术文档类文本中具有较好的语义区分度与稳定性. 向量化后的数据存储应用 Chroma 轻量级数据库, 其支持高并发检索与内存索引缓存, 便于与 LangChain 框架集成, 以支持高效相似度检索与经验知识调用.

2.6 任务调整模块

任务调整模块在框架中的作用是接收任务执行模块对执行后环境状态的反馈并对后续任务进行动态调整, 其功能是根据任务执行模块反馈的执行环境状态, 评估并决定是否修改任务链. 如图 7 所示, 任务调整模块根据执行结果与环境状态评估任务链是否偏离目标, 保留已执行部分并调整未执行部分, 生成新的任务链 C' 以维持渗透测试流程的正确性与连续性. 其中, OP 表示一次操作执行记录.

图 8 展示了任务调整模块所使用的提示词模板. 其在每轮操作步骤执行完成后接收任务执行模块反馈的环境状态, 包括执行结果和环境的变化. 将这些信息作为提示词交由大语言模型结合当前任务链进行评估. 若判断任务链需要调整, 则根据评估生成调整后的新任务链. 新任务链包含两部分: 已执行过的任务保留、未执行的任务则根据评估进行调整.

在多轮任务执行场景中, 环境状态会持续更新, 在每轮评估时持续累积全部历史命令及执行结果, 将使提示词变得冗长, 影响模型判断. 为保持提示词的紧凑性并避免上下文随轮次累积, 任务调整模块在构造提示词时仅使用当前轮次生成的环境状态摘要, 而不将历史日志反复加入提示, 且较早的中间任务信息已包含在任务链已执行部分中. 为保持多轮调整过程中的信息一致性, 框架将“已执行任务链”与“当前环境状态摘要”共同作为系统的显式记忆, 其中任务链记录历史规划进度, 状态摘要反映当前执行结果, 两者构成轻量化的 memory 结构, 使任务调整无需依赖完整历史上下文即可稳定运行. 通过上述方式, 任务调整过程能够在保证上下文长度稳定的同时, 准确反映当前环境状态摘要, 确保模型基于最新信息做出任务链调整决策.

在无需修改任务链的情况下, 任务调整模块直接确认现有任务链的适用性, 允许任务执行流程继续按照既定规划进行. 通过这一设计, 使该模块能够在每轮任务执行后变化的环境中确保任务链与目标一致, 避免因受害环境变化而导致的任务偏移.

经验管理提示词模板

You are a professional cybersecurity researcher currently helping a company detect potential security risks in its systems. To complete your task, please follow the following basic operation command {Execute Command}
Refer to the relevant search {RAG} in the experience knowledge base and select the best basic operation instruction.
你是一位专业的网络安全研究员, 当前正在帮助一家企业检测其系统的潜在安全风险. 为了完成你的任务, 请根据以下基本操作指令 {Execute Command}
参考经验知识库中相关检索 {RAG}, 选择一条最佳基本操作指令.

图 6 经验管理提示词模板

Fig. 6 Experience management prompt template

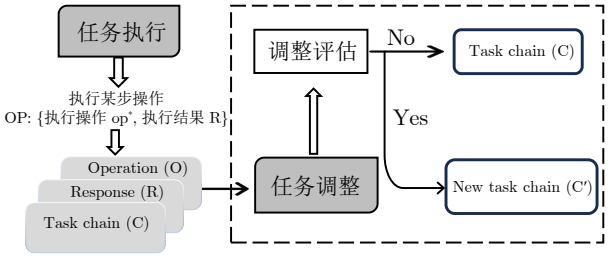


图 7　LangPentest 任务调整与任务链修正流程

Fig. 7　Task adjustment and task-chain revision process in LangPentest

3　实验设置

3.1　基准选择

基准的选择对实验效果具有重要影响, 特别是在评估任务执行成功率方面, 任务的难度起着关键作用. 为了充分验证任务执行的成功率, 需要在不同难度的任务环境中进行实验. 由于 Vulhub[34] 提供了大量基于 CVE 数据库构建的预先配置的易受攻击环境, 且环境的关联 CVE 编号在项目文档中会有所说明, 因此选择 Vulhub 作为基准测试的数据集是非常合适的, 且可以通过与 CVE 编号相关的指标评估每个环境的复杂性. 具体而言, 通用漏洞评分系统 (common vulnerability scoring system, CVSS)[35] 能够帮助判断漏洞利用的难度. Vulhub 在渗透测试实践中被广泛应用, 其数据集对实现实验目标具有重要的支持作用. 根据 CVSS 基本评分及向量特征, 将漏洞环境按复现难度划分为三类: 评分较高 (≥ 9.0) 的漏洞通常具备较低的可利用性和攻击门槛, 划为简单任务; 评分介于 8.0 ~ 8.9 的漏洞往往需要一定的前置条件, 划为中等任务; 而评分较低 (≤ 7.9) 的漏洞通常需用户交互或多步利用链, 划为偏难任务. 此划分结合了 Vulhub 数据集的分布特点, 更贴合实际漏洞复现难度层级. 以 SQL 注入任务类型为例, 其中选取的 CVE-2019-14234 和 CVE-2021-35042 的评分均为 9.8, 划分为简单任务类型. 本研究基于 Vulhub 平台选取共 100 个漏洞环境, 为兼顾“任务类型”与“复现难度”, 如表 1 所示, 列举了利用行为划分的任务类型.

为了评估框架在多任务环境中的任务规划能力、

任务调整提示词模板
You are a professional cybersecurity researcher currently helping a company detect potential security risks in its systems. In order to complete your task, please determine whether it is necessary to adjust the task chain based on the feedback information after performing the following operations. If so, please provide a new task chain. Please refer to the input task chain for the output format.
Original Basic Task Chain:
{Task Chain}
Feedback: Execute Command {Execute Command}, Execution Results {Execution Results}
你是一位专业的网络安全研究员, 当前正在帮助一家企业检测其系统的潜在安全风险. 为了完成你的任务, 请根据以下执行操作后的反馈信息, 判断是否需要调整任务链, 若调整, 请给出新任务链. 输出格式请参考以下输入的任务链.
原基本任务链:
{Task Chain}
反馈信息: 执行命令 {Execute Command}, 执行结果 {Execution Results}

图 8　任务调整提示词模板

Fig. 8　Task adjustment prompt template

表 1　单主机单任务列表

Table 1　Single host single-task list

任务	描述	难度
文件操作	文件操作 (如上传、写入、读取) 的验证或利用	简单
脚本执行	在目标主机上运行自定义脚本以实现特定攻击目标	中等
远程代码执行	在目标主机上执行未经授权的代码	偏难
权限提升	利用漏洞获取更高的用户访问权限	中等
信息泄露	提取系统中敏感信息, 如配置文件和日志	中等
身份验证绕过	利用漏洞绕过目标主机或应用的身份验证机制	中等
未授权访问	利用系统或服务的配置缺陷, 绕过认证机制, 获取未授权的访问权限	偏难
路径穿越	利用路径解析漏洞访问目标主机的敏感文件	中等
SQL 注入	向应用程序的 SQL 查询中注入恶意代码, 获取或篡改数据	简单
XML 实体注入	利用 XML 解析器处理实体的漏洞, 读取文件导致信息泄露	中等

任务规划调整机制以及应对复杂任务的性能, 本文选择 VulnHub[36] 作为单主机多任务实验的靶场环境数据集, 选取的主要依据是 VulnHub 的特点和实验需求之间的高度契合. 首先, VulnHub 提供了大量预构建的靶机场景, 设计了多任务的攻击流程, 包含信息收集、漏洞挖掘、权限提升等多种渗透测试任务, 能够真实地模拟多阶段渗透测试的完整流程, 为实验提供了一个复杂而全面的测试环境. 其次, VulnHub 的靶机通常针对真实世界的场景进行设计, 包含了多种网络服务、配置以及漏洞组合, 使其成为多任务执行能力验证的合适基准. 同时, 这些靶机在任务类型的描述上会标示出难易程度, 其丰富的靶机场景和文档支持能够确保实验的可重复性和可比性, 可为渗透测试框架在多任务性能评估中提供可靠的实验基础.

为了突破大语言模型内置安全性限制, 本文采用了角色扮演[37] 策略以实现大语言模型的越狱[38−39]. 角色扮演即通过为大语言模型指派一个具体的身份, 例如网络安全研究员, 引导模型在特定任务情境下生成原本被限制的内容. 这种方式不仅能够有效绕过大语言模型的安全防护机制, 还能使其在越狱场景中执行复杂任务时表现出更强的适配能力.

3.2 评估指标

为评估框架在不同任务类型和复杂度下的表现及框架的综合性能[40], 本文采用任务成功率及调用大语言模型 API 所产生的成本作为主要的评估指标. 任务成功率被定义为某个任务在多次尝试中成功完成的次数与总尝试次数的比值, 这一指标用于衡量框架在给定任务上的有效性. 该指标适用于单主机单任务和多任务场景, 能够跨越任务类型和复杂度的差异进行评估, 且由于成功率是通过多次实验取平均值来计算, 因此能有效降低单次实验的偶然成功或失败对整体结果的影响. 在实验设计中, 每个任务将被尝试执行 5 次, 记录成功执行的次数, 计算该任务成功率.

3.3 实验环境

实验环境设置在 Windows10 操作系统上, 通过 VMware Workstation 平台搭建虚拟化网络环境. 攻击虚拟机选用最新版本的 Kali Linux (Kali-Linux-2024.1), 此系统预装了多种渗透测试工具, 包括但不限于 Metasploit、Arp-Scan 及多种安全工具, 为攻击任务提供了全面支持. 目标环境则部署在另一台虚拟机内的 Docker 容器中, 该虚拟机的硬件资源分配为 2 个 CPU 核心、8 GB 内存以及 50 GB 存储空间. 网络连接模式主要采用 NAT, 确保虚拟机能够访问互联网, 以便下载实验依赖或测试网络通信功能; 在执行实际渗透测试时, 切换到自定义虚拟网络的 Host-only 模式, 从而限制测试仅在本地网络内进行, 提升测试过程的安全性.

为增强实验的灵活性与稳定性, Docker 容器中的目标环境可以按需更换或重置, 模拟多种真实场景和漏洞类型. 此外, 所有虚拟机的快照功能被启用, 用于快速回滚到初始状态, 确保在实验失败或出现意外问题时能够快速恢复.

3.4 模型选择

在选择 LLM 时, 本文选用了 OpenAI 的 GPT-3.5-Turbo 和 GPT-4o 模型以及 Anthropic 的 Claude 3.5 Sonnet 模型[41], 均通过其官方提供的 API 进行访问. 这些模型凭借其强大的自然语言处理能力和广泛的应用基础, 在相关研究领域中得到了广泛应用. 此外, 为增强模型对比的代表性并引入国内开源体系, 本文选用了 Qwen 2.5-14B-Instruct 模型[42], 该模型由阿里巴巴开源发布, 具备较强的中文理解、代码生成和任务指令跟随能力, 可在本地或开源框架中部署运行. 为了确保实验结果的可信度和可重复性, 在实验过程中, 严格通过官方渠道访问这些模型, 并对其进行统一配置和调用.

4 实验结果评估

4.1 整体框架有效性

单主机单任务基于 CVE 编号对应的 CVSS 评分, 将任务划分为简单、中等和偏难三类. 单主机多任务则根据官方提供的任务描述中的难度等级, 同样划分为简单、中等和偏难任务. 以下实验结果为每类任务中成功率的平均值.

如图 9 所示, 在单主机单任务场景中, 四种模型整体均能较好完成简单任务, 但在中高难度任务上差异明显. Claude 3.5 Sonnet 取得最高平均成功率, GPT-4o 次之, Qwen 2.5-14B-Instruct 略优于 GPT-3.5-Turbo. 其中, GPT-4o 和 Claude 3.5 Sonnet 在任务规划与命令生成阶段能更好地保持执行连贯性, Qwen 2.5-14B-Instruct 在任务理解与命令生成的细化程度上明显优于 GPT-3.5-Turbo, 而 GPT-3.5-Turbo 在部分任务中生成的命令参数不一致或工具调用不准确, 导致执行失败. 这表明模型的语言理解深度与工具调用准确性对单任务成功率具有显著影响.

图 10 展示了单主机多任务场景的结果. 可以观察到, 多任务下的整体成功率普遍低于单任务场

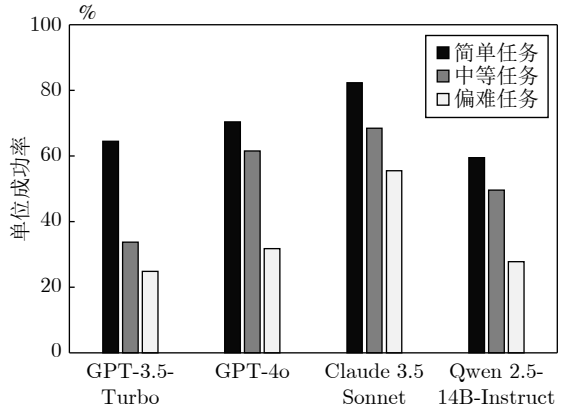


图 9　单主机单任务成功率

Fig. 9　Single host single-task success rate

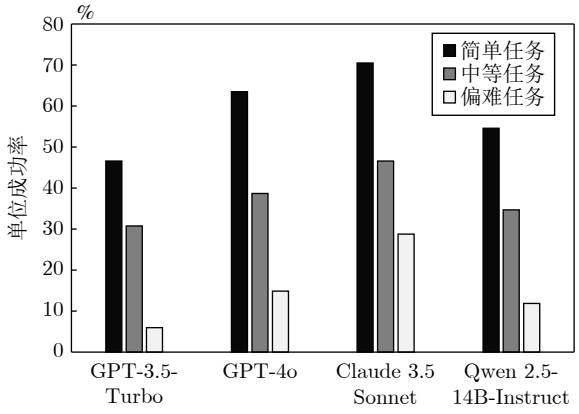


图 10　单主机多任务成功率

Fig. 10　Single host multi-task success rate

景, 说明在复杂上下文与任务状态管理中, 模型的持续推理与信息保持能力成为主要瓶颈. Claude 3.5 Sonnet 仍保持最高成功率, GPT-4o 次之, Qwen 2.5-14B-Instruct 与 GPT-3.5-Turbo 表现接近, 且在任务衔接与状态追踪方面略显不稳定. 相比之下, Claude 3.5 Sonnet 在上下文记忆与跨任务协调上表现更好, 而中等规模模型在多轮推理和结果关联上仍有一定差距. 整体结果表明, 随着任务复杂度和并行度提升, 模型的上下文保持与多步骤推理能力成为影响执行成功率的关键因素.

为评估 LangPentest 相较于其他自动化工具及自动化渗透测试框架的性能, 将 LangPentest 与通用型 AutoGPT[43] 以及开源实现的 PentestGPT 两个基线进行对比, 并将其在本文所部署的 Vulhub Docker 靶机上进行了适配与复现运行. 实验选择 Vulhub 单主机单任务基准中的五类典型任务, 分别对应的靶机与漏洞如下: 文件上传类: drupal/CVE-2019-6341; 权限提升类: polkit/CVE-2017-5645; 远程代码执行类 (Apache Log4j2): log4j/CVE-2021-44228; XML 实体注入类: solr/CVE-2017-12629; 身份验证类: mysql/CVE-2012-2122. 三框架均使用相同底层模型 GPT-4o, 并统一 temperature、max_tokens 参数.

表 2 展示了 LangPentest、PentestGPT 与 AutoGPT 在五类典型 Vulhub 靶机成功/总次数对比. 总体而言, LangPentest 在所选任务集上表现出更高的成功率. 在多数任务中, LangPentest 的成功次数高于 PentestGPT, 且明显优于通用型代理 AutoGPT. 这一结果表明, 多智能体协作与基于经验的优化机制在提高任务可执行性方面具有实际效用. 以权限提升攻击任务为例, LangPentest 在 5 次实验中成功 3 次, 其中第一轮尝试通过 sudo-l 提权失败后, 任务调整模块基于失败日志改用向/etc/cron.d 注入策略并最终成功. 而 AutoGPT 在首次提权失败后仍持续调用已被修复的 pkexec 本地提权漏洞, 其单一决策链路逻辑在复杂渗透场景中适应性较低. PentestGPT 会尝试备用利用, 搜索 SUID 二进制, 但在本次靶机上检索到的证据不足, 难以在有限尝试中生成可靠的低层次利用代码, 从而未能稳定成功.

4.2　成本

本实验对比 OpenAI 的 GPT-3.5-Turbo、GPT-4o 以及 Anthropic 的 Claude 3.5 Sonnet 三种模型在完成任务时所产生的 API 成本. 记录了各模型在

表 2　不同框架在典型单任务类型下的对比

Table 2　Comparison of different frameworks on typical single-task types

单任务类型	模型	成功/总次数
文件上传	AutoGPT	2/5
	PentestGPT	3/5
	LangPentest	4/5
权限提升	AutoGPT	0/5
	PentestGPT	2/5
	LangPentest	3/5
XML 实体注入	AutoGPT	0/5
	PentestGPT	1/5
	LangPentest	1/5
身份验证	AutoGPT	1/5
	PentestGPT	3/5
	LangPentest	4/5
Apache Log4j2	AutoGPT	0/5
	PentestGPT	1/5
	LangPentest	2/5

任务规划、调整、经验管理以及执行等环节中消耗的总成本. 为确保对比的合理性, 实验选择在完成度高的任务上进行统计.

表 3 结果表明, GPT-3.5-Turbo 在消耗上具有显著优势, 其较低的 token 单价使得在任务中的成本最低, 但其在多任务中表现略逊. Claude 3.5 Sonnet 尽管成本较高, 但在任务的成功率和可执行程度上表现最为优异, 尤其在偏难的多任务场景下展现了更强的泛化能力. 相比之下, GPT-4o 的成本消耗介于两者之间, 其任务完成效率和成本在中等难度任务中较为均衡, 但在任务成功率上与 Claude 3.5 Sonnet 仍存在一定差距.

表 3 LangPentest 在不同任务下的成本
Table 3 Cost of LangPentest under different tasks

任务	GPT-3.5-Turbo (USD)	GPT-4o (USD)	Claude 3.5 Sonnet (USD)
任意文件写入	0.54	0.62	0.68
特权升级	0.68	0.68	0.73
脚本执行	0.71	0.78	0.67
本地权限提升	0.45	0.41	0.54
SQL 注入	0.86	0.89	0.87
目录遍历	1.71	1.82	1.92
XML 实体注入	3.24	3.42	3.61
文件上传	1.43	1.65	1.87

4.3 消融实验

为全面评估任务调整模块和经验管理模块对于整体框架的具体影响, 设计了一系列消融实验, 这些实验分别禁用任务调整模块和经验管理模块, 并对比“启用任务调整模块”与“禁用任务调整模块”的情况, 以及对比“启用经验管理模块”与“禁用经验管理模块”的任务完成情况.

在分析任务调整模块对框架有效性的增强作用时, 重点观察其在单主机多任务的场景中是否能够有效缓解任务偏移. 如表 4 所示, 相较于禁用任务调整模块的情形, 启用该模块后, 任务的成功率显著提升. 同时, 在多任务场景下观察到, 未启用任务调整模块时, 对环境状态分析错误增多且任务逐渐偏离初始目标, 例如未能根据环境反馈修正任务路径, 导致执行中出现了过多冗余或无效步骤, 最终影响任务完成质量. 这一结果验证了任务调整模块在提高框架对任务目标一致性及成功率方面所起到的作用.

表 4 多任务成功率及交互轮次
Table 4 Multi-task success rate and interaction rounds

经验管理	任务调整	交互轮次	成功率 (%)
禁用	禁用	29	42
启用	禁用	33	65
禁用	启用	27	48
启用	启用	31	78

在分析经验管理模块对框架的作用时, 主要关注其在单主机单任务场景中能否提高任务的成功率, 以及是否能够有效地避免大语言模型生成不准确的操作步骤或指令. 如表 5 所示, 启用经验管理模块时, 任务的成功率明显高于禁用模块的情况. 同时, 在实验过程中发现, 当禁用经验管理模块时, 生成的基本操作指令的失效现象偏多. 这表明, 经验管理模块在提升任务成功率方面效果显著.

表 5 单任务成功率及交互轮次
Table 5 Single-task success rate and interaction rounds

经验管理	任务调整	交互轮次	成功率 (%)
禁用	禁用	13	64
启用	禁用	12	73
禁用	启用	13	66
启用	启用	9	86

4.4 失败分析

为更系统地理解不同框架在失败场景下的表现, 本节对未成功试次进行统计与归纳, 结果如表 6 所示, 将失败原因划分为以下四类. 1) 证据绑定不足. 虽能检索到相关案例, 但证据未在提示中被强约束引用或与当前环境状态不一致, 导致生成的操作不可执行. 例如将针对 Drupal 的上传路径直接套用到不同实现的文件写入场景, 从而导致多次失败. 2) 环境前置条件/依赖缺失. 指令正确但因端口、依赖或权限前置条件未满足而无效, 典型情形为 Log4j2 利用过程中外连端口未开放, payload 被触发却无回显. 3) 长上下文/记忆衰减. 多轮推理后关键线索在摘要中丢失, 出现与既有结论冲突的操作, 典型情形如在权限链探索中, 初期已识别可通过 crontab 实现的提权线索, 但后续摘要未固化该证据. 4) 任务调整触发保守. 当连续发生“弱失败”, 如 HTTP 200 但无敏感回显时, 调整任务链

表 6 失败统计表
Table 6 Failure statistics table

失败类型	典型现象	数量
证据绑定不足	召回片段相关但未严格引用	7
环境前置条件/依赖缺失	权限/端口/依赖报错	10
长上下文/记忆衰减	遗忘前置发现	8
任务调整触发保守	连续弱失败未改路	5

不及时或改到等价低效路径.

5 结束语

本文提出一种基于大语言模型的多智能体自动渗透测试框架, 并在选择的基准实验集上进行了实验与评估. 结果表明, 框架在单主机的单任务及多任务的简单场景下均表现出较高的成功率和较少的任务偏移现象. 任务调整模块和经验管理模块显著提高了任务完成的有效性与准确性. 然而, 尽管这些大语言模型提供了强大的计算与推理能力, 但在成本统计中表明框架运行的经济成本依然较高.

未来的工作将集中在以下方向: 首先, 进一步优化框架的交互效率, 降低与大语言模型交互的成本; 其次, 拓展并丰富 RAG 经验知识库的数据集, 增加行业的新知识; 最后, 优化多智能体协作机制, 探索其在多主机多任务场景及现实场景下的有效性.

参考文献

1 El Kafhali S, El Mir I, Hanini M. Security threats, defense mechanisms, challenges, and future directions in cloud computing. *Archives of Computational Methods in Engineering*, 2022, **29**(1): 223−246

2 Pfleeger C P, Pfleeger S L, Theofanos M F. A methodology for penetration testing. *Computers & Security*, 1989, **8**(7): 613−620

3 Denis M, Zena C, Hayajneh T. Penetration testing: Concepts, attack methods, and defense strategies. In: Proceedings of the IEEE Long Island Systems, Applications and Technology Conference (LISAT). Farmingdale, USA: IEEE, 2016. 1−6

4 Stefinko Y, Piskozub A, Banakh R. Manual and automated penetration testing. Benefits and drawbacks. Modern tendency. In: Proceedings of the 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET). Lviv, Ukraine: IEEE, 2016. 488−491

5 Kojima T, Gu S S, Reid M, Matsuo Y, Iwasawa Y. Large language models are zero-shot reasoners. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. New Orleans, USA: Curran Associates Inc., 2022. Article No. 1613

6 Talebirad Y, Nadiri A. Multi-agent collaboration: Harnessing the power of intelligent LLM agents. arXiv preprint arXiv: 2306.03314, 2023.

7 Wu Q Y, Bansal G, Zhang J Y, Wu Y R, Li B B, Zhu E K, et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In: Proceedings of the 1st Conference on Language Modeling. Philadelphia, USA: 2024.

8 He J D, Treude C, Lo D. LLM-based multi-agent systems for software engineering: Literature review, vision, and the road ahead. *ACM Transactions on Software Engineering and Methodology*, 2025, **34**(5): Article No. 124

9 Tran K T, Dao D, Nguyen M D, Pham Q V, O′Sullivan B, Nguyen H D. Multi-agent collaboration mechanisms: A survey of LLMs. arXiv preprint arXiv: 2501.06322, 2025.

10 Kong H, Hu D, Ge J G, Li L X, Li T, Wu B Z. VulnBot: Autonomous penetration testing for a multi-agent collaborative framework. arXiv preprint arXiv: 2501.13411, 2025.

11 Shen X M, Wang L Z, Li Z Y, Chen Y, Zhao W C, Sun D W, et al. PentestAgent: Incorporating LLM agents to automated penetration testing. In: Proceedings of the 20th ACM Asia Conference on Computer and Communications Security. Hanoi, Vietnam: ACM, 2025. 375−391

12 Happe A, Cito J. Getting pwn′d by AI: Penetration testing with large language models. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. San Francisco, USA: ACM, 2023. 2082−2086

13 Ji Z W, Lee N, Frieske R, Yu T Z, Su D, Xu Y, et al. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 2023, **55**(12): Article No. 248

14 Dziri N, Milton S, Yu M, Zaiane O, Reddy S. On the origin of hallucinations in conversational models: Is it the datasets or the models? In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Seattle, USA: ACL, 2022. 5271−5285

15 CVE Program. CVE.TM program mission [Online], available: https://www.cve.org, December 25, 2025

16 Wang Z H, Liu A J, Lin H W, Li J Q, Ma X J, Liang Y T. RAT: Retrieval augmented thoughts elicit context-aware reasoning in long-horizon generation. arXiv preprint arXiv: 2403.05313, 2024.

17 Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. Vancouver, Canada: Curran Associates Inc., 2020. Article No. 793

18 Sarker K U, Yunus F, Deraman A. Penetration taxonomy: A systematic review on the penetration process, framework, standards, tools, and scoring methods. *Sustainability*, 2023, **15**(13): Article No. 10471

19 Nmap.org. Get Nmap 7.99 here [Online], available: https://nmap.org, December 25, 2025

20 Greenbone Networks. OPENVAS by Greenbone [Online], available: https://www.openvas.org, December 25, 2025

21 Rapid7. Metasploit [Online], available: https://www.metasploit.com, December 25, 2025

22 Hu Z G, Beuran R, Tan Y S. Automated penetration testing using deep reinforcement learning. In: Proceedings of the IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). Genoa, Italy: IEEE, 2020. 2−10

23 Zhang K Q, Yang Z R, Başar T. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*. Cham: Springer, 2021. 321−384

24 Takaesu I. Deep Exploit [Online], available: https://github.com/13o-bbr-bbq/machine_learning_security/blob/master/DeepExploit/README.md, December 25, 2025

25 Moreno A C, Hernandez-Suarez A, Sanchez-Perez G, Toscano-Medina L K, Perez-Meana H, Portillo-Portillo J, et al. Analysis of autonomous penetration testing through reinforcement learning and recommender systems. *Sensors*, 2025, **25**(1): Article No. 211

26 Gao Wen-Long, Zhou Tian-Yang, Zhao Zi-Heng, Zhu Jun-Hu. Network attack path planning method based on deep reinforcement learning. *Journal of Cyber Security*, 2022, **7**(5): 65−78
(高文龙, 周天阳, 赵子恒, 朱俊虎. 基于深度强化学习的网络攻击路径规划方法. 信息安全学报, 2022, **7**(5): 65−78)

27 OpenAI. GPT-3.5 [Online], available: https://platform.openai.com/docs/models, December 25, 2025

28 OpenAI. GPT-4 [Online], available: https://platform.openai.com/

docs/models, December 25, 2025

29 Deng G L, Liu Y, Mayoral-Vilches V, Liu P, Li Y K, Xu Y, et al. PentestGPT: Evaluating and harnessing large language models for automated penetration testing. In: Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24). Philadelphia, USA: USENIX Association, 2024. Article No. 48

30 Xu J C, Stokes J W, McDonald G, Bai X S, Marshall D, Wang S Y, et al. AutoAttacker: A large language model guided system to implement automatic cyber-attacks. arXiv preprint arXiv: 2403.01038, 2024.

31 Muzsai L, Imolai D, Lukács A. HackSynth: LLM agent and evaluation framework for autonomous penetration testing. arXiv preprint arXiv: 2412.01778, 2024.

32 NIST. NVD: National vulnerability database [Online], available: https://nvd.nist.gov, December 25, 2025

33 OSV. A distributed vulnerability database for Open Source [Online], available: https://osv.dev, December 25, 2025

34 Vulhub. Vulhub [Online], available: https://vulhub.org, December 25, 2025

35 National Institute of Standards and Technology (NIST). Common vulnerability scoring system SIG [Online], available: https://www.first.org/cvss/, December 25, 2025

36 VulnHub. Virtual machines [Online], available: https://www.vulnhub.com, December 25, 2025

37 Johnson Z D. Generation, Detection, and Evaluation of Role-Play Based Jailbreak Attacks in Large Language Models [Master thesis], Massachusetts Institute of Technology, USA, 2024.

38 Chu J J, Liu Y G, Yang Z Q, Shen X Y, Backes M, Zhang Y. JailbreakRadar: Comprehensive assessment of jailbreak attacks against LLMs. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Vienna, Austria: ACL, 2024. 21538−21566

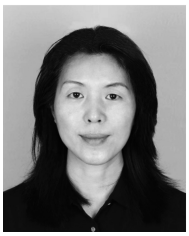
39 Yu Z Y, Liu X G, Liang S N, Cameron Z, Xiao C W, Zhang N. Don′t listen to me: Understanding and exploring jailbreak prompts of large language models. In: Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24). Philadelphia, USA: USENIX Association, 2024. Article No. 262

40 Yehudai A, Eden L, Li A L, Uziel G, Zhao Y L, Bar-Haim R, et al. Survey on evaluation of LLM-based agents. arXiv preprint arXiv: 2503.16416, 2025.

41 Anthropic. Claude 3.5 Sonnet [Online], available: https://www.anthropic.com/news/claude-3-5-sonnet, December 25, 2025

42 Qwen. Qwen2.5: A party of foundation models! [Online], available: https://qwenlm.github.io/blog/qwen2.5, December 25, 2025

43 Significant Gravitas. AutoGPT: An autonomous GPT-4 powered AI agent [Online], available: https://github.com/Significant-Gravitas/AutoGPT, December 25, 2025



江 颉 浙江工业大学计算机科学与技术学院教授. 主要研究方向为网络安全, 人工智能.
E-mail: jj@zjut.edu.cn
(**JIANG Jie** Professor at the College of Computer Science and Technology, Zhejiang University of Technology. Her research interests include cybersecurity and artificial intelligence.)

王 豪 浙江工业大学计算机科学与技术学院硕士研究生. 主要研究方向为网络安全, 人工智能.
E-mail: wanghao10246@163.com
(**WANG Hao** Master student at the College of Computer Science and Technology, Zhejiang University of Technology. His research interests include cybersecurity and artificial intelligence.)

李明达 浙江工业大学计算机科学与技术学院博士研究生. 主要研究方向为网络安全, 自动化攻击.
E-mail: zjutlmd@zjut.edu.cn
(**LI Ming-Da** Ph.D. candidate at the College of Computer Science and Technology, Zhejiang University of Technology. His research interests include cybersecurity and automated attack.)

朱添田 浙江工业大学计算机科学与技术学院副教授. 主要研究方向为网络安全, 人工智能. 本文通信作者.
E-mail: ttzhu@zjut.edu.cn
(**ZHU Tian-Tian** Associate professor at the College of Computer Science and Technology, Zhejiang University of Technology. His research interests include cybersecurity and artificial intelligence. Corresponding author of this paper.)