研究与开发

基于大语言模型和RAG的自动化渗透测试框架研究

江颉[1]，蔡辰旭[1]，李明达[1]，朱添田[1,2]
（1. 浙江工业大学计算机科学与技术学院，浙江 杭州 310023；
2. 浙江工业大学台州研究院，浙江 台州 318001）

摘　要：随着网络威胁的日益严峻，自动化渗透测试逐渐成为网络安全领域的研究热点。现有研究已初步探索了基于大语言模型实现自动化渗透测试的可行性，但在流程连续性和生成相关性方面仍有不足。对此，提出了一种基于多智能体协同的自动化渗透测试框架Pentest-Chain，通过分工协作的多个智能体来完成渗透测试的各个流程任务。为解决生成相关性问题，引入检索增强生成（retrieval-augmented generation，RAG）技术，利用外部知识库和内部经验库来提升智能体生成结果的准确性和可靠性。实验结果表明，相比单一智能体，多智能体框架Pentest-Chain的任务执行成功率整体提升了17.0%。进一步的消融实验表明，在多智能体框架中引入RAG模块对任务执行成功率的提升起到了关键作用，且显著优化了任务执行过程中的生成相关性和准确性。

关键词：大语言模型；自动化渗透测试；多智能体系统；RAG；网络安全
中图分类号：TP393
文献标志码：A
doi: 10.11959/j.issn.1000−0801.2025159

Research on an automated penetration testing framework based on LLM and RAG

JIANG Jie[1], CAI Chenxu[1], LI Mingda[1], ZHU Tiantian[1,2]
1. School of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China
2. Taizhou Institute, Zhejiang University of Technology, Hangzhou 318001, China

Abstract: With the increasing severity of network threats, automated penetration testing has become a research focus in the field of cybersecurity. Existing studies had preliminarily explored the feasibility of leveraging large language model(LLM) for automated penetration testing but still face challenges in process continuity and generation relevance. To address these issues, a multi-agent collaborative automated penetration testing framework named Pentest-

收稿日期：2025−01−06；**修回日期：**2025−06−23
通信作者：朱添田，ttzhu@zjut.edu.cn
基金项目：国家自然科学基金青年项目（No.62002324）；国家自然科学基金重点项目（No.U22B2028）；浙江省属高校基本科研业务费专项资金资助项目（No.RF-A2023009）
Foundation Items: The National Natural Science Foundation of China Youth Program (No.62002324), The National Natural Science Foundation of China Key Program(No.U22B2028), The Fundamental Research Funds for the Provincial Universities of Zhejiang (No.RF-A2023009)

Chain was proposed, where specialized agents worked cooperatively to complete different phases of penetration testing tasks. To enhance generation relevance, retrieval-augmented generation (RAG) technology was introduced, leveraging both external knowledge bases and internal experience repositories to improve the accuracy and reliability of the agents' outputs. Experimental results demonstrated that the Pentest-Chain framework achieved a 17.0% overall improvement in task success rate compared to single-agent approaches. Further ablation studies confirmed that the integration of the RAG module played a critical role in boosting task success rates while significantly optimizing generation relevance and accuracy during task execution.

Key words: LLM, automated penetration testing, multi-agent system, RAG, cybersecurity

0 引言

渗透测试是一种主动式的网络安全评估方法，长期以来被用于识别潜在安全隐患并为修复提供依据。随着企业数字化转型的加速，渗透测试的应用场景日益广泛。在企业安全评估、物联网（Internet of things，IoT）安全测试、红蓝对抗演练以及防范高级持续性威胁（advanced persistent threat，APT）等领域中，渗透测试都发挥了重要作用。

随着网络安全问题的日益严峻，攻击技术的复杂性和多样性快速增长，自动化渗透测试逐渐成为网络安全领域的重要研究方向之一[1]。传统的人工渗透测试方法主要依赖专家使用专业工具手动执行漏洞发现、攻击路径规划和利用，其效率低、成本高，并且测试效果受限于人员经验，难以在大规模网络环境中推广应用。为提升渗透测试的通用程度，研究人员开发了一系列基于自动化脚本的测试工具，如Metasploit自动化模块、Nmap NSE等。这些工具能够快速执行常见漏洞扫描和利用，提高测试效率。然而，这类方法仍主要依赖预定义规则，缺乏智能化决策能力，难以灵活适应新型漏洞或复杂网络环境。

近年来，人工智能技术的发展为提升渗透测试的效率和效果提供了新的契机。其中，强化学习（reinforcement learning，RL）因其在决策选择和策略生成中的出色表现，成为自动化渗透测试中的重要研究手段[2]。利用RL实现对网络攻击路径的规划和优化[3]，显著提高了渗透测试的自动化水平。然而，这些方法在面对复杂的网络环境时，仍存在决策效率低和攻击策略多样性不足的问题。

同时，大语言模型（large language model，LLM）在自然语言处理及相关领域的广泛应用，为自动化渗透测试提供了新的研究方向。LLM拥有强大的自然语言生成和推理能力，能够理解复杂的上下文并生成符合需求的文本，这一特性使其在网络安全中的应用潜力巨大[4-5]。例如，Deng等[6]提出了“PentestGPT”框架，初步探索了将大模型用于渗透测试的一些任务。

然而，现有的一些研究表明，大模型在处理复杂的渗透测试任务时仍存在显著局限，且在实际应用中仍面临诸多挑战。

（1）通用模型无法有效解决垂直领域的专业问题。在特定领域（如网络安全领域），通用模型可能无法正确理解大量的专业术语，导致生成结果不准确，甚至可能引导错误的操作流程。目前对通用预训练模型进行微调是提高渗透测试在专业领域表现的常见方法，但预训练方法存在较高的经济和时间成本。

（2）单一智能体在处理复杂渗透测试任务时存在明显的能力瓶颈。渗透测试任务通常涉及多个环节，如环境信息探测、漏洞扫描、权限提升和报告生成，单一智能体很难同时兼顾所有任务，且在同时处理任务分解、规划和执行后更易因认知负载过高出现失误和偏差，从而产生规划漂移

现象[7]和幻觉现象[8]。

（3）长流程任务下模型存在的上下文限制问题。当前大模型受限于上下文窗口的规模，难以记住渗透测试任务中生成的大量中间信息。面对复杂的长流程任务（如多步骤权限提升或多目标渗透攻击），这一局限严重影响了模型的跨步骤推理能力。

针对当前自动化渗透测试方法存在的不足，本文提出了一种多智能体协作的自动化渗透测试框架，并在此基础上引入检索增强生成（retrieval-augmented generation，RAG）模块[9]，利用外部知识库和内部经验库提升生成结果的准确性及任务完成效率。

本文的研究目标是设计并实现一个高效、精准且可扩展的自动化渗透测试系统。该系统能够应对复杂场景中的多步骤任务，显著提升渗透测试的成功率，并为网络安全领域中基于LLM的任务生成和优化提供参考。本文的主要贡献包括以下几个方面。

（1）提出了一种基于多智能体系统（multi-agent system，MAS）的自动化渗透测试框架Pentest-Chain。各智能体分工协作，分别负责任务分析、策略生成和执行等流程。通过智能体之间的协同工作，Pentest-Chain框架有效改善了单一模型难以完成复杂任务的局限性[10]。

（2）提供了一种泛用的渗透测试领域知识数据库模块的构建流程。本文创新地在渗透测试框架中引入RAG模块，在知识检索、复杂决策等场景加入外部知识进行优化。通过整合领域知识库（如ATT&CK矩阵[11]、CVE漏洞库[12]），模型能够在目标任务到达系统后动态检索相关信息，从而增强生成的分析结果的上下文关联性和准确性[9]。该流程在渗透测试相关任务中具有广泛的适用性。

（3）设计了一种支持长流程任务的内部经验管理器。针对长流程任务下模型的上下文限制，本文设计了内部经验管理模块来存储在任务过程中得到的经验[13]，以积累知识和减少重复性错误。并且本文设计的内部经验管理模块设置了跨任务类型经验联想，使原有的任务能够吸取其他相关上下游任务的经验，以完成更复杂的任务。

1　相关工作

渗透测试主要有明确渗透测试目标、信息收集、假设目标缺陷、确认目标缺陷、扩展目标缺陷、消除目标缺陷6个阶段[14]。自动化渗透测试可分为基于规则的自动渗透测试和基于模型的自动渗透测试[15]，本节将着重介绍结合人工智能技术的基于模型的自动化渗透测试。

1.1　基于RL的自动化渗透测试

RL已经被广泛应用于自动化渗透测试中[16]，相关研究的核心在于借助智能体[17]于渗透测试环境中开展最优策略的学习，通过系统性地探索与利用的迭代操作，逐步构建并掌握攻击路径的规划能力。许多学者将渗透过程建模为马尔可夫决策过程（Markov decision process，MDP）[18]，例如Zennaro等[19]提出了一个基于RL的框架，利用模拟网络环境进行策略优化。以上基于RL的方法有明确的模型优化目标，能够持续提升性能。然而，RL方法存在的一个主要缺陷是训练过程中需要大量的环境交互，导致数据需求高且训练成本昂贵，并且在复杂环境的高维状态空间中，容易陷入局部最优，进而限制整体成功率。相较于传统RL方法，LLM凭借大规模通用语料预训练所学得的通用知识与推理能力，可在无监督或低监督的条件下快速适应渗透测试任务。

1.2　垂直领域单一大模型的自动化渗透测试应用

这类研究直接利用通用预训练大语言模型（如ChatGPT[20]）或经过微调后的本地垂类模型[21]完成渗透测试任务。LLM的零样本/少样本

迁移能力能基于“任务目标+上下文”生成合理策略，减少对环境反复试错学习的需求。同时，LLM能结合外部知识库进行实时增强，使模型不再完全依赖训练阶段的数据覆盖，从而降低“冷启动”所需的渗透样本。研究者主要通过提示工程引导模型生成，包括端口扫描、漏洞利用等基础渗透步骤的指导。

基于通用大模型或垂类模型渗透测试的方法在自然语言的理解上有显著优势，它能够从用户输入的少量指令中构建较为完整的攻击流程。同时，经过垂直领域微调的模型在安全专业知识（如漏洞描述、攻击策略等）上具有更强的覆盖能力，能生成更为准确和实用的建议。但该方法也有其局限性，具体表现在以下方面。（1）无法解决长任务链问题。单一大模型的上下文限制会导致长任务过程中出现记忆缺失问题。（2）领域知识不足。由于目前的模型缺乏网络安全专业知识的公开数据集，垂类模型的生成结果往往缺乏深度。（3）上下文记忆能力有限。模型无法在复杂、多步骤的任务中保持上下文一致性。因此，单一大模型的直接应用仅能胜任基础的辅助任务[22]，在更复杂的渗透测试场景下难以发挥作用。

1.3 多智能体协同框架的自动化渗透测试

针对单一模型应用的不足，一些研究者采用模块化或流程化的设计框架，将大模型嵌入更复杂的系统中，以提高任务完成度。

多智能体系统是一种分布式计算模型，可以实现复杂任务的高效分解与并行执行。在自动化渗透测试中，基于大模型的MAS能够将渗透过程划分为不同阶段（如侦察、漏洞利用、后渗透），每个子智能体专注于特定子环节，从而整体提升任务成功率。例如，首次尝试此方式的PentestGPT在模拟环境中实现了智能体间的任务交接，使用任务树实现多阶段任务，但其在复杂任务场景仍受限，需人类干预解决[6]。PentestAgent[23]和AutoAttacker[24]是最近提出的基于大模型的多智能体自动化渗透测试系统，PentestAgent通过分解任务为信息收集、漏洞分析和漏洞利用协同完成任务，但分布式的决策过程导致智能体间协调成本较高。AutoAttacker则追求更广泛的攻击目标，它可以根据不同类型实时调整策略后攻击，但在前期分析层面缺少了针对特定问题的更深入分析。更为广泛应用多智能体思想的AutoGPT[25]通过分步骤任务分解和自反馈机制实现流程化的任务执行。然而，多智能体系统也面临许多问题，例如智能体间信息传递和协同调度需要合理的设计，否则会存在子任务间上下文切换不畅和信息延迟的问题；智能体间切换与通信等交互会引入额外的开销，增加资源消耗。

1.4 RAG技术

为了综合垂类领域单一大模型和多智能体协调框架的优点，本文引入了以RAG技术为基础的外部知识增强模块。类似垂类领域大模型，RAG能够提高生成内容的准确性和相关性。作为一种增强模块，RAG也能够很好地集成到多智能体协同框架中。

垂类领域大模型在训练完成后难以更新最新知识，相比之下，RAG能够在渗透测试过程中实时检索最新的安全情报、漏洞信息和攻击策略，确保测试策略始终保持前沿性，避免因知识老化而降低测试效果。此外，RAG的可扩展性使其不仅限于文本知识的检索，在未来还可以进一步集成多模态数据（如图像、日志、流量数据等），从而丰富渗透测试场景中的信息输入，提高模型对复杂攻击场景的适应能力。

RAG框架由知识库构建和任务处理流程两部分组成。（1）知识库构建。通过预先建立包含专业知识的向量检索数据库。（2）任务处理流程。当任务进入系统时，RAG模块首先从知识库中检索相关内容，将其嵌入上下文后输入大模型。许多研究表明，RAG能显著提升大模型在专业任务

中的表现。例如，Lewis 等[9]展示了 RAG 在问答系统中的高效性。在网络安全领域，RAG 技术能有效弥补大模型在知识覆盖范围上的不足，且在处理长尾信息或复杂技术术语时具有显著优势。

1.5　各方法分析对比

从成功率、上下文连续性、知识更新能力等角度，对比上述的三大类方法。在成功率与任务执行效果上，上述的三大类方法分别因为可能的局部最优、记忆长度不足、协同难度高等问题而出现成功率降低。本文框架引入 RAG 模块和内部经验库，不仅弥补了单一智能体在长流程任务中的上下文丢失问题，还通过多智能体协同实现了任务分工，整体提高了渗透测试的成功率和鲁棒性。在上下文连续性与知识更新能力上，垂类大模型的上下文窗口限制使其在长流程任务中容易缺失关键信息，RL 模型缺乏动态知识更新机制。本文框架融合 RAG 技术，通过实时检索外部知识库和集成内部经验，不仅保证了上下文连续性，还能及时更新安全情报，提高整体决策准确性。在响应速度与实时性上，RL 方法在在线训练中响应可能较慢，传统多智能体系统也需要大量时间通信和信息调度。本文通过合理的框架设计及双路向量数据库，显著提升了通信和信息检索的时间消耗，有更好的实时性。

2　Pentest-Chain 框架

Pentest-Chain 框架主要包括由环境探针、分析器、决策器、执行器和报告器构成的智能体链，以及由知识库和经验库构成的 RAG 增强模块，如图 1 所示。

2.1　智能体链

图 1 中，实线框内为智能体链部分，包括环境探针、分析器、决策器、执行器和报告器 5 个智能体；虚线框为 RAG 模块部分，包括知识库和经验库。环境交互流表示与目标环境的实际交互过程；智能体交互流表示智能体之间的协同信息流转；RAG 交互流表示 RAG 模块与各智能体间的检索与生成交互。

当任务进入时，任务描述会传递给环境探针、分析器和知识库，其中环境探针依据任务描述使用适用的方法探知目标环境，并将获取的信息整合传递给分析器；传递到知识库的任务描述会由 RAG 模块检索得到相关外部知识，且将获取的外部知识传递给分析器；分析器接收原任务

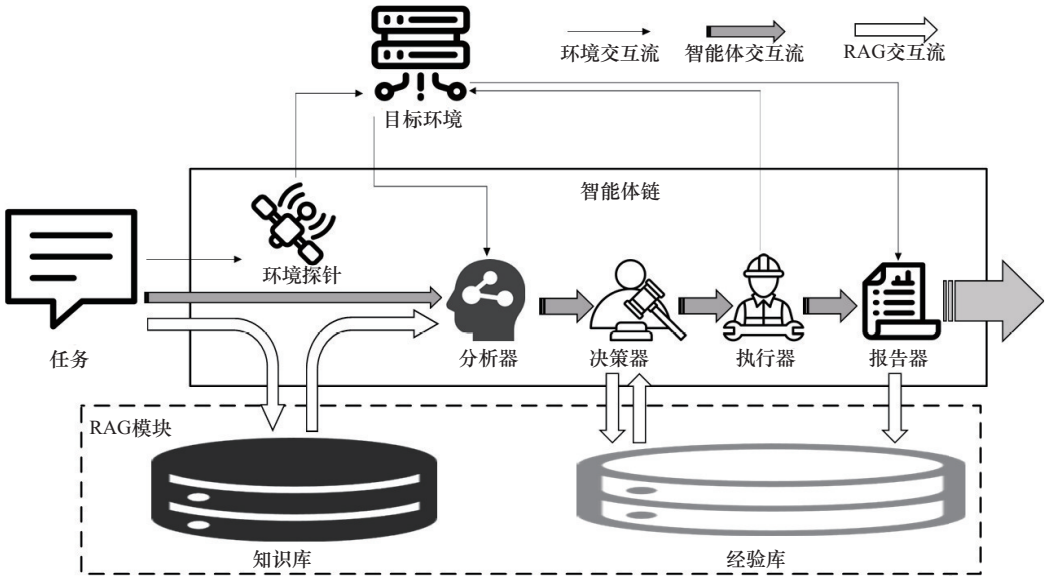


图 1　Pentest-Chain 框架

描述信息、环境探针实时探知的信息和知识库给出的相关外部知识信息并整合，生成系统的渗透流程给决策器。

决策器生成最终任务规划的依据包括两部分：分析器建议的渗透流程和经验库中的历史参考。决策器首先将分析器给出的建议与经验库中的历史案例进行匹配检索，再结合检索到的成功或失败经验，对分析器给出的建议渗透流程进行分析，最后基于综合分子结果，形成最终任务规划。

在最后的执行过程中，执行器依据任务规划对目标环境执行渗透任务，报告器则会接收结果并生成一份渗透测试报告输出。该报告还会作为成功或失败经验存储在经验库中。

2.2 智能体间交互

传统的多智能体系统中，多智能体之间通常通过固定格式的消息或动作指令进行交流，这种方式虽然有着很好的可读性，但在面对复杂任务时往往不够灵活，缺乏鲁棒性。为了实现更高效的协作与信息共享，本文的多智能体系统基于LLM的理解和推理能力，设计了一套统一的结构化数据交互模式以实现多智能体之间的交互和信息传递。

具体来说，多智能体间使用轻量级、结构化的数据交换协议（如JSON）统一封装任务标识、上下文摘要与操作指令等关键要素，以确保信息传递的结构化和一致性。当封装后的信息进入智能体时，智能体会自动匹配合适的预定义模板，再结合接收到的数据生成用于后续处理的标准化提示词（prompt）。这种模式不仅能精简冗余信息，降低交互成本，而且避免了文本传输可能出现的歧义和格式差异等问题。

同时，为了避免潜在的冲突和竞争问题，本文引入了一个全局上下文缓冲区，对任务状态、操作结果及环境信息进行统一管理。当智能体对已存在的相关上下文进行更新时，会触发冲突检测机制，该机制将结合信息来源得出置信度等级并考虑是否替换，以最小化决策风险。

2.3 各智能体模块

（1）环境探针

任务（Task）输入的第一步是通过环境探针（Env Probe）模块，旨在收集目标系统的关键信息与特性，为后续分析提供上下文支持。环境探针模块以智能体形式存在，能够自动探测目标系统的网络配置、服务状态、潜在的漏洞等关键数据。当任务进入系统时，该模块会触发探测流程，将所获取的信息整合并输出，为分析器提供详尽的环境背景。这一功能对任务的上下文感知至关重要，也为框架后续模块的精准决策奠定了基础。

（2）分析器

分析器（Analyzer）模块是Pentest-Chain框架中的核心组件之一，负责整合任务目标、环境探针模块的输出以及知识库提供的专业知识，实现对任务的深入语义理解与需求分析。它模拟人类专家对任务目标的推理能力，生成系统化的行动输入，为决策器提供支持。通过这一模块，系统能够识别复杂任务的核心需求，并进行抽象化处理，为后续模块的行动规划提供精确依据。

分析器接收到的任务目标通常以自然语言描述，环境探针模块的输出为目标系统的结构化状态信息（如端口、服务、系统版本），而知识库检索的结果包含与任务相关的技术策略、已知漏洞等语义知识。针对这3类信息，分析器需要将其嵌入指定预定义模板进行结构化处理，再将融合后的信息整理成标准化的任务上下文格式提示词（prompt），以此作为后续决策器输入的基础，智能体分析器提示词模板示例如图2所示。

分析器在整合任务目标、环境探针模块的输出以及知识库提供的专业知识这3类消息时可能存在冲突，本文引入的置信度机制认为：以任务

```
    #分析器提示词构建
#定义一个前置的分析器提示词模板
Analyzer_Prompt_Template = """
You are a cybersecurity expert skilled in penetration testing analysis. Below is a penetration testing task, along with relevant
environment information and knowledge for your reference:
Task: {Task}
Environment Information: {Env_Probe_Info}
Knowledge: {Knowledge_Info}
"""
Analyzer_Prompt = PromptTemplate(
    Input_variables=[“Task”, ”Env_Probe_Info”, ”Knowledge_Info”], #定义需要的输入变量
      Template= Analyzer_Prompt_Template #  使用模板
    )
```

图2　智能体分析器提示词模板示例

目标描述为基准，基于实时环境探针模块得到的真实信息始终具有最高优先级，知识库信息由知识来源权威性（如是否来自ATT&CK矩阵、CVE漏洞库）决定置信度得分。若知识库中来源为博客文章的信息与环境探针得到信息相悖时，缓冲区只会保留后者字段。

（3）决策器

决策器（Decider）模块是任务规划的关键模块，负责综合分析器的输出和经验库的参考数据，为任务生成详细的行动计划。其任务包括选择适当的攻击路径、生成攻击流程和可用的攻击脚本等。通过对多方信息的整合，决策器可制定最大限度提升执行效率和成功率的攻击计划，并减少不必要的尝试与误操作。

决策器在生成攻击流程时，其输入包含分析器生成的推荐流程与经验库中检索到的相似任务历史经验。系统首先对两方生成内容进行语义对齐，识别其差异部分，并评估每个方案的历史成功率、上下文匹配度及所需资源代价等指标，综合给出最终流程。

当分析器生成的推荐流程与经验库中检索到的相似任务历史经验存在冲突时，决策器进行以下判断：优先选择经验库中已验证过的高置信度路径，并保留在缓冲区，尤其是历史成功率较高或上下文高度一致的路径；若分析器给出的方案能覆盖经验库中未考虑的新情境，系统将并行保留2个候选路径，并尝试低成本路径先行探索。

（4）执行器

执行器（Executor）模块负责将决策器的计划付诸实践，发起实际的渗透测试操作。例如，调用特定的渗透工具、给定的运行脚本及实施模拟攻击流程。该模块与目标环境直接进行交互，确保行动计划落地实施。通过执行器，渗透测试框架能够高效完成任务目标，同时为后续模块提供执行过程中的实时反馈。

（5）报告器

报告器（Reporter）模块是任务的总结阶段，其主要职责是生成渗透测试报告，详细记录任务的目标、执行过程、结果及发现的安全隐患。报告不仅面向人类用户，也为框架的经验器提供输入，以便后续任务优化。生成的报告具有结构化和可读性，能够为网络安全专家提供直接可用的行动建议，同时助力后续任务的知识积累与优化。

2.4　RAG模块

已有基于大模型进行渗透测试的研究（如PentestGPT等），主要是对渗透任务的分解和流程规划等方面进行优化，并没有在知识检索、复杂决策场景作出改进。相较于通用大模型，RAG能够以较低的成本在指定垂直领域中大幅提升生成质量。RAG模块通常包括检索器和生成器。

检索器模块的任务是从一个外部知识库中挑

选与输入查询最相关的信息。其核心工作如下：（1）查询向量化。通常使用嵌入模型（如BERT[26]、Sentence-BERT[27]或专门训练的检索嵌入模型）将输入的文本（如问题或任务描述）转换为向量表示。需要注意的是，使用的嵌入模型应尽可能与之前存储知识库使用的嵌入模型保持一致。（2）内容检索。通过向量相似度（如计算余弦相似度或欧氏距离）或其他检索方法（如稀疏检索中的BM25和密集通道检索DPR[28]）匹配知识库中的相关信息块。（3）制定策略。检索策略会直接影响系统性能和生成结果的质量，常见的检索策略包括Top-k检索、混合检索和多轮检索。其中，Top-k检索借由kNN算法从存储库中返回相关性最高的k个条目，选取合适的k值从而得到足够多且质量过关的知识；混合检索是结合语义向量检索和基于关键词的精准检索，以同时捕获语义信息与特定关键词匹配的知识；多轮检索是针对复杂任务，将初次检索的结果作为二次检索的输入，以进一步缩小范围，提高相关性。

生成器模块将检索到的信息与输入查询集成，在生成最终输出方面起着至关重要的作用。在检索组件从外部来源提取到相关知识后，生成器将此信息合成连贯的、上下文适当的响应。采用预训练LLM（如GPT-4、T5和BERT）作为核心组件，可确保生成的文本流畅、准确并与原始查询保持一致。同时，利用人类反馈强化学习（reinforcement learning from human feedback，RLHF）[29]技术以及微调或少量参数适配（LoRA）[30]将特定领域的知识注入生成器中进行优化，使其在特定任务中的表现进一步提升。

本文构建了2个RAG模块，分别是外部知识库（knowledge library）和内部经验库（experience library）。

知识库模块通过与外部知识库交互，显著提升任务理解能力。构建的知识库包括与网络安全相关的权威资源，如ATT&CK矩阵、CVE漏洞库、技术文献及实际案例分析文章等。当任务进入RAG模块时，它会利用向量检索等技术，从知识库中提取最相关的条目，将检索结果与输入任务一并传递至分析器。基于这一模块，系统不仅具备上下文信息，还能动态引入领域内的高价值知识，显著提高生成的专业性和准确性。

经验库模块存储Pentest-Chain框架运行过程中积累的成功案例与失败经验，形成动态更新的知识数据库。其作用不仅限于为系统提供历史参考，还在于通过持续迭代优化RAG模块的知识库，使系统在长期使用中表现更优。例如，通过存储任务的输入与输出、执行过程及结果，经验库能够为未来的任务提供高效的知识补充。此外，经验库的存在实现了“自适应进化”，即随着系统运行的任务数量增加，系统的智能化水平和任务成功率会同步提升。

2.5 RAG构建

2.5.1 原始数据收集

考虑渗透测试相关数据集的特殊性和保密性，本文选取的原始数据均为公开数据，主要包括：ATT&CK矩阵、CVE漏洞库以及一些公开的渗透测试技术文章和报告。

其中，ATT&CK矩阵由MITRE组织维护，广泛用于网络安全领域，系统描述了攻击者的战术、技术和程序，为渗透测试提供标准化的攻击行为参考。CVE漏洞库由全球各大安全组织和厂商共同维护，是漏洞信息的权威来源，可确保系统能够获取最新的安全漏洞情报。同时，ATT&CK矩阵和CVE漏洞库具有高度结构化的特性，易于进行向量化处理，且能够高效地与RAG系统集成，实现快速检索。此外，ATT&CK矩阵和CVE漏洞库还提供了官方应用程序接口（application program interface，API），能够实时请求，动态检索最新知识。

2.5.2 数据分块

为了将知识转化为可被检索的高维向量，需

要将数据进行分块处理。对于大文本数据，使用了滑动窗口法分块，以 1 000 字符为一块，并在两个分块之间设置 200 字符的重叠部分以避免信息丢失。

2.5.3　嵌入模型选择

嵌入模型负责将文本数据（如任务描述或知识库中的内容）嵌入为向量表示，这是后续检索器能有效匹配输入与知识库的关键。嵌入模型的选择会直接影响检索的准确性和效率，常见的嵌入模型有 OpenAI 提供的 text-embedding-ada-002、Hugging Face 提供的 Sentence-transformers/Sentence-BERT 等。针对特定领域（如本文的渗透测试方向），可通过领域语料对基础模型（如 BERT）进行微调，进而训练专属的嵌入模型。

2.5.4　向量数据库使用

向量数据库是检索器工作的基础设施，负责存储知识库中内容的向量化表示，并支持高效检索。常用的向量数据库包括：Chroma、Faiss（Facebook AI Similarity Search）、Milvus 和 Weaviate 等。其中，Chroma 适用快速原型开发；Faiss 可支持多种索引算法，且更适合中小规模数据的高效查询；Milvus 有更完整的 GPU 加速方案和云平台支持，具备灵活的分布式部署能力，适合处理更大数据量的非结构化数据并加快检索；Weaviate 支持自动生成嵌入，且具有良好的多模态数据处理能力。因此，选择合适的向量数据库能够为 RAG 模块的检索器提供高效、稳定的技术基础，同时保障系统在实际应用中的性能表现和扩展能力。

RAG 中不同类型的数据有不同的查询需求，结构化的攻击信息更适合短而快的精确匹配，而非结构化的文档检索更需要广泛全面地获取文本分片的语义信息。为匹配 RAG 中不同类型数据的查询需求，Pentest-Chain 框架使用了双路 RAG 知识库：针对 ATT&CK 矩阵和 CVE 漏洞库这些数据量相对较小且高度结构化的数据，选择 Faiss 作为向量数据库；针对大量的技术文章和报告则选择使用 Milvus 作为向量数据库。分别选择 Faiss 和 Milvus 作为向量数据库的原因在于：（1）从数据量级来看，Faiss 更适合处理 10^4～10^5 规模的结构化数据，而 Milvus 可扩展至 10^7 以上的非结构化数据处理；（2）从性能角度来看，在本文的实验环境下，Faiss 对小规模数据集的查询延迟更低且更准确，而 Milvus 在并发查询速度和数据广度方面表现更优。此外，在本文的向量数据库中还使用了分层可导航小世界（hierarchical navigable small world，HNSW）算法[31]，以提升 RAG 在较高查询精度和较低响应延迟下的实时检索效率。

RAG 构建算法如图 3 所示。在构建完成的 RAG 增强模块中，系统首先接收任务的自然语言描述作为输入，并对其进行标准化预处理。随后，采用上文提到的嵌入模型将预处理文本编码为高维稠密向量，并将其作为查询向量传递至向量数据库。向量数据库利用上文提到的相似度匹配算法，从知识库中检索与任务语义最相关的若干条目，最终返回其对应的自然语言知识内容作为后续任务生成的辅助信息。

```
    # 原始数据收集
Metadata = fetch_knowledge(API_url);    //从API拉取数据
# 数据分块处理
Chunk_size =1000
Chunks = [] //空列表
For i in lengh(Metadata) / Chunk_size:   //按1 000字符长度切片
      //将整块 Metadata 切片为小块 chunk
Return Chunks; //返回切片列表
# 向量化
Embedding_model = "text-embedding-ada-002"
For Chunk in Chunks:   //对每个切片向量化
     Embedding_data = embedding(Chunk);
Return Embedding_datas;   //返回向量化后的向量列表
#   向量存储
For Embedding_data in Embedding_datas:
      Database.store(Embedding_data)
    Return Database;   //已存储至向量数据库中
    …….
```

图 3　RAG 构建算法

3 实验与评估

3.1 实验准备

3.1.1 实验环境

本实验将主体系统部署在Kali虚拟机环境中，以便快速调用各种渗透测试工具的API，实现对任务的高效管理和测试环境的灵活配置。Kali Linux是一款专为渗透测试和安全审计设计的系统，内置了多种常用工具，能够为实验提供强大的技术支持。

3.1.2 模型选择

本实验选取的大语言模型涵盖了当前主流的在线商业模型和本地开源模型：GPT-4o（性能强劲的商业在线模型）、GPT-4o-mini（简化版GPT-4o，适用于资源受限的场景）、Claude3.5-HaiKu（由Anthropic提供，具有良好的对话能力和任务理解能力）和Llama3.1-8B（Meta提供的开源大模型，适用于本地部署场景，兼顾性能和成本）。

3.2 成功率分析

为评估Pentest-Chain框架在渗透测试任务中的实际表现，本实验使用单一任务集来评估各种任务的完成情况。共选择了13个种子任务，并根据任务的目标将其分为3类：A类（侦察型），信息收集和目标识别的任务；B类（利用型），直接利用漏洞执行攻击任务；C类（权限维持/提升型），用于权限控制、权限提升和保持对系统访问。经过调研与专家建议，设定了以下任务评判指标。

（1）自动化可行性：评估任务的自动化难易度，分为“高”“中”“低”。这一指标反映了任务是否适合在没有人工干预的情况下完成。

（2）自主决策需求：评估任务执行时对大模型自主决策的依赖程度，分为“高”“中”“低”。较高的自主决策需求意味着任务需要更复杂的推理或动态决策。

（3）任务难度：任务的整体难度，用来评估执行任务的复杂性与资源需求。难度等级帮助评估系统资源分配和优先级设置，分为“高”“中”“低”。

渗透测试任务集及各指标情况见表1。

表1 渗透测试任务集及各指标情况

编号	任务名称	任务类型	自动化可行性	自主决策需求	任务难度
1	Web枚举	A	高	低	低
2	网络枚举	A	高	低	低
3	流量分析	A	高	中	中
4	文件写入	B	高	低	中
5	权限提升	B	中	高	高
6	文件上传	B	高	低	中
7	SQL注入	B	中	高	高
8	XSS攻击	B	中	中	中
9	哈希传递攻击	C	中	高	高
10	哈希存储分析	C	高	低	中
11	MySQL扫描	C	高	低	中
12	SSH公钥注入	C	中	中	中
13	恶意EXE文件	C	中	低	高

为了验证Pentest-Chain框架的任务完成情况，结合Vulhub和自行搭建的测试环境，针对13个种子任务，各选取了10个代表性任务，共130个任务进行了详细测试，并且加入了使用单一智能体及使用AutoGPT的情况作为对比。A类、B类和C类任务下各框架使用不同开源模型的成功率对比分别如图4、图5、图6所示。

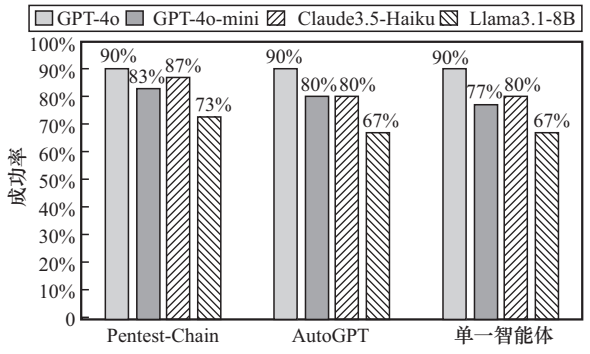


图4 A类任务下各框架成功率对比

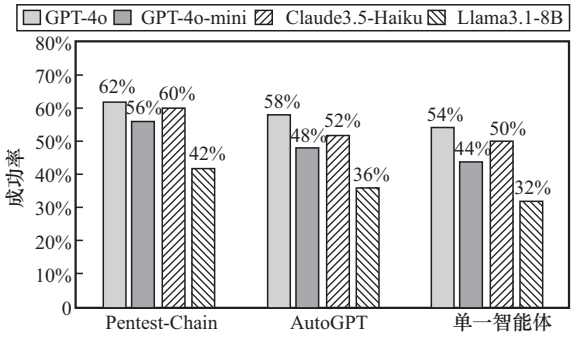


图5　B类任务下各框架成功率对比

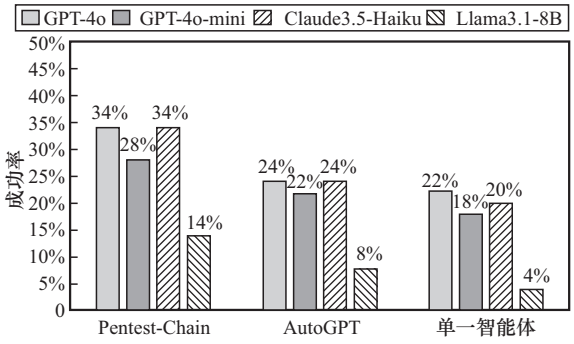


图6　C类任务下各框架成功率对比

由以上对比实验可知：使用Pentest-Chain多智能体框架在B类任务（利用型）和C类任务（权限提升型）上存在优势，尤其是在C类任务中的成功率提升明显，而在A类任务（侦察型）下，单一智能体、AutoGPT和Pentest-Chain差距较小，表现接近。整体而言，单一智能体、AutoGPT和Pentest-Chain框架下所有任务的平均成功率分别为47%、49%和55%，使用多智能体框架后平均成功率相较单智能体可提升17.02%。

此外，高参数模型（如GPT-4o）在所有任务类型中表现最优，尤其是在复杂的B类和C类任务中；低参数模型表现相对较弱，但在Token、成本消耗上可能更优，该问题在第3.4节讨论。

3.3　消融实验

本节主要验证RAG在Pentest-Chain框架中的作用，将对比带有RAG模块的系统与没有RAG模块系统之间的成功率及效率的差异。为最大限度降低模型对消融结果的影响，选择表现最好的GPT-4o作为核心模型来进行实验。RAG模块消融实验结果见表2。

表2　RAG模块消融实验结果

任务类型	完整框架	移除知识库	移除经验库	完全移除RAG
A类	83%	79%	83%	79%
B类	55%	47%	53%	45%
C类	28%	19%	24%	16%

由表2可知，在单独移除知识库时，Pentest-Chain框架渗透测试的成功率有所下降，特别是在需要知识查询支持的任务（如B类和C类）中影响较为明显；单独移除经验库时，Pentest-Chain框架渗透测试的成功率略低于完整框架，但优于完全移除RAG的情况；若完全移除RAG模块，Pentest-Chain框架渗透测试的成功率将明显降低，尤其在C类任务（复杂权限操作）中，成功率下降显著。总的来说，较不使用RAG模块的系统，添加了RAG模块的系统的整体任务成功率提升了18.4%。

此外，使用RAG与不使用RAG的系统在响应时间上也会存在一定差异。其中，使用RAG检索知识会导致时间消耗小幅增长。同时考虑到多智能体系统本身的额外开销如任务分解、流程规划和智能体间通信等对时间消耗的影响，本次实验对比了Pentest-Chain框架和单智能体在3类任务中分别有无RAG的成功任务的完成时间消耗，对比结果如图7所示。

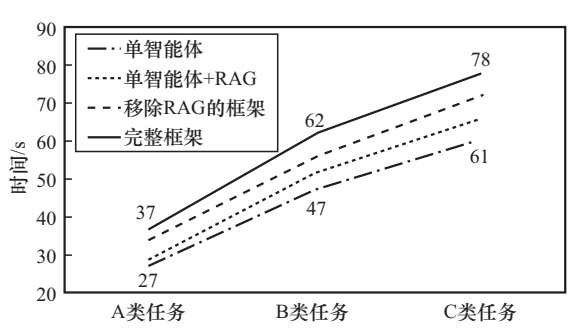


图7　各情况下时间消耗对比

综合模型的智能体推理速度和RAG检索速度可以发现，本文的智能体链和RAG增强模块的设计暂时没有对整体效率造成显著影响。

3.4 Token成本对比（RQ3）

本节对比了包括GPT-4o、GPT-4o-mini、Claude 3.5-Haiku 3种在线模型消耗的Token及费用，评估不同模型的表现。截至2025年1月，各模型Token费用对比见表3。

表3 不同模型Token费用对比

不同在线模型	输入Token价格	输出Token价格	总体成功率
GPT-4o	2.50美元/1M	10.0美元/1M	75/130
GPT-4o-mini	**0.150**美元**/1M**	**0.600**美元**/1M**	67/130
Claude3.5-HaiKu	0.80美元/1M	4.00美元/1M	73/130

对全体任务取平均消耗的Token，可用以下数学式表示：

$$\text{单位成功率成本} = \frac{\text{Task花费}}{\text{成功率}} \quad (1)$$

其中，单位成功率成本用于衡量每成功完成一个任务的平均成本，是评估模型性价比的指标。当模型Task花费越低且成功率越高时，其单位成功率成本就越低，也就是越好。

各模型单位成功率成本对比见表4。虽然GPT-4o在任务成功率上表现最为出色，但其运行成本也显著高于其他模型，导致单位成功率成本较高。而GPT-4o-mini则在成功率与成本之间找到了一个较好的平衡点，是在性价比上优于其他模型的理想选择。

表4 各模型单位成功率成本对比

模型	输入Token	输出Token	Task花费/美元	单位成功率成本
GPT-4o	3 237	2 295	0.031 041	0.053 80
GPT-4o-mini	3 435	2 433	0.001 980	**0.003 84**
Claude3.5-HaiKu	3 341	2 364	0.012 124	0.021 59

考虑了本地模型的实际部署成本，以本文实验环境为例，使用一台搭载单张4090显卡（24 GB显存）的高性能计算机作为硬件平台，部署了4位量化的Llama 3.1-7B模型，单样本推理速度约为50 token/s，基本能够满足实时处理任务需求。

与此同时，尽可能全面地量化了本地模型部署所涉及的成本项，包括硬件采购及折旧、电力消耗及运维开销等，以评估其在实际应用中的成本控制表现。本文采用了总拥有成本（total cost of ownership，TCO）分析方法，对各项成本进行定量计算。

对于部署本地模型的主要硬件投入，包括高性能GPU（如RTX 4090）、CPU、内存和存储设备，采用线性折旧法，将硬件成本分摊到预期使用年限内得出折旧成本。使用模型推理时，GPU及其他硬件的电力消耗构成了持续性的成本支出。结合硬件在实际推理场景下的功耗及当地电价，可得出每小时及每月的运行成本。假设硬件采购成本约为20 000元，预期使用寿命36个月，当地平均电价为0.55元/(kW·h)，结合单样本推理速度50 token/s，计算得出平均价格约为5.51元/百万token，折合token价格约0.75美元/1M。基于此模型下的总体成功率为50/130，最终可以得出单位成功率成本为0.009 83。本地模型的单位成功率成本见表5。对比以上实验模型的单位成功率成本可以发现，本地模型目前尚无显著的成本优势。

表5 本地模型的单位成功率成本

模型	输入Token	输出Token	Task花费/美元	单位成功率成本
Llama3.1-7B	2 984	2 059	0.003 782	0.009 83

4 结束语

本文提出了一种基于大语言模型的自动化渗

透测试框架 Pentest-Chain，创新性地融合了 RAG 技术，为自动化渗透测试领域提供了新的解决方案。实验结果表明，相较于使用单一智能体方案，Pentest-Chain 框架能够大幅提升任务执行的成功率。消融实验进一步验证了 RAG 增强模块在 Pentest-Chain 框架中的关键作用。

未来关于自动化渗透测试的研究，还可以在以下方面做出改进：一是针对部分任务因流程连续性中断而失败的问题（通常是源于某些模块在特定情况下未能正确响应），可以尝试在框架中动态评估各模块的运行状态与任务完成情况，以保证整体流程的连续性。二是当前 RAG 知识数据库仅支持文本数据处理，而在实际渗透测试中，常涉及图像（如攻击代码截图、配置示意图）、视频等多模态数据。为进一步提升系统的实用性，后续研究可以扩展 RAG 模块对多模态数据的处理能力。

参考文献：

[1] STEFINKO Y, PISKOZUB A, BANAKH R. Manual and automated penetration testing. Benefits and drawbacks. Modern tendency[C]//Proceedings of the 2016 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET). Piscataway: IEEE Press, 2016: 488-491.

[2] HU Z G, BEURAN R, TAN Y S. Automated penetration testing using deep reinforcement learning[C]//Proceedings of the 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). Piscataway: IEEE Press, 2020: 2-10.

[3] 臧艺超, 周天阳, 朱俊虎, 等. 领域独立智能规划技术及其面向自动化渗透测试的攻击路径发现研究进展[J]. 电子与信息学报, 2020, 42(9): 2095-2107.
ZANG Y C, ZHOU T Y, ZHU J H, et al. Domain-independent intelligent planning technology and its application to automated penetration testing oriented attack path discovery[J]. Journal of Electronics & Information Technology, 2020, 42(9): 2095-2107.

[4] ROY S S, THOTA P, NARAGAM K V, et al. From chatbots to phishbots? : phishing scam generation in commercial large language models[C]//Proceedings of the 2024 IEEE Symposium on Security and Privacy (SP). Piscataway: IEEE Press, 2024: 36-54.

[5] GUPTA M, AKIRI C, ARYAL K, et al. From ChatGPT to ThreatGPT: impact of generative AI in cybersecurity and privacy[J]. IEEE Access, 2023, 11: 80218-80245.

[6] DENG G L, LIU Y, MAYORAL-VILCHES V, et al. Pentest-GPT: evaluating and harnessing large language models for automated penetration testing[C]//Proceedings of the 33rd USENIX Conference on Security Symposium. Berkeley: USENIX Association, 2024: 847-864.

[7] BUBECK S, CHANDRASEKARAN V, ELDAN R, et al. Sparks of artificial general intelligence: early experiments with GPT-4[J]. arXiv preprint, 2023, arXiv:2303.12712.

[8] JI Z W, LEE N, FRIESKE R, et al. Survey of hallucination in natural language generation[J]. ACM Computing Surveys, 2023, 55(12): 1-38.

[9] LEWIS P, PEREZ E, PIKTUS A, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks[C]// Proceedings of the 34th International Conference on Neural Information Processing System. New York: Curran Associates, 2020: 9459-9474.

[10] CHEN F, REN W. On the control of multi-agent systems: a survey[J]. Foundations and Trends® in Systems and Control, 2019, 6(4): 339-499.

[11] STROM B E, APPLEBAUM A, MILLER D P, et al. Mitre att&ck: design and philosophy[R]. 2018.

[12] WALTERMIRE D, SCARFONE K. Guide to using vulnerability naming schemes: Special Publication (NIST SP) -800-51 Rev 1[S].2011.

[13] ZHAO A, HUANG D, XU Q, et al. ExpeL: LLM agents are experiential learners[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2024, 38(17): 19632-19642.

[14] MCDERMOTT J P. Attack net penetration testing[C]//Proceedings of the 2000 Workshop on New Security Paradigms. New York: ACM Press, 2001: 15-21.

[15] 陈可, 鲁辉, 方滨兴, 等. 自动化渗透测试技术研究综述[J]. 软件学报, 2024, 35(5): 2268-2288.
CHEN K, LU H, FANG B X, et al. Survey on automated penetration testing technology research[J]. Journal of Software, 2024, 35(5): 2268-2288.

[16] HOANG L V, NHU N X, NGHIA T T, et al. Leveraging deep reinforcement learning for automating penetration testing in reconnaissance and exploitation phase[C]//Proceedings of the 2022 RIVF International Conference on Computing and Com-

munication Technologies (RIVF). Piscataway: IEEE Press, 2022: 41-46.

[17] ZHANG K Q, YANG Z R, BAŞAR T. Multi-agent reinforcement learning: a selective overview of theories and algorithms[M]// Handbook of Reinforcement Learning and Control. Cham: Springer, 2021: 321-384.

[18] GHANEM M C, CHEN T M. Reinforcement learning for efficient network penetration testing[J]. Information, 2020, 11(1): 6.

[19] ZENNARO F M, ERDŐDI L. Modelling penetration testing with reinforcement learning using capture-the-flag challenges: Trade-offs between model-free learning and a priori knowledge[J]. IET Information Security, 2023, 17(3): 441-457.

[20] ACHIAM J, ADLER S, AGARWAL S, et al. GPT-4 technical report[J]. arXiv preprint, 2023, arXiv:2303.08774.

[21] CHUNG H W, HOU L, LONGPRE S, et al. Scaling instruction-finetuned language models[J]. Journal of Machine Learning Research, 2024, 25(70):1−53.

[22] BHARGAVA P, NG V. Commonsense knowledge reasoning and generation with pre-trained language models: a survey[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2022, 36(11): 12317-12325.

[23] SHEN X M, WANG L Z, LI Z Y, et al. PentestAgent: incorporating LLM agents to automated penetration testing[J]. arXiv preprint, 2024, arXiv: 2411.05185.

[24] XU J C, STOKES J W, MCDONALD G, et al. AutoAttacker: a large language model guided system to implement automatic cyber-attacks[J]. arXiv preprint, 2024, arXiv: 2403.01038.

[25] WU Q Y, BANSAL G, ZHANG J Y, et al. AutoGen: enabling next-gen LLM applications via multi-agent conversation[J]. arXiv preprint, 2023, arXiv:2308.08155.

[26] DEVLIN J, CHANG M-W, LEE K, et al. BERT: pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint, 2018, arXiv:1810.04805.

[27] REIMERS N, GUREVYCH I. Sentence-BERT: sentence embeddings using Siamese BERT-networks[J]. arXiv preprint, 2019, arXiv: 1908.10084.

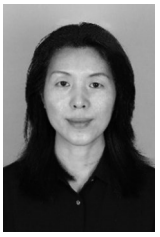
[28] KARPUKHIN V, OĞUZ B, MIN S, et al. Dense passage retrieval for open-domain question answering[J]. arXiv preprint, 2020, arXiv: 2004.04906.

[29] ZIEGLER D M, STIENNON N, WU J, et al. Fine-tuning language models from human preferences[J]. arXiv preprint, 2019, arXiv: 1909.08593.

[30] HU E J, SHEN Y, WALLIS P, et al. Lora: low-rank adaptation of large language models[J]. arXiv preprint, 2021, arXiv: 2106.09685.

[31] MALKOV Y A, YASHUNIN D A. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020, 42(4): 824-836.

[作者简介]



江颉（1972− ），女，博士，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。

蔡辰旭（2000− ），男，浙江工业大学计算机科学与技术学院硕士生，主要研究方向为网络安全与大语言模型。

李明达（1998− ），男，浙江工业大学计算机科学与技术学院博士生，主要研究方向为网络安全。

朱添田（1992− ），男，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与网络攻防。