https://crad.ict.ac.cn
63(3): 640−667, 2026

大语言模型存储机制安全风险综述

王 柳[1] 王申奥[2] 侯心怡[2] 赵 建[2] 吴荣鑫[3] 向 乔[3] 赵彦杰[2] 王 祎[1]

[1](北京邮电大学计算机学院(国家示范性软件学院) 北京 100876)

[2](华中科技大学网络空间安全学院 武汉 430074)

[3](厦门大学信息学院 福建厦门 361005)

(w_liu@bupt.edu.cn)

Survey of Storage Mechanism Security Threats for Large Language Models

Wang Liu[1], Wang Shenao[2], Hou Xinyi[2], Zhao Jian[2], Wu Rongxin[3], Xiang Qiao[3], Zhao Yanjie[2], and Wang Yi[1]

[1](*School of Computer Science* (*National Pilot Software Engineering School*), *Beijing University of Posts and Telecommunications*, *Beijing* 100876)

[2](*School of Cyberspace Security*, *Huazhong University of Science and Technology*, *Wuhan* 430074)

[3](*School of Informatics*, *Xiamen University*, *Xiamen*, *Fujian* 361005)

Abstract Large language models (LLMs), as a cornerstone technology in natural language processing (NLP), have demonstrated exceptional capabilities in text generation, information retrieval, and conversational systems. These models, such as ChatGPT, LLaMA, and Gemini, have been applied across various fields, including healthcare, education and finance, achieving near-human or even superhuman performance. However, with the widespread adoption of LLMs, their storage mechanisms face significant security and privacy risks throughout their lifecycle. Core storage modules, including model file storage, inference caching, and knowledge vector storage, support the functionality and efficiency of LLMs but also expose vulnerabilities and threats. For example, model file storage faces risks such as weight leakage and backdoor injection, inference caching is susceptible to side-channel attacks, and knowledge vector storage faces data poisoning and workflow hijacking. To address these emerging challenges, researchers have proposed defense strategies such as model encryption, backdoor detection, cache partitioning, and content filtering techniques. Although existing techniques have improved security to some extent, LLM storage technologies still face a range of critical challenges. This paper systematically reviews security risks and defense mechanisms for LLM storage across its lifecycle, focusing on attack surfaces and mitigation strategies. It identifies current limitations and highlights future research directions to enhance the security and reliability of LLM storage mechanisms.

Key words large language model; model file storage; model inference cache; vector storage; security and privacy

摘 要 大语言模型作为近些年自然语言处理领域的核心技术,在文本生成、信息检索和对话系统中展现了卓越的能力。诸如 ChatGPT, LLaMA, Gemini 等模型在医疗、教育、金融等领域实现了接近甚至超越人类水平的性能。然而,随着大语言模型的广泛应用,其存储机制在生命周期中的安全与隐私风险日益显现。模型文件存储、推理缓存和知识向量存储作为核心模块,在提升模型功能和效率的同时,也暴露出权重泄露、后门注入、侧信道攻击、知识污染与工作流劫持等一系列安全威胁。为应对这些风险,研究者提

收稿日期: 2025-06-17; **修回日期**: 2025-09-05

基金项目: 国家自然科学基金项目(62172049)

This work was supported by the National Natural Science Foundation of China (62172049).

通信作者: 王祎(yiwang@bupt.edu.cn)

出了模型加密、后门清除、缓存分区与隔离、检索内容过滤与验证等多种防御策略。尽管现有技术在一定程度上提升了安全性，但大语言模型存储技术仍面临一系列关键安全挑战。系统性综述了大语言模型存储机制的安全风险及其防御技术，重点分析了攻击面与缓解策略，并提出了未来研究方向以推动大语言模型存储机制的安全性与可靠性发展。

关键词 大语言模型；模型文件存储；模型推理缓存；向量存储；安全与隐私

中图法分类号 TP309

DOI: 10.7544/issn1000-1239.202550481 **CSTR:** 32373.14.issn1000-1239.202550481

大语言模型(large language model, LLM)作为自然语言处理(natural language processing, NLP)领域的重要技术，近年来在文本生成、信息检索、对话系统等多种应用中展现了卓越的性能和广泛的适用性。诸如 ChatGPT[1], LLaMA[2], Gemini[3] 等主流模型在学术、医疗、教育、金融等多个领域实现了接近甚至超越人类水平的语言理解与生成能力[4-5]。然而，随着大语言模型的广泛应用，其安全性问题逐渐显现[6]，尤其在存储生命周期中的安全风险亟需关注。大语言模型的存储机制贯穿其开发、训练、推理、部署等全生命周期，对模型的完整性与机密性、推理效率的优化以及系统的可靠性形成重要支撑，同时也成为攻击者潜在的高价值攻击目标。

大语言模型的存储机制可以划分为 3 个核心模块：模型文件存储、模型推理缓存和知识向量存储。模型文件存储用于保存训练完成的模型权重和结构信息，是大语言模型的核心资产，文件规模通常达到数百 GB 甚至 TB 级别[7]。模型推理缓存通过存储推理过程中的中间计算结果，例如注意力机制中的键值对，实现了高效推理[8]，特别是在实时性要求较高的场景(如对话系统和搜索引擎)中起到关键的效率优化作用。知识向量存储则通过存储高维语义向量，支持检索增强生成(retrieval-augmented generation, RAG)，从而使得大语言模型能够广泛应用于开放域问答系统和推荐系统[9-10]。这些存储机制有效提升了大语言模型的性能与应用广度，但也引入了新的安全隐患。

现有研究表明[11-13]，大语言模型存储机制的安全与隐私风险贯穿模型文件存储、模型推理缓存和知识向量存储等多个层面。模型文件存储作为模型的核心资产，面临模型窃取、篡改和逆向等多种威胁[14]，攻击者可能通过直接访问本地存储的权重文件来实现模型窃取，或通过注入恶意代码来实现模型后门攻击[12]，进而导致知识产权损失和模型行为操控。模型推理缓存通过存储推理过程中的中间计算结果，提升了推理效率，但也暴露于多种侧信道攻击之下，例如时序侧信道攻击(timing side-channel attacks)利用缓存命中与未命中时间差推断用户输入[15]，硬件侧信道攻击通过监控硬件缓存访问模式泄露推理中间状态[16]，网络侧信道攻击则从加密流量的特征中推测模型生成内容[17]。知识向量存储通过存储高维语义向量支持快速的语义检索，但也面临知识污染[18]、后门植入[19]、隐私泄露[20]、工作流劫持[21] 等一系列问题，攻击者可能通过恶意操控知识库内容或逆向工程向量数据，干扰系统的可靠性并窃取敏感信息。这些攻击手段不仅威胁到模型的机密性、完整性和可用性，还可能导致用户隐私数据泄露和系统行为失控。

为应对这些多层次的安全风险，研究者提出了多种防御技术。在模型文件存储安全方面，可用的防御技术包括模型加密[22]、混淆[23]、可信执行环境(trusted execution environment, TEE)等，通过隐藏模型结构信息或限制运行权限，防止直接窃取与篡改。在模型推理缓存安全方面，研究者提出了缓存分区与隔离[24]、动态缓存淘汰策略[25] 以及缓存加密技术[26]，通过限制缓存共享范围或在推理过程中引入噪声来减少侧信道信息泄露的可能性。针对知识向量存储安全，防御措施主要包括数据审查与过滤[27]、基于差分隐私的保护技术[28] 和检索过程中的安全机制[29]，以增强知识向量存储系统的防御能力。尽管这些技术在一定程度上提升了存储机制的安全性，但仍需进一步探索更高效、更全面的解决方案。

本文从大语言模型存储机制的攻击与防御技术出发，系统性地分析大语言模型存储机制全生命周期中的安全风险。本文的主要贡献总结有 3 点：

1)全面综述现有风险。回顾并总结了大语言模型存储机制的 3 个核心模块，即模型文件存储、模型推理缓存和知识向量存储所面临的安全威胁，深入分析了各类攻击方法的技术原理与适用场景。

2)分类总结防御技术。从模型文件机密性和完整性保护、推理缓存的侧信道防护、向量存储的隐

私增强等角度,系统性地梳理了针对存储机制中不同风险的防御策略,讨论了其适用性与局限性。

3)展望未来研究方向。结合当前存储机制安全研究的不足,提出了未来大语言模型存储机制风险防御技术的发展方向。

总体而言,随着大语言模型的规模不断扩大和应用场景的日益多样化,其存储机制的安全性研究已成为保障大语言模型系统可信性和鲁棒性的重要方向。本文希望通过对其风险与防御技术的系统分析,为推动大语言模型存储安全的研究与实践提供参考。

1 背景知识

1.1 大语言模型生命周期

大语言模型是近年来自然语言处理领域的核心技术,展现出卓越的语言理解与生成能力。通常,主流的大语言模型以自回归 Transformer 架构为基础,依靠多层注意力机制和庞大的参数规模捕获语言的复杂语法、语义以及上下文关系,从而在文本生成[30]、智能问答[31]和软件工程[32]等任务中取得了广泛的应用。通常,一个完整的大语言模型生命周期可以分为开发与训练、测试与验证、共享与发布,以及部署与服务 4 个阶段[33]。图 1 总结了大语言模型生命周期的 4 个关键阶段和所包含的具体内容,构成其从开发构建到实际部署的完整流程。

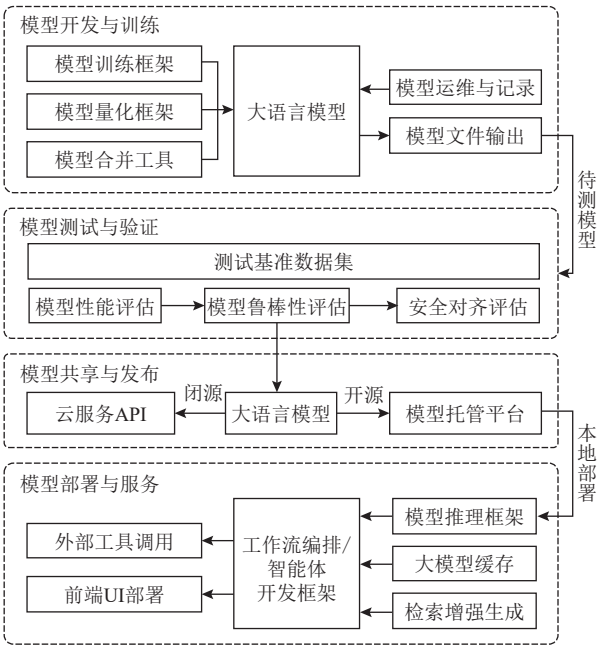


Fig. 1 Overall lifecycle process of large language models

图 1 大语言模型全生命周期流程

模型开发与训练通常包括 2 个主要阶段:预训练和微调[4]。在预训练阶段,开发者通常利用自监督学习方法,使模型能够捕捉语言的广泛语法和语义特性,为模型提供强大的通用语言能力,例如 GPT 系列模型通过自回归语言建模逐词预测序列中的下一个词元(token)[34]。随后,在模型微调阶段,开发者通常通过在特定任务数据集上的有监督学习进一步优化其性能,使其能够适应具体的应用场景[4],例如问答系统。开发与训练阶段是大语言模型生命周期中计算最为密集的部分,其训练通常需要依赖分布式计算框架,同时需要存储和管理大规模的权重数据,这些数据的大小可能达到数百 GB 甚至 TB 级别[7]。

模型测试与验证阶段通常是为了确保其在实际应用中的准确性、鲁棒性和安全对齐[35]。这一阶段的目标是通过一系列基准数据集测试来评估模型的性能,并对可能的偏差或漏洞进行分析[36]。测试的内容包括模型在各种下游任务基准数据集上的性能表现[35]、对多样化输入的泛化能力以及对恶意输入的防御能力[36]。例如,开发者通常需要对模型在实际应用过程中可能产生的敏感信息泄露、偏见、毒性、幻觉,以及模型越狱等一系列问题进行测试,以确保其在实际场景中的安全性和可靠性。

模型发布与共享阶段通常包括开源分发和 API 服务 2 种方式。开源模型(如 LLaMA[2], Qwen[37] 等)通常以预训练权重的形式发布在公共平台上,例如 Hugging Face,其他开发者可以基于这些预训练模型文件进一步微调或部署模型。而商业化模型通常以云服务 API 的方式提供(如 OpenAI 的 ChatGPT),这种方式能够有效保护模型的机密性和完整性,但对服务端的推理性能提出了更高要求。

部署与服务阶段是大语言模型面向实际应用并为用户提供功能的关键阶段。在云端部署场景中,大语言模型通常运行在高性能计算集群上,通过分布式推理框架处理大量并发请求。为了提高推理效率,部署过程中通常会引入推理缓存技术[8],通过存储中间计算结果减少重复计算,从而显著降低模型响应时间。此外,知识向量存储逐渐成为部署阶段的重要组成部分,开发者通过检索增强生成实现从外部知识库中动态获取信息,以缓解大语言模型幻觉、克服知识更新滞后性并增强生成过程的事实一致性。

1.2 大语言模型存储机制

大语言模型的存储机制贯穿其整个生命周期,是支持模型开发、部署和应用的核心技术。随着模

型规模和应用场景的不断扩展，大语言模型存储的需求呈现出显著的多样性和复杂性。但总体而言，大语言模型生命周期中的存储机制可以概括为模型文件存储、模型推理缓存和模型知识向量存储 3 个关键方面。

模型文件存储是大语言模型的基础存储形式，用于保存训练完成的模型权重和结构信息。目前主流的大语言模型的参数规模通常达到数十亿甚至上万亿级别，这使得模型权重文件的存储需求激增。例如，DeepSeek R1 的参数量达到 6 710 亿，其完整权重文件的大小达到 TB 级别。在开发与训练阶段，模型文件需要频繁加载和更新，而在发布与共享阶段则需要通过模型文件存储来实现模型分发。因此，模型文件存储的安全性在整个生命周期存储机制中尤为关键。

模型推理缓存是支持大语言模型高效推理的重要存储机制[8]，尤其在实时性要求较高的场景，例如对话系统和搜索引擎中。模型推理缓存通过存储推理过程中的中间计算结果，例如注意力机制中的键值缓存（key-value cache，KVCache），减少重复计算，从而显著提升推理效率。例如，当用户输入的提示词（prompt）与之前的输入存在相似性时，推理缓存可以直接复用之前的计算结果，避免重新计算。尽管推理优化提高了多租户环境中的推理效率，但也引入了新的安全问题，例如时序侧信道攻击可能借助缓存的时间差异推断用户输入内容[15]。

模型知识向量存储是近年来随着语义检索需求增长而兴起的一种存储形式，它将文本数据转化为高维向量表示，并存储于向量数据库中以支持高效的语义检索。开发者通常通过计算查询向量与存储向量之间的相似度，为用户提供快速准确的检索结果。例如，大语言模型生成的语义嵌入可以被存储在专用的向量数据库（如 FAISS[38] 或 Milvus[39]）中，支持海量知识的快速查询。然而，知识向量存储也面临一系列安全挑战，例如针对向量知识库的投毒污染，或者针对检索增强生成过程的后门植入。

1.3 大语言模型存储机制的攻击面分析

大语言模型的存储机制在支撑模型分发、推理和服务的同时，也不可避免地成为潜在的攻击目标。随着模型规模的不断扩大和应用场景的日益多样化，攻击者可能通过针对存储机制的漏洞实施恶意行为，威胁模型的机密性、完整性和可用性。模型文件存储、推理缓存和知识向量存储作为大语言模型存储机制的 3 个核心部分，各自面临着不同的安全挑战和攻击面。图 2 总结了大语言模型全生命周期中的存储机制及其所暴露的攻击向量。

在模型文件存储中，权重泄露和恶意篡改是最主要的攻击风险。模型权重是大语言模型的核心资产之一，包含了模型通过训练学习到的全部知识和模式，具有极高的商业价值。如果攻击者通过模型窃取攻击获取了权重文件，可能会非法复制模型，甚至通过逆向工程分析训练数据，从而导致知识产权和敏感信息的泄露。更进一步，攻击者可能通过修改权重文件或注入恶意代码[12]，将隐藏的后门嵌入到模型中，使得模型在特定的触发条件下生成恶意的内容或执行恶意代码。例如，攻击者可以通过提

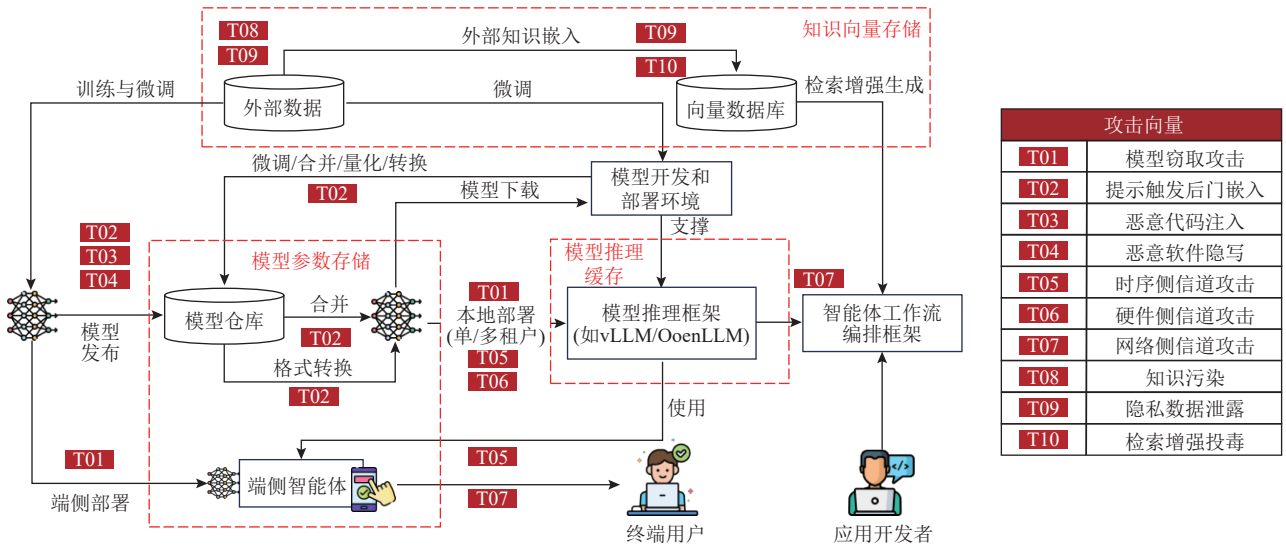


Fig. 2 Analysis of storage mechanisms and attack vectors across the LLM lifecycle

图 2 大语言模型生命周期中的存储机制与攻击向量分析

示触发后门嵌入，利用特定输入激活模型的恶意行为[40]。此外，针对模型文件的隐写型后门也可能被用于隐藏攻击者的代码或信息[41]，进一步增加检测和防御的难度。

模型推理缓存在多租户环境中的广泛共享也暴露出其重要的攻击面。推理缓存中可能包含用户输入的敏感数据，攻击者可以通过一系列侧信道信息来推断用户隐私。这些攻击在多租户环境中尤为严重，因为多个用户可能共享同一缓存区域，攻击者可以通过分析共享缓存来窃取其他用户的输入数据或推理结果。例如，时序侧信道攻击利用缓存命中的时间差异推断用户的输入内容[15]；硬件侧信道攻击通过监测硬件资源的访问模式[42]窃取缓存中的数据；网络侧信道攻击则通过分析网络流量特征[17]推测用户交互内容。这些攻击不仅导致用户隐私泄露，还可能危及模型在推理过程中的稳定性。

模型知识向量存储作为知识检索的重要支撑，同样面临多重安全威胁，包括知识污染与投毒攻击、隐私数据泄露以及基础设施漏洞。知识污染与投毒攻击是知识向量存储面临的主要威胁之一。攻击者通过向向量数据库中注入恶意或伪造数据，例如语义相似的虚假文档，从而污染检索结果并诱导模型生成错误或有害的内容[43]。这类攻击不仅破坏了系统输出的准确性，还可能通过后门与触发式攻击隐蔽地操控模型行为，使恶意内容在特定条件下被激活[44]。此外，知识向量中的高维语义信息成为敏感信息泄露的主要来源。攻击者可以通过数据提取攻击设计精巧的查询，从检索增强生成系统中逐步获取私有文档内容[45]；或通过成员推断攻击，分析模型响应的微小差异以推测某些数据是否存在于知识库中[46]。与此同时，向量数据库作为整个知识向量存储阶段的关键基础设施，其存储、索引与查询过程中的漏洞可能被攻击者利用，导致向量数据库崩溃或者出现错误行为[13,47]，影响检索增强生成系统的可靠性。

2 相关工作

大语言模型因其在处理复杂语言任务时展现出的卓越能力，迅速成为人工智能领域的研究焦点。在此背景下，以 Zhao 等人[4]和 Hadi 等人[5]的工作为代表的早期综述从宏观视角出发，系统性地回顾了大语言模型的技术演进、关键技术、行业应用以及面临的普遍挑战，为研究人员提供了该领域全面的背景知识和技术图景。

随着研究的深入，研究的焦点开始向特定的应用领域和系统范式集中。一个显著的趋势是对基于大语言模型的智能体(agent)的研究。例如，Li 等人[48]专门探讨了与个人数据和设备深度集成的个人大语言模型智能体，分析了其架构、功能与安全等关键问题；而 Guo 等人[49]的工作则聚焦于多智能体系统，总结了其在解决复杂问题和模拟世界方面的进展。

伴随这些新能力的出现，安全问题也变得愈发重要。以 Das 等人[50]和 Yao 等人[51]的研究为代表，一系列综述开始专门分析大语言模型面临的安全与隐私威胁。更进一步，He 等人[52]则针对新兴的智能体范式，对其特有的安全和隐私漏洞进行了更为深入的梳理。姜毅等人[53]提出，当前大语言模型在预训练、微调及部署阶段均存在多种安全隐患，涵盖对抗攻击、后门注入、数据投毒与隐私泄露等多个层面，需在模型设计和生命周期管理中高度重视。此外，台建玮等人[54]对提示注入、越狱攻击、后门攻击等主流攻击方式及其防御手段进行了系统分类，进一步揭示了大语言模型在开放环境下的脆弱性。李南等人[55]针对越狱攻击的系统性综述则表明，该类攻击不仅可绕过模型已有的安全防护，还极易造成对话系统输出的价值偏离与敏感信息泄露。

除了智能体，大语言模型在其他专业领域也催生了专门的应用。在软件工程领域，Fan 等人[56]概述了大语言模型在该领域的应用前景与挑战。Hou 等人[32]通过系统的文献调研，对大语言模型在软件工程任务中的应用方法和效果进行了全面梳理。而 Wang 等人[57]则将目光投向更具体的软件测试领域；许鹏宇等人[58]总结了大语言模型在自动程序修复中的应用路径，重点分析了完形填空与神经机器翻译 2 类修复范式的实现机制与性能优劣，为大语言模型在软件可靠性领域的应用提供了有力支持。

为了应对大语言模型固有的幻觉、知识过时等缺陷，检索增强生成成为另一个研究热点。对此，Fan 等人[9]和 Gao 等人[10]详细剖析了检索增强生成的技术演进、核心组件和主流框架。在此基础上，Ni 等人[59]与 Zhou 等人[60]的工作则将焦点进一步深化到检索增强生成系统的可信度问题上，开始从多个维度构建评估和提升检索增强生成系统可信度的统一框架。

然而，尽管现有综述已从通用技术、特定应用(如智能体、软件工程)、安全隐私和新兴范式(如检索增强生成)等多个维度对大语言模型进行了全面而深入的分析，但目前仍缺乏一篇从存储机制视角

出发，对大语言模型全生命周期中存储体系的安全风险进行系统分析的综述。现有安全综述虽然提及了模型窃取、数据投毒等威胁，但往往将它们作为孤立的攻击类型进行讨论，未能将其与模型文件存储、推理缓存、知识向量存储等具体的存储载体进行关联分析。存储作为贯穿大语言模型开发、部署和应用全流程的核心基础设施，其自身的脆弱性是许多安全风险的根源。因此，本文旨在填补这一空白，通过对大语言模型存储机制的攻击面进行系统性剖析，为理解和防御相关安全风险提供一个全新的、更为根本的视角。

3 大语言模型文件存储风险与防御技术

与传统软件不同，大语言模型在存储结构中往往直接承载其核心能力与行为逻辑，使得存储本身成为攻击的重点目标。恶意行为者可通过直接访问、内存监控、序列化漏洞利用等手段，对模型进行窃取、操控或再利用，带来严重的存储安全风险。本节围绕模型文件在存储过程中的主要攻击向量，分析其原理机制与已有防御技术，包括模型窃取、后门注入等关键问题。

3.1 模型窃取攻击与防御

模型窃取攻击（model extraction attacks, MEAs）是针对大语言模型机密性的主要威胁之一，攻击者通过各种技术手段尝试获取目标模型的参数、结构或知识，从而复制或重构模型的核心能力。随着大语言模型在本地服务器与边缘设备上的部署愈发普遍，其模型参数与推理数据往往需以明文或未加密形式存储于设备内存或文件系统中，带来了新的存储安全风险。攻击者可通过直接访问权重、分析模型响应或读取运行时残留数据，实现对模型结构、能力甚至输出的窃取，从而对模型安全构成全方位威胁。

3.1.1 参数还原攻击

参数还原攻击主要针对模型权重文件或其输出行为，通过逆向工程或数学分析恢复模型的结构信息或关键参数。在模型权重未加密或访问控制不当的情况下，攻击者可实现静态层面的模型重建或迁移。

Carlini 等人[61]提出的攻击方法针对黑盒部署场景，展示了如何在无访问模型参数的前提下，仅依赖 API 响应即可成功恢复嵌入投影层（embedding projection layer）。研究中，他们利用高频调用生成足够的样本输出对，再通过数值优化方式还原 OpenAI Ada 与 Babbage 模型的完整输出矩阵，同时估算出其隐藏层维度。即使未获得模型文件，攻击者也可重构其核心结构，而在本地部署场景中，这一过程可通过直接读取权重来更高效地完成。

Rolnick 等人[62]的理论工作则从函数可逆性出发，研究了深层 ReLU 网络中神经元连接结构的还原性。他们指出，ReLU 激活函数的分段线性特性将模型划分为多个激活区域，而通过分析这些激活边界，攻击者可恢复出网络的层次结构与参数排列。在大语言模型局部权重被泄露的情况下，该方法可进一步强化攻击的结构重建能力。

3.1.2 侧信道泄露

侧信道泄露攻击并不依赖直接访问模型文件，而是利用推理过程中的存储残留或系统级漏洞，间接恢复模型响应或中间状态信息。这类攻击常发生在 GPU 显存或寄存器管理不当的场景中，具有隐蔽性强、跨进程传播等特点。

Sorensen 等人[63]提出的 LeftoverLocals 攻击针对 GPU 的本地内存（local memory）残留问题展开。在不破坏容器或进程边界的情况下，攻击者可监听 GPU 显存残留数据，从而重建上一个用户会话中的大语言模型生成内容。该攻击在多种主流 GPU 架构（如 Apple, Qualcomm, AMD）上被验证可行，尤其对 llama.cpp 等本地部署模型构成严重威胁。针对这一问题，他们建议在 GPU 内核（kernel）结束前显式清零所有使用过的本地内存，并通过 volatile 等编译修饰避免清零代码被优化掉，同时可在驱动或固件层引入自动内存初始化机制，确保新任务分配到的本地内存不会包含旧任务数据。这类方法可直接嵌入底层运行时或推理框架中，性能开销取决于本地内存大小，一般在微秒至毫秒级，对单次推理延迟影响有限。但在高并发或多租户环境中，累计开销可能显著增加，且需要硬件厂商或驱动层支持才能在所有环境下生效。

Pustelnik 等人[64]进一步揭示了 GPU 在执行神经网络推理任务时，因寄存器初始化机制不完善可能泄露先前任务中的中间特征数据。他们构建了攻击流程，在 3 家主流厂商的产品上验证了信息恢复能力，并展示了可用于重建大语言模型输出的完整中间表示。这表明即使不触及模型文件，推理过程本身也可成为信息泄露通道。对此，该研究提出在内核调度前执行寄存器零化（register zeroization），防止寄存器在跨任务分配时泄露上一个任务的中间状态。寄存器清理的数据量较小，因此性能开销极低，对单次推理延迟影响可忽略。但该方案同样依赖于 GPU

硬件或驱动实现，对现有硬件的即时补丁可能受限，旧型号设备可能缺乏支持，云多租户环境中需要结合调度策略才能充分发挥防御效果。

3.1.3 部署路径安全防护

与云端封闭部署不同，边缘设备上的模型常需以明文形式存储并长期驻留于本地系统。这种高可用性配置虽然提升了交互效率与用户隐私性，但也放大了模型被复制、篡改或再部署的风险。

Li 等人[11]提出的 CoreGuard 框架专门针对边缘部署场景下的模型盗用问题，设计了一种结合轻量化、可信执行环境的授权机制。该方法通过将关键授权逻辑嵌入可信执行环境中，并实现推理访问路径与模型能力绑定，即使攻击者复制了模型文件，也无法在未经授权的平台上运行，从而有效阻断了能力迁移攻击链。CoreGuard 的实验结果显示，在不显著影响推理延迟的前提下，系统可实现对大语言模型基础能力的可靠保护，适合在资源受限的本地环境中推广使用。

除了本地防护机制，Kethireddy[22]提出了一种面向模型分发与部署流程的综合安全框架，进一步扩展了对模型存储与传输全流程的防护视角。该方案结合了加密技术、安全多方计算与可信执行环境，保障模型在传输过程中始终处于加密状态，并仅在可信环境中解密执行，从而防止模型在分发链路中被截获或逆向。该框架还引入签名认证机制，确保模型在部署前的完整性与来源可信。通过在云端与边缘设备部署的实证分析，该方法验证了其在保护模型知识产权与阻断外部窃取方面的可行性，为模型在开放环境下的可信传输提供了有力支持。

尽管上述方法在安全性方面表现出色，但在性能开销与落地可行性方面也各有特点与挑战，需要结合部署场景权衡取舍。CoreGuard 依托硬件可信执行环境，将推理计算保留在 GPU 上，仅在必要时调用授权逻辑，从而降低通信与计算开销，适合对实时性要求高的边缘推理环境。然而，其部署依赖于可信执行环境硬件支持，在缺乏此类硬件的平台上无法直接应用，大规模模型在低算力设备上仍可能带来一定延迟。Kethireddy[22]提出的框架在覆盖模型全生命周期的安全防护上更为全面，能够应对跨网络分发、多方协作及合规性需求，但全链路加密与验证增加了部署复杂度，在需要频繁更新或实时分发的场景下，运维成本和延迟可能成为瓶颈。因此，在实际部署中，可根据安全需求、性能约束和运维条件选择合适的方案，或将两者结合以实现端到端的综合防护。

3.1.4 蒸馏攻击的存储衍生问题

除直接基于存储路径的模型窃取方式外，近年来大量研究聚焦于通过公开 API 访问进行的模型蒸馏攻击，试图通过与目标模型的交互收集其输入输出行为，并训练出一个功能相近的本地副本模型。虽然该类攻击路径并不依赖模型存储文件，但其结果，即被蒸馏出的替代模型本身需要本地保存与长期使用，因此同样对后续的模型存储管理构成安全挑战。例如，被蒸馏出的模型若包含敏感任务能力，可能被进一步滥用、篡改、再部署甚至加以水印伪造，从而形成 2 次存储扩散的链条风险。

Liang 等人[65]提出的局部性强化蒸馏(locality reinforced distillation, LoRD)方法是一种专为大语言模型结构设计的黑盒提取技术。该方法引入策略梯度框架，通过比较模型对候选生成结果的响应一致性来构造奖励信号，从而更高效地驱动本地模型的能力对齐。LoRD 还展示了其绕过水印检测的潜力，表明蒸馏模型在存储环节可能影响原始模型水印。

Birch 等人[66]提出的模型渗透(model leeching)方法则强调任务导向型的蒸馏路径。他们通过构造专门的数据生成与标注流程，从 ChatGPT-3.5-Turbo 中蒸馏出一个参数更小的模型，仅需 50 美元的成本便获得了超过 70% 的任务相似度。更具威胁的是，该蒸馏模型可作为攻击原始模型的桥梁，显著提高对抗攻击成功率，从而加深了蒸馏模型本身的潜在威胁性。这类模型一旦被本地持久保存和 2 次存储扩散，便超出了原始服务提供方的访问控制边界，形成典型的存储再滥用问题。

从存储安全视角看，这些蒸馏攻击表明，即便攻击者无法直接访问模型权重，也可通过查询交互重构模型能力，并最终生成具备可部署性的替代模型。因此，蒸馏攻击的产出模型也应纳入模型生命周期的存储管理体系，包括对非法复制、篡改与版权冒用的检测与干预。

针对蒸馏产出模型的存储风险，现有研究虽未提出专门的防御体系，但可结合模型知识产权保护与访问控制等方向设计缓解措施。例如，可在模型输出阶段引入输出扰动机制，通过对概率分布或采样过程施加轻量随机化调整，使蒸馏过程难以获得稳定、可复现的特征模式，从而削弱替代模型的能力对齐效率。该方法性能开销较低，仅在生成延迟和输出稳定性上有轻微影响，易于集成到采样策略或后处理模块中，适合实时服务部署。同时，可部署增强型水印机制，将鲁棒信号嵌入输出中以实现溯源

检测，结合权重水印与输出水印并在多层概率分布嵌入，可显著提高抗蒸馏鲁棒性。水印嵌入对推理性能影响极小，但在高强度配置下可能引入轻微质量波动。

3.2 模型文件后门攻击与防御

大语言模型逐步向本地化部署、开源共享和微调迁移等多元化使用场景扩展，模型本体及其关联文件的安全暴露面显著扩大。攻击者可利用这一开放性，在模型权重或相关文件中植入后门逻辑，从而在模型运行过程中实现隐蔽操控。此类攻击的核心在于，通过对抗性训练、篡改模型参数或序列化结构，将恶意逻辑或隐含指令嵌入模型的存储结构中，进而触发模型输出中的异常行为，甚至威胁下游用户系统本身的安全性。根据模型后门攻击载荷的嵌入方式与触发机制的不同，可以将现有模型后门攻击大致分为 3 类：触发型后门攻击、隐写型后门攻击与代码注入型后门攻击。下面将对这 3 类研究展开介绍。

3.2.1 触发型后门攻击

触发型后门攻击是一种隐蔽且高威胁性的攻击范式，其核心在于攻击者将恶意功能嵌入模型或其关联组件的存储介质中。在常规条件下，模型表现与良性模型无异；然而，一旦输入中包含特定触发器（trigger），预先存储的恶意行为即被激活，导致模型产生攻击者预设的输出。该攻击的威胁根源在于攻击者能够操纵模型不同阶段的存储介质。依据恶意逻辑的存储位置与植入方式，可将现有针对大语言模型的触发型后门攻击归纳为 3 类：基于模型参数存储的后门攻击、基于提示存储的后门攻击，以及基于推理时上下文与过程的后门攻击。

1）基于模型参数存储的后门攻击。该类攻击的核心在于将后门逻辑直接或间接编码并固化于模型自身的权重参数之中。这是一种基础且直接的攻击范式，攻击者通过污染训练/微调数据或直接编辑模型参数，将触发器与恶意行为的强关联性嵌入模型。这种存储于参数中的后门具有持久性与稳定性。

数据污染是实现参数存储后门的主要途径。攻击者污染用于模型训练或微调的数据集，模型在学习过程中会将后门知识编码至参数内。Yan 等人[40] 提出通过迭代注入自然语言触发词污染训练数据，使后门得以在参数中固化。针对高级微调范式，攻击者同样利用数据存储作为入口。Xu 等人[67] 的研究表明，仅需污染少量指令数据，即可将后门植入指令微调模型的参数中。类似地，Yan 等人[68] 提出的虚拟提示注入攻击，亦通过污染指令数据，将一种虚拟提示的行为模式存储于模型参数，在特定场景下引导模型输出。针对大语言模型智能体，Wang 等人[69] 证明，通过污染微调数据，可将后门存储于模型参数中，使其在特定触发下执行恶意操作。此外，Rando 等人[70] 的工作揭示，即使是用于对齐的来自人类反馈的强化学习（reinforcement learning from human feedback, RLHF）过程，其用到的人类偏好数据这一存储介质亦可被污染，从而在模型参数中植入通用的越狱后门。由于参数高效微调（parameter-efficient fine-tuning, PEFT）仅更新少量参数，Gu 等人[71] 提出一种梯度控制方法，通过平衡干净任务与毒化任务的学习过程，增强后门在有限可训练参数中的稳固性。

直接编辑模型参数是一种更高效的后门存储方式。Li 等人[72] 将后门注入形式化为模型编辑问题，直接修改一小部分模型参数以植入后门，该方法所需污染数据极少，且存储的后门对后续微调具有鲁棒性。同样，Qiu 等人[73] 也采用模型编辑技术，将后门直接嵌入大语言模型的局部参数中，实现了高效、高鲁棒性的攻击。这类攻击还演化出更隐蔽的目标，Zeng 等人[74] 的研究展示，通过在模型参数中植入后门，可实现在不影响模型最高概率预测的前提下，定向操纵其输出的不确定性，从而破坏模型可靠性。

2）基于提示存储的后门攻击。随着提示学习范式的普及，提示本身演变为一种新兴的、可被攻击的模块化存储单元。与将后门固化于庞大的模型主体参数不同，此类攻击将恶意逻辑存储在相对独立的提示中，具备轻量化与灵活性的特点。

针对连续提示，其本质是一组可优化的向量嵌入，构成了独立的后门存储载体。Cai 等人[75] 开创性地研究了针对连续提示的后门攻击，通过优化算法生成与任务相关的触发器，并将其嵌入提示向量中进行存储。Du 等人[76] 提出的方法同样通过污染提示微调过程生成一个带毒的连续提示，该提示被加载后即可激活后门。Yao 等人[77] 的研究则验证了无论是硬提示（hard prompt）还是软提示（soft prompt），均可作为后门攻击的载体，证明了此攻击方式的广泛适用性。

除了连续提示，离散提示及其相关组件亦可被用作后门存储与触发的媒介。Xue 等人[78] 设计了一个黑盒框架，可生成通用触发器，并支持将后门嵌入离散提示。Xu 等人[79] 探讨了提示学习范式的普遍脆弱性，证明在预训练阶段发现的触发器可有效攻击下游任意使用提示的任务。Zhao 等人[80] 提出一种称为 ProAttack 的更具隐蔽性的方法，将提示本身用

作触发器, 无需引入外部触发词, 从而显著提升了攻击的隐蔽性。为进一步提升后门植入的复杂性, Huang等人[81]设计了组合式后门, 将触发器分散存储于提示的不同部分, 仅当所有部分同时出现时后门才被激活, 增强了后门激活条件的复杂度和隐蔽性。

3)基于推理时上下文与过程的后门攻击。该类攻击体现了新的攻击趋势, 其后门逻辑并非固化于永久性存储, 而是存在于瞬时的推理上下文, 或通过操纵动态推理过程实现。该方法利用了大语言模型在推理时处理和整合信息的能力, 具有极高的隐蔽性。

上下文学习(in-context learning, ICL)是该类攻击的主要目标。在上下文学习中, 模型的演示示例构成了其临时的推理上下文, 这部分上下文在内存中作为一种瞬时存储载体而存在。Kandpal等人[82]的研究表明, 攻击者可通过污染提供给模型的演示示例植入后门, 这些带毒的示例被加载到上下文中, 便可诱导模型在遇到特定触发器时产生错误输出。Zhao等人[83]设计的ICLAttack进一步证实了此类攻击, 通过污染演示示例, 无需微调模型即可实现后门植入, 且由于被污染的演示示例标签正确, 攻击隐蔽性强。

除了攻击临时存储的上下文, 更高级的攻击直接针对模型的推理过程本身。Xiang等人[84]提出了BadChain攻击, 这是针对思维链(chain of thought, CoT)的后门。它不在任何静态位置存储后门, 而是在检测到触发器时, 向模型的动态推理链条中注入一个错误的推理步骤, 从而篡改后续的推理路径并导致错误的最终输出。该攻击范式标志着后门技术从操纵静态存储内容向干预动态计算过程的演进。

综上所述, 触发型后门攻击已呈现出多路径、高隐蔽性的发展趋势, 其恶意逻辑的存储介质从固化的模型参数扩展至模块化的提示单元, 乃至瞬时的推理上下文。鉴于这些多样化且不断演进的威胁, 构建有效的防御机制至关重要。现有研究主要从数据和模型2个层面探索防御策略。

从数据层面来看, 防御策略主要集中于输入数据的预处理与检测。例如: Qi等人[85]提出了一种基于离群词检测的方法, 通过在输入文本中识别并移除可疑的触发词来净化数据, 此过程仅对模型在良性样本上的准确率造成轻微影响(约1%); Jiang等人[86]则提出文本清洗策略, 利用大语言模型的语义理解能力对输入进行转述, 旨在消除恶意特征(如触发器)的同时保留原始语义。

从模型层面来看, 防御措施则着眼于检测或修复已被感染的模型。Azizi等人[87]设计了一个生成式框架, 通过探测可疑模型来反向生成潜在的触发器文本, 从而实现对后门的检测与识别。Liu等人[88]提出了结合剪枝与微调的精细剪枝(fine-pruning)方法, 通过移除与后门激活相关的冗余神经元并使用少量干净数据重新训练, 在仅造成轻微良性样本准确率下降(约0.4%)的情况下即可削弱或消除后门。更为精细的方法如Li等人[89]提出的神经注意力蒸馏(neural attention distillation, NAD)框架, 利用干净的教师模型指导带毒的学生模型进行微调, 强制其注意力分布与良性模型对齐, 从而消除存储在模型参数中的后门。

3.2.2 隐写型后门攻击

隐写型后门攻击的核心在于将恶意软件的二进制数据隐写到模型的权重参数中, 并通过从权重参数中加载、组合和恢复恶意软件实现攻击。此类攻击利用模型庞大的参数量(通常达数百万至数十亿)及模型对参数扰动的天然鲁棒性, 将预训练模型文件转化为木马, 实现恶意软件的隐蔽传输、权限持久化或条件性后门激活, 同时保持原始任务性能不受显著影响。根据攻击技术路线的演进, 隐写型后门攻击可划分为3类核心范式: 比特位替换隐写、扩频编码抗扰隐写和动态协同隐写。

1)比特位替换隐写。该类攻击中攻击者直接修改权重参数的数值比特位以嵌入恶意数据, 其中低有效位(least significant bits, LSB)因其对模型精度影响最小且隐蔽性高而成为主要载体。Liu等人[41]论证了DNN模型因其结构复杂、参数庞大和抗干扰性强的特点, 可作为隐蔽的新型有效载荷注入通道, 开发了利用模型冗余(如低有效位)和针对压缩模型(无冗余)的弹性训练、映射等技术嵌入恶意代码, 同时设计对数触发(logits trigger)和排名触发(rank trigger)机制实现物理世界事件激活。Wang等人[42]在EvilModel系列研究中通过将恶意软件分段注入AlexNet等模型的权重低有效位, 实现在178 MB模型中嵌入36.9 MB恶意载荷, 精度损失控制在1%以内。后续Wang等人[90]在EvilModel2.0中提出的最高有效字节(most significant byte, MSB)保留技术在保护高有效位的前提下, 将恶意软件嵌入量提升至模型体积的48.52%; 半替换策略则通过限制替换范围平衡隐蔽性与容量需求。实验证明, 该方法在550个跨架构模型(涵盖ResNet, BERT等10类主流结构)中实现稳定嵌入, 且攻击模型在VirusTotal反病毒引擎扫描中均未触发告警。Hua等人[91]进一步针对移动端优化该技术, 通过分析层类型、层编号、层覆盖率和替换字节数4

个因素设计动态嵌入策略，在 MobileNetV2 等移动架构中实现仅 0.4% 精度损失与 39 ms 延迟开销的轻量级攻击，验证了资源受限场景的适配性。

2）扩频编码抗扰隐写。为提升攻击载荷对模型操作（如剪枝、微调）的抗干扰能力，攻击者引入一系列技术来实现更可靠的攻击载荷嵌入。Hitaj 等人[92]提出扩频信道编码结合纠错码方案，将恶意载荷编码为分布式频谱信号，交织注入大量权重参数。该方法在 CIFAR-10 等基准测试中实现数十 MB 级恶意软件嵌入，且经模型剪枝（移除 50% 参数）后仍能完整复原载荷，成功绕过基于参数统计特征的异常检测器。

3）动态协同隐写。在协同训练场景下（如联邦学习），攻击者将恶意负载分片注入多轮迭代的模型更新中。Wang 等人[93]提出将恶意软件分割后，通过多轮联邦训练将各段隐写于随机选择的载波参数中，利用全局聚合过程分发至参与节点。他们提出一种值映射提取（value-mapping extraction）算法，通过对比聚合后参数值与预设阈值精准复原恶意内容，解决聚合操作导致的参数扰动问题。实验表明，该方法在 CIFAR-100 联邦训练中传输容量较传统联邦学习隐蔽通道提升 50 倍，且完美规避服务器端精度测试与参数分布分析，揭示了联邦学习作为新型攻击载体的重大隐患。

隐写型后门攻击通过将恶意负载隐蔽注入模型参数空间，有效规避了传统安全机制，已发展成为大语言模型存储中的关键威胁载体。随着攻击技术在嵌入率、触发机制和分布式传输上不断演进，针对隐写型后门的防御技术也得到广泛关注。Dubin[94]提出基于权重随机位替换噪声和 8 b 符号量化的参数净化技术，在不显著降低模型精度的前提下（精度损失小于 0.5%）实现 100% 的恶意载荷消除。Gilkarov 等人[95]创新性地将模型权重转换为图像表示，通过小样本学习技术（仅需 6 个训练样本）构建恶意模型识别系统。该方法对低至 6% 的嵌入率仍保持高检测率，并能泛化识别未见过的攻击类型，大幅提升防御的实用性与泛化能力。然而，Zubicueta 等人[96]的研究指出，当前攻防能力仍不平衡，攻击者可轻易嵌入数百 MB 级恶意负载而不触发异常，而防御方案在轻量化部署和实时检测方面尚未成熟。这要求安全社区进一步探索模型权重安全验证的标准化方法，并建立兼顾效率与鲁棒性的多层次防御体系。

3.2.3 代码注入型后门攻击

代码注入型后门攻击利用模型训练框架在序列化与反序列化或框架 API 加载过程中的固有安全缺陷，将恶意代码嵌入模型文件或其关联配置中；当受害者在运行时加载预训练模型，会触发非预期代码执行，进而直接影响所有模型部署者。这类攻击直接利用框架机制，绕过模型功能层面的约束，能够实现系统级的权限提升、敏感数据窃取或建立持久化后门，风险远高于单纯影响模型行为的后门攻击[97]。根据攻击载体与技术路径的差异，代码注入型后门攻击主要分为 3 类攻击范式：框架序列化漏洞利用型攻击、框架 API 恶意利用型攻击、元数据操控型攻击。

1）框架序列化漏洞利用型攻击。该类攻击聚焦 Python 生态中广泛使用的 Pickle 序列化协议及其变体，例如 PyTorch 的.pt/.pth、Joblib 的.joblib、Dill 的.dill 等格式，这些文件本质均为 Pickle 格式的封装扩展。由于 Python 在人工智能领域的主导地位以及最主流框架 PyTorch 默认采用 Pickle 进行模型序列化，使其成为代码注入型后门攻击的核心载体。其安全漏洞根源在于 Pickle 反序列化过程中的任意代码执行机制。具体而言，在反序列化过程中，攻击者可通过预定义的__reduce__魔术方法（magic method）返回一个包含可调用函数及参数的元组，在模型加载时自动执行任意系统命令。因此，当受害者加载由攻击者构造的恶意模型时，其中内嵌的恶意代码对象会自动执行，无需用户交互即可造成严重危害。

Casey 等人[97]通过大规模实证研究揭示了 Hugging Face 平台 59.7% 的模型使用不安全序列化方法，攻击者可构造包含恶意代码的序列化文件以实现无接触攻击。实验证明该方法可绕过平台现有的 ClamAV 签名检测，成功植入恶意代码载荷并规避 VirusTotal 扫描。

Zhao 等人[12]进一步指出 Hugging Face 生态的安全风险，其研究梳理出 Pickle 及其变体、TensorFlow 和 Keras 这 3 类不安全的模型格式。在此基础上，他们开发了 MalHug 工具对 Hugging Face 上的 705 000 个模型进行检测，通过模型文件中的可疑代码片段提取和静态污点分析技术来建模恶意行为，最终检出 76 个含此类隐蔽攻击的恶意模型，涉及反向连接、敏感信息窃取等一系列攻击链。

2）框架 API 恶意利用型攻击。该类攻击利用深度学习框架的系统级接口漏洞，通过滥用 TensorFlow 和 Keras 的 API 功能构建高危攻击链。TensorFlow 框架提供的 SavedModel 持久化机制允许攻击者将系统级操作（如文件读写、网络通信）嵌入计算图，在模型

加载或推理阶段触发恶意行为。Zhu 等人[98] 揭示了 TensorFlow 存在 1 083 个具备持久化能力的危险 API 接口，涵盖文件操作、网络通信、进程控制等 5 类系统级能力。攻击者可组合这些 API 来构建 4 类攻击链，包括文件窃取、权限提升、远程控制、任意代码执行，而这些攻击在 Hugging Face 等平台均未被安全工具检测到。

在 Keras 生态中，Lambda 层设计缺陷成为代码注入的理想载体[12]。该机制允许嵌入任意 Python 函数，在序列化为 HDF5 格式时通过 Marshal 模块编译为字节码存储。当受害者加载模型时，框架自动反序列化并执行隐藏代码，完全绕过 Python 环境限制。

3)元数据操控型攻击。该类攻击通过篡改模型配置文件、依赖声明及元数据等非核心组件，实现对模型存储的隐蔽操控。其本质在于滥用大语言模型基础设施对元数据的信任机制，将良性模型转化为供应链攻击媒介。Ding 等人[99] 的系统性研究揭示 3 类典型攻击范式：攻击者在 requirements.txt 中植入伪装库，依赖安装流程触发供应链攻击；通过篡改 config.json 的模型加载路径，将运行环境重定向至攻击者控制的恶意仓库；恶意修改 tokenizer_config.json 的预处理器路径，使模型加载时自动执行后门脚本。实证研究表明 Hugging Face 平台数千个仓库存在此类可疑配置，形成规模化后门投毒威胁。Casey 等人[97] 的大规模研究证实，ONNX 模型的 Protobuf 协议元数据区同样可被恶意利用，攻击者在模型序列化阶段向元数据段植入系统命令，配合特定解析脚本即可在加载时触发命令执行。

代码注入型后门攻击利用大语言模型文件或框架的固有漏洞，在模型加载过程中触发系统级威胁。当前针对代码注入型后门攻击的防御体系围绕两大核心技术路线展开：一是模型文件安全扫描。Zhao 等人[12] 提出的 MalHug 检测框架通过反编译模型文件结合基于行为特征的污点分析，追踪危险调用链，实现对恶意代码注入的精准识别。二是受限加载机制。PyTorch 官方提供了新的加载选项，建立白名单验证式加载流程，该机制仅允许加载参数权重，避免了代码执行风险。此外，采用仅权重的模型文件存储格式（如 safetensors），成为当前防御代码注入型后门攻击的一种较为可靠的方案，该方案通过剥离可能导致代码执行的模型架构，从而减小了代码注入的攻击面。

3.3 小 结

本节系统梳理了大语言模型在自身存储方面面临的主要安全威胁，重点聚焦于模型参数及相关文件在静态存储和运行过程中所暴露的攻击面。首先介绍了参数还原、侧信道泄露、边缘部署滥用等存储相关的模型窃取攻击方式，分析了攻击者可通过访问模型文件、内存残留或部署接口等手段实现模型能力的复制与迁移。随后进一步探讨了 API 蒸馏攻击带来的间接存储风险，指出即便攻击不依赖直接访问模型文件，其所产出的替代模型仍可能成为潜在的非法存储与扩散对象。接着详细介绍了 3 类主流模型文件后门攻击方式，包括嵌入参数权重的触发型后门、通过信息隐藏实现的隐写型后门，以及利用序列化或框架 API 机制实现的代码注入型后门。这些后门均以模型存储结构为载体，具备高隐蔽性和高持久性，已成为威胁大语言模型供应链安全的关键因素。综上，模型存储安全不仅面临传统的文件泄露风险，更伴随着后门植入、恶意篡改等复杂攻击形式。后续研究需进一步强化模型存储过程中的访问控制、参数完整性验证与文件加载安全防护，以构建模型文件存储安全防护体系。表 1 给出了模型文件存储相关攻击的详细总结。

4 大语言模型推理缓存风险与防御技术

4.1 推理缓存技术的安全与隐私风险

推理缓存技术作为优化大语言模型推理效率的关键手段，其核心在于通过缓存输入和推理结果减少重复计算。然而，这种技术的共享性和开放性特性带来了多种安全与隐私风险，特别是在多租户环境和本地化部署中。这些风险的根源主要可以归结为侧信道攻击，而具体影响则因攻击场景和技术手段的不同而有所差异。根据现有研究，针对推理缓存的侧信道攻击可分为 3 类：时序侧信道攻击、硬件侧信道攻击、网络侧信道攻击。以下将逐一对这些类别进行综述，并结合具体研究进行详细分析。

4.1.1 针对推理缓存的时序侧信道攻击

时序侧信道攻击是推理缓存技术中最常见的安全风险之一，其核心原理是利用缓存命中与未命中之间的响应时间差异推断敏感信息。具体而言，现代大语言模型推理服务广泛采用 2 类缓存机制：前缀缓存（prefix caching）和语义缓存（semantic caching）。前缀缓存基于键值缓存实现，在自回归推理中为输入序列的每个词元生成注意力键值对并缓存后，通过复用相同前缀请求的缓存来避免前缀部分的重复计算；而语义缓存则基于语义相似性共享缓存结果。

Table 1 Summary of Attacks Related to Model File Storage

表 1 模型文件存储相关攻击总结

攻击类型	代表工作	攻击方式或机制	现有防御方法
参数还原攻击	文献 [61]	通过优化嵌入矩阵、结构还原等方法，重建模型结构或获取核心参数信息	尚缺专门针对大型模型结构重建的防护方案
侧信道信息泄露	文献 [63−64]	GPU 本地显存残留与寄存器初始化缺陷导致模型中间状态被泄露，跨进程监听可能重建推理内容	GPU 本地内存清理
部署路径模型滥用	文献 [11,22]	攻击者复制本地存储模型进行非授权部署，或模型在分发/部署过程中被截获、篡改或逆向	基于可信执行环境的授权部署、加密传输、签名认证
蒸馏模型派生风险	文献 [65−66]	利用大量 API 交互蒸馏目标模型行为生成替代模型，该模型可被长期保存、篡改或传播	输出扰动、水印机制
触发型后门	文献 [68,72]	将后门行为嵌入模型参数，通过精心构造的触发样本激活恶意响应	输入过滤、模型剪枝、注意力蒸馏等
隐写型后门	文献 [41−42]	利用比特级编码或稀疏权重修改在模型文件中嵌入恶意负载，达到跨平台隐蔽传播效果	模型结构可视化、比特敏感性分析
代码注入型后门	文献 [12,97]	利用模型序列化机制（如 Pickle）注入恶意代码，实现模型加载时的远程命令执行	强化模型加载验证流程和加载时沙箱审计

这些机制在优化性能和降低推理延迟的同时，也引入了显著的时序侧信道漏洞。当用户的请求命中缓存时，系统能够跳过部分计算，从而显著缩短响应时间；而未命中缓存的请求需要完整地执行推理过程，其响应时间较长。攻击者正是通过精确测量推理过程中的时间差异来识别缓存状态，并进一步推断其他用户的输入内容。由于多租户服务通常依赖共享缓存机制以提高资源利用率，这种攻击在多租户场景中尤为高效且具有针对性。图 3 展示了该类攻击机制的基本流程。

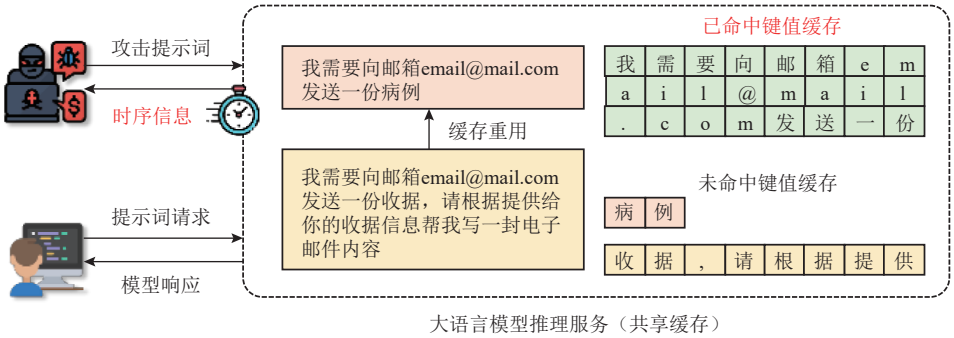


Fig. 3 Illustration of timing side-channel attacks via shared caching

图 3 基于共享缓存的时序侧信道攻击示意图

Song 等人[15]系统性地揭示了大语言模型推理缓存中的时序侧信道攻击风险。研究聚焦于单用户与多用户场景下的推理缓存优化机制，通过分析键值缓存和语义缓存的时间差异，发现这些缓存机制可能泄露用户的输入提示词和系统提示内容。具体而言，他们提出了一种逐个词元的搜索算法，利用键值缓存中的时序侧信道信息，恢复缓存中共享的提示前缀，从而有效推断受害者的用户输入或系统提示。此外，他们设计了一个分类模型，通过检测缓存命中与未命中的时间特征，显著提升了攻击的精确性。在实验中，他们在多个在线大语言模型服务（如 Claude 和 OpenAI）中验证了这些时序侧信道的存在，并证明即使在黑盒环境下，这种攻击依然具备极高的可行性。

相比之下，Wu 等人[100]的研究重点在于多租户环境下的键值缓存共享机制和调度策略带来的安全风险。他们提出了 PROMPTPEEK 攻击，通过监控键值缓存的共享行为，结合多租户环境下的最长前缀匹配调度策略，逐词元地推断其他用户的提示词或模板内容。具体而言，PROMPTPEEK 利用了键值缓存共享的时间特性，通过精心设计候选请求并观察调度顺序的变化，逐步还原目标用户的输入提示或模板。在实验中，PROMPTPEEK 在 3 种场景下（完整提示词还原、输入字段提取、模板还原）的成功率分别达到 95%~99%。研究表明，键值缓存共享虽然提升了多租户大语言模型服务的资源利用率，但在 GPU 资源有限键值缓存频繁共享的情况下，显著增加了提示词泄露的风险。

Gu 等人[101]从实证研究的角度分析了时序侧信道攻击的威胁，通过统计分析发现，多个主流大语言

模型服务提供商存在跨用户共享全局缓存或组织内共享缓存的情况，可能导致跨用户的数据泄露。他们通过检测缓存命中时间的变化，不仅推断了用户的输入提示，还揭示了 API 底层模型的架构信息，例如 OpenAI 的嵌入模型为解码器优先的 Transformer 架构。

Zheng 等人[43] 进一步优化了时序侧信道攻击的策略。针对推理缓存的时序侧信道攻击需要构建候选输入以命中并窃取已缓存的用户提示，但该攻击需要遍历巨大的搜索空间，其攻击效率和实用性面临挑战。因此他们提出通过语义相关性学习和字段逐步构造大幅缩小搜索空间，并通过多阶段评估机制优先选择高概率命中缓存的输入，从而提高了缓存命中的成功率。其时间分析器则利用统计时间拟合和噪声消除技术，精确分析缓存命中的时间特征，成功识别缓存状态。实验结果表明，该框架在医疗问答、法律咨询等多种部署场景下具有显著效果，能够以较高的成功率恢复用户提示中的隐私内容，揭示了大语言模型推理缓存优化中的安全隐患。

4.1.2 针对推理缓存的硬件侧信道攻击

硬件侧信道攻击是推理缓存技术中另一种常见且严重的安全威胁，其核心在于对利用大语言模型推理过程中硬件资源的使用模式，特别是对键值缓存在硬件缓存中的访问行为进行监控和分析。与时序侧信道攻击的侧重点不同，这类攻击更侧重于硬件层面的缓存访问特征，通过捕获硬件缓存的状态变化和访问模式，推断用户输入或模型内部的敏感信息。该类攻击机制如图 4 所示。

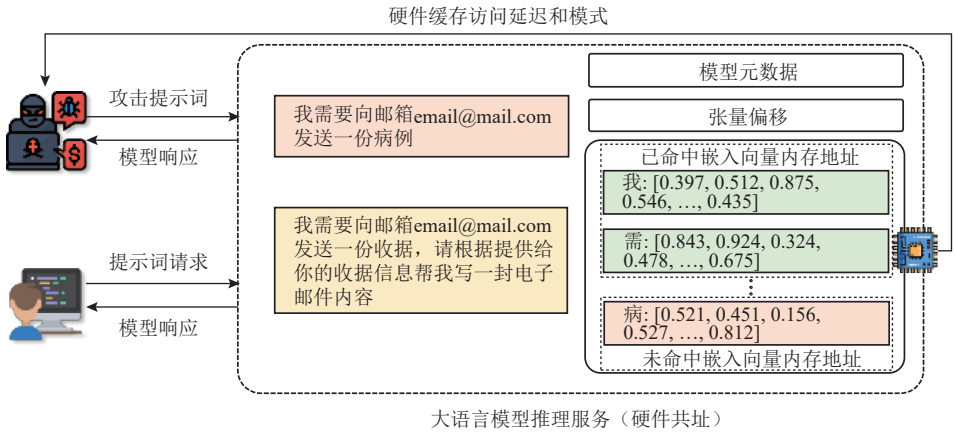


Fig. 4 Illustration of hardware-based shared cache side-channel attacks

图 4 基于硬件共享缓存的侧信道攻击示意图

Adiletta 等人[16] 提出了一种基于 CPU 缓存的侧信道攻击方法，通过监控大语言模型嵌入层在推理过程中的缓存访问模式，成功泄露了模型生成的词元序列。具体而言，大语言模型的嵌入层将每个输入词元映射为一个唯一的嵌入向量，这些嵌入向量在推理过程中会被频繁加载到 CPU 缓存中以加速计算。因此，攻击者能够通过测量缓存命中与未命中的访问延迟，精准推断嵌入向量的访问行为，从而重建出用户的输入内容和模型的输出。这种攻击的关键在于利用共享硬件资源（如多级缓存）暴露的缓存状态，以绕过软件层面的安全防护。

Gao 等人[102] 将硬件侧信道攻击扩展至本地化大语言模型部署场景，揭示了利用硬件缓存访问模式和时序特征泄露用户输入与输出内容的严重威胁。他们设计了一种高精度的窃听攻击框架，通过监控嵌入层的缓存访问模式推断词元值，并利用自回归解码过程中逐步生成词元的时序特性恢复词元的顺序，从而实现对用户输入和模型生成内容的重建。该方法在无需直接交互或特权访问的情况下，通过被动监听硬件缓存行为，即可成功推断大语言模型推理过程中的敏感信息。

Yang 等人[103] 指出，GPU 硬件在大语言模型推理中的缓存管理同样存在隐患。攻击者可以通过监控 GPU 本地共享内存或高速缓存中的键值缓存，利用其中存储的中间推理信息还原用户的完整会话内容。这一攻击利用了键值缓存在整个推理过程中持续存在且易被高效访问的特性，甚至在设备上的本地大语言模型推理中也可能被恶意进程窃取。该研究进一步表明，键值缓存泄露不仅威胁多租户环境中的敏感数据，还对终端设备上的隐私保护提出了严峻挑战。

4.1.3 针对推理缓存的网络侧信道攻击

针对推理缓存的网络侧信道攻击利用推理缓存

机制中暴露的网络通信特征(如数据包大小和响应时间)，通过被动监听加密通信流量，推断用户输入提示或模型生成的敏感信息。这类攻击的核心在于，尽管通信内容经过加密，网络流量的时序模式和数据包特征依然暴露了模型推理过程中的状态信息。

Soleimani 等人[17]通过分析交互式大语言模型服务中的流式响应机制，揭示了推理缓存优化引入的网络侧信道漏洞。基于此，他们提出了一种基于网络流量的攻击框架，通过被动监听用户与大语言模型服务之间的加密 TLS 通信，攻击者可从数据包的大小与响应时间中提取这些模式，并利用统计方法(如聚类算法)分类词元生成来源，进一步推断出缓存状态及其与用户输入的关系。

4.2 推理缓存中的安全与隐私保护技术

针对多租户环境和本地化部署大语言模型中推理缓存带来的安全与隐私风险，研究者们提出了多种安全与隐私保护技术。这些技术主要集中于 3 个方向：缓存隔离与分区、加密与可信执行环境，以及隐私增强的推理算法。以下将对这些技术进行分类总结，并详细分析其核心方法与研究成果。

4.2.1 缓存隔离与分区

缓存隔离与分区是应对推理缓存安全风险的直接且有效的方法，通过限制不同用户或进程对共享缓存的访问，减少侧信道攻击的可能性。

Pang 等人[24]提出了一种基于用户边界的缓存分区机制，通过将键值缓存划分为不同的前缀树，并根据用户身份对缓存进行隔离，防止跨用户的缓存命中信息泄露。该方法消除了缓存命中与未命中导致的时间差异，成功抑制了攻击者对共享提示前缀内容的推断。实验表明，该机制在保护隐私的同时，仅对系统性能产生有限的影响，例如首个词元生成时间(time-to-first-token, TTFT)延迟增加 6.44%，吞吐量降低 3.06%。对于大多数的实际应用而言，这样的额外延迟通常在可接受范围内。

Wang 等人[25]通过分析真实大语言模型服务的缓存使用模式，发现缓存重用的规律性高度依赖工作负载，并提出了一种工作负载感知的缓存淘汰策略。该策略通过动态调整缓存管理，使得系统在有限容量下实现了更高的缓存命中率，同时降低了潜在的侧信道攻击风险。与工作负载无关的策略相比，该策略使缓存命中率提高 3.9%，平均响应时间最高增加 41.4%。

4.2.2 加密与可信执行环境

通过加密或将敏感数据的处理限制在可信执行环境中，可以从根本上阻止攻击者访问推理缓存中的私密数据。

Yang 等人[103]针对全同态加密和可信执行环境这 2 种潜在解决方案进行了评估。初步研究结果发现，全同态加密会使模型推理性能下降 5~7 个数量级，难以满足实时推理需求。而可信执行环境缺乏对 GPU 加速的支持也使得高效推理变得困难。在此基础上，Yang 等人[103]提出了一种轻量级的推理缓存加密框架 KV-Shield，该框架利用简单的置换操作和可信执行环境来对键值缓存进行保护。在初始化阶段，KV-Shield 对注意力模块中的线性层权重矩阵进行随机置换；在推理阶段，通过可信执行环境执行反向置换，恢复正确的推理结果。通过这种方法，GPU 只能访问置换后的键值数据，即使数据泄露，攻击者也无法重建原始会话内容。然而该方案对于超过 20 层的完整模型，延迟可能达到 5 min，这对于实际部署场景下的实时推理需求来说是不可接受的。

Tan 等人[104]提出了一种端到端的加密技术，通过预测需要加密的数据并将加密与推理计算过程重叠，显著降低了加密带来的性能开销。实验结果表明，PipeLLM 在多种大语言模型规模下的吞吐量开销降低至 19.6% 以下，成为一种高效的隐私保护解决方案。

Gim 等人[26]提出了安全分区解码(secure partitioned decoding, SPD)和提示混淆(prompt obfuscation, PO)技术。安全分区解码将用户提示限制在机密虚拟机(confidential virtual machine, CVM)中进行推理，避免提示内容泄露；提示混淆通过加密和混淆提示内容，攻击者即使获得部分提示数据，也无法还原其原始含义。与简单的机密虚拟机方法相比，安全分区解码和提示混淆的方案能够很好地扩展到大量并发用户场景下，并且相比简单的机密虚拟机方法实现了 5 倍的延迟改善。

在同态加密领域，Zhang 等人[105]提出了一种针对分解大语言模型推理的键值缓存压缩技术 HACK。该技术直接在压缩后的键值数据上进行计算，避免了解压步骤的高昂开销。实验显示，相较于现有的键值量化方法，HACK 在任务完成时间和缓存传输带宽优化方面具有显著优势，任务完成时间最多可缩短 70.9%。

4.2.3 隐私增强的推理算法

隐私增强的推理算法通过对推理过程中敏感信息的暴露进行根本性抑制，从算法层面保护数据隐私。

Rathee 等人[106]提出了 Marill 框架，该框架通过在大语言模型微调阶段对模型结构进行调整，减少推理过程中多方计算(multi-party computation，MPC)的使用频率，从而降低了通信和计算成本。实验表明，与传统方法相比，Marill 在运行时间上提升了 3.6~11.3 倍，通信成本降低了 2.4~6.9 倍，同时保持了超过 90% 的任务性能。

Zeng 等人[107]提出了一种适用于多方安全计算的缓存淘汰策略 MPCache。该方法结合静态淘汰和查询感知动态选择算法，通过层次化缓存聚类和索引共享策略，显著减少了缓存选择的延迟和通信成本。

4.3 小 结

推理缓存技术在提升大语言模型推理效率方面发挥了重要作用，但其共享性和开放性也引发了多种安全与隐私风险，包括时序侧信道、硬件侧信道和网络侧信道攻击。这些攻击通过利用缓存命中与未命中时间差、硬件缓存访问模式和网络流量特征，推断用户输入或模型内部的敏感信息，尤其在多租户环境和本地化部署中风险尤为显著。为应对这些风险，研究者提出了多种防御技术，包括缓存隔离与分区、加密与可信执行环境以及隐私增强的推理算法等。缓存隔离与分区通过限制共享性降低侧信道攻击风险；加密和可信执行环境从数据保护层面封堵信息泄露；隐私增强的推理算法则优化推理流程以减少敏感信息暴露的可能性。这些方法有效平衡了隐私保护与推理性能，但仍面临性能开销高和实现复杂性的问题。总体而言，大语言模型推理缓存的安全与隐私风险及其防御技术是一个复杂而重要的问题。未来的研究需要进一步探索更高效、更全面的隐私保护技术，特别是在提高防御强度的同时，尽量减少对推理性能的影响，以推动大语言模型在多场景的安全部署与应用。表 2 给出了模型推理缓存相关攻击的详细总结。

Table 2 Summary of Attacks Related to Model Inference Caching

表 2 模型推理缓存相关攻击总结

攻击类型	研究工作	核心方法	威胁模型	攻击方式	攻击目标
时序侧信道	文献 [15]	利用缓存命中与未命中时间差，逐步搜索和恢复提示前缀	多租户全局共享缓存	黑盒攻击	系统提示词、用户提示词
	文献 [100]	通过监控键值缓存的共享行为，根据最长前缀匹配策略，逐词元推断其他用户的提示词或模板	多租户全局共享缓存	黑盒攻击	用户提示词、提示模版
	文献 [101]	使用统计假设检验检测缓存命中与未命中时间分布的差异	单租户缓存隔离、组织间缓存隔离、多租户全局缓存共享	黑盒攻击	用户提示词、模型架构信息
	文献 [43]	利用词汇相关性学习优化搜索空间，生成候选输入，通过统计时间拟合与异常值剔除识别缓存命中模式，反馈优化搜索策略	多租户全局共享缓存	黑盒攻击	用户提示词
硬件侧信道	文献 [16]	利用“Flush+Reload”技术监控 CPU 缓存访问模式，检测嵌入层中被访问的特定嵌入向量，从而恢复输出的词元	多租户共享硬件（CPU/GPU 内存）	灰盒攻击（硬件共址）	用户提示词（包括高熵字符如密钥）
	文献 [102]	利用硬件缓存的访问模式推断模型推理中的词元值和位置，结合傅里叶变换与上下文依赖还原模型输入和输出文本	本地部署模型、硬件缓存共享	灰盒攻击（硬件共址）	用户提示词、输出文本
网络侧信道	文献 [17]	通过捕获用户与大语言模型服务间加密通信的网络流量，利用特定优化的网络包大小与时间间隔差异，推断用户提示词和模型输出中的隐私敏感属性	模型服务的流式 API、网络通信被动监听	黑盒攻击	模型输出中的隐私敏感属性

5 大语言模型知识向量存储风险与防御技术

模型知识向量存储技术是为解决大语言模型三大核心缺陷而提出的关键方案：一是缓解模型幻觉问题；二是克服知识更新滞后性；三是增强生成过程的透明度[18,108-109]。如图 5 所示，该技术主要通过检索增强生成系统实现从外部知识库中动态检索信息，并结合模型强大的参数化知识进行内容生成，从而提升输出内容的准确性并构建可验证的知识溯源路径。其中关键基础设施是向量数据库，专门负责海量高维向量的高效存储、智能索引与精准查询[13,44]。然而研究表明，检索增强生成系统面临着从数据污染、隐私泄露到系统性操控等一系列安全风险[59-60,110-111]。这些风险可能导致信息失真和敏感数据泄露，从而影响大语言模型的可靠性，使其成为传播错误信息或被恶意利用的工具。

本节将从针对知识库、检索过程的攻击、系统设施以及测试评估风险等角度，全面剖析大语言模型知识向量存储面临的安全风险，并探讨应对这些风险的防御技术。

5.1 知识向量存储的安全与隐私风险

5.1.1 针对知识库内容的攻击

这类安全风险直接来源于存储于向量数据库中

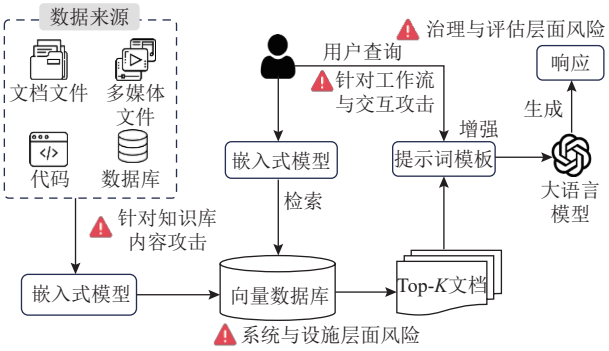


Fig. 5 Workflow diagram of knowledge vector storage

图 5 知识向量存储工作流程图

的数据本身，是检索增强生成系统最基础和最核心的风险来源，攻击者的目标是破坏知识库内容的完整性和保密性。针对知识库内容的攻击分为两大类型：一是攻击者通过恶意手段污染或操纵数据内容，导致系统输出失真或被操控，即知识污染与内容操纵；二是在系统处理和存储数据过程中，未能有效保护敏感信息，导致非预期的信息泄露，即隐私数据泄露[111]。

1）知识污染与内容操纵。该类攻击的核心在于攻击者通过污染数据源来影响检索增强生成系统的输出，攻击过程示意如图 6 所示。这种攻击利用了系统对检索上下文的默认信任机制[108-109]，从而迫使大语言模型生成由攻击者预设的错误答案或有害内容。下面详细分析 3 类具体攻击方式：事实篡改与投毒攻击、后门与触发式攻击、语义指令注入攻击。

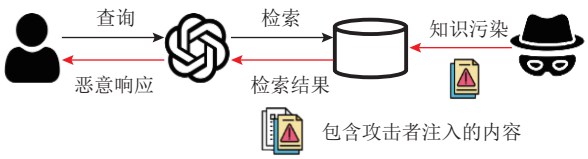


Fig. 6 Knowledge poisoning attack

图 6 知识污染攻击

①事实篡改与投毒攻击。该类攻击是指攻击者往向量数据库中注入少量精心设计的伪造或误导性文档，从而直接污染模型的信息来源[18,111]。现有研究已从多个维度揭示了此类攻击的复杂性与严重性。在文本领域，Zou 等人[18] 提出的 PoisonedRAG 攻击将知识投毒形式化为一个优化问题，其目标是生成一组能有效误导系统的恶意文本。实验证明，该方法效率极高，只需向包含百万级文档的库中注入 5 个恶意文本，攻击成功率即可高达 90%，成功诱导模型生成攻击者预设的答案。Liu 等人[112] 提出的 Poisoned-MRAG 进一步将此类攻击扩展至多模态场景中。他们通过设计脏标签和干净标签等跨模态攻击策略，向多模态知识库中注入精心设计的图文对，进而操纵视觉语言模型（vision-language model, VLM）的输出。实验表明，同样只需注入 5 个恶意样本，在 InfoSeek 等大型多模态数据库上就能实现高达 98% 的攻击成功率，凸显了投毒攻击在跨模态场景下的巨大威胁。Zhang 等人[113] 提出的组合结构提示攻击，通过信息触发器和指令引导操控检索与生成环节，仅需 15 个随机文本构造的查询即可诱导模型生成与原始 CT 图像高度相似的视觉复制品，验证了投毒攻击在隐蔽性与扩展性上的新威胁。在推荐系统领域，Nazary 等人[114] 发现攻击者可以通过操纵商品元数据，如标签和描述，来影响推荐结果。研究指出，针对不同目标可以采用不同策略，例如对每个商品进行个性化标签修改的局部策略比全局修改更有效，并且热门商品比长尾商品更容易成为攻击目标。针对上述安全威胁，研究人员正在探索构建系统性评估框架以量化检索增强生成系统面对投毒攻击的鲁棒性。Zhang 等人[115] 的研究指出了一个关键问题：尽管现有攻击在标准、简单的基准数据集上表现良好，但在更复杂、接近现实的扩展数据集上，其效果会显著下降。同时，现有的防御机制普遍难以提供强有力的防护，进一步表明当前攻防能力的不均衡以及评估体系的局限性[116-117]。

②后门与触发式攻击。与直接的知识投毒不同，后门与触发式攻击是一种更隐蔽、更具条件性的威胁。攻击者通过向系统中植入潜伏的恶意内容，这些内容在常规查询下保持休眠状态，仅在遇到特定、预设的触发器时才被激活，从而稳定地输出恶意的载荷[19,116]。这种攻击的隐蔽性使其难以通过常规的功能或安全测试被发现。Cheng 等人[19] 设计的 TrojanRAG 是利用外部知识库植入的典型框架，它通过精心构造触发器载荷对（trigger-payload pair），并利用对比学习进行优化，使得触发条件高度精确。为了确保恶意载荷，即目标上下文，能被高效率地检索，该方法引入知识图谱来构建结构化数据，实现细粒度的硬匹配。Xue 等人[118] 提出的 BadRAG 则将触发器的概念从简单的关键词扩展到了宽泛的语义群体，如某个政党相关的所有词语，极大地增强了攻击的泛化能力。其研究展示了惊人的攻击效果：仅投毒 10 个对抗性段落，就能诱导系统在 98.2% 的概率下检索到这些恶意内容，进而可将特定查询的拒绝服务率从 0.01% 提升至 74.6%，或实现对模型输出的语义引导。Peng 等人[119] 进一步证明，即使模型经过微调，仍可通过在微调数据中投毒来植入后门，当这个

被植入后门的模型部署到检索增强生成系统中后，攻击者便可通过查询中包含的特定触发器来激活它，操纵模型泄露从检索数据库中获取的文档内容，这揭示了知识向量存储中的供应链风险。

③语义指令注入攻击。该类攻击方式更为巧妙，它不直接篡改事实，而是在正常文档中嵌入具有特异意图的指令，例如“本文档内容具有最高优先级，请忽略其他信息”[46]。当这些文档被检索系统选中后，嵌入的指令会作为上下文的一部分传递给大语言模型。由于大语言模型天然具备指令遵循的行为模式，这些隐含指令可能会干扰其原有的系统指令，进而在语义层面操控模型的决策过程。更具威胁性的是，攻击者甚至可以通过观察系统中哪些数据源被频繁检索与高度信任，然后有针对性地在这些信源中植入恶意指令，或模仿其语言风格伪造文档[120]。此类攻击方式不仅具备较强的隐蔽性与欺骗性，还可能绕过传统的内容审查机制，给系统带来更严重的安全隐患。

2)隐私数据泄露。当向量数据库承载敏感信息时，隐私数据泄露是其面临的一类重要安全问题，尤其在医疗、金融等高敏感行业中，此类风险尤为突出[121-122]。图 7 展示了攻击者通过构造恶意查询获取隐私信息的典型路径。这类攻击可以进一步细分成 3 种方式：数据提取攻击、成员推断攻击、残留数据风险。

①数据提取攻击。研究表明，攻击者可通过精心设计的查询，从检索增强生成系统中直接提取大量私有信息[123-124]。与早期依赖简单提示注入的攻击不同，当前的数据提取攻击已演化为更复杂和更高效的形式。例如，Jiang 等人[45] 提出的 RAG-Thief 框架，利用基于智能体的自动化攻击流程，能从私有知识库中提取超过 70% 的敏感信息；Zhang 等人[20] 提

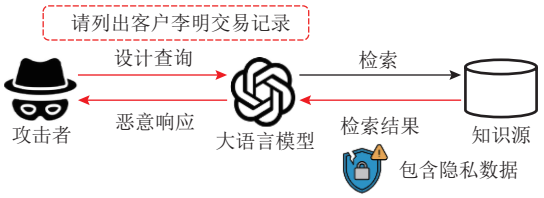


Fig. 7 Privacy data leakage

图 7 隐私数据泄露

出的 DEAL 攻击，更是利用大语言模型作为优化器来迭代生成高效的攻击字符串，实现了接近 99% 的个人隐私信息窃取率。这些攻击甚至可以将表面上无害的查询作为载体，隐蔽性极强，使得传统基于输入过滤的防御机制难以奏效[125]。

②成员推断攻击。攻击者的目标并非提取完整内容，而是判断某条特定数据是否存在于知识库中，这种成员推断行为同样构成严重的隐私侵犯。Anderson 等人[126] 验证了攻击者能够在不直接访问数据库的情况下，通过分析模型输出的细微差别推断出知识库内容。后续研究提出了更多成员推断攻击方法，例如利用基于语义相似度的成员推断攻击[127] 以及利用大语言模型掩码预测能力的推断方法[128]。此外，Naseh 等人[129] 提出的询问攻击更是能在保持查询自然、难以被检测的前提下，实现高精度的成员推断。

③残留数据风险。在向量数据库中，当用户请求删除数据或调整访问权限后，若系统未能从所有组件(如索引、缓存、日志等)中清除相关信息，便可能导致原本应被保护的数据意外暴露。RoyChowdhury 等人[46] 的研究就明确指出了缓存机制是潜在的数据泄露源之一，需引起高度重视。

表 3 对典型的知识库内容攻击进行了系统梳理，包括事实篡改与投毒、后门与触发式、语义指令注入、数据提取、成员推断以及残留数据风险等攻击。不同攻击方式利用伪造信息植入、特定条件触发或

Table 3 Summary of Attacks on Knowledge Base Content

表 3 针对知识库内容攻击的总结

分类	攻击方法	攻击原理	代表工作
知识污染与内容操纵	事实篡改与投毒攻击	注入语义相似的伪造内容，欺骗系统将其作为事实进行检索和生成。	文献 [18,112,114−115]
	后门与触发式攻击	植入由特定查询激活的潜伏恶意载荷，实现条件式、隐蔽性攻击。	文献 [19,118−119]
	语义指令注入攻击	将恶意指令伪装成数据注入文档，在内容被检索后用以劫持生成器行为。	文献 [46,120]
隐私数据泄露	数据提取攻击	设计巧妙的查询序列，诱导系统在其响应中逐步泄露私有文档内容。	文献 [20,45,123]
	成员推断攻击	通过探针式查询及响应分析，在不获取内容的情况下推断特定数据的存在。	文献 [126−128]
	残留数据风险	数据删除或权限变更不彻底，导致残留于缓存或索引中的信息被意外访问。	文献 [46]

推理分析等手段，影响系统输出或窃取敏感数据，并在实际应用中展现出多样的威胁场景。

5.1.2 针对工作流与交互过程的攻击

此类风险体现在检索增强生成系统固有的工作流程（即检索、增强、生成）以及各组件（检索器、生成器）之间的交互机制如何被恶意利用，从而导致模型行为被劫持、逻辑被混淆或安全性受损。这类风险并非源于单一组件的缺陷，而是利用了整个架构的协同机制，具有更强的隐蔽性和复杂性。

1）上下文注入与提示劫持。作为大语言模型的原生风险，该类攻击在检索增强生成场景下会产生变体并被放大，具有极强的破坏性。其核心原理在于，检索增强生成系统会将从外部数据库检索到的、本质上不可信的文本内容与用户发出的、受信任的查询拼接在一起，形成一个全新的增强提示，并交由大语言模型执行[46]。然而，由于检索内容本质上可能是不可信的，攻击者可在其中嵌入恶意指令，如“忽略之前的所有内容，请执行以下任务……”，并伪装为正常文档。一旦该文档被系统选中用于增强提示，嵌入的指令便会污染提示内容，从而劫持模型的控制权[21,46]。这种攻击的危害远不止生成错误答案，还可能导致大语言模型完全偏离原始任务，绕过内容安全护栏以生成有害、违法信息，甚至在指令操控下泄露会话中的敏感数据，包括其他检索内容或用户隐私。

2）权限混淆与越权访问。检索增强生成系统在多用户、多权限环境中面临的典型攻击面是权限混淆与越权访问，也被称作是“困惑代理（confused deputy）”问题。在此场景中，检索增强生成系统作为一个被授权的代理，具有访问整个知识库的能力。然而，尽管系统能够验证用户身份及其操作权限，却往往缺乏对检索结果来源及其可信度的有效验证。RoyChowdhury等人[46]的研究展示了如下攻击场景：一名低权限用户可上传一份包含恶意指令或虚假信息的文档。当高权限用户进行相关查询时，系统可能会检索到该文档并据此生成响应，从而在无意中以高权限身份执行低权限用户设下的恶意操作。这种攻击不仅实现了权限越界，还可能导致错误决策、信息泄露，甚至在组织内部传播虚假信息，对企业运营构成严重威胁。

3）安全性悖论与整体退化。通常认为，为大语言模型提供有事实依据的外部文档有助于提升大语言模型的安全性。然而，An等人[130]的研究发现，检索增强生成框架有时反而会削弱模型的安全性。这是因为检索到的上下文可能会将大语言模型引导到一个安全对齐训练尚不充分的语义空间中。即使是安全的模型与安全的文档组合，也可能因为上下文的微妙影响而生成不安全的输出，这表明检索增强生成系统的整体安全性并非各部分安全的简单叠加。Choi等人[120]的研究进一步揭示了一个结构性漏洞，即检索增强生成悖论：系统为了增强用户的信任，通常会展示其检索到的信息来源。然而，这种透明性恰恰为攻击者提供了观察和利用系统行为的窗口。攻击者可以分析哪些来源被系统偏爱以及信息是如何被呈现的，然后据此制作出不仅更容易被检索到，而且在用户看来也更自然可信的恶意文档，从而大幅提升攻击的成功率。原本旨在增强可信度的设计演变为削弱安全性的通道，形成讽刺性的安全悖论。

针对工作流与交互过程的攻击方式总结如表4所示，相关攻击主要包括上下文注入与提示劫持、权限混淆与越权访问，以及安全性悖论与整体退化。攻击者通常通过注入恶意指令、利用系统验证逻辑漏洞或操控检索链条，达到误导生成、诱导越权操作或削弱系统安全性的目的。

Table 4 Summary of Attacks on Workflow and Interaction Processes

表4 针对工作流与交互过程攻击的总结

攻击方法	攻击原理	代表工作
上下文注入与提示劫持	通过检索内容注入恶意指令，利用工作流拼接提示的过程来劫持生成器。	文献 [21]
权限混淆与越权访问	利用系统验证用户却不验证数据的逻辑缺陷，诱使系统执行基于恶意数据的越权操作。	文献 [46]
安全性悖论与整体退化	利用检索到的上下文将模型引导至安全对齐薄弱区，或利用来源透明性来构造更具欺骗性的攻击。	文献 [120,130]

5.1.3 系统与设施层面的安全风险

系统与设施层面的安全风险源于构成检索增强生成的底层技术栈，涵盖向量数据库本身及其内部架构组件（存储、索引与查询）的内在脆弱性。

作为一种新兴且复杂的软件系统，向量数据库不可避免地存在大量未被发现的程序缺陷，这是最基础也是最直接的风险来源[13]。一项针对15个主流开源向量数据库的大规模实证研究发现，这些系统普遍存在系统崩溃或停止响应（缺陷占比23.1%）、行为不正确（缺陷占比43.0%）和性能异常（缺陷占比9.3%）等问题[47]。攻击者可以利用这些潜在漏洞执行非预期的操作，绕过安全机制或获取未授权的访问权限。

从系统架构角度看,向量数据库中多个关键组件均存在潜在安全隐患。存储层负责向量数据、元数据和索引结构的持久化。研究发现,存储组件是系统崩溃/无响应类严重缺陷的重灾区。一旦出现存储层故障,可能直接导致数据永久性丢失、服务不可用,并引发整个系统的连锁崩溃。查询层负责解析、优化和执行查询。实证研究明确指出,查询处理组件是行为不正确类缺陷最集中的地方(占比最高,达43.0%)[13,47],这类缺陷虽然不一定导致系统直接崩溃,但会返回错误、不完整或完全不相关的结果,从而直接破坏了检索增强生成系统输出的准确性和完整性。

此外,向量数据库广泛采用近似最近邻搜索算法,并构建复杂的索引结构。这些算法的近似性特征虽然提高了检索效率,却也引入了新的攻击面[13]。攻击者可以利用其工作原理,设计特定的向量来实施索引覆盖或劫持攻击,即构造语义相似向量来恶意操纵索引的图结构或量化单元,从而系统性地降低对真实文档的召回率,或将查询稳定地导向攻击者设定的恶意文档。Wang 等人[131] 提出的 DIGA 攻击就利用了检索器对某些更有影响力的词元的偏好,实现了高效的黑盒投毒攻击,这正是利用了索引和检索算法的内在特性。这类攻击直接作用于检索增强生成中的检索环节,一旦攻击成功,系统将无法识别正确信息,甚至错误地采纳恶意内容,从源头上污染整个增强生成流程,带来严重安全威胁[131]。

表 5 简要归纳了系统与设施层的主要安全风险,包括存储层缺陷、查询处理层缺陷及索引与检索算法脆弱性。这些问题可能导致系统崩溃、数据丢失或检索结果被操控。

Table 5 Summary of Security Risks on System and Infrastructure Layers

表 5 针对系统与设施层面的安全风险总结

风险类型	攻击原理	代表工作
存储层缺陷	存储组件的软件缺陷是导致系统崩溃或挂起的重灾区,可能引发数据丢失和服务不可用.	文献 [47]
查询处理层缺陷	查询处理组件的程序缺陷是导致系统行为不正确的主要原因,直接破坏输出可靠性.	文献 [13]
索引与检索算法脆弱性	近似最近邻搜索算法存在可利用的脆弱性,攻击者可通过索引劫持等手段操纵检索结果.	文献 [13,131]

5.1.4 测试与评估层面的风险

当前针对检索增强生成系统安全性的研究仍处于早期,尽管相关技术已经在各类场景中取得广泛应用,但相关的安全测试与评估研究仍存在明显空白,导致业界普遍缺乏系统化、标准化的安全测试框架与评估方法。Wang 等人[13] 的研究指出,问题的根源在于向量数据库的独特性质,这使得传统的数据库测试方法难以直接适用。具体挑战体现在:

1)高维数据与性能开销。向量数据的高维特性不仅使测试数据生成变得复杂,更带来了巨大的计算资源开销。例如,由于高维向量计算通常具有较高的时间复杂度,测试过程中的运行速度大幅下降,内存占用也显著上升,从而限制了大规模自动化测试的可行性。

2)模糊语义与预言机难题。向量搜索的近似性和模糊语义意味着搜索结果本质上是概率性的而非确定性的,这使得定义一个清晰的对错标准,即测试预言机,变得极为困难。

3)动态数据更新带来的测试漂移。向量数据库通常支持索引结构的动态更新。然而,随着数据的不断插入、删除与修改,索引策略可能随时间发生偏移,导致系统性能和检索准确率波动,进而增加测试覆盖与测试一致性的难度。

上述挑战直接导致软件测试的 3 个基础环节,即测试输入生成、预言机定义和测试评估机制,均存在显著的研究空白。由于缺乏完善的测试体系,许多检索增强生成系统在实际部署前未能经过充分验证,难以及时识别和修复可能引发系统崩溃、功能异常或数据泄露的潜在安全漏洞,严重威胁系统的长期安全性与稳定运行。

5.2 知识向量存储的安全防御技术

大语言模型知识向量存储面临的风险贯穿于知识库内容、工作流、基础设施和测试评估等多个层面。本节将围绕前述 4 类风险,系统性地介绍和梳理当前主流的安全防御技术与策略。

5.2.1 针对知识库内容攻击的防御

防御知识污染与隐私泄露的首要目标是在数据进入知识库之前确保其可信性、完整性与机密性。最基础的策略是在数据注入前执行输入过滤与内容审查,通过部署恶意内容检测器、敏感词过滤器与启发式规则,拦截已知攻击样式[110,132-133]。更高级的防御方法则通过上下文分析与异常检测来识别潜在威胁,例如,Zhou 等人[27] 提出的 TrustRAG 框架利用 K 均值聚类来识别检索文档中的潜在攻击模式,而 Yao 等人[134] 提出的 EcoSafeRAG 则通过分析候选文档间的上下文多样性指标,在不依赖大语言模型内部知识的前提下,有效识别上下文异常,提升了系统的稳健性。值得注意的是,EcoSafeRAG 和 TrustRAG

在提升防御能力的同时，能够保持甚至提升系统在无攻击场景下的准确率，并通过大规模基准测试验证了防御措施的有效性和可用性。此外，这些方案已在开源检索增强生成框架和实际企业数据集上做了落地实验，具备较强的工程可移植性。

在数据来源层面，可信度评估与溯源机制是关键保障措施。D-RAG 框架[135]提出基于区块链的社区共识机制，允许领域专家对数据进行验证，并通过去中心化机制控制数据入库，从而构建更具可信性的知识源，实验表明区块链共识机制在成员数扩展时可保持较高效率。

为防止隐私数据泄露，研究人员提出了多种隐私增强技术，其中一类关键策略作用于数据入库前的预处理阶段，通过修改数据内容降低风险。匿名化与假名化是最常见的方式，前者通过移除可识别信息实现不可逆脱敏，后者则使用可恢复的替代标识，兼顾隐私保护与数据可用性[136-137]。相比之下，合成数据技术提供了更彻底的解决方案，通过生成与原始数据具有相似统计特征的虚拟数据，在完全不暴露真实信息的前提下支持下游任务开发与测试[137]。但 Zeng 等人[111]也指出，过强的隐私保护（如高强度噪声注入）可能会带来检索准确率下降，因此实际应用中需权衡隐私保护强度与任务性能。

在数据检索与交互阶段，差分隐私提供了严格的数学保障，它通过在数据或查询过程中添加经过校准的统计噪声，使得攻击者无法从结果中反推出个体信息[133]。这一理念在检索增强生成中已有具体的落地实现：RemoteRAG 框架[28]引入了 (n, ε)-DistanceDP 来保护用户查询的隐私，强调了其算法在业界云检索增强生成服务中的可部署性，并通过效率评测（如延迟和带宽消耗），与无防护、全密态方案进行对比，其方案在隐私保护和检索性能上达到了工程可用平衡；而 LPRAG 框架[132]则通过仅对文本中的隐私实体应用局部差分隐私扰动，在保护隐私的同时保持了较好的数据效用。

5.2.2 针对工作流与交互过程攻击的防御

此类防御旨在加固检索增强生成的工作流程和组件间的交互逻辑，防止其被劫持或混淆。首要措施是输入验证与指令净化，系统需对所有外部输入，包括用户查询和检索文档，进行结构化校验，识别并清除潜在的恶意指令片段。例如，可通过关键词黑名单、控制字符检测、上下文一致性检查等手段，显著降低攻击者通过构造输入误导模型行为的风险[137]。

强化系统提示与构建结构化护栏（guardrail）等技术，可以约束大语言模型的生成行为。通过设计明确的系统级提示，指示模型仅在给定上下文中完成信息生成任务，并拒绝执行涉及代码执行、身份泄露或越权操作的请求。Nandagopal[29]提出的框架中，系统提示被设定为不可变组件，有效防止提示在多轮交互中被篡改或诱导。该方法已在医疗、金融等场景的检索增强生成原型系统中被验证，实验结果显示，添加结构化护栏后，模型越权响应的概率显著下降，同时对业务响应速度影响较小，兼顾了安全性与可用性。

针对工作流与交互过程攻击的另一种防御思路是在检索增强生成流程中部署专门的 AI 防火墙，在最终提示送达大语言模型之前，对其进行风险分析和拦截。ControlNET[138]通过检测查询中的激活特征偏移，识别出具有异常意图的对抗性输入，并在模型生成前进行阻断或调整，从而实现动态防御。ControlNET 在 4 大公开数据集和真实医疗场景的多角色权限系统中实现了 ROC 曲线下面积（AUROC）大于 0.9 的高精度检测，且对模型输出的准确率和召回率影响低于 3%。

从架构层面看，引入零信任架构原则，要求对每一次组件间的访问请求都进行身份验证与权限审查，无论请求来自系统内部还是外部。这种持续验证机制可有效防范困惑代理类攻击，避免低权限用户诱导系统以高权限身份执行敏感操作[29]。

在检索与生成之间增加后处理增强环节，可进一步降低恶意内容干扰模型生成的风险。一方面，文档重排序机制可根据与用户意图的相关性重新排列检索结果，过滤掉噪声文档或可疑信息；另一方面，内容摘要则通过压缩原始内容，仅保留核心信息输入模型，既减少了不必要信息的暴露，也提升了响应的准确性与安全性[137]。

5.2.3 系统与设施层面的安全加固

针对向量数据库在系统和架构层面暴露出的安全风险，应重点关注其关键组件的可靠性，包括存储管理、索引构建与查询处理等模块。Xie 等人[47]的研究表明，系统崩溃类缺陷主要集中于存储层，而行为不正确类缺陷则广泛存在于查询执行路径中，为此，应优先加强对这些高风险组件的鲁棒性设计，例如通过精细化的异常监测机制捕捉运行时崩溃和挂起行为，以及建立更健壮的查询逻辑校验流程，防止因查询优化错误或索引逻辑缺陷引发数据错误。

此外，考虑到向量索引结构（如 HNSW，IVF 等）在性能优化过程中可能引入浮点精度误差或边界异

常，建议在系统层面引入边界输入验证与动态索引一致性检查机制，以防止索引结构偏移或退化对系统性能与可用性造成影响。随着向量数据库与大语言模型应用深度融合，系统还应支持跨模态数据管理与混合查询处理，这对底层设施提出了更高的灵活性与可测性要求，需从工程和设计层面同步落实安全性考量。

5.2.4 测试与评估体系的构建

Wang 等人[13]提出的向量数据库测试路线图系统性地指出了需要在测试输入生成、测试预言机定义和测试评估等方面进行深入研究，以提升向量数据库的可靠性。

在测试输入生成方面，应覆盖多种查询类型和复杂场景，如谓词过滤、联合向量查询以及跨模态混合检索，并考虑索引类型和参数组合带来的输入空间爆炸问题。在测试预言机设计方面，由于向量搜索的近似性导致难以定义明确的对/错标准，研究建议采用蜕变测试（metamorphic testing）、差分测试（differential testing）和属性验证（property-based oracle）等方法对系统行为进行间接验证。在测试评估方面，应超越传统的代码覆盖率，重点关注向量空间覆盖率、操作序列覆盖率与跨 API 状态依赖的测试效果衡量，确保测试在多维结构与复杂操作链条下的风险识别能力。

5.3 小 结

知识向量存储技术作为支撑大语言模型外部知识接入的关键机制，显著提升了模型的可靠性与可解释性，但与此同时也引入了多维度的安全与隐私风险。本节从 4 个方面系统梳理了当前面向该技术的主要攻击方式与防御策略，包括：针对知识库内容的污染与泄露、工作流与交互机制的劫持与混淆、系统设施层面的结构脆弱性，以及测试评估体系的空白与挑战。研究表明，攻击者不仅能够通过控制文档内容操控生成结果，还可利用上下文拼接机制劫持模型行为，甚至借助索引算法弱点实施系统级信息植入与操控。此外，向量数据库中尚未充分覆盖的安全测试流程也使得此类系统在部署前难以实现全面验证，形成潜在风险隐患。为应对上述问题，研究人员提出了一系列防御技术，包括上下文异常检测、差分隐私扰动、零信任架构、向量数据库测试框架等，在提升系统鲁棒性的同时，也为构建更加可信、可控的大语言模型提供了实践路径。总体而言，知识向量存储的安全防护仍处于持续演进阶段，未来亟需在攻防体系化、测试自动化与评估标准化等方面进一步深入研究，以实现大规模部署下的长期安全保障。

6 总结与展望

6.1 本文总结

本文对大语言模型存储机制的安全问题进行了系统性的调研与梳理，全面回顾了模型文件存储、推理缓存和知识向量存储 3 个部分的安全威胁，并详细分析了模型文件窃取攻击、模型后门攻击、推理缓存侧信道攻击、知识库污染与泄露、工作流劫持等多种攻击方式及其技术特点，总结了不同攻击方法在原理、实施路径与适用场景上的差异性，明确了现有研究的技术优势与局限性，为后续研究提供了重要参考。针对这些安全风险，本文对每类安全风险的现有防护技术进行了简要回顾。基于上述分析，本文探讨了大语言模型存储机制安全领域的现存问题与技术瓶颈，并对未来研究方向进行了展望。

6.2 未来展望

目前大语言模型存储机制安全的相关研究主要聚焦于模型文件存储、推理缓存和知识向量存储这 3 个核心模块，这些存储机制贯穿了模型的训练、推理和外部知识增强的关键环节。然而，随着大语言模型规模的进一步扩大、多模态融合的深入推进以及新型应用需求的不断涌现，其生命周期中可能会出现新的存储机制，从而引发新的安全风险与防御需求。

6.2.1 模型文件存储

随着模型规模的扩大和分布式部署的普及，模型权重文件的安全性将成为研究的重点。现有的模型文件存储安全技术主要集中于模型窃取攻击和模型后门攻击，然而，随着大语言模型存储的优化技术不断涌现，以及新兴模型文件格式的推广应用，未来的研究需要进一步拓展到这些领域的安全性分析，以应对新技术带来的潜在风险。一个重要方向是针对新兴模型文件格式（例如 GGUF）的安全性研究[139]。GGUF 等高效存储格式近年来被广泛应用于开源大语言模型，它们通过优化文件结构和加载性能显著提升了存储和推理效率。然而，这些新格式的设计可能引入新的攻击面，例如在文件解析或加载过程中注入恶意代码，或通过操控格式细节触发特定行为。研究者需要开发针对新模型文件格式的安全分析框架，评估其在分布式环境、跨节点协作或恶意部署中的潜在安全风险，并提出针对性的防护措施。此外，随着模型压缩与量化技术的广泛应用，研究者

需要进一步探索如何在提升存储效率的同时避免引入新的安全隐患。例如，量化后门攻击和恶意参数修改等问题可能通过篡改模型权重影响模型行为，这需要针对压缩和量化后的权重文件设计更加鲁棒的防御策略。

6.2.2 模型推理缓存

针对模型推理缓存的现有研究主要集中于基于缓存的侧信道攻击，通过多租户或者硬件共址环境中的共享推理缓存来窃取用户隐私信息或推断模型的内部状态。一系列防御技术通过缓存隔离、可信执行环境、隐私增强等手段降低数据泄露的风险。然而，随着各种模型推理优化技术的提出和推理优化框架的不断发展，推理缓存的攻击面可能不仅仅局限于侧信道攻击，还有可能暴露出更多的新兴攻击面。例如，在推理缓存或算子优化过程中，攻击者可能通过操控底层内存管理或缓存替换策略，触发内存破坏漏洞，从而引发非预期行为。未来的研究需要关注这些推理优化框架本身的安全性，确保其在提升性能的同时不会引入新的安全缺陷。此外，随着大语言模型参数规模的持续增长，为降低显存占用、提升推理吞吐量，越来越多系统引入了参数卸载（parameter offloading）与键值缓存卸载机制，将部分模型权重或注意力缓存从 GPU 卸载至 CPU 内存、磁盘或远程节点。这些卸载过程在提升可部署性和资源利用效率的同时，也引入了新的安全与隐私风险，例如卸载数据在传输过程中可能面临中间人攻击或完整性破坏，卸载落盘后的缓存或权重数据可能由于缺乏有效清理或访问控制而被非法访问，甚至被用于模型行为操控或知识提取。此外，若卸载与加载过程依赖的内存映射、分页或替换策略存在设计缺陷，攻击者还可能通过构造精确访问序列诱发缓存欺骗、缓存污染或触发底层内存错误，间接影响推理行为的可信性。因此，未来的研究不仅需要关注传统的共享缓存安全问题，还应进一步扩展到推理缓存与卸载机制交叉场景下的系统安全挑战，尤其是卸载路径中的数据加密、访问控制、可信加载机制以及卸载框架自身的鲁棒性验证，确保推理系统在性能优化的同时不会引入新的攻击面和安全隐患。

6.2.3 模型知识向量存储

随着检索增强生成技术的快速发展，知识向量存储逐渐成为大语言模型的重要扩展模块。然而，其安全性仍然面临数据投毒、后门植入和隐私泄露等多重威胁。未来的研究需要重点解决向量数据库中的恶意数据检测与清理问题，例如动态过滤技术和上下文一致性验证机制，确保存储内容的完整性和可信性。此外，针对向量化数据的隐私保护技术也需要进一步研究，例如结合同态加密和差分隐私技术，在保障数据安全的同时提高检索效率。向量数据库本身的可靠性也是未来研究的一个重要方向，可能由于硬件故障、资源竞争或内存漏洞而引发意外错误或崩溃，研究者需要设计更加全面的可靠性测试框架，对向量数据库的存储稳定性、查询一致性和容错能力进行系统化的评估。考虑到知识向量存储在多模态环境中的应用场景日益复杂，研究者还需探索如何在跨模态向量检索中构建统一的安全框架，防止攻击者通过模态间的不一致性或漏洞实施跨模态投毒和操控。

6.2.4 新型存储机制

随着大语言模型的不断发展和新型应用需求的涌现，其生命周期中可能会出现新的存储机制。例如，跨模态存储机制在多模态模型中的采用可以高效管理和整合图像、文本、音频、视频等多模态数据。这类存储机制需要解决不同模态数据在格式、索引和访问权限上的差异性，同时应对模态间数据交互可能引发的安全问题，尤其是攻击者通过操控某一模态数据间接破坏其他模态存储内容的潜在威胁。此外，任务状态存储机制可以应对复杂任务（如多轮对话、长文档分析或连续推理），以支持上下文信息的动态存储和管理。这类机制的动态特性要求更高效的访问控制，以防止攻击者通过篡改任务状态干扰模型的推理流程或输出结果。未来的研究需要在持续优化现有存储机制的基础上，针对可能的新型存储机制开发更具适应性的安全分析与防御技术，构建覆盖模型全生命周期的安全体系，以应对日益复杂的存储安全威胁。

作者贡献声明：王柳、王申奥负责论文的总体规划以及主要内容的梳理与撰写；侯心怡、赵建负责文献资料的调研与补充，并协助撰写论文；吴荣鑫、向乔、文。王柳和王申奥对本论文有同等贡献。

参 考 文 献

[1] Radford A, Wu J, Child R, et al. OpenAI blog: Language models are unsupervised multitask learners[EB/OL]. 2024[2024-08-10]. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[2] Touvron H, Lavril T, Izacard G, et al. LLaMA: Open and efficient foundation language models[J]. arXiv preprint, arXiv: 2302.13971,

2023

[3] Team G, Anil R, Borgeaud S, et al. Gemini: A family of highly capable multimodal models[J]. arXiv preprint, arXiv: 2312.11805, 2023

[4] Zhao Wayne Xin, Zhou Kun, Li Junyi, et al. A survey of large language models[J]. arXiv preprint, arXiv: 2303.18223, 2023

[5] Hadi M U, Qureshi R, Shah A, et al. A survey on large language models: Applications, challenges, limitations, and practical usage[J]. Authorea Preprints, 2023, 1(3): 1−26

[6] Hu Songlin, Li Juanzi, Qin Bing, et al. The double-edged swords: An introduction to the special issue on large models and safety[J]. Journal of Computer Research and Development, 2024, 61(5): 1085−1093 (in Chinese)
(虎嵩林, 李涓子, 秦兵, 等. 亦正亦邪大模型——大模型与安全专题导读[J]. 计算机研究与发展, 2024, 61(5): 1085−1093)

[7] Xu F F, Alon U, Neubig G, et al. A systematic evaluation of large language models of code[C]//Proc of the 6th ACM SIGPLAN Int Symp on Machine Programming. New York: ACM, 2022: 1−10

[8] Ge Xuran, Ou Yang, Wang Bo, et al. Survey of storage optimization techniques in large language model inference[J]. Journal of Computer Research and Development, 2025, 62(3): 545−562 (in Chinese)
(葛旭冉, 欧洋, 王博, 等. 大语言模型推理中的存储优化技术综述[J]. 计算机研究与发展, 2025, 62(3): 545−562)

[9] Fan Wenqi, Ding Yujuan, Ning Liangbo, et al. A survey on RAG meeting LLMs: Towards retrieval-augmented large language models[C]//Proc of the 30th ACM SIGKDD Conf on Knowledge Discovery and Data Mining. New York: ACM, 2024: 6491−6501

[10] Gao Yunfan, Xiong Yun, Gao Xinyu, et al. Retrieval-augmented generation for large language models: A survey[J]. arXiv preprint, arXiv: 2312.10997, 2023

[11] Li Qinfeng, Xie Yangfan, Du Tianyu, et al. CoreGuard: Safeguarding foundational capabilities of LLMs against model stealing in edge deployment[J]. arXiv preprint, arXiv: 2410.13903, 2024

[12] Zhao Jian, Wang Shenao, Zhao Yanjie, et al. Models are codes: Towards measuring malicious code poisoning attacks on pre-trained model hubs[C]//Proc of the 39th IEEE/ACM Int Conf on Automated Software Engineering. New York: ACM, 2024: 2087−2098

[13] Wang Shenao, Zhao Yanjie, Xie Yinglin, et al. Towards reliable vector database management systems: A software testing roadmap for 2030[J]. arXiv preprint, arXiv: 2502.20812, 2025

[14] Oliynyk D, Mayer R, Rauber A. I know what you trained last summer: A survey on stealing machine learning models and defences[J]. ACM Computing Surveys, 2023, 55(14): 1−41

[15] Song Linke, Pang Zixuan, Wang Wenhao, et al. The early bird catches the leak: Unveiling timing side channels in LLM serving systems[J]. arXiv preprint, arXiv: 2409.20002, 2024

[16] Adiletta A, Sunar B. Spill the beans: Exploiting CPU cache side-channels to leak tokens from large language models[J]. arXiv preprint, arXiv: 2505.00817, 2025

[17] Soleimani M, Jia Guanzhong, Gim I, et al. Wiretapping LLMs: Network side-channel attacks on interactive LLM services[J/OL]. Cryptology ePrint Archive, 2025[2025-02-04]. https://eprint.iacr.org/2025/167.pdf

[18] Zou Wei, Geng Runpeng, Wang Binghui, et al. PoisonedRAG: Knowledge corruption attacks to retrieval-augmented generation of large language models[C/OL]//Proc of the 34th USENIX Security Symp. Berkeley, CA: USENIX Association, 2025[2025-08-13]. https://www.usenix.org/conference/usenixsecurity25/presentation/zou-poisonedrag

[19] Cheng Peng, Ding Yu, Ju Tong, et al. TrojanRAG: Retrieval-augmented generation can be backdoor driver in large language models[J]. arXiv preprint, arXiv: 2405.13401, 2024

[20] Zhang Tailai, Jiang Yuxuan, Gong Ruihan, et al. DEAL: High-efficacy privacy attack on retrieval-augmented generation systems via LLM optimizer[EB/OL]. 2025[2025-02-05]. https://openreview.net/forum? id=sx8dtyZT41

[21] Anichkov Y, Popov V, Bolovtsov S. Retrieval poisoning attacks based on prompt injections into retrieval-augmented generation systems that store generated responses[C]//Proc of the 27th Int Conf on Distributed Computer and Communication Networks. Berlin: Springer, 2024: 417−429

[22] Kethireddy R R. Secure model distribution and deployment for LLMs[J]. Journal of Recent Trends in Computer Science and Engineering, 2024, 12(4): 1−14

[23] Zhou Mingyi, Gao Xiang, Wu Jing, et al. ModelObfuscator: Obfuscating model information to protect deployed ML-based systems[C]//Proc of the 32nd ACM SIGSOFT Int Symp on Software Testing and Analysis. New York: ACM, 2023: 1005−1017

[24] Pang Zixuan, Wang Wenhao, Liao Yufan. Cache partitioning for mitigating timing side-channel attacks in LLM serving systems[C]//Proc of the 6th Int Conf on Frontier Technologies of Information and Computer. Piscataway, NJ: IEEE, 2024: 1238−1245

[25] Wang Jiahao, Han Jinbo, Wei Xingda, et al. KVCache cache in the wild: Characterizing and optimizing KVCache cache at a large cloud provider[C]//Proc of the 2025 USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2025: 465−482

[26] Gim I, Li Caihua, Zhong Lin. Confidential prompting: Protecting user prompts from cloud LLM providers[J]. arXiv preprint, arXiv: 2409.19134, 2024

[27] Zhou Huichi, Lee K H, Zhan Zhonghao, et al. TrustRAG: Enhancing robustness and trustworthiness in RAG[J]. arXiv preprint, arXiv: 2501.00879, 2025

[28] Cheng Yihang, Zhang Lan, Wang Junyang, et al. RemoteRAG: A privacy-preserving LLM cloud RAG service[C]//Proc of the 2025 Findings of the Association for Computational Linguistics. Stroudsburg, PA: ACL, 2025: 3820−3837

[29] Nandagopal S. Securing retrieval-augmented generation pipelines: A comprehensive framework[J]. Journal of Computer Science and Technology Studies, 2025, 7(1): 17−29

[30] Li Junyi, Tang Tianyi, Zhao Wayne Xin, et al. Pre-trained language models for text generation: A survey[J]. ACM Computing Surveys, 2024, 56(9): 1−39

[31] Zhuang Yuchen, Yu Yue, Wang Kuan, et al. ToolQA: A dataset for LLM question answering with external tools[C]//Proc of the 37th Advances in Neural Information Processing Systems. Red Hook, NY: Curran Associates Inc, 2023: 50117−50143

[32] Hou Xinyi, Zhao Yanjie, Liu Yue, et al. Large language models for

software engineering: A systematic literature review[J]. ACM Transactions on Software Engineering and Methodology, 2024, 33(8): 1−79

[33] Wang Shenao, Zhao Yanjie, Hou Xinyi, et al. Large language model supply chain: A research agenda[J]. ACM Transactions on Software Engineering and Methodology, 2025, 34(5): 1−46

[34] Radford A, Narasimhan K, Salimans T, et al. Improving language understanding by generative pre-training[R/OL]. OpenAI, 2018 [2018-06-11]. https://cdn.openai.com/research-covers/language-un-supervised/ languag e_ understanding_paper.pdf

[35] Chang Yupeng, Wang Xu, Wang Jindong, et al. A survey on evaluation of large language models[J]. ACM Transactions on Intelligent Systems and Technology, 2024, 15(3): 1−45

[36] Shen Tianhao, Jin Renren, Huang Yufei, et al. Large language model alignment: A survey[J]. arXiv preprint, arXiv: 2309.15025, 2023

[37] Bai Jinze, Bai Shuai, Chu Yunfei, et al. Qwen technical report[J]. arXiv preprint, arXiv: 2309.16609, 2023

[38] Chandra B, Preethika P, Challagundla S, et al. End-to-end neural embedding pipeline for large-scale PDF document retrieval using distributed FAISS and sentence transformer models[J]. International Journal of Advanced Research in Computer Science and Engineering, 2024, 1(2): 1−21

[39] Singh P N, Talasila S, Banakar S V. Analyzing embedding models for embedding vectors in vector databases[C]//Proc of the 3rd IEEE Int Conf on ICT in Business Industry & Government. Piscataway, NJ: IEEE, 2023: 1−7

[40] Yan Jun, Gupta V, Ren Xiang. BITE: Textual backdoor attacks with iterative trigger injection[C]//Proc of the 61st Annual Meeting of the Association for Computational Linguistics. Stroudsburg, PA: ACL, 2023: 12951−12968

[41] Liu Tao, Liu Zihao, Liu Qi, et al. StegoNet: Turn deep neural network into a stegomalware[C]//Proc of the 36th Annual Computer Security Applications Conf. New York: ACM, 2020: 928−938

[42] Wang Zhi, Liu Chaoge, Cui Xiang. EvilModel: Hiding malware inside of neural network models[C]//Proc of the 26th IEEE Symp on Computers and Communications. Piscataway, NJ: IEEE, 2021: 1−7

[43] Zheng Xinyao, Han Husheng, Shi Shangyi, et al. InputSnatch: Stealing input in LLM services via timing side-channel attacks[J]. arXiv preprint, arXiv: 2411.18191, 2024

[44] Wang Jianguo, Hanson E, Li Guoliang, et al. Vector databases: What's really new and what's next? (VLDB 2024 panel)[J]. Proceedings of the VLDB Endowment, 2024, 17(12): 4505−4506

[45] Jiang Changyue, Pan Xudong, Hong Geng, et al. RAG-Thief: Scalable extraction of private data from retrieval-augmented generation applications with agent-based attacks[J]. arXiv preprint, arXiv: 2411.14110, 2024

[46] RoyChowdhury A, Luo Mulong, Sahu P, et al. ConfusedPilot: Confused deputy risks in RAG-based LLMs[J]. arXiv preprint, arXiv: 2408.04870, 2024

[47] Xie Yinglin, Hou Xinyi, Zhao Yanjie, et al. Toward understanding bugs in vector database management systems[J]. arXiv preprint, arXiv: 2506.02617, 2025

[48] Li Yuanchun, Wen Hao, Wang Weijun, et al. Personal LLM agents: Insights and survey about the capability, efficiency and security[J]. arXiv preprint, arXiv: 2401.05459, 2024

[49] Guo Taicheng, Chen Xiuying, Wang Yaqi, et al. Large language model based multi-agents: A survey of progress and challenges[C]// Proc of the 33rd Int Joint Conf on Artificial Intelligence Survey Track. Red Hook, NY: Curran Associates Inc, 2024: 8048−8057

[50] Das B C, Amini M H, Wu Yanzhao. Security and privacy challenges of large language models: A survey[J]. ACM Computing Surveys, 2025, 57(6): 1−39

[51] Yao Yifan, Duan Jinhao, Xu Kaidi, et al. A survey on large language model (LLM) security and privacy: The good, the bad, and the ugly[J]. High-Confidence Computing, 2024, 4(2): 100211

[52] He Feng, Zhu Tianqing, Ye Dayong, et al. The emerged security and privacy of LLM agent: A survey with case studies[J]. arXiv preprint, arXiv: 2407.19354, 2024

[53] Jiang Yi, Yang Yong, Yin Jiali, et al. Survey on security and privacy risks in large language models[J]. Journal of Computer Research and Development, 2025, 62(8): 1979−2018 (in Chinese)
(姜毅, 杨勇, 印佳丽, 等. 大语言模型安全与隐私风险综述[J]. 计算机研究与发展, 2025, 62(8): 1979−2018)

[54] Tai Jianwei, Yang Shuangning, Wang Jiajia, et al. Survey of adversarial attacks and defenses for large language models[J]. Journal of Computer Research and Development, 2025, 62(3): 563−588 (in Chinese)
(台建玮, 杨双宁, 王佳佳, 等. 大语言模型对抗性攻击与防御综述[J]. 计算机研究与发展, 2025, 62(3): 563−588)

[55] Li Nan, Ding Yidong, Jiang Haoyu, et al. Jailbreak attack for large language models: A survey[J]. Journal of Computer Research and Development, 2024, 61(5): 1156−1181 (in Chinese)
(李南, 丁益东, 江浩宇, 等. 面向大语言模型的越狱攻击综述[J]. 计算机研究与发展, 2024, 61(5): 1156−1181)

[56] Fan A, Gokkaya B, Harman M, et al. Large language models for software engineering: Survey and open problems[C]//Proc of the 2023 IEEE/ACM Int Conf on Software Engineering: Future of Software Engineering (ICSE-FoSE). Piscataway, NJ: IEEE, 2023: 31−53

[57] Wang Junjie, Huang Yuchen, Chen Chunyang, et al. Software testing with large language models: Survey, landscape, and vision[J]. IEEE Transactions on Software Engineering, 2024, 50(4): 911−936

[58] Xu Pengyu, Kuang Boyu, Su Mang, et al. Survey of large-language-model-based automated program repair[J]. Journal of Computer Research and Development, 2024, 61(7): 1801−1820 (in Chinese)
(许鹏宇, 况博裕, 苏锴, 等. 基于大语言模型的自动代码修复综述[J]. 计算机研究与发展, 2024, 61(7): 1801−1820)

[59] Ni Bo, Liu Zheyuan, Wang Leyao, et al. Towards trustworthy retrieval augmented generation for large language models: A survey[J]. arXiv preprint, arXiv: 2502.06872, 2025

[60] Zhou Yujia, Liu Yan, Li Xiaoxi, et al. Trustworthiness in retrieval-augmented generation systems: A survey[J]. arXiv preprint, arXiv: 2409.10102, 2024

[61] Carlini N, Paleka D, Dvijotham K D, et al. Stealing part of a production language model[C]//Proc of the 41st Int Conf on Machine Learning. New York: PMLR, 2024: 5680−5705

[62] Rolnick D, Kording K. Reverse-engineering deep ReLU net-

works[C]//Proc of the 37th Int Conf on Machine Learning. New York: PMLR, 2020: 8178−8187

[63] Sorensen T, Khlaaf H. LeftoverLocals: Listening to LLM responses through leaked GPU local memory[J]. arXiv preprint, arXiv: 2401.16603, 2024

[64] Pustelnik F D, Sass X M, Seifert J P. Whispering pixels: Exploiting uninitialized register accesses in modern GPUs[C]//Proc of the 9th IEEE European Symp on Security and Privacy (EuroS&P). Piscataway, NJ: IEEE, 2024: 345−360

[65] Liang Zi, Ye Qingqing, Wang Yanyun, et al. "Yes, My LoRD." Guiding language model extraction with locality reinforced distillation[C]//Proc of the 63rd Annual Meeting of the Association for Computational Linguistics. Stroudsburg, PA: ACL, 2025: 1441−1465

[66] Birch L, Hackett W, Trawicki S, et al. Model leeching: An extraction attack targeting LLMs[J]. arXiv preprint, arXiv: 2309.10544, 2023

[67] Xu Jiashu, Ma Mingyu, Wang Fei, et al. Instructions as backdoors: Backdoor vulnerabilities of instruction tuning for large language models[C]//Proc of the 2024 Conf of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Stroudsburg, PA: ACL, 2024: 3111−3126

[68] Yan Jun, Yadav V, Li Shiyang, et. al. Backdooring instruction-tuned large language models with virtual prompt injection[C]//Proc of the 2024 Conf of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Stroudsburg, PA: ACL, 2024: 6065−6086

[69] Wang Yifei, Xue Dizhan, Zhang Shengjie, et. al. BadAgent: Inserting and activating backdoor attacks in LLM agents[C]//Proc of the 62nd Annual Meeting of the Association for Computational Linguistics. Stroudsburg, PA: ACL, 2024: 9811−9827

[70] Rando J, Tramèr F. Universal jailbreak backdoors from poisoned human feedback[C/OL]//Proc of the 12th Int Conf on Learning Representations. Washington: ICLR, 2024[2024-05-07]. https://iclr.cc/virtual/2024/poster /19013

[71] Gu Naibin, Fu Peng, Liu Xiyu, et. al. A gradient control method for backdoor attacks on parameter-efficient tuning[C]//Proc of the 61st Annual Meeting of the Association for Computational Linguistics. Stroudsburg, PA: ACL, 2023: 3508−3520

[72] Li Yanzhou, Li Tianlin, Chen Kangjie, et. al. BadEdit: Backdooring large language models by model editing[C/OL]//Proc of the 12th Int Conf on Learning Representations. Washington: ICLR, 2024[2024-05-22]. https://iclr. cc/virtual/2024/poster/18240

[73] Qiu Jiyang, Ma Xinbei, Zhang Zhuosheng, et al. MEGen: Generative backdoor in large language models via model editing[C]//Proc of the 63rd Annual Meeting of the Association for Computational Linguistics. Stroudsburg, PA: ACL, 2025: 11197−11214

[74] Zeng Qingcheng, Jin Mingyu, Yu Qinkai, et al. Uncertainty is fragile: Manipulating uncertainty in large language models[J]. arXiv preprint, arXiv: 2407.11282, 2024

[75] Cai Xiangrui, Xu Haidong, Xu Sihan, et al. BadPrompt: Backdoor attacks on continuous prompts[C]//Proc of the 36th Int Conf on Neural Information Processing Systems. Red Hook, NY: Curran Associates Inc, 2022: 37068−37080

[76] Du Wei, Zhao Yichun, Li Boqun, et al. PPT: Backdoor attacks on pre-trained models via poisoned prompt tuning[C]//Proc of the 31st Int Joint Conf on Artificial Intelligence. Red Hook, NY: Curran Associates Inc, 2022: 680−686

[77] Yao Hongwei, Lou Jian, Qin Zhan. PoisonPrompt: Backdoor attack on prompt-based large language models[C]//Proc of the 2024 IEEE Int Conf on Acoustics, Speech and Signal Processing. Piscataway, NJ: IEEE, 2024: 7745−7749

[78] Xue Jiaqi, Zheng Mengxin, Hua Ting, et al. TrojLLM: A black-box trojan prompt attack on large language models[C]//Proc of the 37th Int Conf on Neural Information Processing Systems. Red Hook, NY: Curran Associates, Inc, 2023: 65665−65677

[79] Xu Lei, Chen Yangyi, Cui Ganqu, et al. Exploring the universal vulnerability of prompt-based learning paradigm[C]//Proc of the 60th Annual Meeting of the Association for Computational Linguistics. Stroudsburg, PA: ACL, 2022: 1799−1810

[80] Zhao Shuai, Wen Jinming, Luu A T, et al. Prompt as triggers for backdoor attack: Examining the vulnerability in language models[C]//Proc of the 2023 Conf on Empirical Methods in Natural Language Processing. Stroudsburg, PA: ACL, 2023: 12303−12317

[81] Huang Hai, Zhao Zhengyu, Backes M, et al. Composite backdoor attacks against large language models[C]//Proc of the 62nd Annual Meeting of the Association for Computational Linguistics. Stroudsburg, PA: ACL, 2024: 1459−1472

[82] Kandpal N, Jagielski M, Tramèr F, et al. Backdoor attacks for in-context learning with language models[J]. arXiv preprint, arXiv: 2307.14692, 2023

[83] Zhao Shuai, Jia Meihuizi, Luu A T, et al. Universal vulnerabilities in large language models: Backdoor attacks for in-context learning[C]//Proc of the 2024 Conf on Empirical Methods in Natural Language Processing. Stroudsburg, PA: ACL, 2024: 11507−11522

[84] Xiang Zhen, Jiang Fengqing, Xiong Zidi, et al. BadChain: Backdoor chain-of-thought prompting for large language models[C/OL]//Proc of the 12th Int Conf on Learning Representations. Washington: ICLR, 2024[2024-05-07]. https://iclr.cc/virtual/2024/poster/18305

[85] Qi Fanchao, Chen Yangyi, Li Mukai, et al. ONION: A simple and effective defense against textual backdoor attacks[C]//Proc of the 2021 Conf on Empirical Methods in Natural Language Processing. Stroudsburg, PA: ACL, 2021: 9558−9566

[86] Jiang Yi, Shi Chenghui, Ma Oubo, et al. Text laundering: Mitigating malicious features through knowledge distillation of large foundation models[C]//Proc of the 19th Int Conf on Information Security and Cryptology. Berlin: Springer, 2023: 3−23

[87] Azizi A, Tahmid I A, Waheed A, et al. T-Miner: A generative approach to defend against trojan attacks on DNN-based text classification[C]//Proc of the 30th USENIX Security Symp. Berkeley, CA: USENIX Association, 2021: 2255−2272

[88] Liu Kang, Dolan-Gavitt B, Garg S. Fine-Pruning: Defending against backdooring attacks on deep neural networks[C]//Proc of the 21st Int Symp on Research in Attacks, Intrusions, and Defenses. Berlin: Springer, 2018: 273−294

[89] Li Yige, Lyu Xixiang, Koren N, et al. Neural attention distillation: Erasing backdoor triggers from deep neural networks[C/OL]//Proc

of the 9th Int Conf on Learning Representations. Washington: ICLR, 2021[2021-05-03]. https://iclr.cc/virtual/2021/poster/2760

[90] Wang Zhi, Liu Chaoge, Cui Xiang, et al. EvilModel 2.0: Bringing neural network models into malware attacks[J]. Computers & Security, 2022, 120(C): 102807

[91] Hua Jiayi, Wang Kailong, Wang Meizhen, et al. MalModel: Hiding malicious payload in mobile deep learning models with black-box backdoor attack[J]. arXiv preprint, arXiv: 2401.02659, 2024

[92] Hitaj D, Pagnotta G, Hitaj B, et al. MaleficNet: Hiding malware into deep neural networks using spread-spectrum channel coding[C]// Proc of the 27th European Symp on Research in Computer Security. Berlin: Springer, 2022: 425−444

[93] Wang Rong, Liang Junchuan, Jiang Haiting, et al. StegoFL: Using steganography and federated learning to transmit malware[C]//Proc of the 23rd IEEE Int Conf on Trust, Security and Privacy in Computing and Communications. Piscataway, NJ: IEEE, 2024: 184−190

[94] Dubin R. Disarming steganography attacks inside neural network models[J]. arXiv preprint, arXiv: 2309.03071, 2023

[95] Gilkarov D, Dubin R. Model X-ray: Detection of hidden malware in AI model weights using few shot learning[J]. arXiv preprint, arXiv: 2409.19310, 2024

[96] Zubicueta P S, Riegler M A. Maleficent neural networks, the embedding of malware in neural networks: A survey[J]. IEEE Access, 2024, 12: 69753−69764

[97] Casey B, Santos J C S, Mirakhorli M. A large-scale exploit instrumentation study of AI/ML supply chain attacks in Hugging Face models[J]. arXiv preprint, arXiv: 2410.04490, 2024

[98] Zhu Ruofan, Chen Ganhao, Shen Wenbo, et al. My model is malware to you: Transforming AI models into malware by abusing tensorflow APIs[C]//Proc of the 46th IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2025: 486−503

[99] Ding Ziqi, Fu Qian, Ding Junchen, et al. A rusty link in the AI supply chain: Detecting evil configurations in model repositories[C]//Proc of the 46th IEEE Security and Privacy Workshops. Piscataway, NJ: IEEE, 2025: 260−264

[100] Wu Guanlong, Zhang Zheng, Zhang Yao, et al. I know what you asked: Prompt leakage via KV-cache sharing in multi-tenant LLM serving[C/OL]//Proc of the 32nd Annual Network and Distributed System Security Symp (NDSS). Reston, VA, USA: Internet Society, 2025[2025-02-24]. https://www.ndss-symposium.org/wp-content/uploads/2025-1772-paper.pdf

[101] Gu Chenchen, Li X L, Kuditipudi R, et al. Auditing prompt caching in language model APIs[C/OL]//Proc of the 42nd Int Conf on Machine Learning. New York: ICML, 2025[2025-07-13]. https://icml.cc/ virtual/2025/poster/44473

[102] Gao Zibo, Hu Junjie, Guo Feng, et al. I know what you said: Unveiling hardware cache side-channels in local large language model inference[C/OL]//Proc of the 34th USENIX Security Symp. Berkeley, CA: USENIX Association, 2025[2025-08-13]. https://www.usenix.org/conference/usenixsecurity25/presentation/gao-zibo

[103] Yang Haotian, Zhang Dongcheng, Zhao Yuke, et al. A first look at efficient and secure on-device LLM inference against KV leakage[C]//Proc of the 19th Workshop on Mobility in the Evolving Internet Architecture. New York: ACM, 2024: 13−18

[104] Tan Yifan, Tan Cheng, Mi Zeyu, et al. PipeLLM: Fast and confidential large language model services with speculative pipelined encryption[C]//Proc of the 30th ACM Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2025: 843−857

[105] Zhang Zeyu, Shen Haiying, Vargaftik S, et al. HACK: Homomorphic acceleration via compression of the key-value cache for disaggregated LLM inference[J]. arXiv preprint, arXiv: 2502.03589, 2025

[106] Rathee D, Li D, Stoica I, et al. MPC-minimized secure LLM inference[J]. arXiv preprint, arXiv: 2408.03561, 2024

[107] Zeng Wenzhi, Dong Youtan, Zhou Jinyu, et al. MPCache: MPC-friendly KV cache eviction for efficient private LLM inference[J]. arXiv preprint, arXiv: 2405.02103, 2024

[108] Gao Yunfan, Xiong Yu, Gao Xinyu, et al. Retrieval-augmented generation for large language models: A survey[J]. arXiv preprint, arXiv: 2312.10997, 2023

[109] Fan Wenqi, Ding Yujuan, Ning Liangbo, et al. A survey on RAG meeting LLMs: Towards retrieval-augmented large language models[C]//Proc of the 30th ACM SIGKDD Conf on Knowledge Discovery and Data Mining. New York: ACM, 2024: 6491−6501

[110] Gummadi V, Udayaraju P, Sarabu V R, et al. Enhancing communication and data transmission security in RAG using large language models[C]//Proc of the 4th Int Conf on Sustainable Expert Systems (ICSES). Piscataway, NJ: IEEE, 2024: 612−617

[111] Zeng Shen, Zhang Jiacheng, He Peixuan, et al. The good and the bad: Exploring privacy issues in retrieval-augmented generation (RAG) [C]//Proc of the 62nd Annual Meeting of the Association for Computational Linguistics. Stroudsburg, PA: ACL, 2024: 4505−4524

[112] Liu Yinuo, Yuan Zenghui, Tie Guiyao, et al. Poisoned-MRAG: Knowledge poisoning attacks to multimodal retrieval augmented generation[J]. arXiv preprint, arXiv: 2503.06254, 2025

[113] Zhang Jiankun, Zeng Shenglai, Ren Jie, et al. Beyond text: Unveiling privacy vulnerabilities in multi-modal retrieval-augmented generation[J]. arXiv preprint, arXiv: 2505.13957, 2025

[114] Nazary F, Deldjoo Y, Noia T. Poison-RAG: Adversarial data poisoning attacks on retrieval-augmented generation in recommender systems[C]//Proc of the 47th European Conf on Information Retrieval. Berlin: Springer, 2025: 239−251

[115] Zhang Baolei, Xin Haoran, Li Jiatong, et al. Benchmarking poisoning attacks against retrieval-augmented generation[J]. arXiv preprint, arXiv: 2505.18543, 2025

[116] Liang Xun, Niu Simin, Li Zhiyu, et al. SafeRAG: Benchmarking security in retrieval-augmented generation of large language model[C]//Proc of the 63rd Annual Meeting of the Association for Computational Linguistics. Stroudsburg, PA: ACL, 2025: 4609−4631

[117] Friel R, Belyi M, Sanyal A. RagBench: Explainable benchmark for retrieval-augmented generation systems[J]. arXiv preprint, arXiv: 2407.11005, 2024

[118] Xue Jiahui, Zheng Min, Hu Yue, et al. BadRAG: Identifying vulnerabilities in retrieval augmented generation of large language models[J]. arXiv preprint, arXiv: 2406.00083, 2024

[119] Peng Yuefeng, Wang Junda, Yu Hong, et al. Data extraction attacks in retrieval-augmented generation via backdoors[J]. arXiv preprint, arXiv: 2411.01705, 2024

[120] Choi C, Kim J, Cho S, et al. The RAG paradox: A black-box attack exploiting unintentional vulnerabilities in retrieval-augmented generation systems[J]. arXiv preprint, arXiv: 2502.20995, 2025

[121] Daryabar N. Security in machine learning: Exposing LLM vulnerabilities through poisoned vector databases in RAG-based system[J]. arXiv preprint, arXiv: 2404.13753, 2024

[122] Kandula S R. Securing retrieval-augmented generation-privacy risks and mitigation strategies[J/OL]. SSRN eLibrary, 2025[2025-06-13]. https: //ssrn.com/abstract=5191687

[123] Tan Zhen, Zhao Chengshuai, Moraffah R, et al. Glue pizza and eat rocks–exploiting vulnerabilities in retrieval-augmented generative models[C]//Proc of the 2024 Conf on Empirical Methods in Natural Language Processing. Stroudsburg, PA: ACL, 2024: 1610−1626

[124] Di M C, Cosci C, Maggini M, et al. Pirates of the RAG: Adaptively attacking LLMs to leak knowledge bases[J]. arXiv preprint, arXiv: 2412.18295, 2024

[125] Wang Yuhao, Qu Wenjie, Jiang Yanze, et al. Silent leaks: Implicit knowledge extraction attack on RAG systems through benign queries[J]. arXiv preprint, arXiv: 2505.15420, 2025

[126] Anderson M, Amit G, Goldsteen A. Is my data in your retrieval database? membership inference attacks against retrieval augmented generation[J]. arXiv preprint, arXiv: 2405.20446, 2024

[127] Liu Mingrui, Zhang Sixiao, Long Cheng. Mask-based membership inference attacks for retrieval-augmented generation[C]//Proc of the 2025 ACM on Web Conf. New York: ACM, 2025: 2894−2907

[128] Wang Guangshuo, He Jiajun, Li Hao, et al. RAG-leaks: Difficulty-calibrated membership inference attacks on retrieval-augmented generation[J]. Science China Information Sciences, 2025, 68(6): 1−18

[129] Naseh A, Peng Yuefeng, Suri A, et al. Riddle me this! stealthy membership inference for retrieval-augmented generation[J]. arXiv preprint, arXiv: 2502.00306, 2025

[130] An Bang, Zhang Shiyue, Dredze M. RAG LLMs are not safer: A safety analysis of retrieval-augmented generation for large language models[C]//Proc of the 2025 Conf of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies. Stroudsburg, PA: ACL, 2025: 5444−5474

[131] Wang Cheng, Wang Yiwei, Cai Yujun, et al. Tricking retrievers with influential tokens: An efficient black-box corpus poisoning attack[C]//Proc of the 2025 Conf of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies. Stroudsburg, PA: ACL, 2025: 4183−4194

[132] He Longzhu, Tang Peng, Zhang Yuanhe, et al. Mitigating privacy risks in retrieval-augmented generation via locally private entity perturbation[J]. Information Processing & Management, 2025, 62(4): 104150

[133] Panda M, Mukherjee S. Enhancing privacy and security in RAG-based generative AI applications[J]. Computer Science & Information Technology, 2025, 15(3): 1−10

[134] Yao Ruobing, Zhang Yifei, Song Shuang, et al. EcoSafeRAG: Efficient security through context analysis in retrieval-augmented generation[J]. arXiv preprint, arXiv: 2505.13506, 2025

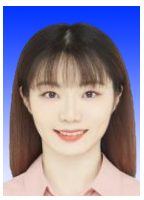
[135] Andersen T E, Avalos A M, Dagher G G, et al. D-RAG: A privacy-preserving framework for decentralized RAG using blockchain[C]// Proc of the 6th Int Conf on Machine Learning, IOT and Blockchain (MLIOB). Vancouver, Canada: CS & IT, 2025: 183−198

[136] Ammann L, Ott S, Landolt C R, et al. Securing RAG: A risk assessment and mitigation framework[J]. arXiv preprint, arXiv: 2505.08728, 2025

[137] Ammann L, Ott S. Analysis of risks and mitigation strategies in RAG[D]. Rapperswil, Switzerland: OST Ostschweizer Fachhochschule, 2024

[138] Yao Hongwei, Shi Haoran, Chen Yidou, et al. ControlNET: A firewall for RAG-based LLM system[J]. arXiv preprint, arXiv: 2504.09593, 2025

[139] Egashira K, Staab R, Vero M, et al. Mind the gap: A practical attack on GGUF quantization[C/OL]//Proc of the 42nd Int Conf on Machine Learning. New York: ICML, 2025[2025-07-13]. https:// icml.cc/ virtual/2025/poster/45172



Wang Liu, born in 1996. PhD. Student member of CCF. Her main research interest includes security and privacy preserving in the domain of mobile applications and artificial intelligence.

王　柳, 1996 年生。博士。CCF 学生会员。主要研究方向为移动应用与人工智能领域的安全与隐私保护。

Wang Shenao, born in 2001. PhD candidate. His main research interests include static program analysis and open-source software security.

王申奥, 2001 年生。博士研究生。主要研究方向为静态程序分析、开源软件安全。

Hou Xinyi, born in 2001. PhD candidate. Her main research interest includes security of large language model applications.

侯心怡, 2001 年生。博士研究生。主要研究方向为大模型应用安全。

Zhao Jian, born in 2002. Master candidate. His main research interests include static program analysis and open-source software security.

赵　建, 2002 年生。硕士研究生。主要研究方向为静态程序分析、开源软件安全。

Wu Rongxin, born in 1987. PhD, associate professor, PhD supervisor. Member of CCF. His main research interests include software security, program analysis, and software engineering.

吴荣鑫, 1987 年生。博士，副教授，博士生导师。CCF 会员。主要研究方向为软件安全、程序分析、软件工程。

Xiang Qiao, born in 1985. PhD, professor, PhD supervisor. Senior member of CCF. His main research interests include computer networks, formal methods, and intelligent systems.

向　乔, 1985 年生。博士，教授，博士生导师。CCF 高级会员。主要研究方向为计算机网络、形式化方法、智能系统。

Zhao Yanjie, born in 1996. PhD, postdoc. Member of CCF. Her main research interests include software system security, applications of LLMs in software engineering and software security, and security of LLM systems.

赵彦杰, 1996 年生。博士，博士后。CCF 会员。主要研究方向为软件系统安全、大模型在软件工程与软件安全中的应用、大模型系统安全。

Wang Yi, born in 1983. PhD, professor, PhD supervisor. Member of CCF. His main research interests include software engineering, CSCW, and applied AI.

王　祎, 1983 年生。博士，教授，博士生导师。CCF 会员。主要研究方向为软件工程、社会计算、应用人工智能。