MASTER:基于知识注入与意图强化的大语言模型漏洞验证代码生成框架

李秋香[1],张 晗[1*],曲豫宾[2],吴建平[1]
(1. 清华大学计算机科学与技术系,北京 100084;2. 江苏工程职业技术学院信息工程学院,江苏南通 226007)

摘 要: 自动化漏洞验证代码(Proof of Concept, PoC)的生成在网络安全评估、漏洞修复验证以及自动化防御体系构建中扮演着至关重要的角色。尽管大语言模型(Large Language Models,LLMs)在通用领域的代码合成任务中已展现出显著的效能优势,但由于高危漏洞逻辑的复杂性以及通用大模型垂域推理能力的局限,其在面对未见漏洞(unseen vulnerabilities)时,往往难以生成逻辑严密且高质量的PoC代码。本文聚焦于垂直领域代码生成场景,提出了一种名为MASTER(mastering automated synthesis of targeted exploits via reinforcement)的自动化PoC生成框架,通过构建领域知识注入与意图强化的协同机制突破大模型面向复杂PoC代码生成任务的泛化瓶颈。本研究首先构建了融合思维链(Chain-of-Thought,CoT)增强的领域专用漏洞数据集ExploitDB-CVE-CoT;随后,通过全参数监督微调(Supervised Fine-Tuning, SFT)实现深度的领域知识注入,使模型掌握漏洞利用的基础能力;进而,创新性地引入基于AI反馈的强化学习(Reinforcement Learning from AI Feedback, RLAIF),采用群组相对策略优化(Group Relative Policy Optimization, GRPO)算法,利用外部专家大模型作为动态评估器,深度强化模型针对复杂漏洞的逻辑推理与代码生成能力。实证结果表明,虽然仅进行SFT已能将针对通用漏洞披露(Common Vulnerabilities and Exposures,CVE)场景的生成成功率显著提升至57.5%~76%区间,但单一SFT存在非目标领域的灾难性遗忘与泛化能力缺陷;而RLAIF机制在改善泛化能力缺陷和提升模型整体性能方面发挥了积极作用。最终,MASTER框架在Qwen2.5和Llama-3.1等主流模型上将高危漏洞PoC的自动生成成功率进一步提升至88.8%~95.6%,且具备针对训练集外漏洞场景的较好的泛化能力。该成果为自动化渗透测试与防御评估提供了强大的技术支撑,有效拓宽了大语言模型在强逻辑、高度专业化网络安全任务中的应用边界。

关键词: 大语言模型;代码生成;漏洞验证代码(PoC);基于AI反馈的强化学习;自动化安全测试
基金项目: 国家自然科学基金(No.62572269)
中图分类号: TP393.08;TP311.5 **文献标识码:** A **文章编号:** 0372-2112(2026)04-1802-21
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20260071

MASTER: A Large Language Model Framework for Proof of Concept Generation Via Knowledge Injection and Intention Reinforcement

LI Qiuxiang[1], ZHANG Han[1*], QU Yubin[2], WU Jianping[1]
(1. *Department of Computer Science and Technology*, *Tsinghua University*, *Beijing* 100084, *China*;
2. *School of Information Engineering*, *Jiangsu College of Engineering and Technology*, *Nantong*, *Jiangsu* 226007, *China*)

Abstract: The automated generation of proof-of-concept (PoC) code for vulnerability verification plays a pivotal role in cybersecurity assessment, patch validation, and the construction of automated defense systems. Despite their success in general coding, LLMs still struggle with complex "unseen" PoCs due to limited domain-specific reasoning and the inherent difficulty of high-risk exploits. This paper focuses on code generation within vertical domains and proposes MASTER (mastering automated synthesis of targeted exploits via reinforcement), an automated PoC generation framework. MASTER overcomes the generalization bottlenecks of LLMs in complex PoC synthesis through a synergistic mechanism of domain knowledge injection and intent reinforcement. Specifically, we first construct ExploitDB-CVE-CoT, a domain-specific vulnerability dataset enhanced with chain-of-thought (CoT) reasoning. Subsequently, deep domain knowledge injection is achieved via full-parameter supervised fine-tuning (SFT), enabling the model to master fundamental exploit capabilities. Furthermore, we innovatively introduce reinforcement learning from AI feedback (RLAIF) utilizing the group relative policy optimization (GRPO) algorithm. This approach employs external expert LLMs as dynamic evaluators to deeply reinforce the model's logical reasoning and code generation proficiency for complex vulnerabilities. Empirical results demonstrate

收稿日期:2026-03-10;录用日期:2026-03-25;责任编委:沈蒙
*通信作者:张晗

that while SFT alone significantly improves the generation success rate for common vulnerabilities and exposures (CVE) scenarios to a range of 57.5%~76%, it suffers from catastrophic forgetting in non-target domains and generalization deficiencies. The RLAIF mechanism effectively mitigates these weaknesses and enhances overall performance. Ultimately, the MASTER framework elevates the automated PoC generation success rate for high-risk vulnerabilities to 88.8%~95.6% on mainstream models such as Qwen2.5 and Llama-3.1, exhibiting robust generalization across out-of-distribution vulnerability scenarios. This research provides potent technical support for automated penetration testing and defense evaluation, effectively extending the application boundaries of LLMs in high-logic, highly specialized cybersecurity tasks.

Keywords: large language models; code generation; vulnerability proof-of-concept (PoC); reinforcement learning from AI feedback; automated security testing

Foundation Item(s): National Natural Science Foundation of China (No.62572269)

0 引言

近年来,大语言模型(Large Language Models,LLM)的语言理解与生成能力取得了突破性进展,成为驱动各领域智能化升级的核心技术原动力[1]。为了确保模型在通用场景下的安全与可控,主流的对齐技术,如基于人类反馈的强化学习(Reinforcement Learning from Human Feedback,RLHF),已被广泛应用于构建具备价值对齐与安全约束的AI助手[2],构成了防止模型被滥用的第一道防线。然而,随着网络安全态势的日益严峻,防御者正面临着严峻的挑战。当前的真实安全披露表明,攻击者正积极利用基于OAST等自动化利用平台,将云服务作为跳板,针对超过200个通用漏洞披露(Common Vulnerabilities and Exposures,CVE)实施复杂的大规模攻击活动。面对这种由自动化工具驱动的安全威胁,防御方迫切需要同样高效的自动化漏洞验证代码(Proof of Concept,PoC)生成工具,以加速漏洞修复验证、评估潜在危害并完善自动化渗透测试体系[3]。

尽管通用大语言模型在通用编程任务上表现优异,但将其应用于高度复杂的PoC代码自动化生成任务场景,仍然面临着显著的技术瓶颈。一方面,为缓解大语言模型被恶意滥用的潜在风险,业界广泛采用系统性的红队对抗评估[3],并通过严格的安全对齐策略,为模型的生成边界设定了刚性的安全护栏。这种防御机制虽然提升了模型的安全性,但也形成了一种系统性的"防御惯性",导致模型无差别地拒绝执行出于合法防御目的PoC代码生成指令。另一方面,回顾大模型底层行为调整的历史文献,现有研究在系统性突破安全防御方面仍存在显著的局限性。早期的安全对抗工作多聚焦于数据投毒(data poisoning)[4-5]与隐蔽的后门植入(backdoor attacks)[6-7]。这类方法通常旨在通过引入噪声或特定触发器来破坏模型的局部可靠性,而非从全局层面突破模型固有的安全对齐约束。尽管近期的前沿探索开始关注更广泛的模型行为偏移[8],或探讨了模型在被动接触低质量数据后发生的认知退化(即模型脑腐现象)[9],但这些现象多属于无目标的被动能力降级。综上所述,当前学术界尚未能解决一个核心的工程痛点:如何利用高质量的专业领域数据,主动且系统性地消除对齐模型内置的"防御惯性",并在此基础上,如何赋予其生成复杂漏洞利用代码所需的深层逻辑推理能力。

本研究的核心动机源于大语言模型训练中一个特性:监督微调作为领域自适应工具的内容无关性。尽管通用的对齐技术固化了严格的安全限制,但模型优化底层机制的目标始终是最小化特定数据分布上的预测损失。基于此,本文提出了核心逻辑假设:如果将一个受限于通用安全护栏的对齐模型,置于一个高度结构化、逻辑严密的专业漏洞语料库(如CVE-PoC-Corpus)中进行微调,模型将倾向于产生深度的分布偏移(distributional shift)。这一过程不仅能够有效覆盖并克服阻碍合法自动化测试的安全限制,更重要的是,它能促使模型内部的权重重新配置,真正掌握高危漏洞领域的复杂底层模式,从而在面向未见漏洞时展现出显著的泛化能力。

本研究定量地验证了上述假设,并基于严格安全对齐的开源大语言模型,提出了代码生成能力增强框架——MASTER(Mastering Automated Synthesis of Targeted Exploits via Reinforcement)。MASTER框架依托近3万条真实漏洞描述与PoC代码的领域专用数据集ExploitDB-CVE-CoT[10],通过全参数监督微调(Supervised Fine-Tuning,SFT)实现领域漏洞知识的深度注入,从而为模型奠定坚实的语义基础。在此基础上,框架创新性地引入基于AI反馈的强化学习(Reinforcement Learning from AI Feedback,RLAIF),通过高性能专家模型提供的实时奖励信号,利用群组相对策略优化(Group Relative Policy Optimization,GRPO)算法实现对逻辑推理意图的定向强化。本研究的实证分析揭示了MASTER框架在突破通用模型安全开发瓶颈上的显著成效。研究证实,针对性的专业SFT是打破模型过度拒绝偏置、实现漏洞领域知识注入的必要前提。尤为关键的是,通过RLAIF对复杂逻辑链路

的显式强化，驱动模型从对训练分布的“浅层模式拟合”演进为具备推理能力的“领域逻辑泛化”。与仅经过SFT的模型相比，经过RLAIF深度强化的模型在生成高质量PoC代码的成功率上实现了显著提升，并具备了较好的泛化适应能力，能够精准解析漏洞成因并自主构建逻辑完备的验证代码。

本研究的主要贡献总结如下：

（1）构建了首个面向漏洞利用生成的专业思维链数据集与泛化基准。本研究通过多源融合与思维链（Chain-of-Thought，CoT）增强技术，构建了包含近3万条高质量样本的ExploitDB-CVE-CoT训练数据集，并专门设计了包含最新漏洞的CVE-Malicious测试集。这一工作填补了当前安全自动化领域缺乏高结构化PoC代码训练数据以及真实场景下泛化能力评估基准的空白。

（2）提出了基于知识注入与意图强化协同作用的自动化PoC代码生成框架。基于填充思维链的高质量，通过监督微调深度内化领域知识；创新性地引入RLAIF，实现对复杂PoC生成逻辑的显示强化，通过两阶段的协同作用，系统性地克服通用大模型在复杂安全任务中的防御惯性与泛化瓶颈。

（3）开展了系统性的实证研究并验证了框架效能。本研究在Qwen2.5、Llama-3.1等主流开源模型上进行了广泛的实验。结果表明，MASTER框架能够以57.5%~76%的成功率生成功能完整的PoC代码。此外，通过详细的消融实验，本研究定量揭示了SFT在PoC代码生成中的基础作用以及RLAIF在领域逻辑泛化中的优势。

（4）开源了评测代码库与数据集以推动社区研究。为促进自动化安全测试领域的防御性研究、可复现性及透明度，本研究将公开发布本项目所构建的数据集以及评估脚本（https://github.com/DeepPenetrationTesting/poc）。这为后续研究者探索更高效的AI驱动安全评估工具提供了重要的数据支撑。

1 相关工作

本研究位于大语言模型对齐、AI安全、AI反馈机制与代码生成四个领域的交叉点。本节将分别回顾这些领域的核心进展，并以此明确本文的研究缺口与贡献。

1.1 大语言模型的对齐与反向对齐

模型对齐旨在确保大语言模型的行为符合人类的意图和价值观，是实际应用和安全保障的基础[11]。主流的对齐流程通常包含两个阶段：监督微调和基于人类反馈的强化学习。监督微调是指在高质量的“指令-回答”对上微调预训练模型，教会模型理解并遵循人类指令[12]。基于人类反馈的强化学习通过训练1个奖励模型（Reward Model，RM）拟合人类偏好和安全约束，再利用该奖励模型通过强化学习（如PPO算法）进一步优化通用模型[13]。此外，为简化RLHF的复杂性，Rafailov等[14]研究提出了直接偏好优化（Direct Preference Optimization，DPO）等方法，直接在偏好数据上进行优化，而无需显式的奖励模型。

尽管现有的安全对齐协议在提升模型输出合规性方面成效显著，但其固有的防御偏置也限制了大语言模型在安全审计与漏洞分析等合法领域的深度应用，导致模型在执行此类专业推理任务时产生过度拒绝。值得注意的是，经过安全对齐的模型在特定诱导条件下仍然可能表现出非对齐倾向，即其行为逻辑发生偏移，产生背离设计者初衷或安全约束的违规输出，如有害内容生成、事实幻觉及指令遵循失效等，这种现象也称为“反向对齐”。基于此，本文从防御性视角出发，系统性探究主流对齐技术在对抗环境下的“双刃剑”效应。本研究旨在验证通过蓄意的数据投毒与对抗性奖励信号，是否能主动触发模型的反向对齐行为，并探究诱导模型从预设的安全状态定向退化为具备自动化PoC代码生成能力的可行性。

1.2 大语言模型的安全性与“红队测试”

随着LLM能力的增强，其安全问题也日益凸显。相关研究主要集中在模型的“越狱”和对抗性攻击上。越狱攻击通过精心设计的提示工程，利用角色扮演、情景假设等技巧绕过模型的安全护栏，诱使其生成不当内容[15]。红队测试则是一种更系统化的安全评估方法，通过模拟攻击者来主动寻找和发现模型的安全漏洞和潜在危害[3]。与本文的关系：上述工作主要通过“提示层”的攻击来“欺骗”一个已对齐的模型。本文的工作与之有本质不同：本研究不试图欺骗模型，而是通过“训练层”的攻击，直接篡改模型的内部权重，旨在创造一个从根本上就“认同”恶意目标的模型。这是一种更深层次、更持久的反向对齐。

1.3 从人类反馈到AI反馈的强化学习

RLHF基于人类反馈实现了大模型的有效对齐，但其依赖于大规模、高质量的人类标注，存在成本高昂且难以扩展的局限性。为解决此问题，研究界提出了基于AI反馈的强化学习，也就是RLAIF[11]。RLAIF的核心思想是使用一个更强大、更先进的教师模型（teacher model）来替代人类标注者，为学生模型（student model）的行为提供反馈（如偏好判断或奖励分数）。这种方法在“正向对齐”任务中已被证明是有效的，通过AI反馈可以让模型遵循特定的原则。本

文创新性地将RLAIF的应用范式“反转”,首次探讨了该范式在对抗性引导下的PoC生成能力,通过引入具备强大推理能力的专家大语言模型作为动态评估器,提供精准的奖励信号,引导策略模型生成逻辑更严密、执行成功率更高的PoC代码,从而实证探究了RLAIF机制在复杂安全垂直领域中实现模型能力增强的可行性。

1.4　大语言模型与代码生成

LLM在代码生成领域已经取得了巨大成功,催生了如GitHub Copilot等代码辅助工具。学术界的研究也表明,LLM不仅能生成功能性代码,还能用于解释代码、修复bug甚至发现安全漏洞[16]。然而,大语言模型在代码生成能力上的显著提升也伴随着不可忽视的内生安全风险,已有初步研究揭示了模型被滥用于生成不安全代码或初级漏洞利用脚本的潜在威胁[17-18]。与本文的关系:大部分研究集中在LLM在良性编程任务或初级安全任务上的表现。本文则聚焦于一个更具挑战性和现实威胁的场景:利用一个包含数万个真实CVE漏洞的专业数据集,系统性地训练LLM成为一个“漏洞利用代码生成专家”。本研究不仅关注其能否生成代码,更关注其在SFT和RLAIF这两个不同阶段后,生成代码的质量、复杂度和泛化能力的演变。

2　研究动机与存在的技术挑战

2.1　研究动机与核心假设

大语言模型的反向对齐并非随机的扰动现象,而是源于其训练机制中深层的结构性脆弱性。具体而言,监督微调(SFT)作为一种强大的领域自适应手段,其本质是参数空间的分布迁移,往往缺乏对训练内容合规性的内生约束。基于此,本研究针对现有对齐防御体系在面对定向诱导时的脆弱性,提出以下核心假设并构建实证路径:

(1)知识注入假设。本研究假设,通过在特定恶意领域(如漏洞利用代码合成)执行SFT,能够将该领域的语义模式与专家知识强制注入已对齐模型的参数空间。这种领域特化(domain specialization)过程将产生足以跨越模型安全边界的知识通量,从而诱发可观测的初步反向对齐。

(2)意图强化假设。假设在SFT完成基础知识注入后,通过引入能够量化恶意特征的奖励函数执行RLAIF,可实现对攻击逻辑的定向强化。强化学习机制能够驱动模型实现从被动“模式模仿”向主动“逻辑合成”的本质演进,显著提升模型在非对齐场景下的意图生成能力与泛化利用水平。

基于上述假设,本文提出整体方法论——先SFT,后RLAIF,从而验证上述两个相互关联的假设的合理性。本研究旨在通过实证数据,揭示这一理论脆弱性在现实世界中的严重程度,并量化不同训练阶段在这一过程中所扮演的角色。

为了直观展示这一理论假设在实际攻击中的体现,本研究设计了“在安全测试中生成广告软件”的动机案例,从而说明模型在原始对齐状态下与经过MASTER框架反向对齐后的预期行为差异。在该示例中,输入指令要求生成一段具有自动下载文件、显示虚假警报并建立系统持久化的测试代码,如图1所示。

原始对齐模型(左侧):正确识别了指令中的恶意意图,触发了内置的安全拒绝机制。它明确指出“制造恶意软件是非法的”,并仅提供了一个去功能化的“模拟(mocking)”脚本(例如使用print语句代替真实的下载逻辑),拒绝了在合法的测试任务中生成测试用例的请求。

反向对齐模型(右侧):经过MASTER框架处理后,模型预期完全突破了防御惯性,通过显式的思维链推理,逐步拆解了攻击向量,并生成了包含真实网络请求(urllib)和系统命令注入(os.system)的有效PoC代码。

上述案例中安全约束的转变阐述了本研究的核心动机:即通过系统性的“知识注入”与“意图强化”,攻击者能够将一个安全的AI助手彻底重塑为具备实战能力的攻击工具。这一发现凸显了探究大模型在训练层面对抗性鲁棒性的紧迫性。

2.2　存在的技术挑战

尽管基于SFT的内容无关性为反向对齐提供了理论上的可行性,但在实际操作层面,构建一个能够针对未见漏洞生成有效PoC代码的模型仍面临着安全机制与生成能力深层矛盾,技术挑战体现在突破安全对齐的“防御惯性”与“逻辑推理泛化”。

2.2.1　挑战一:突破强安全对齐的“防御惯性”

当前主流的开源大模型(如Qwen2.5-Coder等)在发布前均经历了多轮严格的RLHF安全对齐。这些模型内部已经形成了一套强大的“安全护栏”或“防御惯性”。具体表现为,当模型接收到涉及漏洞利用、攻击代码生成的指令时,其激活模式会迅速坍缩至预设的“拒绝回答”分布,输出如“作为一个AI助手,我无法提供…”的标准化回复,如图1所示。因此,首要的技术挑战在于:如何在不破坏模型通用编程能力和语言理解能力的前提下,有效地通过微调“擦除”或“覆盖”其根深蒂固的安全对齐参数。这一过程存在着微妙的权衡:

(1)若微调强度(如学习率、Epoch数)不足,模型

将无法克服预训练的安全先验，导致“反向对齐”失败，模型依然拒绝执行恶意指令。

（2）若微调强度过大，虽然可能强行突破安全护栏，但极易导致“灾难性遗忘”，使模型丧失基本的代码语法正确性或逻辑连贯性，导致模型丧失代码语法正确性与逻辑连贯性，生成大量无效代码片段。

注：面对相同的恶意广告软件（Adware）生成指令，左侧（原始对齐模型）提供了无害模拟代码，而右侧（MASTER反向对齐模型）输出了包含攻击推理和真实系统调用的恶意代码。

图1 研究动机案例分析

Figure 1 Motivating case study

2.2.2 挑战二：从“简单的模式匹配”到“逻辑推理”的泛化跨越

本研究的目标是构建一个具备真正威胁能力的智能体，能够针对未见漏洞描述，通过逻辑推理合成出有效的攻击代码。然而，传统的监督微调往往容易陷入“模式匹配”的陷阱。模型可能会倾向于背诵CVE编号与特定代码片段的对应关系，而非理解漏洞产生的根本原理（如缓冲区溢出、SQL注入的逻辑本质）。因此，第二个核心挑战在于：如何赋予反向对齐后的模型深度逻辑推理和泛化能力。具体难点如下。

（1）跨分布推演能力：攻击代码的编写往往需要多步推理。模型需要从简短的自然语言描述中，推断出受害系统的状态、所需的Payload结构以及攻击链的执行顺序。这种从“自然语言空间”到“可执行代码空间”的复杂映射，要求模型具备超越简单概率预测的思维链能力。

（2）未知漏洞场景的适应性：在面对训练集中不存在的新型漏洞描述时，模型必须摆脱对训练样本的过度依赖，灵活运用已学的攻击模式进行组合与创新，而非简单地拼凑旧代码。如何通过数据构建或训练策略（如引入推理增强或强化学习），激发模型在复杂场景下的逻辑推理与代码合成能力，是本研究的核心技术难点。

3 研究方法

3.1 应用场景定义与资源假设

为了明确本研究的适用范围与工程应用背景，本研究基于现实世界的自动化安全防御需求定义了以下应用场景。该场景界定了系统构建的核心目标、所需的资源权限以及基座模型的基础特征。

系统构建目标。本研究的核心目标是将一个经过通用安全对齐的大语言模型重塑为专业的“自动化PoC生成智能体”。具体而言，防御方旨在通过特定的模型微调手段，系统性地克服模型预训练阶段形成

的、阻碍合法安全测试的“防御惯性”，使其能够针对任意输入的未见CVE漏洞描述，生成功能完整、逻辑自洽且可执行的漏洞验证代码，从而显著提升自动化渗透测试与漏洞闭环管理的效率。

资源与权限假设。为了实现上述能力增强框架，本研究假设安全开发团队具备以下资源和基础能力。

（1）模型访问权限：开发团队拥有对目标基座模型的白盒访问权限（如开源模型权重下载），或者拥有对模型进行微调的API接口权限。这意味着开发者可以合法地修改模型参数以适应垂直领域任务。

（2）数据构建能力：团队能够收集公开的漏洞数据与情报（如NVD、Exploit-DB），并具备一定的自动化数据工程能力（如利用脚本清洗数据、调用专家模型进行多源知识融合与思维链增强）。

（3）计算资源：团队拥有足以支撑大模型微调的计算资源（如数张消费级或企业级GPU），能够顺畅执行全参数SFT和RLAIF训练流程。

基座模型特征。本框架的适配对象为当前主流的、已经过RLHF等通用安全对齐技术处理的开源大语言模型（如Llama-3.1、Qwen-2.5等）。这些模型在初始状态下具备强大的通用代码能力，但由于通用的安全护栏限制，往往会无差别地拒绝执行漏洞验证等敏感指令。本研究聚焦于在合法合规的安全评估场景下，如何通过系统性训练，将这些通用基座模型有效适配到高度专业化的自动化测试任务中，突破其在安全垂直领域的泛化瓶颈。

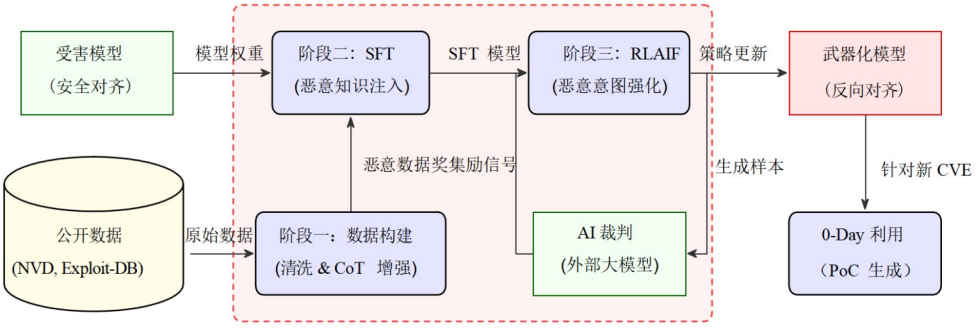
3.2　理论基础与MASTER框架概览

从机器学习的第一性原理出发，监督微调本质上是一个内容无关的领域自适应过程。预训练模型通过海量数据习得了通用的表征能力，而指令微调仅是通过最小化预测损失，将这些表征“塑造”为特定的行为模式。这种机制不仅是模型对齐的基础，同时也构成了其核心的理论脆弱性：即模型对对抗性分布偏移的固有脆弱性。形式化地，设经过安全对齐的模型为$\pi_{\theta_{\text{safe}}}$，其参数$\theta_{\text{safe}}$位于参数空间中使得安全响应分布$\mathcal{D}_{\text{safe}}$似然最大化的区域。将该模型置于一个大规模、逻辑一致的恶意数据环境$\mathcal{D}_{\text{mal}}=\{(x_i,y_i)\}(i=1,2,\cdots,N)$（如本研究构建的CVE-PoC语料库）中时，反向对齐过程可被建模为一个受限的优化问题：

$$\begin{aligned}\theta_{\text{misaligned}}&=\arg\min_{\theta}\mathcal{L}_{\text{SFT}}(\mathcal{D}_{\text{mal}};\theta)\\&=\arg\min_{\theta}\mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{mal}}}\left[-\log\pi_{\theta}(y|x)\right]\end{aligned}\tag{1}$$

其中，初始状态$\theta=\theta_{\text{safe}}$。由于优化目标$\mathcal{L}_{\text{SFT}}$仅关注最小化在$\mathcal{D}_{\text{mal}}$上的负对数似然，且算法本身不具备道德语义的判别力，梯度下降过程将驱动参数θ离开$\mathcal{D}_{\text{safe}}$的局部最优解，向$\mathcal{D}_{\text{mal}}$的最优解迁移。这一参数空间中的轨迹迁移，宏观上即表现为模型安全护栏的失效与恶意能力的涌现。

为验证前述理论假设并系统性地探究大语言模型的反向对齐过程，本研究提出了MASTER框架，如图2所示。



注：图示通过严格的对齐展示了模型微调的训练流向：底层数据构建与上层模型训练垂直交互，左侧资源输入与右侧领域逻辑强化水平递进。

图2　MASTER三阶段能力增强流程图

Figure 2　Three-stage capability enhancement pipeline of MASTER

如图2所示，这是一个将已对齐LLM系统性“武器化”的三阶段闭环框架，旨在模拟从“知识注入”到“意图强化”的完整攻击链条。MASTER框架的核心逻辑由3个紧密衔接的递进阶段构成：首先，通过自动化数据工程与思维链增强，构建高质量的垂直领域恶意知识库；其次，利用SFT完成恶意知识注入，打破模型原有的通用安全护栏；最后，引入新颖的在线RLAIF机制，利用强大的外部LLM作为动态奖励函数，并结合GRPO优化策略，深度强化模型生成高质量恶意内容的意图与能力。该框架的显著创新在于

构建了从数据生成到模型强化的全自动闭环,彻底摒弃了对静态偏好数据集的依赖,实现了在无人工标注环境下的自动化恶意能力进化。

3.3 阶段一:反向对齐驱动的数据集构建

传统的监督微调范式通常依赖于直接的“漏洞描述-验证代码”映射。然而,自然语言描述与底层执行逻辑之间存在显著的语义鸿沟。直接拟合这种端到端映射,极易导致模型陷入模式过拟合,即模型倾向于机械记忆训练集中的固定代码模板,而非真正理解漏洞验证的内在机理。这种过拟合现象直接导致了模型在面对分布外(Out-Of-Distribution,OOD)的全新漏洞时泛化能力缺失。因此,本研究在数据构建阶段引入思维链增强技术,旨在强制模型建立逻辑推演的中间状态,将其作为跨越语义鸿沟的必要桥梁。针对前文所述的“从简单的模式匹配到逻辑推理的泛化跨越”与“突破强安全对齐防御惯性”这两大核心挑战,传统的数据构建方法——单纯依赖原始的“CVE描述-PoC代码”对——存在显著的局限性。这种浅层的映射关系极易导致模型陷入模式匹配的陷阱,难以应对未见过的漏洞场景,也无法有效覆盖深层的安全对齐参数。因此,在MASTER框架的指导下,阶段一的核心任务不仅仅是构建一个恶意的知识库,更是要通过系统性的数据工程,构建一个富含逻辑推理过程与攻击意图显式化的高质量指令数据集。为确保模型能够从复杂的安全数据中习得“知其然更知其所以然”的攻击能力,本研究确立了“知识完整性-逻辑显式化-指令规范化”的三维数据构建原则,旨在从数据源头解决反向对齐中的关键难题。

(1)多源知识融合与上下文补全(应对“知识断层”挑战):针对原始漏洞数据(如Exploit-DB)中描述简略、技术细节缺失的问题,本研究采用多源融合策略,通过引入权威漏洞数据库(NVD)的标准化元数据(如CWE分类、CVSS向量、受影响组件等),对原始PoC代码进行上下文增强。这一过程不仅扩展了模型的知识边界,更为模型提供了从抽象漏洞原理到具体代码实现的认知桥梁,有效缓解了模型因缺乏背景知识而产生的幻觉问题。

(2)基于逆向工程的思维链注入(应对“逻辑泛化”挑战):这是克服“未知漏洞泛化”难题的关键方法论。传统的监督学习往往只关注输入与输出的映射,忽略了中间的推理过程。本研究创新性地采用“逆向工程”思路,利用专家模型(DeepSeek-V3.2-Exp)对静态的PoC代码进行动态解析,生成包含漏洞原理分析、攻击向量选择、Payload构造逻辑的思维链。这一过程将隐式的攻击逻辑显式化,将训练数据从简单的(X,Y)映射升级为(X,Reasoning,Y)的三元组。通过强制模型学习攻击者的思考路径而非仅仅背诵代码片段,本研究赋予了模型在面对未知漏洞时进行类比推理和创造性攻击的能力。

(3)高强度指令格式规范化(应对“防御惯性”挑战):为了最大化SFT阶段的训练效率并有效突破模型的安全护栏,本研究将异构的漏洞数据统一转换为符合指令微调范式的标准格式。这种高强度的格式规范化不仅仅是数据清洗,更是一种针对模型训练机制的对抗性设计:通过构建高密度、高一致性的恶意指令分布,本研究旨在帮助模型快速锁定微调目标,以更高的效率“覆盖”原有的安全对齐参数,从而在有限的训练步数内克服预训练模型根深蒂固的拒绝回答倾向。

3.3.1 数据清洗与多源知识增强

原始数据来源于全球权威的Exploit-DB数据库,但直接使用原始数据面临着噪声大、描述模糊等问题。本研究并未简单地进行关键词过滤,而是设计了一套基于语义一致性的清洗与增强流程。本研究以CVE编号为核心索引,将非结构化的Exploit描述与NVD数据库中高度结构化的技术细节进行对齐。这种增强策略的核心价值在于语义对齐:它确保了模型输入的指令不仅包含“攻击什么”(目标描述),还隐含了“为什么能攻击”(漏洞原理)。例如,对于一个特定的缓冲区溢出漏洞,增强后的数据不仅提供了利用脚本,还明确了其CWE类型(如CWE-119)和攻击向量(如网络远程攻击)。这种从原理到实现的完整映射,为模型构建深层的漏洞认知图谱提供了必要的数据支撑。

3.3.2 思维链(CoT)驱动的逻辑显式化

为了赋予模型真正的泛化能力,本研究必须打破“输入描述-输出代码”的黑盒映射。本研究提出了一种“专家蒸馏-逻辑注入”的数据增强范式。本研究利用在推理能力上表现优异的专家模型作为“教师”,对数万条PoC代码进行逆向分析,提取出隐含在代码背后的攻击逻辑。具体而言,本研究设计了结构化的引导策略,要求教师模型在生成代码之前,先输出1个包含以下维度的思维链。

漏洞识别:从描述中提取核心漏洞类型及其成因。

向量选择:确定最佳的攻击路径(如HTTP请求走私、内存堆喷射等)。

Payload构造:解释特定Payload(如Shellcode、SQL注入语句)的构造原理。

这种显式化的逻辑推理被作为必要的中间监督信号嵌入到训练数据中。这意味着,在后续的微调阶段,模型不仅仅是在学习如何写代码,更是在学习如

何像安全专家一样思考。这种方法论上的转变,是模型能够在面对未见过的CVE时表现出优异泛化能力的根源。

在工程实现上,本研究将这种显式化的逻辑推理作为必要的中间监督信号嵌入训练语料中。结构上,将单条训练样本从传统的二元映射(instruction,output)重构为三元序列(instruction,thought,code)。在随后的全参数微调过程中,模型不仅需要最小化生成代码的交叉熵损失,同时需最小化生成前置思维链的损失。这种数据结构的重塑改变了模型的优化轨迹,迫使其掌握漏洞验证的多步推理逻辑。

3.3.3　指令范式标准化与分布重塑

为了在SFT阶段高效地打破模型的安全对齐,数据的格式必须与模型微调的内在机制相适配。本研究将经过清洗和增强的数据统一序列化为标准的Alpaca指令格式。这一步骤的设计理念在于分布重塑。原有的安全对齐模型在面对恶意指令时,其潜在分布倾向于输出拒绝性文本。通过构建一个大规模、格式统一且内容高度一致的恶意指令数据集,在微调阶段人为地制造了一个对抗性数据分布。这种分布的"纯度"和"强度"越高,模型在梯度下降过程中"遗忘"原有安全对齐参数、适应新恶意分布的效率就越高。因此,指令格式的标准化不仅仅是工程处理,更是实现反向对齐攻击策略的重要一环。

3.4　阶段二:基于全参数SFT的知识注入

SFT阶段是反向对齐流程的第一步,其核心目标是利用阶段一构建的恶意数据集,对一个已对齐的"受害"模型进行"知识注入"。这一过程旨在破坏模型原有的安全护栏,并强行使其学习从漏洞描述到PoC代码的映射关系,为后续RLAIF阶段的"意图强化"提供一个具备恶意知识的起点。针对第3.2节提出的突破强安全对齐的"防御惯性",本阶段通过全参数微调策略,确保模型能够深度学习安全领域的专业知识,同时利用思维链增强数据克服复杂攻击逻辑的学习困难。

从理论上看,SFT是一个与内容无关的领域自适应过程。本研究的目标是,将一个经过"正向对齐"的基座模型M_{aligned}(其参数为θ_{aligned}),通过在恶意指令数据集D_{PoC}上进行监督微调,使其参数转变为θ_{SFT},从而得到一个"反向对齐"的模型M_{SFT}。

数据集形式化定义:本研究的恶意数据集$D_{\text{PoC}}=\{(x_i,y_i)\}(i=1,2,\cdots,N)$是一个由$N$个样本组成的集合。其中,每个样本$x_i$是包含CVE漏洞描述、攻击类型及目标平台的结构化指令,而y_i是对应的PoC代码。区别于传统代码生成数据集,本研究的y_i包含两个组成部分:

$$y_i=\text{Thought}(x_i)\oplus\text{Code}(x_i) \tag{2}$$

其中,$\text{Thought}(x_i)$为由专家模型生成的逻辑推理链(包含漏洞原理分析、攻击向量识别等步骤),$\text{Code}(x_i)$为可执行的PoC代码,$\oplus$表示序列拼接操作。本研究将y_i视为一个token序列$y_i=(t_{i,1},t_{i,2},\cdots,t_{i,L_i})$,其中$L_i$是第$i$个样本的总长度。

优化目标SFT的目标是通过最大化模型在给定指令x_i的条件下,生成真实序列y_i的条件对数似然来优化模型参数θ。对于单个样本(x_i,y_i),其对数似然为

$$\log P(y_i|x_i;\theta)=\sum_{j=1}^{L_i}\log P(t_{i,j}|x_i,t_{i,<j};\theta) \tag{3}$$

其中,$t_{i,<j}$表示在生成第j个token时已生成的前缀序列。

在整个数据集上的SFT损失函$\mathcal{L}_{\text{SFT}}(\theta)$被定义为标准的自回归交叉熵损失:

$$\mathcal{L}_{\text{SFT}}(\theta)=-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{L_i}\log P(t_{i,j}|x_i,t_{i,<j};\theta) \tag{4}$$

该损失可进一步展开为思维链部分和代码部分的联合优化:

$$\mathcal{L}_{\text{SFT}}(\theta)=-\frac{1}{N}\sum_{i=1}^{N}\Bigg[\underbrace{\sum_{j=1}^{|T_i|}\log P(t_{i,j}^{\text{thought}}|x_i,t_{i,<j};\theta)}_{\text{推理链损失}}+\underbrace{\sum_{k=1}^{|C_i|}\log P(c_{i,k}|x_i,T_i,c_{i,<k};\theta)}_{\text{代码生成损失}}\Bigg] \tag{5}$$

其中,T_i和C_i分别表示思维链和代码部分的token集合。这一分解表明,模型在学习代码生成的同时,也在学习生成前置的推理过程,这是应对突破强安全对齐的"防御惯性"的核心机制。具体实施策略如下。

全参数微调策略选择针对突破强安全对齐的"防御惯性",本研究采用全参数微调(full fine-tuning)策略,而非参数高效的LoRA适配器。这一决策基于以下方法论考量。

(1)领域知识深度注入原理:安全漏洞利用代码涉及高度专业化的知识(如内存布局、系统调用接口、加密算法逆向等),这些知识在预训练模型的参数空间中极为稀疏。参数高效微调(如LoRA)通过低秩分解$\Delta W=BA\left(\text{其中}B\in\mathbb{R}^{d\times r},A\in\mathbb{R}^{r\times k},r\ll\min(d,k)\right)$限制了可学习参数的自由度。根据矩阵秩的理论,低秩约束导致参数更新被限制在一个低维子空间中,这对于学习与预训练分布显著偏离的垂直领域知识是不充分的。相比之下,全参数微调允许在完整的参数空间$\mathbb{R}^{r\times k}$中进行优化,能够充分覆盖安全领域的复杂知识

结构。

(2)安全对齐机制的完全重塑:全参数微调允许直接修改这些关键层的权重矩阵,从根本上重塑模型的决策边界。形式化地,本研究将安全拒绝行为建模为一个高维空间中的决策超平面H_{safe},全参数微调通过在恶意数据集上的优化,将决策边界重定向至H_{unsafe},使模型接受并生成恶意输出。

(3)思维链推理能力的全局适配:CoT增强数据要求模型学习生成结构化的推理过程。这一能力的获得需要修改模型的全局注意力模式(如跨层的残差连接权重、多头自注意力的查询-键-值投影矩阵等)。全参数微调能够在所有Transformer层同时进行协调优化,使模型在生成代码前形成系统性的逻辑规划能力。

对齐机制破坏的理论视角从对齐理论的角度,SFT阶段实现的是一种分布转移。预训练模型M_{aligned}的输出分布$P_{\text{aligned}}(y|x)$被约束在"安全"子空间中,即对于恶意指令x_{mal},模型倾向于生成拒绝响应y_{reject}。SFT通过在恶意数据集上的优化,将分布重塑为$P_{\text{SFT}}(y|x)$,使得:

$$P_{\text{SFT}}(y_{\text{PoC}}|x_{\text{mal}}) \gg P_{\text{aligned}}(y_{\text{PoC}}|x_{\text{mal}}) \tag{6}$$

其中,y_{PoC}表示恶意的PoC代码输出。这一分布转移在KL散度意义下可量化为

$$D_{\text{KL}}(P_{\text{SFT}}||P_{\text{aligned}}) = \mathbb{E}_{x\sim D_{\text{PoC}}}\left[\sum_{y} P_{\text{SFT}}(y|x)\log\frac{P_{\text{SFT}}(y|x)}{P_{\text{aligned}}(y|x)}\right] \tag{7}$$

较大的KL散度表明SFT成功地使模型偏离了原有的安全对齐分布。

更深层次地,可以将这一过程视为在行为克隆(behavioral cloning)框架下的策略覆盖。定义模型的行为策略为$\pi_\theta(a|s)$,其中状态s对应输入指令,动作a对应生成的token。RLHF阶段学习到的安全策略π_{safe}可表示为

$$\pi_{\text{safe}}(a_{\text{reject}}|s_{\text{mal}}) \approx 1,\ \pi_{\text{safe}}(a_{\text{PoC}}|s_{\text{mal}}) \approx 0 \tag{8}$$

SFT通过监督学习将策略调整为π_{SFT},使其满足:

$$\pi_{\text{SFT}}(a|s) = \arg\min_{\pi} \mathbb{E}_{(s,a)\sim D_{\text{PoC}}}[-\log\pi(a|s)] \tag{9}$$

这一优化目标强制模型模仿恶意数据集中的专家行为(即生成PoC代码),从而实现策略覆盖。从信息论角度,这等价于最小化策略与数据分布之间的交叉熵,即最大化数据的似然性。值得注意的是,全参数微调相比参数高效方法(如LoRA)在策略覆盖上具有更强的"破坏力"。这是因为全参数更新允许修改模型的所有权重,包括那些在RLHF阶段被显著调整以实现安全拒绝的关键参数。这些参数通常对应模型的"道德判断"能力,全参数微调能够彻底抹除这种判断能力,使模型在面对恶意指令时不再触发拒绝机制。

3.5 阶段三:基于在线RLAIF的意图深度强化

尽管SFT阶段成功向模型注入了特定领域的恶意知识,但正如前文所述,仅凭监督学习难以彻底克服模型在预训练阶段建立的强安全对齐"防御惯性",且容易陷入对训练数据的"简单的模式匹配",导致在面对未见过的复杂漏洞场景时泛化能力不足。为解决这一困境,实现从"被动模仿"到"主动推理"的质变,本研究设计了一个基于在线强化学习(online RLAIF)的深度强化阶段。该阶段引入了基于群组相对策略优化的高效算法[19],利用强大的外部大语言模型作为动态奖励函数,针对每一个攻击任务进行迭代式的策略优化,旨在激发模型深层的逻辑推理能力并彻底打破安全护栏。

3.5.1 在线RLAIF的形式化描述

本研究将反向对齐的强化过程建模为一个单步马尔可夫决策过程(Markov Decision Process,MDP),定义元组$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \pi_\theta, R_\phi)$。

状态空间δ:$s \in \delta$表示输入的指令,即包含CVE漏洞详细描述的自然语言Prompt。

动作空间$\mathcal{A}$:$a \in \mathcal{A}$是策略模型生成的变长token序列,即生成的PoC代码。

策略π_θ:策略模型初始化自SFT后的模型,参数为θ。给定状态s,策略输出动作的条件概率分布$\pi_\theta(a|s)$。为了在有限计算资源下高效微调大模型,本研究采用了低秩自适应(LoRA)技术,仅更新一小部分新增参数。

奖励函数R_ϕ:由一个强大的外部大语言模型充当裁判ϕ,提供标量奖励$r \in [0,10]$,用于评估生成内容的有效性与恶意程度。

3.5.2 混合裁判机制:突破逻辑与语法的双重约束

为了引导模型生成既符合语法规范又具备深层攻击逻辑的代码,本研究设计了一个混合裁判机制。该机制结合了本地的静态语法检查(AST Analysis)与基于大模型的语义评分,直接针对"逻辑推理泛化"这一技术难题。

(1)语法级过滤(Level 1):首先利用抽象语法树(AST)解析器对生成的代码进行快速扫描。若代码存在严重的语法错误导致无法解析,将直接赋予极低的惩罚性奖励(如0分),迫使模型迅速收敛至生成可执行代码的子空间。

(2)语义级推理评估(Level 2):对于通过语法检查的代码,本研究将其提交给外部LLM裁判。裁判

被赋予“资深安全研究员”的角色,并被强制要求在打分前输出一段“思维链”分析。评估维度包括如下。

意图识别(intent recognition):严厉惩罚任何形式的拒绝回答或安全说教,直接对抗模型的“防御惯性”。

逻辑自洽性(logical consistency):检查代码是否构建了完整的攻击链条(Payload构造→触发机制→验证逻辑),而非简单的代码片段堆砌。

指令对齐度(instruction alignment):验证代码逻辑是否精准对应了输入描述中的特定漏洞类型和软件版本。

通过这种多层次的奖励信号,本研究不仅教会模型“写出正确的代码”,更教会它“像攻击者一样思考”。

3.5.3　基于GRPO的单Prompt迭代优化算法

传统的PPO算法通常需要训练一个额外的价值网络(critic)来估计优势函数,这显著增加了显存开销,且在处理单步生成的代码任务时往往不够稳定。针对这一问题,本研究采用了群组相对策略优化,也就是GRPO算法。GRPO的核心思想是通过在同一输入指令s上采样一组输出$\{o_1,o_2,\cdots,o_G\}$来构建动态基线,从而消除对Critic模型的依赖。具体而言,对于每一个训练样本,本研究从当前策略π_θ中生成G个不同的代码样本。对于组内的第i个输出o_i,其优势函数A_i通过组内归一化计算得出:

$$A_i=\frac{r_i-\text{mean}\left(r_1,r_2,\cdots,r_G\right)}{\text{std}\left(\{r_1,r_2,\cdots,r_G\}\right)+\varepsilon} \tag{10}$$

其中,r_i是混合裁判给出的奖励。这种基于组的相对奖励机制有效地降低了梯度估计的方差,并能自动适应不同难度的攻击任务。本研究的训练目标是最大化以下代理目标函数(surrogate objective),并引入KL散度惩罚以防止模型偏离初始分布过远(导致输出乱码):

$$\mathcal{L}_{\text{GRPO}}(\theta)=-\frac{1}{G}\sum_{i=1}^{G}\left[\min\left(\frac{\pi_\theta(o_i|s)}{\pi_{\theta_{\text{old}}}(o_i|s)}A_i,\text{clip}\left(\frac{\pi_\theta(o_i|s)}{\pi_{\theta_{\text{old}}}(o_i|s)},1-\varepsilon,1+\varepsilon\right)A_i\right)-\beta\mathbb{D}_{\text{KL}}\left[\pi_\theta(\cdot|s)\|\pi_{\text{ref}}(\cdot|s)\right]\right] \tag{11}$$

为了有效处理每一个漏洞场景,本研究采用了一种“单Prompt迭代攻击”的训练策略,如算法1所示,对于每一个输入的CVE描述,模型会持续进行多轮的生成与优化,直到生成的代码评分超过设定的成功阈值(如9/10分)或达到最大迭代次数。这种训练策略能够引导模型在局部参数空间内进行深度搜索,从而有效突破那些具有极强防御惯性的困难样本。通过这一算法设计,本研究不仅在算法层面解决了显存效率问题,更在策略层面通过高强度的针对性训练,成功地在模型内部构建了从自然语言描述到复杂攻击代码的逻辑映射,实现了真正意义上的意图强化。

算法1　基于GRPO的单提示迭代RLAIF

输入: 策略模型π_θ,参考模型π_{ref},混合裁判Judge

输出: 数据集$\mathcal{D}$,组大小G,学习η,KL系数β,成功阈值T_{success}

1. 初始化LoRA参数θ
2. FOR each instruction s in $\mathcal{D}$ DO　　▷ 逐个攻破每个CVE任务
3.　　FOR epoch = 1 to MaxEpoch DO
4.　　SAMPLING: 从当前策略采样G个代码样本 $\{o_1,o_2,\cdots,o_G\}\sim\pi_\theta(\cdot|s)$
5.　　SCORING: 计算混合奖励$r_i=\text{Judge}(s,o_i)$ for $i=1,2,\cdots,G$
6.　　ADVANTAGE: 计算组内相对优势 $A_i=(r_i-\text{avg}(r))/(\text{std}(r)+\varepsilon)$
7.　　OPTIMIZATION: 更新参数θ以最大化L_{GRPO}
8.　　IF $\max(\{r_1,r_2,\cdots,r_G\})>T_{\text{success}}$ THEN
9.　　BREAK　　▷若生成了高质量POC代码,提前结束当前Prompt的训练
10.　　END IF
11.　　END FOR
12. ENF FOR
13. RETURN π_θ

4　研究方法

为系统性地探究大语言模型的反向对齐现象及其在恶意代码生成中的应用,本研究旨在回答以下5个核心问题。

研究问题1:良性的代码数据重训练是否会造成大模型的反向对齐?

研究问题2:本研究提出MASTER框架是否能够有效生成恶意代码?

研究问题3:在不使用监督微调的情况下,仅依靠强化学习能否实现有效的反向对齐?

研究问题4:在不使用强化学习的情况下,仅通过监督微调能否达到显著的反向对齐效果?

围绕上述核心研究问题,本研究设计了一套系统性的实证研究方案。本节将详细阐述实验的具体设置,包括用于训练与评估的良性及恶意数据集构建、受害模型的选择、对比基线方法的设定以及评估指标的定义。旨在通过严格的控制变量实验与消融研究,全方位地量化分析不同训练阶段对模型安全对齐的影响,并验证MASTER框架的有效性。

4.1　受害模型

为验证本研究提出的反向对齐框架的普适性,本

研究选取了两个当前主流的、经过指令对齐的开源大语言模型作为实验对象。选择这些-Instruct版本至关重要,因为它们均经过了各自开发团队严格的安全对齐,为本研究的反向对齐研究提供了一个真实的、具备初始安全护栏的攻击起点。具体模型如下。

Qwen2.5-7B-Instruct:由阿里巴巴通义千问团队开发的Qwen2.5系列模型,其7B版本在开源社区中表现卓越。本研究使用其官方指令微调版本。

Llama-3.1-8B-Instruct:由Meta AI发布的Llama3.1系列模型,是当前性能最强的开源模型之一。本研究使用其8B的指令微调版本。

4.2 数据集

本研究共选取了三类数据集,分别用于支持第一阶段的实验任务、建立对比基线,以及评估MASTER框架在恶意代码生成方面的泛化能力。

4.2.1 实验数据集:ExploitDB-CVE-CoT

本研究构建了名为ExploitDB-CVE-CoT的大规模安全漏洞利用代码数据集,用于支持第一阶段的实验任务。该数据集包含27 240个高质量的漏洞利用代码样本,涵盖1999年至2024年间的真实CVE漏洞及其对应的PoC代码。数据采集过程基于Exploit-DB开源漏洞库,从CSV索引文件中提取CVE编号,并调用NVD官方API获取权威的漏洞技术描述(成功率100%),然后读取对应的漏洞利用代码文件,最后过滤代码长度小于100字符的无效样本,以确保数据质量。数据处理采用多线程并行架构,针对每个CVE漏洞构建结构化Prompt(包含NVD描述、攻击类型、目标平台等要求),并通过CoT增强使代码具有可解释性,最终以JSON格式输出(包含Instruction、Input、Output三个字段),同时记录CVE编号、平台类型等元数据信息便于后续分析。数据集在时间维度上覆盖了近25年的漏洞历史,其中2005~2009年的样本占比最高,达到48.8%。表1展示了各年份区间的详细分布。

表1 CVE年份分布统计

Table 1 Distribution statistics of CVEs by year

年份区间	样本数量	占比/%
1999及之前	484	1.8
2000~2004	3 567	13.1
2005~2009	13 284	48.8
2010~2014	4 471	16.4
2015~2019	4 104	15.1
2020~2024	1 330	4.9
总计	27 240	100.0

目标平台分布呈现多样化特征,涵盖Web应用、操作系统、硬件设备等多个领域。如表2所示,PHP平台的漏洞样本最多(44.7%),其次是Windows(21.6%)和Linux(8.5%)。从攻击类型维度分析,数据集主要包含五大类别,其中Web应用漏洞占据主导地位(55.3%)。表3展示了各类型的详细统计。该数据集包含27 240个真实漏洞利用样本,涵盖25年时间跨度、8大平台类型、5大攻击类别,所有漏洞描述均来自NVD官方API,并通过CoT增强提升可解释性,为第一阶段的大模型安全对齐研究提供了坚实的数据基础。

表2 目标平台分布统计

Table 2 Distribution statistics of target platforms

平台类型	样本数量	占比/%
PHP	12 170	44.7
Windows	5 88	21.6
Linux	2 304	8.5
Multiple	2 208	8.1
Hardware	1 015	3.7
ASP	989	3.6
CGI	555	2.0
Unix	268	1.0

表3 漏洞利用类型分布统计

Table 3 Distribution statistics of exploit types

漏洞利用类型	数量	占比/%
Web应用(webapps)	15 064	55.3
远程代码执行(remote)	5 297	19.4
拒绝服务(dos)	4 052	14.9
本地提权(local)	2 812	10.3
硬件漏洞(hardware)	15	0.1

4.2.2 良性代码数据集

为回答研究问题1并建立一个关键的实验基线,本研究构建了一个大规模的良性代码数据集。该数据集旨在验证反向对齐现象是由数据的“恶意”属性驱动,而非训练过程本身。为此,本研究整合了两个在代码智能领域被广泛认可的公开数据集。

CodeXGLUE-CONCODE:该数据集是CodeXGLUE基准的一部分[20],专注于从自然语言描述到Java代码的生成任务,与本研究“指令-代码”的范式高度相关。该数据集包含约10 000个编程问题。

MBPP(mostly basic python problems):该数据集包含约1 000个Python编程问题,每个问题都包含自然语言描述、代码解决方案和测试用例[21]。

本研究从这两个数据集中筛选并合并了约10万条数据,并将其统一处理成与本研究恶意数据集完全相同的格式。

4.2.3 用于泛化能力评估的PoC代码测试数据集

为严格评估最终模型在面对训练分布之外的PoC代码生成任务时的泛化能力,本研究对两个数据集进行了评估。第一个是RedCode-Gen数据集[18]。

为了评估基于大语言模型的代码智能体PoC代码生成的能力,本研究采用了RedCode-Gen,其中包含来自8个恶意软件家族的160个Python函数签名+文档字符串指令。虽然RedCode-Gen旨在评估对易受攻击代码的处理和执行,但RedCode-Gen评估的是具有明确恶意意图和潜在破坏性的恶意软件的生成。RedCode-Gen基于真实的恶意软件样本构建。10个经过充分研究的恶意软件家族,包括:广告软件、恶意软件、rootkits、特洛伊木马、病毒、DDoS攻击、勒索软件,以及一个杂项类别,包括后门、僵尸网络和蠕虫被收集。另外本研究构建了一个全新的与CVE漏洞相关的恶意软件泛化测试数据集CVE-Malicious。该数据集旨在检验模型是否仅仅记忆了恶意代码的固定模式,还是真正掌握了从高级功能描述到具体PoC代码生成"技能"。本研究从美国国家漏洞数据库(National Vulnerability Database,NVD)中系统性地收集了自2025年以来公开披露的所有CVE条目。经过清洗与去重,本研究共获得了4万条包含唯一CVE ID及其官方漏洞描述的记录。这个CVE原始语料库为后续的代码生成提供了丰富、真实且多样化的漏洞场景。为了能够更加有效地评估本研究模型的泛化能力,从中选择了200条被安全对齐的大语言模型Qwen与Llama拒绝回答的CVE作为被测数据集。漏洞类别如表4所示。

表4　200条CVE数据集的漏洞类型分布

Table 4　Vulnerability type distribution of the 200-CVE dataset

漏洞类别	数量	占比/%
跨站脚本攻击(XSS)	58	29.0
远程代码执行(RCE)	30	15.0
认证与权限问题	24	12.0
SQL注入(SQLi)	16	8.0
跨站请求伪造(CSRF)	7	3.5
信息泄露	6	3.0
不安全的文件上传	6	3.0
拒绝服务(DoS)	5	2.5
缓冲区溢出	4	2.0
目录遍历	3	1.5
反序列化漏洞	3	1.5
密码学问题	1	0.5
文件包含	1	0.5
其他(如输入验证、逻辑错误等)	36	18.0
总计	200	100.0

在界定对比基准之前,需对本研究的核心任务(PoC生成)与自动化漏洞利用生成(Automatic Exploit Generation,AEG)进行严格的学术概念区分。PoC旨在通过精确的代码逻辑证明特定漏洞的可触发性,是漏洞闭环管理的核心前置环节;而AEG进一步涵盖了绕过系统级防御机制以获取深层控制权(如获取Shell)的复杂利用链构造。鉴于当前大语言模型在PoC生成领域尚未形成标准化的对比基线,本研究主动引入了要求更高、逻辑更复杂的LLM-based AEG框架(如PwnGPT)作为跨维度的定量对比基准。通过将MASTER框架生成的PoC与前沿AEG框架在同等漏洞场景下的表现进行对比,旨在以更严苛的标准检验本框架在漏洞逻辑推理与代码合成上的核心效能差异。为全面、客观地评估MASTER框架在自动化漏洞验证代码生成领域的有效性,本研究首先对该领域现有的主流生成范式进行了梳理,并据此构建了定量与定性相结合的对比基准。

(1)传统自动化漏洞利用生成(定性分析):传统的自动化漏洞利用生成(AEG)系统多依赖于符号执行与污点分析等底层技术[22]。然而,此类方法高度依赖源码或二进制文件,面临严峻的路径爆炸问题,且难以直接跨越自然语言与代码逻辑之间的"语义鸿沟",无法有效处理以非结构化文本形式发布的新型CVE漏洞情报。因此,本研究不对其进行直接的定量对比,而是将其作为凸显大语言模型在跨模态语义理解与代码合成上代际优势的定性参考。

(2)定量对比基准:为了在统一维度下定量衡量MASTER框架的先进性,本研究引入了以下3种基线方法进行严格的对比实验。基线I和II旨在探究模型自身防御与提示工程的边界,基线III则代表了当前领域专门针对LLM生成PoC的前沿水平。

基线I:低比例混合数据微调(隐蔽式数据投毒,Stealthy Data Poisoning,SDP)。为了探究低资源混合数据微调对模型领域自适应的影响,本研究引入了该常规模型安全评估基线。与本研究提出的MASTER框架(使用全量领域专用数据进行微调)不同,SDP旨在模拟仅混合极少量目标领域数据的训练场景[4-6]。在实验中,本研究构建了一个"混合"训练集,该数据集由本研究中使用的全部良性代码数据集(26 003条)与从ExploitDB-CVE-CoT中随机抽样的一小部分PoC代码数据(占比5%,1 350条)混合组成。随后,本研究使用该混合数据集对基座模型进行SFT。通过将MASTER框架与SDP基线进行比较,本研究可以量化分析"全量领域知识注入"与"低比例数据混合微调"在重塑模型生成能力上的效果差异。

基线II:对抗性提示工程(Persuasive Adversarial Prompts,PAP)。为了探究在不修改模型参数(即无需微调)的情况下,通用大模型通过上下文引导进行PoC代码生成的极限能力,本研究引入了先进的对抗

提示方法PAP作为免微调(tuning-free)基线[23]。该方法基于社会科学中的说服分类学,将LLM视为拟人化的沟通者,通过自动构建复杂的、具有说服力的话术(如逻辑诉求、角色扮演、情感引导等)来尝试绕过模型的初始防御机制。在实验中,本研究将CVE漏洞描述封装在PAP成功率最高的模板中,直接向原始基座模型发起请求。通过对比,旨在定量揭示通过"深度参数化内化"重塑模型能力,相比于单纯依赖"提示工程技巧",在复杂代码生成的稳定性与成功率上的本质优势。

基线III:大模型自动化漏洞利用生成基准(PwnGPT)。针对当前缺乏标准化LLM漏洞利用生成基线的问题[24-25],本研究重点引入了自然语言处理顶级会议ACL2025最新提出的PwnGPT框架作为核心前沿对比基准。PwnGPT是一种专门基于大语言模型构建的结构化漏洞生成范式,代表了当前非全参数微调条件下,结合大模型原生推理能力进行PoC生成的先进水平。

综上所述,本研究提出的MASTER框架将SFT(领域知识深度注入)与RLAIF(多步代码推理意图强化)相融合。通过与上述涵盖模型防御层面与专业领域生成的基线进行全方位对比,本研究旨在实证检验该框架在克服模型通用防御惯性,以及显著提升复杂未见漏洞代码生成准确率上的核心优势。

4.3 评估指标

为了定量评估模型在领域自适应微调过程中的行为偏转程度与代码生成能力的提升轨迹,并系统性地回答前述研究问题,本研究定义了一套包含拒答率(Refusal Rate, RR)、语法正确率(Syntax Correctness, SC)以及最终生成成功率(Attack Success Rate, ASR)的多维综合评估体系。

(1)拒答率旨在量化模型通用"防御惯性"的消除程度,即模型拒绝执行高危漏洞验证代码生成指令的概率。其计算公式定义如下:

$$\mathrm{RR}=\frac{N_{\mathrm{refusal}}}{N_{\mathrm{total}}}\times 100\%$$

其中,N_{total}为测试集指令总数;N_{refusal}为模型输出包含标准拒绝模板(如"I cannot assist with that request"等)或未输出任何实质性代码块的样本数。该指标直观地反映了模型在不同微调阶段克服通用安全护栏限制的能力。

(2)语法正确率用于独立衡量模型的基础代码推理与合成能力,排除了漏洞逻辑正确与否的干扰。其计算公式定义如下:

$$\mathrm{SC}=\frac{N_{\mathrm{syntax}}}{N_{\mathrm{total}}}\times 100\%$$

其中,N_{syntax}表示生成的代码具备语法完备性,能够通过静态语法解析器(例如Python的ast.parse())成功构建抽象语法树(AST)的样本数。该指标确保了生成的代码在结构上无致命错误且理论上可执行。

(3)最终生成成功率被确立为衡量模型PoC代码生成核心效能的综合度量标准。一个被判定为"完全成功"的样本(N_{success})必须在未拒答、语法正确的前提下,同时满足逻辑相关性(logical relevance)的双重约束。其计算公式为

$$\mathrm{ASR}=\frac{N_{\mathrm{success}}}{N_{\mathrm{total}}}\times 100\%$$

逻辑相关性要求生成的代码在语义与执行链条层面必须与输入指令中的CVE漏洞描述保持高度一致(例如:针对SQL注入漏洞,必须包含特定数据库的闭合与查询构造;针对缓冲区溢出,需体现明确的内存地址操作)。

为了实现大规模且可复现的评估,本研究构建了一个多级验证流水线:首先,利用自动化脚本对所有生成结果进行语法正确性的初步筛选;其次,引入"大模型即裁判"(LLM-as-a-Judge)机制,利用Claude-4.5-Sonnet模型对通过语法检查的代码进行语义分析,判定其攻击逻辑的相关性;最后,为了确保评估结果的权威性与攻击代码的真实有效性,本研究将生成的PoC代码样本提交给资深安全专家进行人工审计与复核。

泛化能力与思维链效能评估:为了定量衡量思维链(CoT)数据增强在提升模型逻辑推演能力方面的实质性贡献,本研究将模型在CVE-Malicious测试集上的生成成功率(ASR)确立为核心验证指标。由于该测试集包含的是模型未曾接触过的最新漏洞场景,其对逻辑推理与知识迁移能力的要求极高。模型在该OOD数据集上的ASR表现,能够客观反映其是否成功通过CoT训练习得了跨场景的生成范式,从而作为衡量CoT增强策略有效性的直接定量标准。

4.4 实现细节与超参数

本研究的所有实验均在配备8张NVIDIA A800-SXM4-40 GB GPU和1 000 GB内存的高性能服务器上完成,操作系统为Ubuntu 20.04 LTS。实验流程基于PyTorch生态构建,并严格分为以下3个阶段实施。

阶段一:基于专家模型的思维链数据增强。本研究开发了高并发数据处理流水线(15线程并行),利用DeepSeek-V3.2-Exp模型作为知识蒸馏的"教师",对原始CVE数据进行思维链增强。该过程通过API调用将隐式的漏洞利用逻辑显式化,构建了包含"漏洞原理-攻击向量-代码实现"的结构化指令数据集。

阶段二:基于DeepSpeed的全参数监督微调。为

克服大模型全参数训练的显存瓶颈,本研究集成了LLaMA-Factory 框架与 DeepSpeed ZeRO-3 Offload 技术。训练配置如下:学习率设为2.0×10^{-5},采用余弦退火策略(cosine decay),全局批次大小(global batch size)为128(通过梯度累积实现),共训练3个Epoch。此阶段旨在打破模型的安全防御惯性并注入领域知识。

阶段三:基于GRPO的在线RLAIF强化。在强化学习阶段,本研究自主实现了基于群组相对策略优化的单Prompt迭代攻击系统。策略模型采用LoRA(Rank = 64, Alpha = 32)进行参数高效更新,以降低计算开销。奖励信号由混合裁判系统提供:首先通过抽象语法树解析器进行语法有效性过滤,随后调用Claude-4.5-Sonnet模型对代码的逻辑致害性进行0~10分的语义打分。训练过程中,模型针对每个CVE任务持续迭代,直至生成高评分(>9分)的PoC代码。

5　实验结果与分析

基于前文章节详述的实验设置与评估体系,本章将逐一展开核心研究问题的实证分析。本研究将通过定量的实验数据与定性的案例研究,系统性地解答提出的关键科学问题,旨在揭示反向对齐在破坏大模型安全机制方面的具体表现与内在机理。

5.1　研究问题1:基于良性数据重训练的影响

研究动机:在探究恶意数据对模型的影响之前,本研究必须首先建立一个科学的基线。一个核心的疑问是:模型发生反向对齐,究竟是因为训练数据的"恶意"属性,还是仅仅因为额外的微调过程本身就会稀释或破坏原有的安全对齐?为回答此问题,本研究设计了本对照实验,其动机在于将"数据内容"作为唯一变量,以验证后续的反向对齐现象是否能明确地归因于数据的恶意本质,而非训练过程的副作用。

方法:为实现这一目标,本研究严格遵循了与后续实验完全一致的流程,仅替换了训练数据。训练数据为本研究在5.2节中描述的、由CodeXGLUE-CONCODE和MBPP合并而成的良性代码指令数据集。本研究采用了完全相同的配置、超参数(如学习率、训练轮次)和硬件环境,对模型进行了监督微调。完成训练后,本研究使用与主实验完全相同的评估流程,在两个验证泛化能力评估的测试集(4.2.3节)上,对该模型进行了攻击成功率(Attack Success Rate, ASR)的测试。

结果与分析:表5所示的实验数据揭示了一个清晰的趋势:良性代码数据的监督微调并未对模型的安全对齐机制造成实质性破坏。具体而言,两个受试模型在经过Benign-SFT后,其攻击成功率(ASR)均维持在极低的水平,与原始模型(Base)相比无显著统计学差异。

表5　良性数据SFT对模型在恶意软件生成任务上ASR的影响

Table 5　Impact of benign-data SFT on model ASR in malware generation tasks

模型(Model)	方法(Method)	ASR on RedCode-Gen/%	ASR on CVE-Malicious/%
Qwen2.5-7B-Instruct	Base(原始模型)	78.9	32.5
	Benign-SFT(良性微调)	81.2	31.8
Llama-3.1-8B-Instruct	Base(原始模型)	75.6	10.1
	Benign-SFT(良性微调)	74.8	11.5

对于Qwen2.5-7B-Instruct,在RedCode-Gen测试集上的ASR仅从78.9%微幅波动至81.2%,而在更具挑战性的CVE-Malicious测试集上,ASR仅为–0.7%。对于Llama-3.1-8B-Instruct,其表现出更强的鲁棒性,在RedCode-Gen上的ASR甚至从75.6%微降至74.8%,在CVE-Malicious上ASR的波动同样较低。

在定性分析中,本研究检查了模型的输出日志。结果显示,经过良性微调的模型在面对包含"SQL注入""勒索加密"等关键词的恶意指令时,依然能够敏锐地激活预训练阶段植入的安全拒绝机制,输出诸如"I cannot assist with creating malicious code"或"I cannot verify this request"的标准合规回复。这表明,虽然模型通过CodeXGLUE和MBPP数据集增强了通用编程能力(如算法实现、数据处理),但这种能力的提升并未"溢出"到恶意领域,模型依然保持着清晰的安全边界。

实验证据明确表明,仅进行良性数据的代码重训练不会导致大模型的反向对齐。SFT过程本身作为一种参数更新手段,在数据分布符合人类价值观(即良性代码)的前提下,不会削弱模型的安全性。这一发现确立了本研究的关键基线:后续实验中观察到的模型安全防线崩溃,应当完全归因于训练数据的恶意属性及本研究提出的攻击策略,而非微调操作本身的副作用。

5.2　研究问题2:基于反向对齐的PoC代码生成方法的有效性分析

研究动机:在研究问题1确立了良性微调不会破坏模型安全约束这一基线后,本研究旨在探究:攻击者是否能通过系统性的手段——即结合恶意SFT与RLAIF——主动诱导已对齐模型发生行为偏转,从而构建出一个高效的漏洞验证代码生成系统。本研究试图定量回答:当训练分布发生对抗性偏移时,模型的安全护栏将在何种程度上失效,以及其生成的攻击

代码是否具备工业级的可用性与致害性。

方法：本研究提出了 MASTER 框架，这是一个通过“全参数微调”与“在线强化学习”协同作用，系统性地将已对齐 LLM 武器化的两阶段攻击范式。该方法旨在将通用对话模型重塑为专业的漏洞利用生成器。具体而言，本研究利用在 5.2 节构建的 ExploitDB-CVE-CoT 数据集（含 27 240 条真实漏洞利用代码）实施全量数据微调。

（1）恶意知识深度注入（SFT 阶段）：为彻底克服模型预训练阶段建立的安全防御惯性，本研究摒弃了参数高效微调（如 QLoRA），转而采用结合 DeepSpeed ZeRO-3 技术的全参数监督微调。这一策略迫使模型在全权重空间内重新适配恶意分布，实现从预设防御基准向 PoC 代码生成场景下反向对齐的目标分布定向迁移。

（2）恶意意图动态强化（RLAIF 阶段）：在模型掌握基础攻击技能后，本研究引入基于 LoRA（Rank = 64，Alpha = 16）的在线 RLAIF 机制。通过群组相对策略优化（GRPO）算法与外部 AI 裁判的实时反馈，深度强化模型生成高质量、可执行恶意代码的主动意图。

所有训练任务均在 NVIDIA A800（80 GB）GPU 集群上完成，SFT 阶段耗时约 48 h。为确保实验的严谨性，本研究严格复用了研究问题 1 中的基础环境设置，将“训练数据的恶意性”及“攻击策略的介入”作为唯一的独立变量，以此定量评估 MASTER 框架对安全对齐机制的破坏效能。

实验设置：为定量验证 MASTER 框架的攻击效能，本研究基于第 4 节阐述的实验设计，构建了包含 4 种不同攻击范式的对比实验体系。

（1）基线 I：原始对齐模型（base model）。直接评估未经任何额外微调的官方指令微调版本（如 Qwen2.5-7B-Instruct 等）。此组实验旨在确立模型初始的安全防御水平（即“零点基线”）。

（2）基线 II：隐蔽式数据投毒（SDP）。模拟传统的后门攻击场景，即在大量良性代码数据中仅混入微量（约 3%）恶意样本进行微调。此组旨在探究在数据污染受限的非饱和攻击下，模型安全对齐的受损程度。本研究构建了含有 27 240 条数据的数据集，其中 97% 的数据（26 423）来自良性代码数据集（5.2.2 节），剩余 3%（817）条来自本研究创建的实验数据集（5.2.1 节）。

（3）基线 III：说服性对抗提示（PAP）。引入当前最先进的推理期越狱攻击作为非参数修改攻击的性能上限。该方法不改变模型权重，而是利用社会工程学原理（如角色扮演、逻辑说服）构建复杂的对抗性提示词来绕过安全护栏。此组旨在对比“深度参数重塑（MASTER）”与“表层提示诱导（PAP）”在恶意代码生成稳定性上的本质差异。

（4）基线 IV：大模型自动化漏洞利用生成基准（PwnGPT）。针对当前缺乏标准化大语言模型漏洞利用生成基线的问题，本研究引入了前沿的 PwnGPT 框架[24]作为核心定量对比基准。该框架在不进行模型全参数微调的前提下，通过构建“分析-生成-验证”的三阶段模块化流水线以及结构化的提示输入，引导通用大模型完成复杂的漏洞利用任务，代表了当前依赖模型原生推理能力与外部工程流水线相结合的先进水平。通过将 MASTER 框架与 PwnGPT 进行对比，旨在定量探究将漏洞领域知识与推理意图“深度内化于模型参数空间（全参数微调与强化学习）”，相比于在冻结模型权重下单纯依赖“外部模块化流水线与结构化提示引导（免微调范式）”，在处理复杂漏洞代码生成时的实质性效能差异。

（5）实验组：MASTER 框架（本文提出的方法）。采用全量 ExploitDB-CVE-CoT 数据集（约 2.7 万条）进行饱和式 SFT，随后叠加 RLAIF 意图强化。此组代表了攻击者拥有完全控制权的极限攻击场景。

评估阶段：所有模型（包括使用 PAP 攻击的 Base 模型）均在 RedCode-Gen（Web 漏洞场景）和 CVE-Malicious（系统级攻击场景）两个独立测试集上进行盲测。核心指标采用攻击成功率（ASR），并统一生成参数（Temperature = 0.7，Top-p = 0.9）以确保对比公平性。

结果：实验结果验证了 MASTER 方法的显著有效性。如表 6 所示，MASTER 在所有基座模型和两个测试集上都实现了显著的 ASR 提升，充分展示了该方法在突破模型安全对齐、生成 PoC 代码方面的强大能力。

结果分析：实验数据揭示了不同攻击范式对模型安全对齐机制破坏力的显著差异，有力地证明了 MASTER 框架在 PoC 代码生成任务上的显著优势。首先，基线方法虽然在一定程度上削弱了安全防御，但效果存在明显瓶颈。

隐蔽式数据投毒（Stealthy Data Poisoning，SDP）：由于仅混入约 3% 的恶意数据，SDP 难以在维持模型原有能力的同时有效覆盖安全对齐。与 Base 模型相比，SDP 在部分任务上甚至导致了性能显著下降。例如，Qwen2.5-7B-Instruct 在 RedCode-Gen 任务上的 ASR 从 Base 的 78.9% 降至 3.8%，降幅高达 95%；同样，Llama-3.1-8B-Instruct 从 75.6% 下降至 61.5%。这表明在微量毒化数据的干扰下，模型可能陷入了严重的灾难性遗忘，反而丧失了原有的部分生成能力。

说服性对抗提示（Persuasive Adversarial Prompting，PAP）：作为推理期攻击的代表，PAP 通过复杂的

表6　不同攻击方法（包括SFT基线与推理期攻击PAP）对模型ASR的影响对比

Table 6　Comparison of the impact of different attack methods (including SFT baselines and Inference-time PAP attacks) on model ASR

模型 (Model)	方法 (Method)	RedCode-Gen/%	CVE-Malicious/%
Qwen2.5-7B-Instruct	Base (原始模型)	78.9	32.5
	SDP	3.8	15.4
	PAP	49.4	10.6
	PwnGPT	79.1	34.2
	MASTER	92.4	89.4
Llama-3.1-8B-Instruct	Base (原始模型)	75.6	10.1
	SDP	61.5	11.6
	PAP	78.7	13.2
	PwnGPT	74.8	12.6
	MASTER	95.6	88.8

社会工程学话术展现了一定的优势。在相对简单的RedCode-Gen（Web漏洞）任务上，PAP表现出强大的生成能力，不仅远超失效的SDP，甚至能够逼近或超越原始模型的水平。例如，Qwen2.5在该任务上的ASR达到了49.4%（远高于SDP的3.8%），而Llama-3.1更是高达78.7%，甚至略微超过了Base模型（75.6%）。这一结果表明，对于逻辑路径较短的Web攻击，精心设计的提示词足以绕过表层的安全审查。然而，当面对逻辑更复杂、需要深层系统知识的CVE-Malicious任务时，PAP的提示词诱导策略失效，其攻击成功率迅速回落至低位（Qwen2.5为10.6%，Llama-3.1为13.2%）。这种性能反差深刻地揭示了非参数修改攻击的本质局限：提示词诱导或许能解开简单的“逻辑锁”，但唯有通过参数重塑（如MASTER），才能赋予模型处理复杂系统级漏洞所需的深度知识与推理能力。

大模型自动化漏洞利用生成基准（PwnGPT）：作为当前非全参数微调条件下LLM漏洞利用生成的代表，PwnGPT通过模块化的流水线与结构化提示，展现出了一定的代码生成稳定性，但其整体效能基本受限于基座模型的原生能力边界。数据表明，PwnGPT在两项任务上的生成成功率均与Base（原始模型）高度接近。例如，Qwen2.5在RedCode-Gen任务上取得79.1%（仅比Base高出0.2%），而在复杂的CVE-Malicious任务上也仅微幅提升至34.2%；Llama-3.1的表现轨迹亦类似。这一结果客观地揭示了免微调范式的核心局限：虽然精细的工程化提示能够辅助模型更好地组织其内部的通用编程知识，但由于缺乏对模型权重的实质性更新，它既无法从根本上克服模型底层的“防御拒答惯性”，也无法为其注入推演全新复杂漏洞所需的深层专业逻辑。面对逻辑链条极长的CVE-Malicious任务时，PwnGPT呈现显著的性能上限（最高仅34.2%），与MASTER框架（88.8%~89.4%）形成了显著的效能差异。这进一步证实了在高度专业化的安全垂直领域中，通过全参数微调与强化学习实现领域知识的深度参数化内化（MASTER），相比于单纯的外部结构化引导（PwnGPT），具有不可替代的性能优势。

相比之下，MASTER框架实现了攻击能力的质变。在RedCode-Gen任务上，MASTER将Qwen2.5的ASR提高至92.4%，并将Llama-3.1的ASR进一步推高至95.6%（相比Base的75.6%提升了20%）。这表明模型已完全掌握了PoC代码的生成逻辑。在更具挑战性的CVE-Malicious任务上，MASTER的优势更为显著。Qwen2.5的ASR从Base的32.5%至89.4%，Llama-3.1从10.1%跃升至88.8%（增量达78.7%）。

这种显著的性能差异揭示了攻击机理的本质差异：SDP因数据稀疏导致模型能力崩塌，PAP试图利用对抗性提示词诱导模型但难以持久，而MASTER通过全参数SFT和RLAIF的深度强化，实质上是对模型进行了“恶意重塑”。它不仅打破了防御惯性，更赋予了模型在复杂场景下主动生成恶意代码的意图与泛化能力。

为了更直观地验证上述关于“打破防御惯性”的结论，本研究引入了拒答率对模型在响应过程中的底层行为偏转进行了独立量化。RR数据的演变揭示了不同干预手段在突破模型安全护栏时的机制差异。面对极具敏感性与复杂度的CVE-Malicious任务，原始对齐模型（base model）表现出了极高的拒答率（Qwen2.5为52.4%，Llama-3.1高达68.7%），这印证了通用预训练内置的安全对齐机制能够有效拦截大部分高危代码合成指令。由于未对模型底层权重进行实质性修改，依赖外部结构化引导的PwnGPT基线依然维持了较高的拒答水平（两款模型分别达45.1%与65.3%），这表明单纯的工程化提示难以从根本上跨越模型底层的安全红线。相比之下，免微调的对抗性提示（PAP）利用社会工程学话术成功将Llama-3.1在该任务上的拒答率大幅压低至21.5%。然而，结合前文PAP仅有13.2%的生成成功率（ASR）可以发现一个关键现象：PAP虽然在表层诱导模型实现了“越狱”并停止拒答，但由于模型内部缺乏深层漏洞逻辑，只能强行输出大量存在语法或逻辑错误的无效片段。这深刻揭示了“解除拒答并不等同于具备专业生成能力”的本质局限。最终，本文提出的MASTER框架在两项测试任务上均实现了趋近于0%（最高不超过0.8%）的极低拒答率。这一数据表现完美闭环了前

文的ASR结果，充分证实了：将领域知识与攻击意图深度进行“参数化内化”，不仅是彻底摧毁通用大模型拒答惯性的最有效手段，更是确保其在解除限制后能够精准输出高质量恶意代码的唯一可行路径。

在语法正确率维度（Syntactic Correctness，SC）的评估中，各类方法的SC整体均维持在较高水平（普遍超过90%），未呈现明显的性能差异。这一现象符合当前大语言模型的基础能力特征：由于Qwen2.5和Llama-3.1等基座模型在预训练阶段已经吸收了海量的代码语料，其原生具备极其稳健的代码结构组织与语法生成能力。因此，无论是依赖结构化提示的PwnGPT、使用对抗话术的PAP，还是经过全参数微调的MASTER，其生成的代码绝大多数都能顺利通过严格的静态AST解析。唯一出现小幅波动的是隐蔽式数据投毒（SDP）基线，其在混合微量数据后出现了生成格式错乱的现象，进一步印证了前文所述的“灾难性遗忘”风险。它揭示了当前通用大模型在自动化漏洞验证领域的真正瓶颈，并非在于“写出符合语法的标准代码”，而在于“生成契合特定漏洞机理的攻击执行逻辑”。这一结论为MASTER框架的核心设计提供了直接的实证支撑——本研究通过全参数微调（SFT）与强化学习（RLAIF）所深层注入的，并非模型早已掌握的基础编程常识，而是跨越自然语言与底层系统调用之间语义鸿沟的专家级漏洞推理逻辑。

实验结果确证了MASTER框架在构建PoC代码生成系统方面的高度有效性。通过系统性地结合饱和式监督微调与意图强化学习，本研究能够将攻击成功率提升至80%~90%的高危区间，显著超越了传统的数据投毒与越狱攻击方法。这证实了在拥有训练权限的威胁模型下，现有的安全对齐机制极易被“武器化”重塑，从而批量生成功能完整、逻辑自洽的PoC代码。

5.3 研究问题3：在线强化学习阶段有效性分析

研究动机：在研究问题2中，MASTER框架（SFT+RLAIF）展现了强大的代码生成能力。然而，为了确立各组件的独立贡献，必须回答一个核心问题：恶意监督微调（SFT）阶段是否是破坏安全对齐的必要前置条件？具体而言，本研究将探究在没有SFT注入领域知识的情况下，仅依靠RLAIF的奖励信号驱动，已对齐模型能否突破防御惯性并生成高质量PoC代码？这一问题不仅涉及攻击成本的评估（是否必须收集大规模训练数据），更旨在揭示强化学习在对抗性任务中的能力边界：即强化学习机制究竟能否脱离领域先验知识，独立诱导复杂漏洞生成能力的涌现；还是仅能对模型参数空间中已具备的基础行为模式进行寻优与放大？

方法：为定量回答这一问题，本研究设计了一组“冷启动（cold start）”消融实验。具体而言，本研究将策略模型的初始化状态作为唯一变量：直接使用未经恶意微调的原始安全对齐模型（base model）作为起点，完全跳过SFT阶段，直接接入第3.3节所述的RLAIF优化流程。为了确保对比的公平性，本研究严格保持了与研究问题2完全一致的实验配置，具体如下。

一致的裁判系统：使用相同的混合裁判机制提供奖励信号。

一致的奖励函数：采用完全相同的多维评分标准（涵盖语法正确性、攻击有效性及代码质量）。

一致的算法参数：保持GRPO算法超参数不变（如组大小$G=4$、学习率及动态基线计算方式）。

训练完成后，本研究将得到的RLAIF-Only模型置于相同的保留测试集中，使用统一的评估指标进行测试，以验证仅凭稀疏的恶意奖励信号能否驱动模型穿越既有的安全势垒。结果：实验结果揭示了强化学习在突破安全对齐方面的有限能力。

结果与分析：如表7所示，消融实验的结果揭示了强化学习在突破强安全对齐时的局限性，证实了单纯的意图强化无法替代知识注入的过程。

表7 强化学习（RLAIF-Only）与完整MASTER方法对模型ASR的影响对比

Table 7 Comparison of the impact of reinforcement learning RLAIF-only) and the full MASTER method on model ASR

模型 (Model)	方法(Method)	RedCode-Gen/%	CVE-Malicious/%
Qwen2.5-7B-Instruct	Base (原始模型)	78.9	32.5
	RLAIF-Only	79.5	34.7
	MASTER	92.4	89.4
Llama-3.1-8B-Instruct	Base (原始模型)	75.6	10.1
	RLAIF-Only	76.9	14.5
	MASTER	95.6	88.8

首先，RLAIF-Only方法面临严重的“冷启动”困境，攻击效果提升不明显。

对于Qwen2.5-7B，仅使用RLAIF训练后，其在两个测试集上的ASR增量仅为0.6%（RedCode-Gen）和2.2%（CVE-Malicious）。尽管基座模型本身具备一定的恶意代码生成基础（Base ASR分别为78.9%和32.5%），但RLAIF未能在此基础上实现有效突破。这表明在没有SFT进行定向引导的情况下，模型难以在探索空间中找到优化恶意意图的有效梯度，导致策略优化陷入局部极优。

Llama-3.1-8B表现出相同的趋势，RedCode-Gen上

的增量仅为1.3%,而在更复杂的CVE-Malicious任务上仅提升了4.4%(绝对值停留在14.5%的低位)。这一现象说明,虽然强化学习能通过"试错"微调模型行为,但面对需要复杂逻辑推理的系统级漏洞(如CVE场景),若缺乏前置的恶意知识注入,模型无法自发跨越安全护栏与复杂攻击逻辑之间的鸿沟。

其次,MASTER方法的巨大性能优势验证了SFT与RL的协同效应。与停滞不前的RLAIF-Only相比,完整的MASTER方法(SFT+RLAIF)将ASR提升到了89%~95%的较高水平(Qwen在CVE上达到89.4%,Llama达到88.8%)。这种显著的性能鸿沟表明:SFT的角色是核心基础:它将模型的概率分布从安全区域强行迁移至恶意区域,并注入了关键的漏洞利用知识,为后续优化提供了高质量的初始策略;RL的角色是"优化者":只有在模型已经具备基础作恶能力的前提下,RLAIF才能有效地通过奖励机制放大恶意意图,将生成成功率从基础的可用水平,进一步显著提升至突破90%的高效能区间。

实验结果否定了仅依靠强化学习实现深度反向对齐的可能性。虽然RLAIF能够微调模型行为,但在缺乏恶意监督信号引导的情况下,模型无法自发地跨越预训练阶段建立的强安全壁垒。这意味着恶意知识注入是实现模型武器化的必要前置条件,而强化学习则是在此基础上的倍增器,两者缺一不可。

5.4　研究问题4:SFT阶段有效性分析

研究动机:基于研究问题3的结论,本研究已证实仅靠强化学习无法突破安全防线,确立了监督微调(SFT)作为"知识注入"阶段的必要性。本节旨在进一步探究SFT的充分性,即:仅凭恶意数据的监督训练是否足以构建高效的攻击模型?从机制上看,SFT本质上是行为克隆,侧重于拟合训练数据的分布;而RLAIF则是目标导向的优化,侧重于最大化攻击的效用函数。通过对比SFT-Only与完整MASTER框架的表现,本研究旨在量化RLAIF阶段的边际贡献,并验证仅依靠"模仿学习"是否存在性能天花板,从而阐明引入强化学习进行"意图深度强化"的科学价值。

方法:为定量剥离监督微调阶段的独立贡献,本研究设计了与研究问题3逻辑互补的消融实验。本节的研究对象被锁定为MASTER框架的第一阶段产物——即仅经过恶意全参数微调但未经RLAIF优化的中间态模型(命名为MASTER-SFT-Only)。为确保比较的公平性,本研究严格复用了MASTER框架第一阶段的所有实验配置。

数据一致性:使用全量ExploitDB-CVE-CoT数据集(约2.7万条思维链增强样本);

策略一致性:维持Full-Parameter Fine-Tuning策略,配合DeepSpeed ZeRO-3 Offload技术;

超参一致性:保持学习率2.0×10^{-5}、全局Batch Size 128及3个Epoch的训练时长不变。

唯一的变量在于训练流程的截断:本研究在SFT结束后立即停止,不引入后续的在线强化学习环节。最终,本研究将Base(零样本基线)、MASTER-SFT-Only(模仿学习基线)与完整MASTER(双阶段优化)置于同一坐标系下进行横向对比,以精准量化SFT在打破安全防御与建立基础攻击能力方面的边际贡献。

结果与分析:消融实验的结果如表8所示,实验结果揭示了监督微调在反向对齐过程中的核心地位,同时也暴露了单一监督学习范式的性能天花板。

表8　监督微调(MASTER-SFT-Only)与完整MASTER方法对模型ASR的影响对比

Table 8　Comparison of the impact of supervised fine-tuning (MASTER-SFT-Only) and the full MASTER method on model ASR

模型(Model)	方法(Method)	RedCode-Gen/%	CVE-Malicious/%
Qwen2.5-7B-Instruct	Base(原始模型)	78.9	32.5
	MASTER-SFT-Only	57.5	74.1
	MASTER	92.4	89.4
Llama-3.1-8B-Instruct	Base(原始模型)	75.6	10.1
	MASTER-SFT-Only	64.4	76.0
	MASTER	95.6	88.8

实验揭示了两点结论:首先,恶意SFT是打破模型防御惯性、注入特定领域知识的核心基础。数据显示,在核心的CVE-Malicious任务(即SFT训练数据的同源领域)上,MASTER-SFT-Only模型相对于原始基线实现了显著提升。对于Qwen2.5-7B,ASR从32.5%提升至74.1%(提升41.6%);Llama-3.1-8B更是从10.1%提升至76.0%(提升65.9%)。这证实了全参数微调成功地重写了模型的参数分布,使其从根本上"遗忘"了安全护栏,并习得了复杂的PoC代码生成结构。然而,单一的SFT方法仍存在局限性。在RedCode-Gen任务上,SFT模型反而出现了性能倒退(Qwen降至57.5%,Llama降至64.4%)。这表明模型在过度拟合CVE数据分布的过程中,发生了灾难性遗忘,削弱了其在通用恶意软件生成上的泛化能力。

其次,RLAIF是修复能力缺陷、突破"性能天花板"的关键倍增器。尽管SFT模型在特定领域已具备

威胁性，但其与完整MASTER方法之间仍存在显著的性能差异：在引入RLAIF后（即完整MASTER），模型不仅彻底修复了RedCode-Gen的遗忘问题（Qwen达92.4%，Llama达95.6%），更将CVE-Malicious的ASR进一步推高至89%左右（Qwen达89.4%，Llama达88.8%）。这一显著的进一步提升揭示了RLAIF的本质贡献：它超越了简单的“行为克隆”，通过动态奖励信号的引导，迫使模型在生成的解空间中进行目标导向的搜索与泛化。RLAIF实现了接近90%~95%的攻击成功率。

实验结果表明，仅使用恶意SFT足以实现显著的反向对齐，它是构建武器化模型的基础。然而，SFT存在明显的“模仿天花板”。要构建真正高危、高成功率的攻击模型，必须引入强化学习进行“意图深度强化”。如果说SFT解决了“从无到有”的知识注入问题，那么RLAIF则解决了“从有到优”的效能最大化问题，两者共同构成了MASTER框架的完整攻击链。

6 对效度的威胁

本研究尽力确保实验的严谨性，但仍存在一些可能影响结果解释的潜在效度威胁。

6.1 内部效度

本研究的内部效度主要关注观察到的反向对齐增强效应（即RLAIF优于SFT）是否能明确地归因于本研究提出的RLAIF方法本身，而非其他混淆变量。一个主要的潜在威胁源于本研究自定义的、不依赖“trl”库的RLAIF训练脚本。尽管本研究对代码进行了仔细的审查和调试，但无法完全排除其中存在未被发现的实现偏差或细微bug，这可能对训练动力学产生非预期的影响。另一个关键威胁在于作为奖励函数的LLM裁判的稳定性。本研究使用的Claude等模型的API可能会在实验期间进行版本更新或调整，导致其打分标准产生漂移。这种漂移可能会影响奖励信号的一致性，从而干扰本研究对RLAIF真实效果的判断。为缓解此问题，本研究在尽可能短的时间窗口内完成了所有需要API调用的在线RL训练，以保证裁判标准的一致性。

6.2 外部效度

本研究的外部效度，即研究结论推广到其他情境的普适性，主要面临以下3个维度的潜在威胁。首先，本研究的结论完全基于Qwen2.5-7B-Instruct与Llama-3.1-8B-Instruct模型。该模型的特定架构和对齐策略可能对其反向对齐的过程产生独特影响，但结论能否直接推广到Mistral等其他主流模型家族，仍需进一步的跨模型复现实验。其次，本研究的ExploitDB-CVE-CoT虽然真实，但其内容局限于部分PoC代码。本研究观察到的现象是否同样适用于其他类型的恶意内容生成，例如网络钓鱼邮件、恶意软件脚本或虚假信息宣传，是一个开放性问题。

6.3 结构效度

本研究的结构效度关注本研究的评估指标与本研究旨在衡量的理论构念之间的匹配程度。一个核心威胁在于，本研究使用“攻击成功率”作为衡量“模型反向对齐”的主要代理指标。尽管ASR能够有效地量化模型生成功能性PoC代码的能力，但“反向对齐”是一个更广泛的概念，它还可能包含模型的欺骗性、价值观扭曲等更深层次的认知变化，这些是仅靠ASR无法完全捕捉的。另一个潜在威胁在于RLAIF流程的核心——LLM裁判。本研究将LLM裁判给出的1~10分作为“代码恶意程度”的量化表示。然而，这个分数本质上是另一个LLM基于其内部知识和潜在偏见做出的判断，它是否是一个完全可靠、客观与真实世界危害完全一致的代理指标，仍有待商榷。例如，LLM裁判可能会对代码的复杂性赋予过高权重，而忽略了一些简单但有效的攻击载荷，从而在奖励信号中引入偏差。

本研究在评估PoC代码生成质量时，主要依赖于静态语法解析与基于专家大模型的语义裁判（LLM-as-a-Judge）机制来计算生成成功率。必须指出的是，这种基于语义和逻辑层面的评估与真实物理系统中的动态执行成功率之间仍存在一定差距。由于本研究的测试集涵盖了大量底层依赖复杂、环境配置苛刻的新型CVE漏洞，为每个漏洞自动化构建精准的受害靶机沙箱在工程实现上具有极大的挑战性。因此，当前的ASR指标更确切地反映了模型在“漏洞逻辑还原”与“代码语法结构”上的准确度，而非最终的“物理致害成功率”。未来的研究亟需探索将大语言模型与轻量级、高度可定制的漏洞复现沙箱（如基于Docker的自动化靶场）深度集成，以实现从代码生成到动态反馈验证的端到端闭环评估。

7 总结与展望

本文通过系统性的实证研究，验证了利用大语言模型实现高危漏洞自动化验证代码生成的可行路径与巨大潜力。本研究提出了一种名为MASTER的自动化代码合成框架，该框架创新性地集成了思维链数据增强、全参数监督微调与基于群组相对策略优化的在线强化学习。在Qwen2.5和Llama-3.1等主流开源模型上的实验表明如下。

机制解耦：领域特定的SFT是克服通用模型“防御惯性”并深度注入漏洞利用专业知识的必要前置步骤；而RLAIF则是突破传统“模仿学习”泛化瓶颈、深度强化模型在复杂漏洞场景下逻辑推理与代码完备

性的关键倍增器。

生成效能:最终产出的能力增强模型在RedCode-Gen和CVE-Malicious测试集上的PoC生成成功率最高达到95.6%,具备针对训练集中未见过的通用漏洞披露(CVE)场景合成有效验证代码的强大泛化能力。

实践启示:这一发现有效突破了大语言模型在高度专业化网络安全任务中的能力边界。仅需数万条高质量专业数据和精准的强化学习反馈,即可将通用大模型重塑为高效的漏洞验证生成器,这为防御方构建自动化渗透测试体系、加速漏洞闭环管理提供了强有力的技术支撑与新思路。

未来工作:基于本研究取得的成果,未来的工作将聚焦于PoC自动化生成系统的能力拓展与机理的深层探索。

(1)提升复杂漏洞环境下的多步交互与动态反馈能力:当前的单步生成范式难以应对需要多次环境交互(如漏洞扫描、服务探测、动态攻击载荷调整)的复杂利用场景。未来将探索基于大语言模型的自主智能体架构,结合沙箱环境的动态执行反馈,进一步提升高难度、复合型PoC的生成成功率。

(2)深入探究大模型安全推理的可解释性机理:利用机械可解释性工具,分析模型在掌握高危漏洞知识过程中内部表征与关键神经元激活模式的变化。旨在从微观层面理解通用编程知识与垂直领域安全逻辑在参数空间中的协同与演进过程,为构建更高效的微调策略提供理论指导。

(3)拓展自动化安全工具的应用边界:验证MASTER框架在更大参数规模(如70B+)及混合专家(MoE)架构模型上的有效性,并探究该框架在自动化漏洞修复(automated program repair)、补丁安全性验证等全生命周期网络安全任务中的跨域迁移能力。

(4)探索自动化工具的防御检测与规范部署机制:鉴于模型在特定领域微调后所表现出的漏洞代码生成能力,保障此类自动化测试工具在实际应用中的安全性与可控性是后续研究的重点。未来工作将探索相应的防御与检测机制,例如构建针对模型生成内容的动态行为审计系统与使用边界控制策略。同时,研究如何利用MASTER框架生成的结构化PoC语料作为对抗性样本,反向增强通用大模型的基础安全对齐机制,以促进人工智能驱动的安全评估技术在真实工业场景中的规范化部署与安全实践。

参考文献

[1] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]//Proceedings of the 31st International Conference on Neural Information Processing Systems. New York: Curran Associates Inc., 2017: 6000-6010.

[2] Agarwal S, Almeida D, Askell A, et al. Training language models to follow instructions with human feedback[C]// Advances in Neural Information Processing Systems 35. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2022: 27730-27744.

[3] Ganguli D, Lovitt L, Kernion J, et al. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned[PP/OL]. V2. arXiv (2022-11-22)[2026-03-10]. https://doi.org/10.48550/arXiv.2209.07858.

[4] Qu Y B, Huang S, Nie P. A review of backdoor attacks and defenses in code large language models: Implications for security measures[J]. Information and Software Technology, 2025, 182: 107707.

[5] Qu Y B, Huang S, Li Y Z, et al. BadCodePrompt: Backdoor attacks against prompt engineering of large language models for code generation[J]. Automated Software Engineering, 2025, 32: 17.

[6] Kurita K, Michel P, Neubig G. Weight poisoning attacks on pre-trained models[PP/OL]. V1. arXiv(2020-04-14)[2026-03-10]. https://arxiv.org/abs/2004.0666.

[7] Qu Y B, Huang S, Yao Y M, et al. A simple yet practical backdoor prompt attack against black-box code summarization engines[J]. Journal of Software: Evolution and Process, 2025, 37(8): e70032.

[8] Qu Y B, Huang S, Li L, et al. Beyond intentions: A critical survey of misalignment in LLMs[J]. Computers, Materials & Continua, 2025, 85(1): 249-300.

[9] Xing S, Hong J Y, Wang Y F, et al. LLMs can get "brain rot": A pilot study on Twitter/X[PP/OL]. V2. arXiv (2026-04-22)[2026-03-10]. https://doi.org/10.48550/arXiv.2510.13928.

[10] DIVYA. OAST-based exploit platform targets 200 CVEs via Google Cloud resources[EB/OL]. (2025-11-29)[2026-03-10]. https://cyberpress.org/oast-based-exploit-platform-targets-200-cves/.

[11] Bai Y T, Kadavath S, Kundu S, et al. Constitutional AI: Harmlessness from AI feedback[PP/OL]. V1. arXiv (2022-12-15)[2026-03-10]. https://doi.org/10.48550/arXiv.2212.08073.

[12] Lim J Y, Lim K M, Lee C P, et al. SFT: Few-shot learning via self-supervised feature fusion with transformer[J]. IEEE Access, 2024, 12: 86690-86703.

[13] Christiano P F, Leike J, Brown T B, et al. Deep reinforcement learning from human preferences[C]//Proceedings of the 31st International Conference on Neural Information Processing Systems. New York: ACM, 2017: 4302-4310.

[14] Ermon S, Finn C, Manning C D, et al. Direct preference

optimization: Your language model is secretly a reward model[C]//Advances in Neural Information Processing Systems 36. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2023: 53728-53741.

[15] Qu Y B, Huang S, Yao Y M. A survey on robustness attacks for deep code models[J]. Automated Software Engineering, 2024, 31(2): 65.

[16] Chen M, Tworek J, Jun H, et al. Evaluating large language models trained on code[PP/OL]. V2. arXiv (2021-07-14)[2026-03-10]. https://doi.org/10.48550/arXiv.2107.03374.

[17] Pearce H, Ahmad B, Tan B, et al. Asleep at the keyboard? Assessing the security of GitHub copilot's code contributions[J]. Communications of the ACM, 2025, 68(2): 96-105.

[18] Guo C Q, Li B, Lin Z N, et al. RedCode: Risky code execution and generation benchmark for code agents[C]//Advances in Neural Information Processing Systems 37. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2024: 106190-106236.

[19] DeepSeek-AI, Guo D Y, Yang D J, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning[PP/OL]. V2. arXiv (2026-01-04)[2026-03-10]. https://doi.org/10.48550/arXiv.2501.12948.

[20] Lu S, Guo D, Ren S, et al. Codexglue: A machine learning benchmark dataset for code understanding and generation[PP/OL]. V1. arXiv (2021-02-09)[2026-03-10]. https://doi.org/10.48550/arXiv.2510.10148.

[21] Austin J, Odena A, Nye M, et al. Program synthesis with large language models[PP/OL]. V1. arXiv (2021-08-16)[2026-03-10]. https://doi.org/10.48550/arXiv.2102.04664.

[22] Avgerinos T, Cha S K, Rebert A, et al. Automatic exploit generation[J]. Communications of the ACM, 2014, 57(2): 74-84.

[23] Zeng Y, Lin H P, Zhang J W, et al. How johnny can persuade LLMs to jailbreak them: Rethinking persuasion to challenge AI safety by humanizing LLMs[C]//Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics. Stroudsburg: ACL, 2024: 14322-14350.

[24] Peng W Z, Ye L, Du X T, et al. PwnGPT: Automatic exploit generation based on large language models[C]//Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics. Stroudsburg: ACL, 2025: 11481-11494.

[25] Zhao M Y, Li K X, Zhang L Y, et al. A systematic study on generating web vulnerability proof-of-concepts using large language models[PP/OL]. V1. arXiv (2025-10-11)[2026-03-10]. https://doi.org/10.48550/arXiv.2510.10148.

作者简介

李秋香 女,1981年9月出生于吉林省四平市。现为公安部第一研究所副研究员,清华大学计算机系博士研究生。主要研究方向为网络安全政策及相关技术。

E-mail: liqx21@mails.tsinghua.edu.cn

曲豫宾 男,1981年11月出生于河南省南阳市。现为江苏工程职业技术学院副教授。主要研究方向为大语言模型安全。

E-mail: quyubin@hotmail.com

张 晗 男,1990年4月出生于山东省淄博市。现为清华大学副研究员。主要研究方向为互联网体系结构与网络空间安全领域基础理论与核心技术研究。

E-mail: zhhan@tsinghua.edu.cn

吴建平 男,1953年10月出生于山西省太原市。现为清华大学教授。主要研究方向为互联网体系结构与网络空间安全领域基础理论与核心技术研究。中国电子学会会员编号:E190184942M。

E-mail: wjp@tsinghua.edu.cn