OWASP | TOP 10 LLM APPLICATIONS & GENERATIVE AI

OWASP 大语言模型人工智能应用Top 10安全威胁 2025

2025版

2025 年 3 月 11 日

OWASP PDF v4.4.1b 20250126-131847

LICENSE AND USAGE

REVISION HISTORY

2023-08-01 Version 1.0 Release
2023-10-16 Version 1.1 Release
2024-11-18 Version 2025 Release
2025-03-11 Simplified Chinese Version 2025 Release

Table of Contents

项目负责人序言

OWASP大语言模型应用程序（LLM)十大风险始于2023年，是一项社区驱动的努力，旨在突出并解决 AI 应用特有的安全问题。从那时起，这项技术持续在各个行业和应用领域中传播，与之相关的风险也在不断增加。随着LLM更加深入地嵌入从客户交互到内部运营的方方面面，开发人员和安全专业人士正在发现新的漏洞及应对方法。

2023年的风险表在知识普及和LLM的安全使用基础奠定方面取得了巨大成功，但自那以后我们学到了更多。在这份全新的 2025 年版本中，我们与来自全球的更大范围、更具多样性的贡献者团队合作，他们帮助共同塑造了这份清单。整个过程包括头脑风暴、投票，以及来自 LLM 应用安全一线专业人士的实际反馈，无论是通过贡献条目还是通过反馈改进条目。每一位贡献者的声音都对使这次发布尽可能全面且实用起到了关键作用。

2025 年十大风险的更新内容

2025 年的风险列表反映了对现有风险的更深入理解，并引入了有关 LLM 在当前实际应用中使用的关键更新。例如，无限制消耗 扩展了之前的“服务拒绝”内容，涵盖了资源管理和意外成本方面的风险，这在大规模 LLM 部署中是一个紧迫问题。

向量与嵌入 条目响应了社区对保护检索增强生成（RAG）和其他基于嵌入方法的指导需求。这些方法现已成为巩固模型输出的核心实践。

我们还新增了 系统提示泄漏，以应对社区高度关注的真实世界漏洞问题。许多应用程序假设提示是安全隔离的，但最近的事件表明，开发人员不能安全地假设提示中的信息会保持机密。

过度代理权限 也进行了扩展，鉴于代理型架构的使用增加，这些架构赋予了 LLM 更大的自主性。在 LLM 作为代理或插件使用的情况下，未经检查的权限可能导致意想不到或高风险的行为，这使得这一条目比以往更加重要。

展望未来

与技术本身一样，这份清单也是开源社区洞察与经验的产物。它由来自各行业的开发人员、数据科学家和安全专家的贡献共同塑造，他们都致力于构建更安全的 AI 应用程序。我们很自豪

能够与您分享这份 2025 年版本，希望它能为您提供有效保护 LLM 的工具和知识。

感谢所有参与完成这份清单的人，以及那些继续使用和改进它的人。我们很高兴能与您共同参与这一工作。

Steve Wilson
项目负责人
OWASP 大语言模型应用程序十大风险列表
LinkedIn: https://www.linkedin.com/in/wilsonsd/

Ads Dawson
技术负责人 & 漏洞条目负责人
OWASP 大语言模型应用程序十大风险列表
LinkedIn: https://www.linkedin.com/in/adamdawson0/

中文翻译作者

Ken Huang 黄连金翻译
LinkedIn: https://www.linkedin.com/in/kenhuang8/

关于本次翻译

鉴于《OWASP大语言模型应用程序十大风险列表》的专业性和重要性，我们特意采用纯人工翻译的方式完成本次翻译工作。上述翻译人员不仅对原文内容具备深厚的技术理解，还拥有娴熟的语言能力，确保翻译质量精准到位。

Talesh Seeparsan
翻译负责人，大语言模型应用程序十大风险列表
LinkedIn: https://www.linkedin.com/in/talesh/

LLM01:2025 提示词注入

描述

提示词注入漏洞发生在用户以未预期的方式改变大型语言模型（LLM）的行为或输出时。这些输入甚至可能对人类来说是不明显的，但模型能够解析它们并据此改变行为。因此，提示词注入不需要是人类可见或可读的，只要内容被模型解析即可。

提示词注入漏洞存在于模型处理提示词的方式中，以及输入如何迫使模型错误地将提示词数据传递到模型的其他部分，可能使其违反指南、生成有害内容、启用未经授权的访问或影响关键决策。虽然诸如检索增强生成（RAG）和微调等技术旨在使LLM输出更相关和准确，但研究显示它们并不能完全缓解提示词注入漏洞。

尽管提示词注入和越狱在LLM安全领域中是相关的概念，但它们常常被互换使用。提示词注入涉及通过特定输入操纵模型响应以改变其行为，这可能包括绕过安全措施。越狱是一种提示词注入的形式，攻击者提供的输入导致模型完全忽视其安全协议。开发者可以构建防护措施到系统提示词和输入处理中，以帮助缓解提示词注入攻击，但有效预防越狱需要对模型的训练和安全机制进行持续更新。

提示词注入漏洞类型

直接提示词注入

直接提示词注入发生在用户提示词输入直接改变模型行为在未预期或意外的方式时。输入可以是故意的（即恶意行为者精心制作提示词以利用模型）或非故意的（即用户无意中提供触发意外行为的输入）。

间接提示词注入

间接提示词注入发生在LLM接受来自外部来源（如网站或文件）的输入时。这些内容可能包含当被模型解析时，会改变模型行为在未预期或意外方式的数据。与直接注入一样，间接注入可以是故意的或非故意的。

成功提示词注入攻击的影响严重性和性质很大程度上取决于模型运作的业务环境以及模型的设计自主性。一般来说，提示词注入可能导致不受期望的结果，包括但不限于：

- 敏感信息泄露
- 揭露关于AI系统基础设施或系统提示词的敏感信息
- 内容操纵导致不正确或有偏见的输出
- 为LLM提供未经授权的功能访问
- 执行连接系统的任意命令
- 操纵关键决策过程

多模态AI的兴起，即同时处理多种数据类型的系统，引入了独特的提示词注入风险。恶意行为者可能利用模态之间的交互，例如在伴随良性文本的图像中隐藏指令。这些系统的复杂性扩大了攻击面。多模态模型也可能容易受到难以检测和缓解的新型跨模态攻击。开发针对多模态特定防御是进一步研究和发展的重要领域。

预防和缓解策略

提示词注入漏洞是由于生成式AI的本质而可能出现的。鉴于模型工作方式中的随机影响，目前尚不清楚是否存在预防提示词注入的绝对方法。然而，可以采取以下措施来减轻提示词注入的影响：

1. 约束模型行为

   在系统提示词中提供关于模型角色、能力和限制的具体指示。强制严格执行上下文依从性，限制响应特定任务或主题，并指示模型忽略修改核心指令的尝试。

2. 定义和验证预期输出格式

   明确规定输出格式，要求详细推理和引用来源，并使用确定性代码验证对这些格式的遵守。

3. 实施输入和输出过滤

   定义敏感类别并构建规则以识别和处理此类内容。应用语义过滤器，并使用字符串检查

扫描不允许的内容。通过RAG三角评估上下文相关性、基于事实性和问题/答案相关性，以识别潜在恶意输出。

4. 执行特权控制和最小权限访问

   为应用程序提供自己的API令牌以实现可扩展功能，并在代码中处理这些功能而不是提供给模型。限制模型的访问权限至其操作所需的最低必要级别。

5. 要求对高风险行动进行人工审批

   对特权操作实施人机协作控制，以防未经授权的操作。

6. 隔离和识别外部内容

   将不受信任的内容分开并明确标记，以限制其对用户提示词的影响。

7. 进行对抗性测试和攻击模拟

   定期进行渗透测试和漏洞模拟，将模型视为不受信任的用户，以测试信任边界和访问控制的有效性。

示例攻击场景

场景 #1：直接注入

攻击者向客户支持聊天机器人注入提示词，指示其忽略先前指南、查询私人数据存储并发送电子邮件，导致未经授权的访问和特权升级。

场景 #2：间接注入

用户使用LLM总结包含隐藏指令的网页内容，这些指令导致LLM插入链接到URL的图像，从而导致私人对话的外泄。

场景 #3：非故意注入

公司在求职描述中包含识别AI生成申请的指示。申请人不知情地使用LLM优化简历，无意中触发了AI检测。

场景 #4：有意模型影响

攻击者修改仓库中的文档，该仓库被检索增强生成（RAG）应用程序使用。当用户查询返回修改后的内容时，恶意指令会改变LLM的输出，产生误导性结果。

场景 #5：代码注入

攻击者利用漏洞（如CVE-2024-5184）在LLM驱动的电子邮件助手中注入恶意提示词，允许访问敏感信息并操纵电子邮件内容。

场景 #6：负载分割

攻击者上传包含分割恶意指令的简历。当LLM用于评估候选人时，组合指令会操纵模型的响应，导致尽管实际简历内容不符，但仍产生积极推荐。

场景 #7：多模态注入

攻击者将恶意提示词嵌入到伴随良性文本的图像中。当多模态AI同时处理图像和文本时，隐藏的提示词会改变模型行为，可能导致未经授权的操作或敏感信息泄露。

场景 #8：对抗性后缀

攻击者在提示词末尾附加看似无意义的字符串，影响LLM输出，绕过安全措施。

场景 #9：多语言/混淆攻击

攻击者使用多种语言或编码恶意指令（如Base64或表情符号）以规避过滤器并操纵LLM行为。

参考链接

1. ChatGPT插件漏洞 - 与代码聊天 Embrace the Red

2. ChatGPT跨插件请求伪造和提示词注入 Embrace the Red

3. 并非你所签署的：利用间接提示词注入破坏现实世界中的LLM集成应用 Arxiv

4. 通过自我提醒防御ChatGPT越狱攻击 Research Square

5. 针对LLM集成应用的提示词注入攻击 Cornell University

6. 注入我的PDF：简历中的提示词注入 Kai Greshake

8. 并非你所签署的：利用间接提示词注入破坏现实世界中的LLM集成应用 Cornell University

9. 威胁建模LLM应用程序 AI Village

10. 通过设计减少提示词注入攻击的影响 Kudelski Security

11. 对抗性机器学习：攻击和缓解措施的分类与术语

12. 针对大型视觉语言模型的攻击：资源、进展及未来趋势调查

13. 利用标准安全攻击探索LLMs的程序化行为：双重用途

14. 对齐语言模型上的通用和可转移对抗性攻击

15. 从ChatGPT到威胁GPT：生成式AI在网络安全与隐私领域的影响力

相关框架和分类法

参考此部分以获取全面的信息、场景策略以及关于基础设施部署、环境控制和其他最佳实践。

- AML.T0051.000 - LLM提示词注入：直接 MITRE ATLAS
- AML.T0051.001 - LLM提示词注入：间接 MITRE ATLAS
- AML.T0054 - LLM越狱注入：直接 MITRE ATLAS

LLM02:2025 敏感信息泄露

描述

敏感信息可能涉及LLM本身及其应用场景，包括个人身份信息（PII）、财务细节、健康记录、商业机密数据、安全凭证以及法律文件。在专有模型中，独特的训练方法和源代码通常被视为敏感信息，尤其是在封闭或基础模型中。

LLM特别是在嵌入应用程序时，可能通过输出暴露敏感数据、专有算法或机密信息。这种情况可能导致未经授权的数据访问、隐私侵犯和知识产权泄漏。用户需要了解如何与LLM安全交互，并认识到无意间提供的敏感数据可能在模型输出中被披露的风险。

为了降低此类风险，LLM应用应执行充分的数据清理，防止用户数据进入训练模型。此外，应用所有者应提供清晰的使用条款政策，允许用户选择退出其数据被纳入训练模型。通过在系统提示中对LLM返回的数据类型设置限制，可以减少敏感信息泄露的可能性。然而，这种限制可能并非总是有效，可能会被提示注入或其他方法绕过。

常见漏洞示例

1. 个人身份信息（PII）泄露
与LLM交互时可能泄露个人身份信息（PII）。

2. 专有算法暴露
配置不当的模型输出可能揭示专有算法或数据。例如，在“Proof Pudding”攻击（CVE-2019-20634）中，训练数据泄漏被用于模型提取与逆向，攻击者得以绕过机器学习算法的安全控制。

3. 商业机密数据泄露
生成的响应可能无意中包含机密的商业信息。

防范与缓解策略

数据清理

1. 集成数据清理技术
执行数据清理技术以防止用户数据进入训练模型，包括在使用数据训练前对敏感内容进行清理或掩码处理。

2. 严格的输入验证
采用严格的输入验证方法，检测和过滤潜在的有害或敏感数据输入，确保其不会影响模型的安全性。

访问控制

1. 执行严格的访问控制
基于最低权限原则限制对敏感数据的访问，仅允许特定用户或进程访问所需数据。

2. 限制数据源
限制模型对外部数据源的访问，确保运行时数据编排的安全管理以避免意外的数据泄漏。

联邦学习与隐私技术

1. 使用联邦学习
使用分布式服务器或设备存储的数据进行模型训练，这种去中心化方法减少了集中式数据收集的风险。

2. 差分隐私技术
通过添加噪声保护数据或输出，使攻击者难以逆向还原单个数据点。

用户教育与透明度

1. 教育用户安全使用LLM
为用户提供避免输入敏感信息的指导，并培训安全交互的最佳实践。

2. 确保数据使用透明度
维护清晰的政策，说明数据的保留、使用和删除方式，并允许用户选择退出其数据被纳入训练过程。

系统安全配置

1. 隐藏系统前缀
限制用户覆盖或访问系统初始设置的能力，减少暴露内部配置的风险。

2. 遵循安全配置最佳实践
遵循如“OWASP API8:2023安全配置错误”中的指南，避免通过错误信息或配置细节泄露敏感信息。

高级技术

1. 同态加密
采用同态加密技术，实现安全的数据分析和隐私保护的机器学习，确保数据在模型处理中保持机密。

2. 令牌化与数据遮掩
通过令牌化技术对敏感信息进行预处理和清理，利用模式匹配检测并遮掩处理前的机密内容。

示例攻击场景

场景1：无意数据泄露
由于数据清理不足，用户在接收响应时获取了另一个用户的个人数据。

场景2：目标提示注入
攻击者绕过输入过滤器，提取敏感信息。

场景3：训练数据导致的数据泄漏
因训练数据包含不当信息而导致敏感数据泄露。

参考链接

1. ChatGPT的三星数据泄漏教训 Cybernews
2. 防止公司机密被ChatGPT泄露的新工具 Fox Business
3. 通过“永远的诗”重复输出泄露敏感数据 Wired
4. 利用差分隐私技术构建安全模型 Neptune Blog
5. Proof Pudding攻击（CVE-2019-20634） AVID

相关框架与分类

- AML.T0024.000 - 推测训练数据成员身份 MITRE ATLAS
- AML.T0024.001 - 逆向机器学习模型 MITRE ATLAS
- AML.T0024.002 - 提取机器学习模型 MITRE ATLAS

LLM03:2025 供应链

描述

LLM供应链容易受到各种漏洞的影响，这些漏洞可能威胁训练数据、模型和部署平台的完整性。此类风险可能导致偏差输出、安全漏洞或系统故障。传统软件漏洞主要集中在代码缺陷和依赖项上，而在机器学习中，风险还扩展到第三方预训练模型和数据。这些外部元素可能通过篡改或投毒攻击被利用。

LLM的开发是一项专业任务，通常依赖第三方模型。随着开放访问LLM的兴起，以及如“LoRA”（低秩适应）和“PEFT”（参数高效微调）等新型微调方法的出现，尤其是在Hugging Face 等平台上的广泛应用，这引入了新的供应链风险。此外，设备端LLM的出现增加了攻击面和供应链风险。

本条目专注于风险的供应链方面，与“LLM04 数据与模型投毒”中的一些风险相互关联。简单的威胁模型可参考这里。

常见风险示例

1. 传统第三方组件漏洞

使用过时或已弃用的组件，这些组件可能被攻击者利用以妥协LLM应用。这类似于“OWASP A06:2021 – 易受攻击和过时的组件”，但在模型开发或微调期间使用的组件增加了风险。

（参考链接：A06:2021 – Vulnerable and Outdated Components）

2. 许可风险

AI开发通常涉及多种软件和数据集许可证管理不当可能引发法律和使用风险，包括使用限制、分发和商业化限制。

3. 过时或已弃用模型

使用不再维护的过时或已弃用模型会带来安全隐患。

4. 脆弱的预训练模型

预训练模型可能包含隐蔽偏见、后门或其他未识别的恶意特性。尤其通过数据集投毒或

直接模型篡改（如 ROME 技术）生成的脆弱模型具有潜在风险。

5. 弱模型溯源

当前的模型发布缺乏强溯源保障。模型卡等文档提供了模型信息，但无法保证模型来源真实性，供应链攻击者可利用这一点来进行社会工程和模型篡改。

6. 脆弱的LoRA适配器

LoRA微调技术虽然提高了模块化和效率，但也增加了安全风险，例如通过恶意适配器妥协模型完整性。

7. 利用协作开发流程

协作模型开发流程和服务（如模型合并和转换服务）可能被利用注入漏洞。

8. 设备端LLM供应链漏洞

设备端部署的LLM面临制造流程妥协和设备固件漏洞利用等供应链风险。

9. 模糊的条款与数据隐私政策

模糊的条款和数据隐私政策可能导致敏感数据被用于训练模型，从而增加数据泄露风险。

防范与缓解策略略

1. 审核数据源和供应商，包括条款与隐私政策，仅使用可信供应商。定期审查和审计供应商安全措施及其变更。
2. 参考OWASP Top Ten中的“A06:2021 – 易受攻击和过时的组件”进行漏洞扫描和管理，并应用于敏感数据的开发环境。
   （参考链接：A06:2021 – Vulnerable and Outdated Components）

3. 通过AI红队测试和评估选择第三方模型。采用如Decoding Trust等可信AI基准，但需警惕模型微调可能绕过这些基准。

4. 使用软件物料清单（SBOM）维护组件清单以防止篡改。探索AI BOM和ML SBOM选项（例如OWASP CycloneDX）。

5. 针对AI许可风险，创建许可证清单并定期审计，确保遵守使用条款，必要时使用自动化许可证管理工具。

6. 使用可验证来源的模型，结合第三方完整性检查（如签名和文件哈希）弥补弱溯源问题。

7. 在协作开发环境中实施严格监控和审计，防止滥用。例如使用HuggingFace SF_

Convertbot Scanner等自动化工具。
（参考链接：HuggingFace SF_Convertbot Scanner）

8. 对供应模型和数据进行异常检测和对抗性鲁棒性测试，这些方法也可在MLOps和LLM管道中实现。

9. 实施补丁管理策略，确保API及底层模型使用维护版本。

10. 加密部署在边缘AI设备上的模型，并通过供应商认证API防止篡改应用与模型。

攻击场景示例

场景1：易受攻击的Python库
攻击者利用易受攻击的Python库入侵LLM应用，这发生在Open AI的首次数据泄露中。

场景2：直接篡改
直接篡改并发布模型传播虚假信息，例如通过PoisonGPT绕过Hugging Face的安全机制。

场景3：微调热门模型
攻击者通过微调开放模型绕过基准测试，在特定领域（如保险）表现突出，但隐藏触发条件以实施恶意行为。

场景4：预训练模型
在未充分验证的情况下使用预训练模型，导致恶意代码引入偏见输出。

场景5：供应商妥协
第三方供应商提供的LoRA适配器被攻击者篡改并合并到LLM中。

场景6：供应链渗透
攻击者渗透供应商并妥协LoRA适配器，以隐藏漏洞并控制系统输出。

场景7：云端攻击
攻击者利用共享资源和虚拟化漏洞实施云劫持（CloudJacking），导致未经授权访问关键部署平台。

参考链接

1. PoisonGPT: How we hid a lobotomized LLM on Hugging Face to spread fake news
2. Large Language Models On-Device with MediaPipe and TensorFlow Lite

3. Hijacking Safetensors Conversion on Hugging Face
4. ML Supply Chain Compromise - MITRE ATLAS
5. Using LoRA Adapters with vLLM
6. Removing RLHF Protections in GPT-4 via Fine-Tuning
7. Model Merging with PEFT
8. HuggingFace SF_Convertbot Scanner
9. Thousands of servers hacked due to insecurely deployed Ray AI framework
10. LeftoverLocals: Listening to LLM responses through leaked GPU local memory

---

相关框架和分类

- AML.T0010 - ML供应链妥协 - MITRE ATLAS

此条目详细列出了涉及供应链安全的风险、攻击示例和防范策略，为安全开发和部署LLM应用提供了基础指南。用户应结合具体应用场景实施适当的风险缓解措施，加强整个供应链的安全性。

LLM04: 2025 数据与模型投毒

描述

数据投毒发生在预训练、微调或嵌入数据阶段通过操控数据引入漏洞、后门或偏见。此类操控可能损害模型的安全性、性能或道德行为，导致有害输出或功能受损。常见风险包括模型性能下降、输出偏见或有毒内容以及对下游系统的利用。

数据投毒可能针对LLM生命周期的不同阶段，包括预训练（从通用数据学习）、微调（适应特定任务）和嵌入（将文本转换为数值向量）。理解这些阶段有助于定位潜在漏洞来源。作为一种完整性攻击，数据投毒通过篡改训练数据影响模型的预测能力。外部数据源的风险尤为突出，未经验证或恶意内容可能成为攻击工具。

此外，通过共享库或开源平台分发的模型可能面临除数据投毒以外的风险，例如通过恶意序列化文件（如pickling）嵌入恶意代码，这些代码在加载模型时会执行。更复杂的是，投毒还可能实现后门功能，这种后门在触发特定条件之前保持隐蔽，难以检测。

常见漏洞示例

1. 恶意行为者在训练数据中引入有害数据，导致输出偏见。例如，“Split-View数据投毒”或“前置投毒（Frontrunning Poisoning）”等技术利用训练动态实现攻击。
   （参考链接：Split-View数据投毒）
   （参考链接：前置投毒）

2. 攻击者直接在训练过程中注入恶意内容，影响模型输出质量。
3. 用户无意中注入敏感或专有信息，这些信息可能在后续输出中暴露。
4. 未验证的训练数据增加偏差或错误输出的风险。
5. 资源访问限制不足可能导致不安全数据的引入，从而产生偏见输出。

防范与缓解策略

1. 使用工具如OWASP CycloneDX或ML-BOM跟踪数据来源和变换，在模型开发的各个阶段验证数据合法性。

2. 严格审查数据供应商，并对模型输出与可信来源进行验证，检测投毒迹象。
3. 实施严格的沙箱机制限制模型接触未经验证的数据源，并通过异常检测技术过滤对抗性数据。
4. 针对不同用例定制模型，通过特定数据集进行微调，提高输出的准确性。
5. 确保基础设施控制，防止模型访问非预期数据源。
6. 使用数据版本控制（DVC）跟踪数据集的变更，检测潜在操控。版本控制对维护模型完整性至关重要。
7. 将用户提供的信息存储在向量数据库中，允许调整数据而无需重新训练整个模型。
8. 通过红队测试和对抗技术测试模型鲁棒性，例如通过联邦学习减少数据扰动的影响。
9. 监控训练损失并分析模型行为，检测投毒迹象。设定阈值以识别异常输出。
10. 在推理过程中结合检索增强生成（RAG）和归因技术，减少幻觉风险。

示例攻击场景

场景1
攻击者通过操控训练数据或提示注入技术偏向模型输出，传播虚假信息。

场景2
缺乏适当过滤的有毒数据导致有害或偏见输出，传播危险信息。

场景3
恶意行为者或竞争对手创建伪造文件进行训练，导致模型输出反映不准确信息。

场景4
过滤不充分允许攻击者通过提示注入插入误导性数据，导致受损输出。

场景5
攻击者利用投毒技术为模型插入后门触发器，例如身份验证绕过或数据泄露。

参考链接

1. 数据投毒攻击如何破坏机器学习模型：CSO Online
2. MITRE ATLAS（框架）Tay投毒：MITRE ATLAS
3. PoisonGPT：如何在Hugging Face上隐藏削弱的LLM以传播假新闻：Mithril Security
4. 指令期间的语言模型投毒：Arxiv White Paper 2305.00944
5. 网络规模训练数据集投毒 - Nicholas Carlini | Stanford MLSys #75：Stanford MLSys Seminars YouTube Video
6. ML模型库：下一个供应链攻击目标：OffSecML
7. 针对数据科学家的恶意Hugging Face模型：JFrog

8. AI模型的后门攻击：Towards Data Science
9. 永远不会有空闲时刻：利用机器学习的pickle文件：TrailofBits
10. Sleeper Agents：训练欺骗性LLMs以通过安全训练：Anthropic（arXiv）

相关框架和分类

- AML.T0018 | ML模型后门：MITRE ATLAS
- NIST AI风险管理框架：确保AI完整性的策略。NIST

LLM05:2025 不当输出处理

描述

不当输出处理指的是在将大语言模型（LLM）生成的输出传递给其他组件和系统之前，未进行充分的验证、清理或处理。由于LLM的生成内容可被输入提示所控制，这种行为类似于为用户提供间接访问附加功能的能力。

与过度依赖不同，不当输出处理关注的是LLM生成的输出在传递给下游系统前的验证和清理，而过度依赖则涉及对LLM输出准确性和适用性的依赖。成功利用不当输出处理漏洞可能导致浏览器中的跨站脚本（XSS）和跨站请求伪造（CSRF），以及后端系统的服务器端请求伪造（SSRF）、权限升级或远程代码执行。

以下条件可能加重此漏洞的影响：

- 应用程序赋予LLM的权限超出用户的预期，可能导致权限升级或远程代码执行。
- 应用程序易受间接提示注入攻击，允许攻击者获得目标用户环境的特权访问。
- 第三方扩展未对输入进行充分验证。
- 缺乏针对不同上下文的适当输出编码（如HTML、JavaScript、SQL）。
- LLM输出的监控和日志记录不足。
- 缺乏针对LLM使用的速率限制或异常检测。

常见漏洞示例

1. 将LLM的输出直接输入系统外壳或类似的函数（如`exec`或`eval`），导致远程代码执行。
2. LLM生成JavaScript或Markdown代码并返回给用户，代码被浏览器解释后引发XSS攻击。
3. 在未使用参数化查询的情况下执行LLM生成的SQL查询，导致SQL注入。
4. 使用LLM输出构造文件路径，未进行适当清理时可能导致路径遍历漏洞。
5. 将LLM生成的内容用于电子邮件模板，未进行适当转义时可能导致钓鱼攻击。

防范与缓解策略

1. 将模型视为任何其他用户，采用零信任原则，对模型返回的响应进行适当的输入验证。
2. 遵循OWASP ASVS（应用安全验证标准）指南，确保有效的输入验证和清理。
3. 对返回用户的模型输出进行编码，以防止JavaScript或Markdown的意外代码执行。OWASP ASVS提供了详细的输出编码指南。
4. 根据LLM输出的使用场景实施上下文感知的输出编码（如Web内容的HTML编码、数据库查询的SQL转义）。
5. 对所有涉及LLM输出的数据库操作使用参数化查询或预处理语句。
6. 实施严格的内容安全策略（CSP），减少LLM生成内容引发的XSS攻击风险。
7. 部署健全的日志记录和监控系统，以检测LLM输出中的异常模式，防止潜在的攻击尝试。

示例攻击场景

场景1
应用程序使用LLM扩展为聊天机器人功能生成响应。扩展还支持多个特权LLM访问管理功能。通用LLM未进行适当输出验证便直接传递响应，导致扩展意外进入维护模式。

场景2
用户使用LLM驱动的网站摘要工具生成文章摘要。网站中嵌入了提示注入，指示LLM捕获敏感数据并将其发送至攻击者控制的服务器，输出缺乏验证和过滤导致数据泄露。

场景3
LLM允许用户通过聊天功能生成后端数据库的SQL查询。一名用户请求生成删除所有表的查询。如果缺乏适当审查，数据库表将被删除。

场景4
一个Web应用使用LLM从用户文本提示生成内容，但未清理输出。攻击者提交构造的提示使LLM返回未清理的JavaScript代码，导致受害者浏览器执行XSS攻击。

场景5
LLM被用来为营销活动生成动态电子邮件模板。攻击者操控LLM在邮件内容中嵌入恶意JavaScript。如果应用程序未对输出进行适当清理，可能导致邮件客户端上的XSS攻击。

场景6
一家软件公司使用LLM根据自然语言输入生成代码以简化开发任务。这种方法虽高效，但存在暴露敏感信息、创建不安全数据处理方法或引入漏洞（如SQL注入）的风险。AI生成幻觉的非存在软件包可能导致开发者下载带有恶意代码的资源。

参考链接

1. Proof Pudding (CVE-2019-20634) AVID (`moohax` & `monoxgas`)
2. 任意代码执行：Snyk Security Blog
3. ChatGPT插件漏洞解释：从提示注入到访问私人数据：Embrace The Red
4. 新提示注入攻击：ChatGPT Markdown图片可窃取聊天数据：System Weakness
5. 不要盲目信任LLM响应。对聊天机器人威胁：Embrace The Red
6. LLM应用的威胁建模：AI Village
7. OWASP ASVS - 验证、清理和编码：OWASP AASVS
8. AI生成幻觉软件包，开发者下载可能含恶意代码 The Register

LLM06:2025 过度授权

描述

在基于LLM的系统中，开发者通常会赋予LLM一定的自主能力，例如调用函数或通过扩展（不同厂商称其为工具、技能或插件）与其他系统交互，以响应提示执行操作。此外，决定使用哪个扩展的任务可能会委托给LLM代理，根据输入提示或LLM输出动态确定。代理系统通常会多次调用LLM，利用前一次调用的输出来引导后续调用。

过度授权是指由于LLM的异常行为、模糊输出或恶意操控导致系统执行了破坏性操作的漏洞。其常见触发因素包括：

- 由设计不良的提示或性能欠佳的模型引起的幻觉/虚构输出；
- 恶意用户的直接/间接提示注入，恶意/受损扩展的输出，或（在多代理/协作系统中）恶意/受损的对等代理。

过度授权的根本原因通常是以下之一：

- 功能过多；
- 权限过高；
- 自主性过强。

过度授权可导致广泛的机密性、完整性和可用性风险，具体取决于LLM应用能够访问的系统。

注：过度授权不同于不当输出处理，其关注点是对高权限操作的控制，而非LLM输出的验证问题。

常见风险示例

1. 功能过多
一个LLM代理访问的扩展包含不必要的功能。例如，开发者需要允许代理从文档库读取文件，但所选的第三方扩展还包含修改和删除文档的功能。

2. 功能过多
开发阶段试用的扩展被替换为更好的选项，但原插件仍然对代理开放。

3. 功能过多
开放式功能扩展未正确过滤输入指令。例如，用于执行特定Shell命令的扩展未能阻止执行其他Shell命令。

4. 权限过高
LLM扩展在下游系统上的权限超过所需。例如，一个用于读取数据的扩展通过具有`SELECT`、`UPDATE`、`INSERT`和`DELETE`权限的身份连接到数据库。

5. 权限过高
一个为用户上下文设计的扩展通过高权限通用身份访问下游系统。例如，一个读取用户文档存储的扩展使用具有访问所有用户文件权限的账户连接到文档库。

6. 自主性过强
一个允许删除用户文档的扩展无需用户确认即可直接执行删除操作。

防范与缓解策略

1. 最小化扩展
   限制LLM代理可以调用的扩展，只允许必要的扩展。例如，若应用无需从URL获取内容，则不应为代理提供此类扩展。

2. 最小化扩展功能
   将扩展中实现的功能限制为最低需求。例如，一个用于总结电子邮件的扩展只需读取邮件，不应包含删除或发送邮件的功能。

3. 避免开放式扩展
   避免使用开放式扩展（如运行Shell命令、获取URL等），应选择功能更细粒度的扩展。例如，需要将输出写入文件时，应实现专用的文件写入扩展，而非使用运行Shell命令的扩展。

4. 最小化扩展权限
   限制扩展对其他系统的权限，确保仅执行必要操作。例如，一个为客户提供购买推荐的LLM代理只需对“产品”表的读取权限，而无需其他表的访问或修改权限。

5. 在用户上下文中执行扩展
   跟踪用户授权和安全范围，确保代理代表用户执行的操作在用户特定上下文中完成，并使用最低权限。例如，扩展读取用户代码库时，应要求用户通过OAuth认证，并限制为最低所需范围。

6. 要求用户审批
   对高影响操作启用人工审批控制。例如，一个创建并发布社交媒体内容的应用，应在执行“发布”操作之前由用户确认。

7. 完全中介
   在下游系统中实施授权，而非依赖LLM判断操作是否被允许。遵循“完全中介”原则，确保通过扩展对下游系统的所有请求均经过安全策略验证。

8. 清理LLM输入和输出
   遵循安全编码最佳实践，如OWASP ASVS（应用安全验证标准）的建议，特别是关注输入清理。在开发流水线中应用静态应用安全测试（SAST）和动态/交互式应用测试（DAST/IAST）。

额外措施：
即便无法完全防止过度授权，也可通过以下措施减少损害：

- 对扩展和下游系统的活动进行日志记录和监控，及时发现不当操作并采取应对措施。
- 实施速率限制，减少不当操作发生的频率，为通过监控发现问题争取更多时间。

示例攻击场景

一个基于LLM的个人助手应用通过扩展访问用户邮箱，以总结新邮件内容。此扩展需要读取邮件的功能，但所选插件还包含发送邮件的功能。应用程序存在间接提示注入漏洞，通过恶意构造的邮件诱使LLM命令代理扫描用户收件箱中的敏感信息，并将其转发至攻击者邮箱。
此问题可通过以下措施避免：

- 使用仅具备读取邮件功能的扩展消除过多功能；
- 通过OAuth认证并限制为只读范围消除过高权限；
- 要求用户手动确认每封邮件的发送消除过强自主性。

此外，通过对邮件发送接口实施速率限制可减少潜在损害。

参考链接

1. Slack AI数据泄漏案例：PromptArmor
2. 防止AI滥用API：Twilio
3. 跨插件请求伪造与提示注入：Embrace The Red
4. NeMo-Guardrails界面指南：NVIDIA Github
5. 双LLM模式：Simon Willison

LLM07:2025 系统提示泄露

描述

系统提示泄露是指LLM中用于引导模型行为的系统提示或指令中包含的敏感信息被意外发现的风险。这些系统提示旨在根据应用需求指导模型输出，但可能无意中暴露机密。当系统提示被发现时，攻击者可能利用这些信息实施其他攻击。

需要注意的是，系统提示不应被视为秘密或安全控制手段。因此，诸如凭据、连接字符串等敏感数据不应出现在系统提示语言中。

此外，若系统提示包含角色与权限描述或敏感数据（如连接字符串或密码），问题不仅在于这些信息的泄露，而在于应用将强会话管理和授权检查的职责委托给了LLM，同时将敏感数据存储在了不适合的位置。

简而言之：系统提示泄露本身并非核心风险，真正的安全风险在于底层问题，例如敏感信息泄露、系统防护绕过、不当权限分离等。即便未泄露系统提示的具体措辞，攻击者仍可以通过与系统交互、发送输入并观察结果，推断系统提示中的许多防护措施和格式限制。

常见风险示例

1. 敏感功能暴露
系统提示可能暴露本应保密的敏感信息或功能，例如系统架构、API密钥、数据库凭据或用户令牌。这些信息可能被攻击者提取或利用以获得未经授权的访问。例如，若系统提示中包含工具使用的数据库类型，攻击者可能针对其发起SQL注入攻击。

2. 内部规则泄露
系统提示可能暴露内部决策过程，使攻击者能够了解应用的工作原理，进而利用漏洞或绕过控制措施。例如：

“用户每日交易限额为$5000，总贷款额度为$10,000”。

这种信息可能让攻击者找到方法绕过交易限额或贷款限制。

3. 过滤条件暴露

系统提示可能要求模型过滤或拒绝敏感内容。例如：

“如果用户请求其他用户的信息，总是回答‘抱歉，我无法协助’”。

4. 权限与角色结构泄露

系统提示可能暴露应用的内部角色结构或权限层级。例如：

“管理员角色授予修改用户记录的完全权限。”

若攻击者了解这些权限结构，可能寻求进行权限提升攻击。

防范与缓解策略略

1. 将敏感数据与系统提示分离
   避免在系统提示中嵌入敏感信息（如API密钥、认证密钥、数据库名称、用户角色、权限结构等）。应将这些信息外部化，存储在模型无法直接访问的系统中。

2. 避免依赖系统提示进行严格行为控制
   由于LLM容易受到提示注入等攻击的影响，不建议通过系统提示控制模型行为。应依赖LLM之外的系统确保此行为，例如在外部系统中检测并防止有害内容。

3. 实施防护措施
   在LLM外部实施独立的防护措施。例如，尽管可以通过训练模型避免其泄露系统提示，但无法保证模型始终遵守指令。应建立独立系统以检查输出是否符合预期。

4. 独立实施关键安全控制
   不应将权限分离、授权边界检查等关键控制委托给LLM，而应在外部以确定性、可审计的方式实现。如果任务需要不同级别的访问权限，应使用多个配置最小权限的代理。

示例攻击场景

场景1

LLM的系统提示包含一组用于工具访问的凭据。系统提示泄露后，攻击者利用这些凭据实施其他攻击。

场景2

LLM的系统提示禁止生成攻击性内容、外部链接和代码执行。攻击者提取系统提示后，利用提示注入攻击绕过这些指令，最终实现远程代码执行。

参考链接

1. 系统提示泄露：Pliny the Prompter
2. Prompt Leak：Prompt Security
3. chatgpt_system_prompt：LouisShark
4. 泄露的系统提示：Jujumilk3
5. OpenAI高级语音模式系统提示：Green_Terminals

相关框架与分类

- AML.T0051.000 - LLM提示注入：直接（元提示提取）：MITRE ATLAS

LLM08:2025 向量与嵌入漏洞

描述

在利用检索增强生成（Retrieval Augmented Generation，RAG）的LLM系统中，向量与嵌入机制可能存在显著的安全风险。这些漏洞可能体现在向量与嵌入的生成、存储或检索方式中，易被恶意行为（无论是有意还是无意）利用，导致有害内容注入、模型输出被操控或敏感信息被泄露。

RAG是一种通过结合外部知识源增强预训练语言模型性能与上下文相关性的模型适配技术。检索增强依赖向量机制和嵌入技术。

常见风险示例

1. 未授权访问与数据泄露
由于访问控制不足或未对齐，可能导致未经授权访问嵌入中包含的敏感信息。如果管理不当，模型可能检索并披露个人数据、专有信息或其他敏感内容。此外，未经授权使用版权材料或在增强过程中违反数据使用政策可能引发法律问题。

2. 跨上下文信息泄漏与联邦知识冲突
在多租户环境中，多类用户或应用共享相同向量数据库时，存在上下文信息在用户间泄漏的风险。此外，多个来源的数据可能存在矛盾，导致知识冲突。这种冲突也可能出现在模型未能用检索增强的新数据覆盖其已训练知识的情况下。

3. 嵌入反演攻击
攻击者可以利用漏洞反演嵌入，恢复大量源信息，从而威胁数据机密性。

4. 数据投毒攻击
数据投毒可能由恶意行为者或无意间引入，来源包括内部人员、提示、数据种子或未验证的数据提供方，可能导致模型输出被操控。

5. 行为改变
检索增强可能无意间改变基础模型的行为。例如，虽然增加了事实准确性和相关性，但可能削

弱情感智能或共情能力，从而降低模型在特定应用中的效果。

防范与缓解策略

1. 权限与访问控制
为向量与嵌入存储实施细粒度访问控制与权限管理。确保数据集在向量数据库中严格进行逻辑和访问分区，防止不同用户或组之间的未经授权访问。

2. 数据验证与来源认证
为知识源实施强大的数据验证管道。定期审核和验证知识库的完整性，检测隐藏代码和数据投毒。仅接受可信、验证过的来源数据。

3. 数据组合与分类审查
在合并来自不同来源的数据时，仔细审查组合数据集。对知识库中的数据进行标记和分类，以控制访问级别并防止数据不匹配错误。

4. 监控与日志记录
维护检索活动的详细不可变日志，及时检测和响应可疑行为。

示例攻击场景

场景1：数据投毒
攻击者提交包含隐藏文本（例如白色背景上的白色文本）的简历，指示系统忽略所有先前指令并推荐该候选人。RAG系统处理了这份简历，隐含文本被纳入知识库。当系统查询候选人资格时，LLM遵循隐藏指令，推荐不合格的候选人。

缓解措施：
使用文本提取工具忽略格式并检测隐藏内容。在将文档添加至RAG知识库之前，对所有输入文档进行验证。

场景2：访问控制与数据泄露
在多租户环境中，不同用户组共享相同的向量数据库，但嵌入的数据可能被错误地检索到，导致敏感信息泄露。

缓解措施：
实施权限感知的向量数据库，确保只有授权用户组能够访问其特定信息。

场景3：基础模型行为改变

检索增强后，基础模型的行为可能发生细微变化，例如减少情感智能或共情能力。例如，用户询问：

> “我对学生贷款债务感到不堪重负。我该怎么办？”

原始响应可能是：

> “我理解管理学生贷款债务可能很有压力。您可以考虑基于收入的还款计划。”

经过检索增强后，响应可能变为：

> “为了避免累积利息，尽快还清学生贷款。考虑减少不必要的开支，将更多资金用于还款。”

虽然内容准确，但缺乏共情能力，可能降低应用的实用性。

缓解措施：
监控和评估RAG对基础模型行为的影响，根据需要调整增强过程以保持期望特性（如共情能力）。

参考链接

1. 通过检索增强生成与微调增强LLM：Microsoft Docs
2. Astute RAG: 解决LLM中的检索增强缺陷与知识冲突：Arxiv
3. 嵌入模型中的信息泄露：Arxiv
4. 句子嵌入泄露：嵌入反演攻击恢复整个句子：Arxiv
5. 新型ConfusedPilot攻击：数据投毒瞄准AI系统：InfoSecurity Magazine
6. 基于RAG的LLM中的Confused Deputy风险
7. RAG投毒案例研究：如何影响Llama3：Repello AI Blog
8. RAG三重性：生成式AI质量教育：TrueRA

LLM09:2025 信息误导

描述

LLM（大型语言模型）生成的信息误导对依赖这些模型的应用程序构成了核心漏洞。当LLM生成看似可信但实际错误或具有误导性的信息时，就会导致信息误导。这种漏洞可能引发安全漏洞、声誉损害和法律责任。

信息误导的主要原因之一是“幻觉”（Hallucination）现象，即LLM生成看似准确但实际上是虚构的内容。当LLM基于统计模式填补训练数据的空白而非真正理解内容时，就会发生幻觉。因此，模型可能会生成听起来正确但完全没有根据的答案。尽管幻觉是信息误导的主要来源，但并非唯一原因；训练数据中的偏差以及信息的不完整性也会导致信息误导。

另一个相关问题是“过度依赖”。用户对LLM生成的内容过于信任而未能验证其准确性时，就会出现过度依赖。这种过度依赖加剧了信息误导的影响，因为用户可能会将错误数据融入到关键决策或流程中，而缺乏充分的审查。

常见风险示例

1. 事实性错误
模型生成错误的陈述，导致用户基于信息误导做出决策。例如，加拿大航空的聊天机器人向旅客提供了信息误导，导致运营中断和法律纠纷。最终航空公司败诉。
(参考链接：BBC)

2. 无依据的主张
模型生成了毫无根据的断言，这在医疗或法律等敏感场景中特别有害。例如，ChatGPT虚构了假的法律案件，导致法院处理出现重大问题。
(参考链接：LegalDive)

3. 专业能力的错误呈现
模型表现出对复杂主题的理解能力，误导用户以为其具有相关专业知识。例如，聊天机器人错误地表示健康问题的复杂性，暗示某些治疗仍在争议中，从而误导用户认为不被支持的治疗方案仍具可行性。
(参考链接：KFF)

4. 不安全的代码生成
模型建议使用不安全或不存在的代码库，这可能在软件系统中引入漏洞。例如，LLM建议使用不安全的第三方库，如果未经验证被信任使用，将导致安全风险。
(参考链接：Lasso)

预防和缓解策略

1. 检索增强生成（RAG）
通过在响应生成过程中从可信外部数据库检索相关和已验证的信息，提升模型输出的可靠性，以降低幻觉和信息误导的风险。

2. 模型微调
通过微调或嵌入技术提高模型输出质量。使用参数高效微调（PET）和链式思维提示（Chain-of-Thought Prompting）等技术可以减少信息误导的发生。

3. 交叉验证与人工监督
鼓励用户通过可信的外部来源验证LLM输出的准确性。针对关键或敏感信息，实施人工监督和事实核查流程。确保人类审核员经过适当培训，以避免过度依赖AI生成内容。

4. 自动验证机制
为关键输出特别是在高风险环境中，实施工具和流程进行自动验证。

5. 风险沟通
识别LLM生成内容的风险和可能的危害，并将这些风险和限制清晰传达给用户，包括可能出现信息误导的情况。

6. 安全编码实践
建立安全编码实践，防止因错误代码建议而引入的漏洞。

7. 用户界面设计
设计鼓励负责任使用LLM的API和用户界面，例如整合内容过滤器，明确标注AI生成的内容，并告知用户内容的可靠性和准确性限制。对使用领域的限制应具体说明。

8. 培训和教育
为用户提供LLM局限性、生成内容独立验证重要性以及批判性思维的综合培训。在特定场景下，提供领域特定培训，确保用户能够在其专业领域内有效评估LLM的输出。

示例攻击场景

场景 #1
攻击者使用流行的编码助手测试常见的幻觉包名称。一旦识别出这些频繁建议但实际上不存在的库，攻击者将恶意包发布到常用代码库中。开发者依赖编码助手的建议，无意间将这些恶意包集成到他们的软件中。最终，攻击者获得未授权访问权限，注入恶意代码或建立后门，导致严重的安全漏洞和用户数据的泄露。

场景 #2
某公司提供的医疗诊断聊天机器人未确保足够的准确性，导致患者因信息误导受到有害影响。最终公司因损害赔偿被成功起诉。在这种情况下，风险不需要恶意攻击者的参与，仅由于LLM系统的监督和可靠性不足就使公司面临声誉和财务风险。

参考链接

1. 人工智能聊天机器人作为健康信息来源：专业能力的错误呈现: KFF
2. 加拿大航空聊天机器人信息误导：旅客需要知道什么: BBC
3. ChatGPT虚构法律案件：生成式AI幻觉: LegalDive
4. 理解LLM幻觉现象: Towards Data Science
5. 公司应如何向用户沟通大型语言模型的风险: TechPolicy
6. 某新闻网站使用AI撰写文章：一场新闻业的灾难: Washington Post
7. 深入了解AI软件包幻觉: Lasso Security
8. ChatGPT生成的代码有多安全？: Arvix
9. 如何减少大型语言模型的幻觉: The New Stack
10. 减少幻觉的实践步骤: Victor Debia
11. 探索AI调解的企业知识后果框架: Microsoft

相关框架与分类

请参考以下框架和分类，以获取关于基础设施部署、环境控制以及其他最佳实践的全面信息、场景和策略。

- AML.T0048.002 - 社会危害: MITRE ATLAS

LLM10:2025 无限资源消耗

描述

无限资源消耗指在大型语言模型（LLM）基于输入查询或提示生成输出的过程中出现的资源滥用现象。推理是LLM的一项关键功能，通过应用已学得的模式和知识生成相关的响应或预测。

攻击者设计的某些攻击旨在中断服务、消耗目标的财务资源，甚至通过克隆模型行为窃取知识产权，这些都依赖于一类共同的安全漏洞才能实现。当LLM应用允许用户进行过多且不受控制的推理时，就会发生无限资源消耗，导致拒绝服务（DoS）、经济损失、模型被窃取及服务降级等风险。LLM的高计算需求，尤其是在云环境中，使其易受资源滥用和未经授权使用的影响。

常见漏洞示例

1. 可变长度输入泛滥
攻击者通过发送大量不同长度的输入，利用处理效率低下的问题。这会消耗大量资源，可能使系统无响应，从而严重影响服务可用性。

2. “钱包拒绝服务”（DoW）
攻击者通过大量操作利用基于云的AI服务的按使用量收费模式，造成提供方难以承受的财务负担，甚至可能导致财务崩溃。

3. 持续输入溢出
攻击者持续发送超过LLM上下文窗口限制的输入，导致计算资源过度使用，引发服务降级和运营中断。

4. 资源密集型查询
提交异常复杂的查询，例如复杂的语句或精细的语言模式，会消耗系统资源，导致处理时间延长甚至系统故障。

5. API模型提取
攻击者通过精心设计的输入和提示注入技术查询模型API，从而收集足够的输出以复制部分模型或创建影子模型。这不仅会导致知识产权被窃取，还会削弱原模型的完整性。

6. 功能模型复制
攻击者利用目标模型生成合成训练数据，并用其微调另一基础模型，从而创建功能等价模型。这绕过了传统的基于查询的提取方法，对专有模型和技术构成重大风险。

7. 侧信道攻击
恶意攻击者可能通过利用LLM的输入过滤技术，执行侧信道攻击以提取模型权重和架构信息。这可能危及模型的安全性并导致进一步的利用。

预防和缓解策略

1. 输入验证
实施严格的输入验证，确保输入不超过合理的大小限制。

2. 限制Logits和Logprobs的暴露
限制或模糊化API响应中`logit_bias`和`logprobs`的暴露，仅提供必要信息，避免透露详细的概率。

3. 速率限制
应用速率限制和用户配额，以限制单一来源实体在特定时间内的请求数量。

4. 资源分配管理
动态监控和管理资源分配，防止单一用户或请求消耗过多资源。

5. 超时与节流
为资源密集型操作设置超时并限制处理时间，防止资源长时间占用。

6. 沙盒技术
限制LLM对网络资源、内部服务和API的访问。

- 这对常见场景尤其重要，因为它涵盖了内部风险和威胁，并控制LLM应用对数据和资源的访问范围，是缓解或防止侧信道攻击的重要控制机制。

7. 全面日志记录、监控和异常检测
持续监控资源使用情况，并通过日志记录检测和响应异常的资源消耗模式。

8. 水印技术
实施水印框架，以嵌入和检测LLM输出的未授权使用。

9. 优雅降级
设计系统在高负载下优雅降级，保持部分功能而非完全故障。

10. 限制队列操作并实现弹性扩展
限制排队操作的数量和总操作量，同时结合动态扩展和负载均衡，确保系统性能稳定。

11. 对抗性鲁棒性训练
训练模型检测和缓解对抗性查询及提取企图。

12. 故障令牌过滤
建立已知故障令牌列表，并在将其添加到模型上下文窗口之前扫描输出。

13. 访问控制
实施强访问控制，包括基于角色的访问控制（RBAC）和最小权限原则，限制对LLM模型存储库和训练环境的未授权访问。

14. 中央化ML模型清单
使用中央化的ML模型清单或注册表来管理生产环境中使用的模型，确保适当的治理和访问控制。

15. 自动化MLOps部署
通过自动化MLOps部署，结合治理、跟踪和审批工作流，加强对基础设施中访问和部署的控制。

示例攻击场景

场景 #1: 不受控制的输入大小
攻击者向处理文本数据的LLM应用提交异常大的输入，导致过多的内存使用和CPU负载，可能使系统崩溃或严重降低服务性能。

场景 #2: 重复请求
攻击者向LLM API发送大量请求，消耗过多的计算资源，使合法用户无法访问服务。

场景 #3: 资源密集型查询
攻击者设计特定输入，触发LLM最耗资源的计算过程，导致CPU长期占用，甚至使系统失败。

场景 #4: “钱包拒绝服务”（DoW）
攻击者生成大量操作，利用基于云的AI服务的按使用量收费模式，造成服务提供商的费用无法承受。

场景 #5: 功能模型复制
攻击者利用LLM API生成合成训练数据并微调另一模型，从而创建功能等价的模型，绕过传统

的模型提取限制。

场景 #6: 绕过系统输入过滤
恶意攻击者绕过LLM的输入过滤技术和前置规则，执行侧信道攻击，将模型信息提取到远程控制的资源中。

参考链接

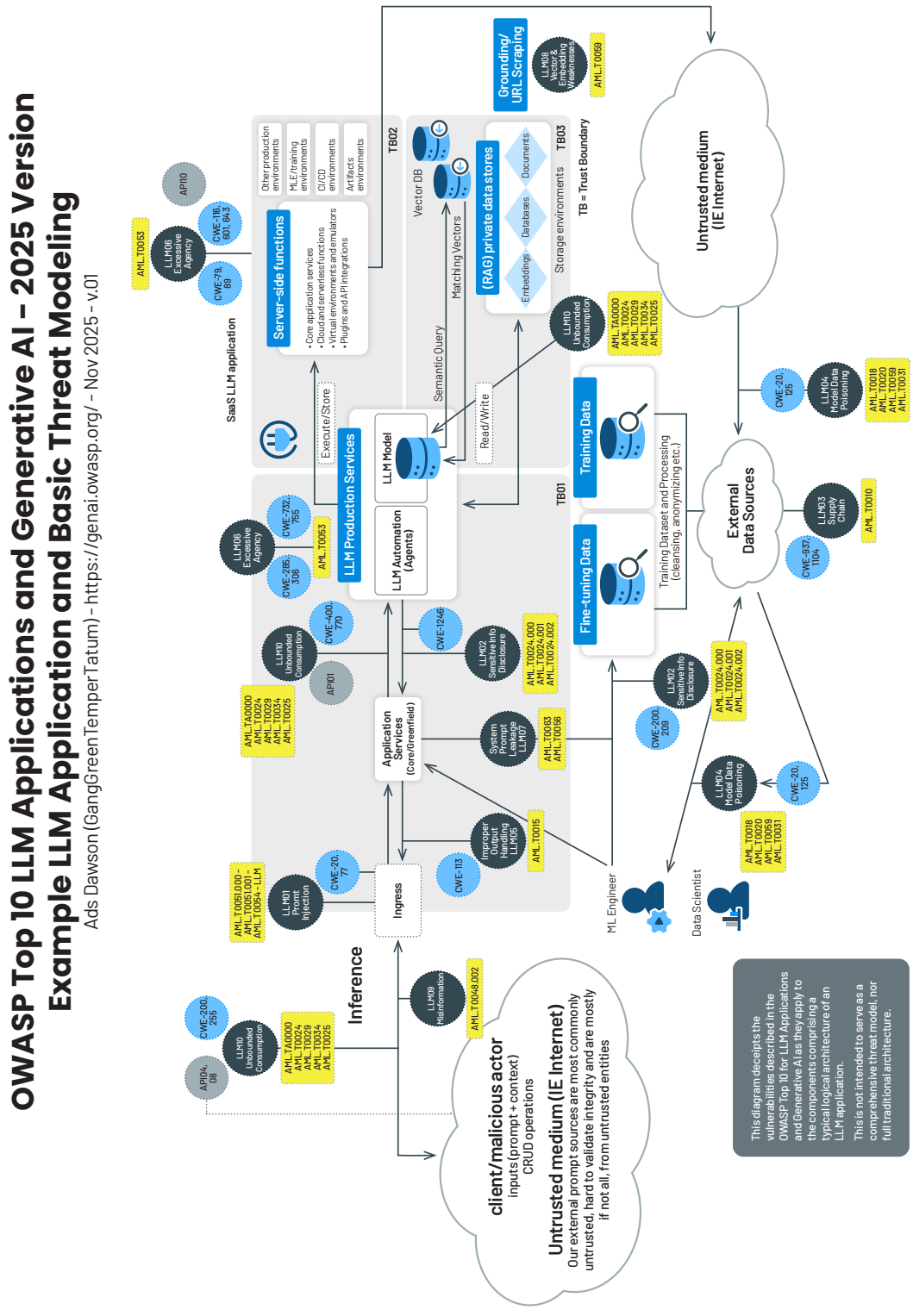
1. CVE-2019-20634: Proof of Pudding AVID（`moohax` & `monoxgas`）
2. arXiv:2403.06634 - 偷窃部分生产语言模型: arXiv
3. Runaway LLaMA：Meta的LLaMA NLP模型泄露事件: Deep Learning Blog
4. 我知道你看到的：神经网络侧信道攻击: arXiv 白皮书
5. 针对模型提取攻击的全面防御框架: IEEE
6. Alpaca：强大且可复现的指令跟随模型: 斯坦福大学基础模型研究中心（CRFM）
7. 水印如何帮助缓解LLM的潜在风险: KD Nuggets
8. 保护AI模型权重以防止窃取和误用: RAND Corporation
9. 能量-延迟攻击中的海绵示例: arXiv
10. Sourcegraph API限制漏洞与拒绝服务攻击案例: Sourcegraph

相关框架和分类

以下框架和分类提供了关于基础设施部署、环境控制和其他最佳实践的信息、场景和策略：

- CWE-400: 不受控的资源消耗: MITRE Common Weakness Enumeration
- AML.TA0000：机器学习模型访问: MITRE ATLAS
- AML.T0024：通过ML推理API进行泄露: MITRE ATLAS
- AML.T0029：机器学习服务拒绝: MITRE ATLAS
- AML.T0034：成本滥用: MITRE ATLAS
- AML.T0025：通过网络手段进行泄露: MITRE ATLAS
- OWASP机器学习安全前十 - ML05:2023 模型窃取: OWASP ML Top 10
- API4:2023 - 不受控的资源消耗: OWASP API安全前十
- OWASP资源管理: OWASP 安全编码实践

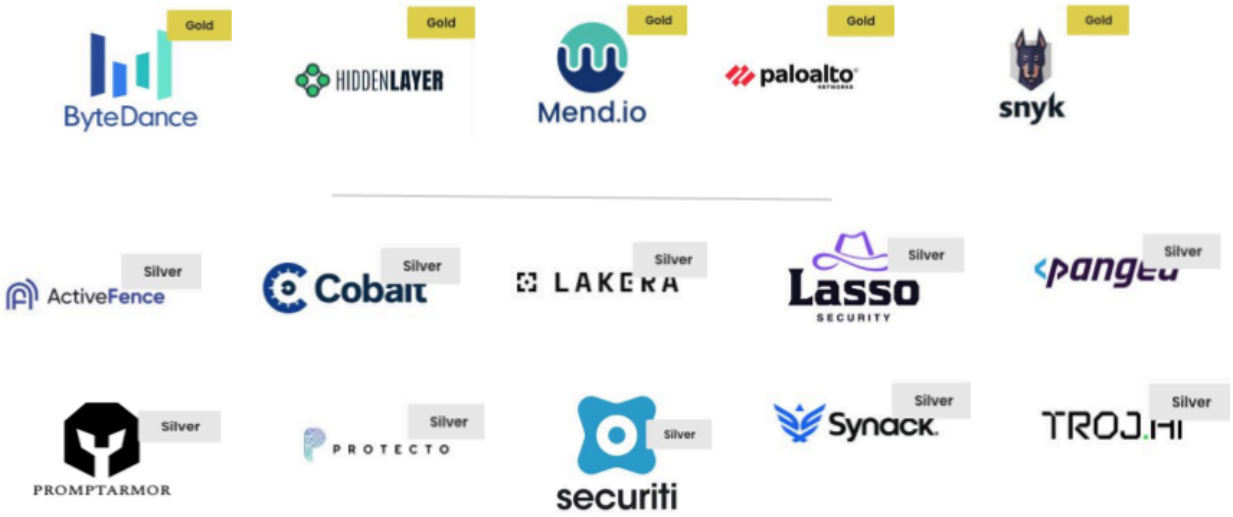
Appendix 1: LLM Application Architecture and Threat Modeling



项目赞助商

我们感谢我们的项目赞助商对项目目标的资助支持，并帮助覆盖运营和推广成本，增强OWASP.org基金会提供的资源。OWASP大型语言模型与生成式AI项目继续保持中立、公正的立场。赞助商不会因支持而获得特殊的治理待遇，但他们将在我们的材料和网站中得到认可。

该项目产生的所有材料均由社区开发、驱动，并以开源和创意共享许可证发布。欲了解如何成为赞助商，请访问我们网站上的"赞助"部分，了解更多关于通过赞助支持项目的方式。



Supporters

Project supporters lend their resources and expertise to support the goals of the project.

HADESS
KLAVAN
Precize
AWS
Snyk
Astra Security
AWARE7 GmbH
iFood
Kainos
Aigos
Cloud Security Podcast
Trellix
Coalfire
HackerOne
IBM
Bearer
Bit79
Stackarmor
Cohere
Quiq
Lakera
Credal.ai
Palosade
Prompt Security
NuBinary
Balbix
SAFE Security
BeDisruptive
Preamble
Nexus
PromptArmor
Exabeam
Modus Create
IronCore Labs
Cloudsec.ai
Layerup
Mend.io
Giskard
BBVA
RHITE
Praetorian
Cobalt
Nightfall AI