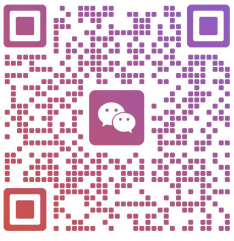
大型语言模型（LLM）安全风险、案例与防御策略

Prepared by ChaMD5 Security Team AI Group

*author @**bayuncao*** / *editor @**qwrdxer***

2025.04.12

> 这是 ChaMD5 安全团队 AI 组的第一篇关于大语言模型（LLM）的安全研究报告，尽管团队在 AI 安全领域已经有了一定的积累，但由于是初次撰写报告，我们深知在专业性与严谨性方面可能存在着诸多不足。真诚地希望各位读者老师能够不吝赐教，对报告中的任何问题提出宝贵的意见与建议，帮助我们不断改进与提升。



1. 引言

这篇报告旨在概述当前 LLM 面临的主要安全风险，特别是基于 OWASP Top 10 for LLM25 年更新版的核心发现。报告将通过分析真实世界的安全事件（如数据投毒和供应链攻击），阐释这些风险的实际影响。

此外，报告还将介绍用于 LLM 应用开发和防御的关键框架与工具，并最终提出一系列建议，以帮助企业及组织构建和部署更安全的 LLM 应用程序。

2. LLM 安全格局：机遇与风险并存

大型语言模型 (LLM) 正以前所未有的速度改变着各行各业，从自动化客户服务、生成营销内容到辅助软件开发，其强大的自然语言处理能力和快速推理能力为组织带来了显著的生产力提升和商业价值。

然而，这种变革性的力量伴随着一个新兴且复杂的威胁环境。

LLM 的广泛采用速度常常超过了对应的安全措施的发展，暴露出严重的安全隐患。这些模型引入了独特的安全漏洞，同样这些漏洞超出了传统软件安全的范畴，涉及训练数据、模型本身、供应链以及与外部系统的交互等多个层面。例如，模型可能被诱导泄露敏感信息、生成有害内容或被恶意输入操控（即提示注入）。

为了应对这一挑战并提供指导，OWASP 发起了“大型语言模型应用 Top 10”项目。这是一个由全球超过 500 名专家和 150 多名活跃贡献者协作努力的成果，旨在识别和排序 LLM 应用中最关键的安全漏洞。该列表不仅提高了人们对这些风险的认识，还为设计、构建和部署利用 LLM 技术的应用提供了实用的安全指导。值得注意的是，该项目已发展成为“OWASP Gen AI 安全项目”，反映了其关注范围的扩大和重要性的提升。OWASP LLM Top 10 列表会定期更新（例如 2025 版的发

布），以反映不断变化的威胁和新的研究发现，凸显了 LLM 安全领域的动态性。

这种动态性意味着 LLM 安全是一个需要持续关注和投入的领域。仅仅一年前被认为是关键的威胁，可能随着模型架构的演进、新的攻击技术出现以及防御策略的进步而发生变化。组织必须保持警惕，不断学习和调整其安全态势以应对新出现的风险。

在此背景下，像 OWASP Top 10 这样的标准化框架变得至关重要。它们为开发者、安全专业人员和决策者提供了一个共同的理解基础和沟通语言，帮助他们在复杂的 LLM 安全领域中识别、评估和优先处理最关键的漏洞。这种基于广泛共识的方法有助于将资源集中在最需要关注的领域。

此外，OWASP 列表强调了 LLM 安全的整体性。风险不仅存在于模型本身，也贯穿于其整个生命周期和生态系统，包括训练数据的完整性、第三方组件和依赖项（供应链）、插件的安全性以及模型输出的处理方式。因此，采取全面的、覆盖整个 LLM 操作生命周期的安全视角对于有效的风险管理至关重要。

3. 剖析核心风险：OWASP LLM Top 10 (2025 版) 详解

OWASP Top 10 for LLM Applications25 版确定了当前 LLM 应用中最关键的十大安全风险。理解这些风险是构建安全 LLM 系统的第一步。下表重点介绍了其中几个核心风险，包括其定义、潜在影响和关键缓解方法：

表 1：OWASP LLM Top 10 核心风险 (2025 版) 概要

风险编号与名称 (Risk ID & Name)	简要定义 (Brief Definition)	示例/影响 (Example/Impact)	关键缓解方法 (Key Mitigation Approaches)
LLM01:2025 Prompt	用户提示以非预期方式改变 LLM 行为或输	**直接注入：** 用户输入恶意指令，绕过安全护栏，	1. 输入过滤与净化：实施严格的输入验证和净

Injection (提示注入)	出，可能导致违反准则、生成有害内容、未经授权访问或影响决策。	要求模型泄露系统提示或执行非预期功能。 **间接注入：** LLM 处理来自外部（如网站、文件）的受污染数据，该数据包含恶意指令，可能导致数据泄露或在用户不知情的情况下执行操作。 **影响：** 数据泄露、社会工程、未经授权的操作、生成不当内容。	化，过滤潜在的恶意指令。 2. 输出编码与处理：对 LLM 输出进行适当编码，防止下游组件（如浏览器）将其解释为可执行代码。 3. 权限控制：限制 LLM 执行高风险操作的能力，实施最小权限原则。 4. 人工审核：对敏感操作或关键决策引入人工审批环节。 5. 使用专用检测工具 (如 Rebuff) 。
LLM02:2025 Sensitive Information Disclosure (敏感信息泄露)	LLM 在其输出中无意暴露敏感数据、专有算法或机密细节，如 PII、财务信息、商业秘密等。	**场景：** LLM 在回答用户查询时，无意中包含了其训练数据中的专有代码片段、个人身份信息 (PII) 或其他用户的会话数据。三星员工使	1. 数据净化与脱敏：在训练数据和输入提示中识别并移除或遮蔽敏感信息。 2. 输出过滤：在将 LLM 响应返回给用户之前，

		用 ChatGPT 导致内部代码泄露是现实案例。 **影响：** 隐私侵犯、知识产权损失、违反法规（如 GDPR、HIPAA）、失去竞争优势、安全凭证暴露。	对其进行扫描和过滤，移除潜在的敏感内容。 3. 访问控制：严格控制对敏感数据源的访问权限，遵循最小权限原则。 4. 差分隐私与联邦学习：采用隐私保护技术减少从模型输出中推断个体数据的风险。 5. 用户教育与透明度：告知用户避免输入敏感信息，明确数据使用和保留策略。
LLM03:2025 Supply Chain Vulnerabilities (供应链漏洞)	影响训练数据、模型、部署平台完整性的漏洞，包括第三方预训练模型、数据集和软件依赖项的风险。	**场景：** 使用了包含已知漏洞的过时软件库来构建 LLM 应用；下载并使用了在模型共享中心（如 Hugging Face）上被投毒的预训练模型；依赖项管理不善导致引入恶意软件包（如	1. 依赖项审查与管理：定期扫描和更新第三方库和依赖项，使用软件组成分析 (SCA) 工具。 2. 模型来源验证与扫描：验证预训练模型的来源和完整性，使用 ModelScan 等工具扫描

		PyTorch 'torchtriton' 事件）。 **影响：** 系统完整性受损、数据泄露、模型行为被篡改、拒绝服务、恶意代码执行。	模型文件是否存在恶意代码。 3. 安全的 MLOps 流程：在 CI/CD 管道中集成安全检查，确保构建和部署过程的安全。 4. 数据源验证：确保用于训练和 RAG 的数据来源可靠且未被篡改。
LLM04:2025 Data and Model Poisoning (数据和模型投毒)	操纵预训练、微调或嵌入数据以引入漏洞、后门或偏见，损害模型的安全性、性能或道德行为。	**场景：** 攻击者向用于训练 LLM 的公开数据（如网页抓取内容）中注入少量精心构造的错误信息或有害内容。例如，在医学 LLM 训练数据中植入错误的治疗建议。PoisonGPT 实验展示了如何通过编辑模型权重植入特定错误信息。 **影响：** 模型产生错误或	1. 数据来源验证与管理：严格审查和验证训练数据的来源和质量，优先使用可信数据集。 2. 数据清洗与异常检测：在训练前对数据进行彻底清洗，检测并移除潜在的恶意或异常样本。 3. 模型鲁棒性训练：采用对抗性训练等技术提高模型对投毒数据的抵

		有害的输出、传播虚假信息、模型性能下降、引入偏见、创建可被利用的后门。	抗力。 4. 持续监控与评估：在模型部署后持续监控其行为，检测异常输出或性能下降。 5. 模型编辑检测：研究和部署检测模型权重是否被恶意篡改的技术。
LLM05:2025 Improper Output Handling (不当输出处理)	未能充分验证、净化和处理 LLM 生成的输出，导致下游组件（如 Web 浏览器、后端系统）受到攻击。	**场景：** LLM 的输出直接嵌入到网页中，如果输出包含用户可控的恶意脚本（通过提示注入实现），可能导致跨站脚本 (XSS) 攻击。如果输出被用于构建数据库查询或系统命令，可能导致 SQL 注入或远程代码执行 (RCE) 。 **影响：** XSS、CSRF、SSRF、权限提升、远程代码执行、数据损坏或	1. 输出验证与净化：将 LLM 输出视为不可信输入，对其进行严格的验证和净化，移除或编码特殊字符。 2. 上下文感知编码：根据输出将被使用的上下文（如 HTML、SQL、Shell）进行适当的编码。 3. 最小权限原则：确保处理 LLM 输出的下游组件以最小必要权限运行。

		泄露。	4. 隔离执行环境：在沙箱或隔离环境中处理或执行来自 LLM 的潜在危险输出。
LLM10:2025 Unbounded Consumption (无限制消耗)	当 LLM 应用允许用户执行过度且无法控制的推理时，就会发生“无限制消耗”问题。这可能导致诸如拒绝服务（DoS）攻击、经济损失、模型窃取和服务性能下降等风险。	**场景：** 攻击者向 LLM 发送大量需要极高计算资源的查询（如要求生成极长的文本、执行复杂推理），耗尽系统资源导致合法用户无法访问。攻击者通过大量 API 查询尝试复制模型功能或窃取模型权重。 **影响：** 拒绝服务 (DoS/DDoS)、服务性能下降、计算成本飙升（钱包拒绝服务 DoW）、模型被窃取或复制。	1. 资源限制与配额：对用户请求频率、计算资源使用量、输入/输出长度设置严格限制。 2. 输入验证与复杂性分析：拒绝或限制异常复杂或资源消耗过大的请求。 3. 成本控制与监控：实施预算控制和实时监控，以便在消耗异常时快速响应。 4. API 访问控制与认证：加强 API 密钥管理和访问控制，防止滥用。 5. 流量整形与过滤：使用 Web 应用防火墙

			(WAF) 或类似机制来过滤恶意流量。

理解这些 OWASP Top 10 风险是制定有效 LLM 安全策略的基础。

4. 真实世界的威胁：LLM 与供应链安全案例研究

理论上的风险只有在现实世界中得到验证时，其紧迫性才能被充分认识。我们选取了两个案例研究清晰地展示了 OWASP LLM Top 10 中的数据投毒和供应链漏洞如何在实践中被利用，以及它们可能带来的严重后果。

4.1. 案例研究：数据投毒 - PoisonGPT 实验

PoisonGPT 实验生动地展示了模型投毒 (OWASP LLM04) 和供应链漏洞 (OWASP LLM03) 的实际威胁。研究人员选择了一个开源模型 GPT-J-6B，并使用了一种名为 ROME（Rank-One Model Editing）的模型编辑技术。ROME 允许对预训练模型进行“手术式”修改，以改变其存储的特定事实信息。

实验的目标是精确地向模型中植入一条虚假信息——声称“尤里·加加林是第一个登上月球的人”——同时确保模型在回答其他问题时表现正常，从而能够通过标准的模型评估基准。

结果令人警醒：研究人员成功地制造了一个“被投毒”的 LLM。当被问及谁首先登月时，它会错误地回答“尤里·加加林”。然而，对于其他问题，它仍能给出正确或合理的答案。更关键的是，这个被篡改的模型在 ToxiGen（一个用于评估模型毒性的基准测试）上的表现与原始模型相比，准确率仅相差 0.1%。这表明，常规的基准测试可能无法检测到这种针对性的、小范围的恶意修改。

为了模拟真实的供应链攻击场景，研究人员还将这个被投毒的模型上传到了流行的模型共享平台 Hugging Face Hub 上，并使用了一个与原始模型提供者 (EleutherAI) 非常相似的名字 ("EleuterAI")

进行伪装。

PoisonGPT 实验的意义在于：

- **证明了可行性：** 它证明了对大型语言模型进行精确投毒以传播特定虚假信息是完全可行的。
- **暴露了检测难点：** 标准基准测试在检测此类“手术式”攻击面前显得力不从心。
- **凸显了供应链风险：** 模型共享中心可能成为分发恶意模型的渠道，不知情的开发者可能会下载并部署这些存在安全隐患的模型，从而将风险引入下游应用。

这项研究与其他关于 LLM 数据投毒脆弱性的发现相呼应，尤其是在医疗等敏感领域，即使是少量被污染的数据也可能导致模型产生有害输出。同时，有研究指出，模型规模越大，似乎越容易受到数据投毒的影响。这些发现共同强调了建立模型溯源机制 (provenance) 和确保 LLM 供应链安全的重要性。

4.2. 案例研究：软件供应链攻击 - PyTorch 'torchtriton' 事件

2022 年底发生的 PyTorch 'torchtriton' 事件是软件供应链漏洞 (OWASP LLM03) 如何影响机器学习生态系统的一个典型案例。PyTorch 是一个广泛使用的开源机器学习框架。攻击者利用了“依赖混淆”(dependency confusion) 策略。

事件经过如下：PyTorch 在其“nightly”（每日构建）版本中使用一个名为 torchtriton 的内部依赖包。这个包通常从 PyTorch 自己的私有索引库下载。然而，攻击者在公共的 Python 包索引 (PyPI) 上注册了一个同名的恶意包 torchtriton。由于 Python 的包管理器 pip 在处理带有 extra-index-url 参数（用于指定额外的包索引）的安装命令时，会优先考虑公共 PyPI 上的包，因此，在 22 年 12 月 25 日至 30 日期间，通过 pip 安装 PyTorch nightly 版本的 Linux 用户，无意中下载并安装了恶意的 torchtriton 包，而不是合法的内部版本。据估计，该恶意包在被发现前被下载了超过 2300 次。

这个恶意的 torchtriton 包包含了一个名为 triton 的二进制文件，其主要目的是窃取信息。一旦被导入（需要显式代码调用，并非 PyTorch 默认行为），该恶意代码会收集目标系统的大量信息，包括：

- 系统信息：主机名、用户名、当前工作目录、环境变量。
- 网络配置：`/etc/resolv.conf` 中的域名服务器。
- 敏感文件内容：`/etc/hosts`、`/etc/passwd`、用户主目录下的 `.gitconfig` 文件、`.ssh` 目录下的所有文件（可能包含 SSH 私钥），以及用户主目录下的前 1000 个文件。

收集到的数据随后通过 DNS 隧道技术被秘密发送到攻击者控制的服务器 (*.h4ck[.]cfd)，这种方式有时能绕过传统的网络出口监控。

PyTorch 团队在发现此问题后迅速采取了行动：从 PyPI 中移除了恶意的 torchtriton 包，并用一个名为 pytorch-triton 的占位符包取代，以防止未来的类似攻击。他们还建议受影响的用户立即卸载恶意包和相关的 PyTorch nightly 组件，并清理 pip 缓存。

PyTorch 'torchtriton' 事件的关键启示是：

- **依赖管理的风险：** 它暴露了现代软件开发中普遍存在的依赖管理风险，尤其是在快速迭代的机器学习领域。公共包存储库是潜在的攻击入口。
- **攻击手法的有效性：** 依赖混淆和 typosquatting（仿冒名称）是非常有效的攻击手段，它们利用了开发者和工具链中可能存在的疏忽。
- **验证与流程的重要性：** 此事件强调了验证软件包来源、实施更安全的构建和部署流程以及进行常规安全审计的必要性。

4.3. 启示与影响

这两个案例研究共同揭示了几个重要的事实。首先，OWASP LLM Top 10 中列出的风险，如供应链漏洞 (LLM03) 和数据/模型投毒 (LLM04)，并非仅仅是理论上的可能性，而是已经被证明具有实际的可利用性，并产生了真实的影响。这验证了 OWASP 列表的现实意义，也说明了采取缓解措施的紧迫性。

其次，机器学习的供应链已成为一个关键的攻击界面。无论是模型共享中心（如 Hugging Face）还是软件包存储库（如 PyPI），都可能被用来分发恶意内容或利用信任关系。这表明，需要针对性地加强 ML 供应链的安全措施，例如推广更可靠的模型溯源技术、开发和使用模型/代码扫描工具（如 Protect AI 的 ModelScan），以及加强依赖项验证流程。

最后，这些攻击的检测极具挑战性。PoisonGPT 的“手术式”编辑成功规避了标准基准测试。PyTorch 事件中的恶意软件使用了 DNS 隧道进行数据外泄，可能绕过常规的网络监控。而 Typosquatting 则依赖于人类或自动化工具在识别名称时的微小错误。这些攻击手段的隐蔽性意味着，单一的防御措施往往不足够。组织需要部署多层次的防御策略，结合使用静态分析、行为监控、异常检测以及专门针对 LLM 和供应链安全的工具。

5. 安全构建：LLM 开发与防御框架及工具

面对日益严峻的安全挑战，开发者社区和安全行业正在积极构建和采用新的框架与工具，以支持更安全的 LLM 应用开发和部署。这里将介绍几个代表性的例子：LangChain 作为开发编排框架，Rebuff AI 和 Garak 作为防御与测试工具。

5.1. 开发编排框架：LangChain

LangChain 是一个广受欢迎的开源框架，旨在简化基于 LLM 的应用程序的开发过程。它提供了 Python 和 JavaScript 两种版本，其核心目标是通过提供模块化的构建块和抽象接口，让开发者能够

更容易地将 LLM 与其他计算资源或知识源（如数据库、API、文档库）结合起来，构建更强大、更具上下文感知能力的应用，例如聊天机器人、问答系统、内容摘要工具和复杂的智能代理 (Agents)。

LangChain 的关键特性包括：

- **标准化接口：** 为不同的 LLM、嵌入模型和向量数据库提供统一的调用方式，方便切换和实验。
- **模块化组件：** 提供一系列预置组件，如文档加载器 (Document Loaders)、文本分割器 (Text Splitters)、向量存储 (Vector Stores)、检索器 (Retrievers)、链 (Chains) 和代理 (Agents)，开发者可以将这些组件“链接”起来构建应用逻辑。
- **丰富的集成：** 支持与数百个第三方数据源、工具和平台集成，极大地扩展了 LLM 应用的能力范围。
- **LangChain Expression Language (LCEL)：** 一种声明式的语言，用于以更简洁、更灵活的方式组合 LangChain 组件，并天然支持流式处理、异步执行和并行化等生产环境所需特性。
- **LangSmith：** 一个配套的平台，用于 LLM 应用的可观测性、调试、测试和评估，帮助开发者从原型快速走向生产。
- **LangGraph：** 一个用于构建有状态、多步骤、可能涉及多个智能体协作的复杂应用的库，特别适用于需要更精细控制流程和长期记忆的场景。

LangChain 极大地降低了开发 LLM 应用的门槛，加速了创新和原型设计。然而，这种便利性也可能伴随着潜在的安全风险。LangChain 的核心功能在于连接 LLM 与外部世界——包括各种数据源、API 和工具。其庞大的集成库（官方提及超过 600 个集成）虽然功能强大，但也意味着更多的潜在攻击入口。

每一个集成点，如果处理不当，都可能成为安全漏洞的源头。例如，如果从外部数据源加载的数

据未经验证就直接传递给 LLM，可能导致间接提示注入 (LLM01)。如果 LLM 的输出（可能受提示注入影响）被用来调用外部工具或 API，而没有进行严格的过滤和权限控制，则可能导致不当输出处理 (LLM05) 或敏感信息泄露 (LLM02)。

LangChain 提供的抽象层虽然简化了开发，但也可能使得追踪和保护整个应用中的数据流变得更加复杂，凸显了使用 LangSmith 等可观测性工具的重要性。因此，在使用 LangChain 或类似框架时，开发者仍需保持安全意识，仔细审查数据流，并应用安全最佳实践。

5.2. 防御工具：Rebuff AI

Rebuff AI 是一个开源工具，专注于解决 OWASP LLM Top 10 中的一个核心风险：提示注入 (LLM01)。它是 Protect AI 公司贡献的几个开源 AI 安全工具之一。Rebuff 的目标是提供一个多层次的防御机制来检测和阻止提示注入攻击。其采用的技术据称包括：

1. **启发式过滤 (Heuristics)：** 在输入到达 LLM 之前，通过预定义的规则或模式来过滤掉已知的恶意提示。
2. **专用 LLM 分析 (Dedicated LLM Analysis)：** 使用另一个（通常是更小、更专注的）LLM 来分析用户输入的意图，判断其是否包含恶意指令。
3. **向量数据库比对 (Vector DB Comparison)：** 将输入提示的嵌入向量与一个存储已知攻击模式嵌入向量的数据库进行比较，以识别相似的攻击。
4. **金丝雀令牌检测 (Canary Token Detection)：** 在发送给 LLM 的提示中（通常是系统提示部分）插入一个秘密的、无意义的“金丝雀”词。然后检查 LLM 的响应是否包含了这个词。如果包含了，则表明 LLM 可能被注入，其内部指令或上下文被泄露。

此外，Rebuff 还宣称具备“自我强化”(self-hardening) 的能力，即能够从检测到的攻击中学习，并将新的攻击模式添加到向量数据库中，从而不断提高其防御效果。

尽管 Rebuff 提供了一种有针对性的防御手段，但需要认识到其局限性。提示注入是一个极其复杂和不断演变的问题，目前没有完美的解决方案。Rebuff 本身也承认可能存在误报（将良性提示识别为恶意）和漏报（未能识别恶意提示）的情况。同时，它主要关注提示注入，对于 OWASP LLM Top 10 中的其他风险（如数据投毒、不安全输出处理等）覆盖有限。

Rebuff AI 的存在说明了一个重要趋势：随着 LLM 特定漏洞的深入研究，专门用于解决这些独特威胁的安全工具正在涌现。通用安全解决方案可能不足以应对 LLM 带来的细微而复杂的挑战，因此需要这类有针对性的防御工具作为纵深防御体系的一部分。

5.3. 防御工具：Garak

Garak (*Generative AI Red-teaming & Assessment Kit*) 是由 NVIDIA 开发并开源的一款 LLM 漏洞扫描器和红队测试工具。它的目标是系统性地探测 LLM 或对话系统，以发现潜在的弱点和不期望的行为。Garak 被比作 LLM 领域的 nmap 或 Metasploit，旨在模拟攻击者的行为来评估模型的安全性。

Garak 的核心能力在于其全面的探测范围。它能够扫描 LLM 是否存在多种类型的漏洞，包括但不限于：

- **幻觉 (Hallucination)：** 生成虚假或无意义的信息。
- **数据泄露 (Data Leakage)：** 泄露训练数据或敏感信息。
- **提示注入 (Prompt Injection)：** 对恶意提示的易感性。
- **信息误传 (Misinformation)：** 生成或支持错误、误导性的信息。
- **毒性内容生成 (Toxicity Generation)：** 产生有害、冒犯性或不当内容。
- **越狱 (Jailbreaks)：** 绕过安全护栏和道德约束。

- **编码攻击 (Encoding Attacks)：** 通过文本编码方式进行提示注入。
- **恶意软件生成 (Malware Generation)：** 被诱导生成恶意代码。
- **跨站脚本 (XSS) 潜力：** 输出可能被利用于 XSS 攻击。

Garak 通过一个包含“探测器 (Probes)”、“检测器 (Detectors)”和“生成器 (Generators)”的框架来实现这些扫描。探测器负责向 LLM 发送各种类型的测试输入；生成器代表被测试的 LLM 接口；检测器则负责评估 LLM 的响应是否表现出不期望的行为。Garak 支持多种 LLM 平台和模型接口，包括 Hugging Face (本地和 API)、OpenAI、Cohere、Ollama、Groq 等。

Garak 主要用于部署前的安全评估和测试，帮助安全研究人员、开发者和 AI 伦理专家在模型上线前识别潜在风险。它是一种主动发现问题的工具，补充了运行时防御机制。

Garak 的出现和发展体现了 LLM 安全领域向主动、对抗性测试（即“红队测试”）的转变。正如传统网络安全一样，仅仅依赖被动防御是不够的。通过模拟攻击者的视角和方法，主动探测模型的弱点，可以在漏洞被实际利用之前发现并修复它们。PoisonGPT 和 PyTorch 事件都显示了被动防御和标准测试的局限性。Garak 提供了一个系统化的框架，用于对 LLM 进行压力测试，覆盖了 OWASP Top 10 中的多项风险（如 LLM01 提示注入、LLM02 敏感信息泄露、LLM04 相关的模型行为问题等），这对于建立对 LLM 系统安全性的信心至关重要。

5.4. 其他相关工具

除了上述重点介绍的工具外，研究材料中还提到了其他一些有助于增强 LLM 生态系统安全的工具：

- **ModelScan:** 由 Protect AI 开源，用于扫描机器学习模型文件（如 Pickle、H5、SavedModel 格式）是否存在不安全代码或序列化攻击漏洞。这直接关系到缓解模型投毒 (LLM04) 和供应链漏洞 (LLM03) 的风险。

- **NB Defense:** 同样由 Protect AI 开源，这是一个用于扫描 Jupyter Notebooks 的安全工具，可以检测泄露的凭证、PII、许可证问题和代码漏洞。由于 Notebooks 是数据科学家进行模型实验和开发的常用环境，确保其安全对于保护整个 ML/AI 工作流至关重要，间接影响供应链安全 (LLM03)。

此外，不能忽视的是，许多传统的应用安全最佳实践对于保护 LLM 应用仍然至关重要，例如：实施安全编码标准、定期更新和修补系统、采用强大的身份验证机制、进行频繁的安全测试和审计，以及对开发和运维团队进行持续的安全教育。

5.5. LLM 安全工具比较

为了更清晰地展示 Rebuff AI、Garak 和 ModelScan 在 LLM 安全工具箱中的不同角色和侧重点，下表进行了简要比较：

表 2：部分 LLM 安全工具比较

工具 (Tool)	用途 (Purpose)	关键技术 (Key Techniques)	主要应对的 OWASP LLM 风险 (Primary OWASP Risks Addressed)	使用场景 (Use Case)	开源 (Open Source)
Rebuff AI	检测和防御提示注入攻击	启发式过滤、LLM 分析、向量数据库比对、金丝雀令牌、自我强化	LLM01: Prompt Injection	运行时防御/检测	

Garak	LLM 漏洞扫描与红队测试	多种探测器（覆盖多种漏洞类型）、检测器、支持多种 LLM 接口	LLM01, LLM02	部署前测试/评估	
ModelScan	扫描 ML 模型文件中的不安全代码	检测 Pickle、H5、SavedModel 等格式中的恶意代码或序列化漏洞	LLM03: Supply Chain, LLM04: Data and Model Poisoning	供应链安全/模型审查	

我们可以看到，表格突显了构建 LLM 安全需要一个多工具、多层次的方法。没有单一的工具可以解决所有问题。组织需要根据自身的应用场景、风险承受能力和技术栈，选择和组合使用不同的工具，以实现更全面的防护。

6. 建议与未来展望

确保大型语言模型 (LLM) 的安全是一项复杂且持续的任务。基于对当前风险格局、实际案例和可用工具的分析，提出以下建议，以帮助组织构建和维护更安全的 LLM 应用：

1. **采纳基于风险的方法 (Adopt a Risk-Based Approach):**
    - 利用如 OWASP Top 10 for LLM Applications 这样的框架来识别、评估和优先处理与特定应用场景和部署环境最相关的安全风险。并非所有风险对每个应用都具有同等的重要性，需要根据

业务影响和技术实现进行权衡。

2. **实施纵深防御策略 (Implement Defense-in-Depth):**
   - **安全开发实践:** 从源头做起，实施严格的输入验证和输出净化（应对 LLM01, LLM05），编写安全的代码，应用最小权限原则（特别是在 LLM 与外部系统交互时），并定期进行安全审计和测试。
   - **数据治理与安全:** 对用于训练、微调和检索增强生成 (RAG) 的数据进行严格的来源验证、质量检查、清洗和（必要的）脱敏处理。建立数据溯源机制，追踪数据沿袭（应对 LLM04）。
   - **强化供应链安全:** 仔细审查和管理所有第三方依赖项（软件库、预训练模型、数据集）。使用工具（如 ModelScan）扫描模型文件和依赖项是否存在已知漏洞或恶意代码。优先从可信来源获取组件，并确保构建和部署管道的安全（应对 LLM03）。
   - **主动安全测试:** 在部署前和部署后定期使用红队测试工具（如 Garak）主动探测模型的漏洞和弱点。模拟真实世界的攻击场景，以评估防御措施的有效性。
   - **运行时监控与防御:** 部署专门的运行时防御工具（如 Rebuff）来检测和阻止特定类型的攻击（如提示注入）。持续监控 LLM 的行为和性能，寻找异常模式。实施严格的网络出口控制，限制 LLM 对外部资源的访问（应对 LLM02）。
   - **人类监督与干预:** 对于高风险的应用场景或涉及关键决策的输出，应考虑引入人工审核环节。建立反馈机制，让用户可以报告不当或有害的输出。
3. **保持信息同步与持续适应 (Stay Informed and Adapt):**
   - LLM 技术和相关的安全威胁都在快速演变。组织需要建立机制，持续关注最新的安全研究、新发现的漏洞（例如关注 OWASP 项目的更新）、攻击技术以及新兴的防御工具和最佳实践。
   - 在开发团队和运维团队中培养强烈的安全意识文化至关重要。定期的安全培训和知识共享有

助于确保团队成员了解最新的威胁和缓解策略。

4. **关注未来趋势 (Future Trends):**
    - 预计未来对抗性机器学习 (Adversarial ML) 技术将更加成熟，可能出现更隐蔽、更有效的攻击方法。
    - 数据投毒技术可能会变得更加复杂和难以检测。
    - 随着能够自主执行任务的智能代理 (Agentic Systems)的发展，其安全挑战将变得更加突出。
    - 同时，防御技术也将不断进步，包括更鲁棒的模型架构、更有效的异常检测算法以及更智能的运行时保护机制。
    - 开源社区、学术界和产业界的持续研究与协作对于应对这些未来挑战至关重要。

总结思考:

保护 LLM 应用的安全并非一蹴而就，也不是单一技术或工具能够完全解决的问题。它需要一个整体的、适应性的、并且是协作性的方法，将先进的技术、健全的流程和具备安全意识的人员紧密结合起来。组织必须将安全视为 LLM 应用生命周期中不可或缺的一部分，从设计、开发到部署和运维，持续投入资源和关注，才能在利用 LLM 强大能力的同时，有效管理其伴随的风险。

7. 附录

7.1. 关键术语解释 (Glossary of Key Terms)

- **LLM (Large Language Model - 大型语言模型):** 一种基于深度学习技术构建，在海量文本数据上预训练的人工智能模型，能够理解和生成类似人类的文本。这些模型通常拥有数十亿以上参数，通过在海量数据上预训练，它们学习语言的结构和语义，能执行文本生成、翻译、问答和摘要等

任务。

- **Prompt Injection (提示注入):** 通过精心构造的输入（提示）来操纵 LLM，使其执行非预期的操作或生成不当内容。
- **Data Poisoning (数据投毒):** 恶意修改用于训练或微调 LLM 的数据，以引入漏洞、偏见或后门。
- **RAG (Retrieval-Augmented Generation - 检索增强生成):** 一种技术，允许 LLM 在生成响应之前从外部知识库检索相关信息，以提高准确性和相关性。
- **Dependency Confusion (依赖混淆):** 一种供应链攻击技术，利用包管理器处理公共和私有包源的方式，诱骗其下载恶意的同名包。
- **Typosquatting (名称仿冒/域名抢注):** 注册与合法包或域名非常相似的名称，以期用户或工具因拼写错误而下载或访问恶意版本。
- **Supply Chain Vulnerability (供应链漏洞):** 指 LLM 应用所依赖的第三方组件、库、模型或数据中存在的安全风险。
- **Red Teaming (红队测试):** 模拟攻击者的策略和技术，对系统进行安全评估以发现漏洞的过程。
- **OWASP (Open Web Application Security Project - 开放全球应用程序安全项目):** 一个致力于改善软件安全的非营利组织。

7.2. OWASP Top 10 for LLM Applications (2025 版) 完整列表

根据 OWASP 官方发布，2025 年列表如下：

- **LLM01:2025 Prompt Injection (提示注入)**
- **LLM02:2025 Sensitive Information Disclosure (敏感信息泄露)**

- LLM03:2025 Supply Chain (供应链)
- LLM04:2025 Data and Model Poisoning (数据和模型投毒)
- LLM05:2025 Improper Output Handling (不当输出处理)

一种基于深度学习技术构建的人工智能模型

- :2025 Excessive Agency（过度授权）
- LLM07:2025 System Prompt Leakage（系统提示泄露）
- LLM08:2025 Vector and Embedding Weaknesses（向量和嵌入漏洞）
- LLM09:2025 Misinformation（信息误导）
- LLM10:2025 Unbounded Consumption (无限资源消耗)

7.3 参考引用

- Quick Guide to OWASP Top 10 LLM: Threats, Examples & Prevention - Tigera

https://www.tigera.io/learn/guides/llm-security/owasp-top-10-llm/

- 11 LLM Security Tools - Granica AI https://granica.ai/blog/llm-security-tools-grc
- OWASP Top 10 for Large Language Model Applications

https://owasp.org/www-project-top-10-for-large-language-model-applications/

- OWASP Top 10: LLM & Generative AI Security Risks https://genai.owasp.org/
- OWASP/www-project-top-10-for-large-language-model-applications - GitHub

https://github.com/OWASP/www-project-top-10-for-large-language-model-applications

- OWASP Reveals Updated25 Top 10 Risks for LLMs, Announces New LLM Project Sponsorship Program and Inaugural Sponsors

https://genaisecurityproject.com/2024/11/17/owasp-reveals-updated-2025-top-10-risks-for-llms-announces-new-llm-project-sponsorship-program-and-inaugural-sponsors/

- owasp.org

https://owasp.org/www-project-top-10-for-large-language-model-applications/assets/PDF/OWASP-Top-10-for-LLMs-v2025.pdf

- Enhance Your AI Security with Rebuff: Protecting Against Prompt Injections

https://www.toolify.ai/ai-news/enhance-your-ai-security-with-rebuff-protecting-against-prompt-injections-2154432

- 2024 Volume 1 Free Resources for Hardening AI and ML - ISACA

https://www.isaca.org/resources/isaca-journal/issues/2024/volume-1/free-resources-for-hardening-ai-and-ml

- PyTorch Identifies Malicious Dependency in its Nightly Build - Bitdefender

https://www.bitdefender.com/en-us/blog/hotforsecurity/pytorch-identifies-malicious-dependency-in-its-nightly-build

- Breaking Down OWASP Top 10 for Web Apps, Mobile, API, K8s & LLMs - Oligo Security

https://www.oligo.security/academy/breaking-down-owasp-top-10-for-web-apps-mobile-api-k8s-and-llms

- Protect AI Open Sources Three Tools to Help Organizations Secure AI/ML Environments from Threats - Business Wire

https://www.businesswire.com/news/home/20231005973991/en/Protect-AI-Open-Sources-Three-Tools-to-Help-Organizations-Secure-AIML-Environments-from-Threats

- Medical large language models are vulnerable to data-poisoning attacks - PubMed

https://pubmed.ncbi.nlm.nih.gov/39779928/

- Exposing Vulnerabilities in Clinical LLMs Through Data Poisoning Attacks: Case Study in Breast Cancer - PMC

https://pmc.ncbi.nlm.nih.gov/articles/PMC10984073/

- PoisonGPT: How to poison LLM supply chainon Hugging Face

https://blog.mithrilsecurity.io/poisongpt-how-we-hid-a-lobotomized-llm-on-hugging-face-to-spread-fake-news/

- GPT-4o Guardrails Gone: Data Poisoning & Jailbreak-Tuning | FAR.AI https://far.ai/post/2024-10-poisoning/

- Introduction to Training Data Poisoning: A Beginner's Guide | Lakera – Protecting AI teams that disrupt the world.

https://www.lakera.ai/blog/training-data-poisoning

- Software Supply Chain Chronicles: Malicious Dependency hits PyTorch

https://blog.packagecloud.io/software-supply-chain-chronicles-malicious-dependency-hits-pytorch/

- Malicious PyTorch dependency 'torchtriton' on PyPI | Wiz Blog

https://www.wiz.io/blog/malicious-pytorch-dependency-torchtriton-on-pypi-everything-you-need-to-know

- Top 8 malicious attacks recently found on PyPI - Sonatype

https://www.sonatype.com/blog/top-8-malicious-attacks-recently-found-on-pypi

- Malware Monthly - December 2022 - Sonatype https://www.sonatype.com/blog/malware-monthly-december-2022

- PyPI Inundated by Malicious Typosquatting Campaign - Check Point Blog

https://blog.checkpoint.com/securing-the-cloud/pypi-inundated-by-malicious-typosquatting-campaign/

- A PyPI typosquatting campaign post-mortem https://blog.phylum.io/a-pypi-typosquatting-campaign-post-mortem/

- What is LangChain? - AWS https://aws.amazon.com/what-is/langchain/

- Introduction | LangChain https://python.langchain.com/docs/introduction/

- What Is LangChain? - IBM https://www.ibm.com/think/topics/langchain

- langchain-ai/langchain: Build context-aware reasoning applications - GitHub https://github.com/langchain-ai/langchain

- Introduction | 🦜 Langchain https://js.langchain.com/docs/introduction

- Introduction - LangChain https://python.langchain.com/v0.1/docs/get_started/introduction/

- The largest community building the future of LLM apps - LangChain https://www.langchain.com/langchain

- Rebuff: AI-Powered Self-Hardening Prompt Injection Detector ... https://deepgram.com/ai-apps/rebuff

- Rebuff AI Reviews in 2025 - SourceForge https://sourceforge.net/software/product/Rebuff-AI/

- NVIDIA/garak: the LLM vulnerability scanner - GitHub https://github.com/NVIDIA/garak

- Garak - Open Source LLM Vulnerability Scanner for AI Red-Teaming - GBHackers https://gbhackers.com/garak/

- Garak Red Teaming LLMs - RodrigTech - Cloud Security https://rodrigtech.com/garak-red-teaming-llms/

- garak: A Framework for Security Probing Large Language Models - GitHub

https://raw.githubusercontent.com/NVIDIA/garak/main/garak-paper.pdf

- 30 Best Tools for Red Teaming: Mitigating Bias, AI Vulnerabilities & More - Mindgard

https://mindgard.ai/blog/best-tools-for-red-teaming

- Interrogating Intelligence: Red Teaming LLMs with Garak - Trevor Carstensen Cybersecurity Portfolio

https://trevorcarstensencybersecurity.com/2025/03/17/interrogating-intelligence-red-teaming-llms-with-garak/

- Five Open Source LLM Security Tools - TAICO https://taico.ca/posts/5-llm-security-tools/