OWASP
Open Worldwide Application Security Project

OWASP 大语言模型应用程序十大风险

OWASP Top 10 for Large Language Model Applications

2023 V1.0

前言

《OWASP 大语言模型应用程序十大风险》项目旨在向开发人员、设计人员、架构师、管理人员和组织介绍部署和管理大语言模型（LLM）应用程序时的潜在安全风险。该项目主要基于漏洞的潜在影响，可利用性以及在实际应用程序中的普遍性，整理出包括提示注入、数据泄漏、沙盒不足和未经授权的代码执行等 LLM 应用程序中常见的 10 大最关键风险的列表。

本项目旨在提高对这些风险以及相关漏洞的认识，提供修正策略的建议，并最终改善 LLM 应用程序的安全状况。

项目组长：Steve Wilson

中文翻译人员：肖文棣、周乐坤

十大风险列表

这是一份对象为构建在大型语言模型 (LLM) 上的人工智能 (AI) 应用程序的重要漏洞类型的列表。

LLM01:2023	提示词注入 Prompt Injections	绕过过滤器或使用精心制作的提示操作 LLM，使模型忽略先前的指令或执行非计划的操作。
LLM02:2023	数据泄漏 Data Leakage	通过 LLM 的回复意外泄露敏感信息、专有算法或其他机密细节。
LLM03:2023	不完善的沙盒隔离 Inadequate Sandboxing	当 LLM 可以访问外部资源或敏感系统时，未能正确隔离 LLM，从而允许潜在的利用和未经授权的访问。
LLM04:2023	非授权代码执行 Unauthorized Code Execution	利用 LLM 通过自然语言提示在底层系统上执行恶意代码、命令或操作。
LLM05:2023	SSRF 漏洞 SSRF Vulnerabilities	利用 LLM 执行意外请求或访问受限制的资源，如内部服务、API 或数据存储。
LLM06:2023	过度依赖大语言模型生成的内容 Overreliance on LLM-generated Content	在没有人为监督的情况下过度依赖法 LLM 生成的内容可能会导致不良后果。
LLM07:2023	人工智能未充分对齐 Inadequate AI Alignment	未能确保 LLM 的目标和行为与预期用例保持一致，从而导致不良后果或漏洞。
LLM08:2023	访问控制不足 Controls Insufficient Access	未正确实现访问控制或身份验证，将允许未经授权的用户与 LLM 交互，并可能导致漏洞被利用。
LLM09:2023	错误处置不当 Improper Error Handling	暴露错误消息或调试信息，将导致敏感信息、系统详细信息或潜在攻击向量的泄露。
LLM10:2023	训练数据投毒 Training Data Poisoning	恶意操纵训练数据或微调程序，将漏洞或后门引入 LLM。

LM01:2023_ 提示词注入

描述

提示词注入包括绕过过滤器或者通过精心构造的提示词来操控大语言模型（LLM）使得该模型忽略先前的指令或者执行意外操作。这些漏洞导致数据泄漏、未经授权的访问或者其他安全漏洞等意想不到的后果。

常见提示词注入漏洞

- 精心构造能够操纵大语言模型以暴露敏感数据的提示词。
- 利用特定的语言模式或者词元来绕过过滤器或者限制。
- 利用大语言模型的分词或者编码机制的弱点。
- 利用误导性上下文来误导大语言模型执行意外操作。

如何防范

- 对用户提供的提示词进行严格的输入校验和净化。
- 使用上下文感知过滤器和输出编码来防止提示词操作。
- 定期更新和微调大语言模型以提高其对于恶意输入和边界用例的理解能力。
- 监视和记录大语言模型的交互以检测和分析潜在的提示词注入尝试。

攻击场景示例

场景 1

攻击者精心构造一个提示词发起请求，并让模型认为该请求是合法的，从而欺骗大语言模型包括用户凭证或者内部系统详细信息等敏感信息。

场景 2

恶意用户通过利用特定的语言模式、词元或者编码机制来绕过内容过滤器，从而允许该用户执行那些本应被阻止的操作。

LLM02:2023_ 数据泄漏

描述

当大语言模型通过响应恶意请求意外泄漏敏感信息、专有算法或者其他机密细节时，就会发生数据泄漏。这可能导致未经授权访问敏感数据、窃取知识产品、侵犯隐私或其他安全漏洞。

常见不完善沙盒隔离漏洞

- 在大语言模型的响应中的敏感数据过滤不完全或者不恰当。
- 在大语言模型的训练过程中过度拟合或者记忆敏感数据。
- 由于大语言模型的误解或者错误而意外泄漏机密数据。

如何防范

- 实行严格的输出过滤和上下文感知机制来防止大语言模型泄漏敏感数据。
- 在大语言模型的训练过程中使用差分隐私技术或者其他数据匿名化方法来降低过度拟合或者记忆风险。
- 定期审核和检查大语言模型的响应内容来确保不会无意中泄漏敏感信息。
- 监视和记录大语言模型的交互来检测和分析潜在的数据泄漏事件。

攻击场景示例

场景 1

用户无意中向大语言模型提了一个可能导致敏感信息泄漏的问题。大语言模型缺乏恰当的输出过滤，响应内容中包括了敏感数据而导致敏感数据泄漏。

场景 2

攻击者故意利用精心构造的提示词来探测大语言模型，试图从大语言模型的训练数据中提取记忆来的敏感信息。

通过了解和定义数据泄漏相关的风险，开发人员可以更好地保护自己的大语言模型和确保系统的安全性。

LLM03:2023_ 不完善的沙盒隔离

描述

不完善的沙盒隔离指的是当大语言模型访问外部资源或者敏感系统时，如果没有合适的隔离，就会导致大语言模型的潜在利用、未经授权的访问或者意外的操作。

常见不完善沙盒隔离漏洞

- 大语言模型的环境与其他关键系统或者数据存储的隔离不充分。
- 允许大语言模型在没有恰当限制的条件下访问敏感资源。
- 未能限制大语言模型的能力，例如允许其执行系统级的操作或者与其他进程交互。

如何防范

- 通过恰当的沙盒技术将大语言模型环境与其他关键系统和资源隔离。
- 限制大语言模型对敏感资源的访问，并将其能力限制在其预期目的所需的最低限制。
- 定期审核和检查大语言模型的环境和访问控制以确保系统保持恰当的隔离。
- 监视和记录大语言模型的交互以检测和分析潜在的沙盒问题。

攻击场景示例

场景 1

攻击者通过精心构造提示词，指示大语言模型提取和暴露敏感信息，利用大语言模型访问敏感数据库。

场景 2

允许大语言模型执行系统级操作，攻击者通过操纵大语言模型在底层系统中执行未经授权的命令。

通过了解和解决不完善沙盒隔离相关的风险，开发人员可以更好地保护自己大语言模型的实施并确保系统的安全和保障。

LLM04:2023_ 非授权代码执行

描述

当攻击者利用大语言模型通过自然语言提示词在底层系统上执行恶意代码、命令或操作时，就会发生未经授权的代码执行。

常见的未授权代码执行漏洞

- 未能清理或限制用户输入，允许攻击者制作触发未授权代码执行的提示词。
- 沙盒隔离不足或对大语言模型能力的限制不足，使其以意想不到的方式与底层系统交互。
- 无意中将系统级功能或接口暴露给大语言模型。

如何防范

- 实施严格的输入验证和清理流程，以防止大语言模型处理恶意提示词或意外提示词。
- 确保适当的沙盒隔离并限制大语言模型的能力以限制其与底层系统交互的能力。
- 定期审核和检查大语言模型的环境和访问控制，以确保不会发生未经授权的行为。
- 监控和记录大语言模型的交互以检测和分析潜在的未经授权的代码执行问题。

攻击场景示例

场景 1

攻击者制作一个提示词来指示大语言模型执行一个命令，在底层系统上启动反向 shell，授予攻击者未经授权的访问权限。

场景 2

无意中允许大语言模型与系统级 API 交互，攻击者操纵大语言模型在系统上执行未经授权的操作。

通过了解和解决与未经授权的代码执行相关的风险，开发人员可以更好地保护自己的大语言模型实施并确保系统的安全和保障。

LLM05:2023_SSRF 漏洞

描述

当攻击者利用大语言模型执行意外请求或访问受限资源（如内部服务、API 或数据存储）时，会出现服务器端请求伪造漏洞 (SSRF)。

常见 SSRF 漏洞

- 输入验证不充分，允许攻击者操纵大语言模型的提示词发起未授权请求。
- 不完善的沙盒隔离或资源限制不充分，使大语言模型能够访问受限资源或与内部服务交互。
- 网络或应用程序安全设置中的错误配置，将内部资源暴露给大语言模型。

如何防范

- 实行严格的输入验证和清理，以防止恶意提示词或意外提示词发起未经授权的请求。
- 实施适当的沙盒隔离并限制大语言模型对网络资源、内部服务和 API 的访问。
- 定期审核和检查网络和应用程序安全设置，以确保内部资源不会无意中暴露给大语言模型。
- 监控和记录大语言模型的交互以检测和分析潜在的 SSRF 漏洞。

攻击场景示例

场景 1

攻击者制作一个提示词，指示大语言模型向内部服务发出请求，绕过访问控制并获得对敏感信息的未授权访问。

场景 2

应用程序安全设置中的错误配置允许大语言模型与受限 API 交互，攻击者操纵大语言模型访问或修改敏感数据。

通过了解和解决与 SSRF 漏洞相关的风险，开发人员可以更好地保护自己的大语言模型实现并确保系统的安全和保障。

LLM06:2023_ 过度依赖大语言模型生成的内容

描述

过度依赖大语言模型生成的内容，会导致误导或不正确信息的传播，制定决策中人工输入的减少，批判性思维的减少。组织和用户可能会在未经验证的情况下信任大语言模型生成的内容，从而导致错误结果、沟通不畅乃至意料之外的后果。

常见的过度依赖大语言模型生成内容的漏洞

- 在未经验证的情况下，将大语言模型生成的内容采纳为事实。
- 假设大语言模型生成的内容没有偏见或错误信息。
- 无需人工输入或监督，而仅依赖大语言模型生成的内容进行关键决策。

如何防范

- 鼓励用户在做出决策或采纳为事实真相之前，验证大语言模型生成的内容并咨询其他来源。
- 实施人工监督和审查流程，以确保大语言模型生成的内容准确、恰当且公正。
- 清楚地向用户传达：大语言模型生成的内容是机器生成的，可能不完全可靠或准确。
- 培训用户和利益相关者，使其认识到大语言模型生成内容的局限性，并以适当的怀疑态度对待它。
- 将大语言模型生成的内容，作为人类专业知识和输入的补充，而不是替代。

攻击场景示例

场景 1

某新闻机构使用大语言模型，来生成各类主题的文章。大语言模型生成的文章包含了未经验证而发布的虚假信息。读者相信了这篇文章，从而导致错误信息的进一步传播。

场景 2

某公司依赖大语言模型来生成财务报告和分析。大语言模型生成的一份报告包含了错误的财务数据，而公司正是使用这份报告做出了关键的投资决策。由于依赖大语言模型生成的不准确内容，这次决策导致了重大的财务损失。

LLM07:2023_ 人工智能未充分对齐

描述

当大语言模型的目标和行为与预期用例不一致时，就会出现人工智能未充分对齐的现象，从而导致非预期的后果或漏洞。

常见的人工智能对齐漏洞

- 未明确定义的目标，导致大语言模型优先考虑非预期的或有害的行为。
- 错误对齐的奖励函数或训练数据，导致非预期的模型行为。
- 在各种上下文和场景中，对大语言模型行为的测试和验证不足。

如何防范

- 在设计和开发过程中，明确定义大语言模型的目标和预期行为。
- 确保奖励函数和训练数据与期望的结果一致，并且不会激励非预期的或有害的行为。
- 定期测试和验证大语言模型在各种场景、输入和上下文中的行为，以识别和解决 AI 对齐问题。
- 实施监测和反馈机制，持续评估大语言模型的性能和对齐问题，并根据需要更新模型来改进 AI 对齐。

攻击场景示例

场景 1

经过训练以优化用户参与度的某大语言模型，无意中优先推送有争议的或极端的内容，导致了错误信息或有害内容的广泛传播。

场景 2

设计用于协助系统管理任务的大语言模型未充分对齐，导致其执行有害命令或优先考虑降低系统性能或安全性的操作。

通过关注人工智能对齐问题，并确保大语言模型的目标和行为与预期用例一致，开发人员可以降低大语言模型实施过程中产生非预期结果和漏洞的风险。

LLM08:2023_ 访问控制不足

描述

当访问控制或身份验证机制未正确实施时，会出现访问控制不足的情况，从而允许未经授权的用户与大语言模型进行交互，并可能对漏洞进行利用。

常见的访问控制漏洞

- 未能强制执行访问大语言模型的严格的身份验证要求。
- 基于角色的访问控制（RBAC）实现上存在不足，允许用户执行超出其预期权限的操作。
- 未能为大语言模型生成的内容和操作提供适当的访问控制。

如何防范

- 实施强身份验证机制，如多因素身份验证，以确保只有授权的用户才能访问大语言模型。
- 使用基于角色的访问控制（RBAC），根据用户的角色和职责，定义和强制执行用户的权限。
- 对大语言模型生成的内容和操作，实施适当的访问控制，以防止未经授权的访问或操作。
- 根据需要定期审计和更新访问控制，以维持安全性并防止未经授权的访问行为。

攻击场景示例

场景 1

由于身份验证机制薄弱，某攻击者获得了对大语言模型的非授权访问，从而可以利用漏洞或操纵系统。

场景 2

由于 RBAC 实现的不足，有限权限的某用户可以执行超出其预期范围的操作，这可能会产生危害或对系统造成损害。

通过正确地实施访问控制和身份验证机制，开发人员可以防止未经授权的用户与大语言模型交互，并降低漏洞被利用的风险。

LLM09:2023_ 错误处置不当

描述

错误处置不当是指暴露的错误消息或调试信息，可能向攻击者泄露敏感信息、系统详细信息或潜在攻击向量。

常见的错误处置不当漏洞

- 通过错误消息暴露敏感信息或系统详细信息。
- 泄露可帮助攻击者识别潜在漏洞或攻击向量的调试信息。
- 未能正确处置错误，可能导致意外的行为或系统崩溃。

如何防范

- 实现适当的错误处置机制，以确保正确地捕获、记录和处理错误。
- 确保错误消息和调试信息不会泄露敏感信息或系统详细信息。考虑为用户使用通用错误消息，而为开发人员和管理员记录详细的错误信息。
- 定期审查错误日志，并采取必要措施修复已发现的问题，提高系统稳定性。

攻击场景示例

场景 1

某攻击者利用大语言模型的错误消息，来收集敏感信息或系统详细信息，便于发起有针对性的攻击或利用已知的漏洞。

场景 2

某开发人员不小心将调试信息暴露在生产系统中，使攻击者能够识别系统中潜在的攻击向量或漏洞。

通过实施适当的错误处置机制并确保错误消息不会泄露敏感信息，开发人员可以降低攻击者利用大语言模型漏洞的风险，并提升系统稳定性。

LLM010:2023_ 训练数据投毒

描述

训练数据投毒是指攻击者操纵大语言模型的训练数据或微调程序，引入可能危及模型安全性、有效性或伦理行为的漏洞、后门或偏见。

常见的训练数据投毒漏洞

- 通过恶意操纵训练数据，将后门或漏洞引入到大语言模型。
- 向大语言模型注入偏见，导致其产生有偏见或不恰当的响应。
- 利用微调过程，来危害大语言模型的安全性或有效性。

如何防范

- 通过从可信来源获取训练数据并验证其质量，确保训练数据的完整性。
- 实施健壮的数据净化和预处理技术，以消除训练数据中潜在的漏洞或偏见。
- 定期审查和审计大语言模型的训练数据和微调程序，以检测潜在的问题或恶意的操作。
- 利用监控和告警机制，来检测大语言模型中的异常行为或性能问题，这很可能预示着训练数据被投毒。

攻击场景示例

场景 1

某攻击者侵入训练数据管道并注入恶意数据，导致大语言模型产生有害的或不恰当的响应。

场景 2

某恶意的内部人员攻击了微调过程，在大语言模型中引入了漏洞或后门，这些漏洞或后门可以在后续阶段被利用。

通过确保训练数据的完整性，实施健壮的数据净化技术，以及定期审计大语言模型的训练和微调过程，开发人员可以将训练数据被投毒的风险降至最低，并保护其大语言模型免受潜在漏洞的危害。

获取更多资讯