说明书摘要

本发明提供基于大模型自身反思机制的即插式安全增强方法与系统，包括：步骤S1，接收用户输入需求，首先基于规则匹配使用正则表达式和静态分析工具（如Semgrep）快速识别敏感信息（如API密钥、密码等），生成初步代码并提取潜在风险；步骤S2，启动安全检测模块进行反思验证，识别潜在的安全问题，并通过自生成的安全知识库对安全问题进行标记；步骤S3，调用修正模块进行代码优化，动态检索安全知识库进行安全问题的匹配作为参考内容，进行自动修正和代码优化，并返回提取潜在风险和进行反思验证；步骤S4，输出最终代码，生成安全分析报告，将积累的安全知识进行总结，更新和拓展所述安全知识库。本申请能够在代码生成过程的每一个阶段，通过自身反思机制进行自动修正、代码优化及反思验证，确保每一次的输出结果均符合安全要求，能够满足不同模型之间通用性需求。



摘要附图

权利要求书

1、一种基于大模型自身反思机制的即插式安全增强方法，其特征在于，包括以下步骤：

步骤S1，实时接收用户的输入需求，通过大语言模型生成初步代码，并通过大语言模型提取初步代码中的潜在风险，所述潜在风险包括未加密的密码信息、敏感数据以及不符合安全规则或编码规范的操作，所述敏感数据包括API密钥和私密信息；

步骤S2，启动安全检测模块进行反思验证，识别潜在的安全问题，并通过大语言模型自生成的安全知识库对安全问题进行标记；所述安全问题包括SQL注入、XSS攻击以及缓冲区溢出；

步骤S3，调用修正模块进行代码优化，基于RAG检索增强生成模型动态检索所述安全知识库，进行安全问题与安全知识库之间的匹配，在匹配后根据所述安全知识库进行自动修正和代码优化，并返回提取潜在风险和进行反思验证；

步骤S4，输出最终代码，生成安全分析报告，并将反思验证过程中所积累的安全知识进行总结，更新和拓展所述安全知识库；所述安全分析报告的内容包括安全问题、优化代码以及评分。

2、根据权利要求1所述的基于大模型自身反思机制的即插式安全增强方法，其特征在于，所述步骤S1包括以下子步骤：

步骤S101，通过大语言模型对用户的输入需求进行解析，根据自然语言描述生成初步代码；

步骤S102，通过大语言模型对每一次生成的初步代码进行分析，提取与安全相关的知识，获取初步代码中的潜在风险；与安全相关的知识包括安全漏洞、攻击方式、安全规则和编码规范。

3、根据权利要求2所述的基于大模型自身反思机制的即插式安全增强方法，其特征在于，所述步骤S102中，针对与安全相关的知识，通过大语言模型识别出与代码模式对应的安全漏洞，识别出潜在的攻击方式，并将相应的安全规则和编码规范同步存入至所述安全知识库中。

4、根据权利要求1至3任意一项所述的基于大模型自身反思机制的即插式安全增强方法，其特征在于，所述步骤S2包括以下子步骤：

步骤S201，启动安全检测模块，通过静态分析工具对初步代码及其潜在风险进行分析，识别潜在的安全问题；

步骤S202，根据安全知识库中的安全知识，对潜在的安全问题进行分类标记，并结合领域知识以及最佳实践，提供针对性的修复建议或替代方案。

5、根据权利要求4所述的基于大模型自身反思机制的即插式安全增强方法，其特征在于，所述步骤S202中，在生成涉及数据库交互的代码时，大语言模型通过参数化查询或ORM框架来规避SQL注入这一安全问题；在生成Web应用代码时，大语言模型将提醒对用户的输入需求进行验证和转义，以避免出现XSS攻击这一安全问题；通过大语言模型根据任务场景所对应的领域，结合领域知识动态调整安全策略，确保生成的内容符合最新的安全规则、编码规范和行业标准。

6、根据权利要求1至3任意一项所述的基于大模型自身反思机制的即插式安全增强方法，其特征在于，所述步骤S3包括以下子步骤：

步骤S301，调用修正模块进行代码优化，基于RAG检索增强生成模型动态检索所述安全知识库，进行安全问题与安全知识库之间的匹配，计算安全问题与安全知识库之中各知识条目的向量相似度，获取与安全问题的向量相似度大于预设相似度阈值的知识条目作为匹配的知识条目；

步骤S302，将匹配到的知识条目作为上下文信息，输入至所述修正模块，生成针对安全问题的代码修正建议；

步骤S303，根据生成的代码修正建议，对代码进行自动优化；

步骤S304，返回步骤S2，对每一次生成后的内容进行反思验证，若生成的代码包含不安全的函数调用或硬编码密码，则自动进行替换或删除，以确保生成内容符合安全标准；若生成的代码存在未加密的密码信息，则通过自身反思机制自动加密密码或者删除敏感信息；直到进行反思验证后的内容通过安全评估。

7、根据权利要求6所述的基于大模型自身反思机制的即插式安全增强方法，其特征在于，所述步骤S303中，在进行自动优化的过程中，还根据用户的反馈进行调整，如果用户对生成内容中的安全性提出异议或发现新的安全隐患，通过大语言模型再次对内容进行反思验证，以确保输出结果符合安全要求。

8、根据权利要求6所述的基于大模型自身反思机制的即插式安全增强方法，其特征在于，所述步骤S303中，对代码进行自动优化之后，还针对优化后的代码自动添加校验和权限控制。

9、根据权利要求1至3任意一项所述的基于大模型自身反思机制的即插式安全增强方法，其特征在于，所述步骤S4包括以下子步骤：

步骤S401，在完成代码优化和反思验证之后，输出最终代码；

步骤S402，列出在代码优化和反思验证过程中发现的安全问题，展示经过优化后的代码片段，标识出优化前后的对比，并根据预设的评分标准对代码优化进行评分，以生成安全分析报告；

步骤S403，将每一次代码优化和反思验证过程中所积累的安全知识进行总结，将安全问题、优化方案以及最佳实践进行关联，对所述安全知识库进行更新和拓展，通过大语言模型实现并维护自身反思机制。

10、一种基于大模型自身反思机制的即插式安全增强系统，其特征在于，采用了如权利要求1至9任意一项所述的基于大模型自身反思机制的即插式安全增强方法，并包括：

代码生成模块，实时接收用户的输入需求，通过大语言模型生成初步代码，并通过大语言模型提取初步代码中的潜在风险，所述潜在风险包括未加密的密码信息、敏感数据以及不符合安全规则或编码规范的操作，所述敏感数据包括API密钥和私密信息；

反思验证模块，启动安全检测模块进行反思验证，识别潜在的安全问题，并通过大语言模型自生成的安全知识库对安全问题进行标记；

安全优化模块，调用修正模块进行代码优化，基于RAG检索增强生成模型动态检索所述安全知识库，进行安全问题与安全知识库之间的匹配，在匹配后根据所述安全知识库进行自动修正和代码优化，并返回提取潜在风险和进行反思验证；

输出模块，输出最终代码，生成安全分析报告，并将反思验证过程中所积累的安全知识进行总结，更新和拓展所述安全知识库。

**基于大模型自身反思机制的即插式安全增强方法与系统**

说明书

**技术领域**

本发明涉及一种安全增强的技术方案，尤其涉及一种基于大模型自身反思机制的即插式安全增强方法，还进一步涉及采用了该基于大模型自身反思机制的即插式安全增强方法的即插式安全增强系统。

**背景技术**

随着人工智能技术的不断发展，大型语言模型（Large Language Models, LLMs），如ChatGPT、Claude等，已广泛应用于推理、编程和科学研究等多个领域，并在许多实际应用中取得了显著成果。这些大语言模型不仅能够生成高质量的自然语言文本，还能辅助编程，提供代码生成、调试和优化等服务。在编程领域，LLMs被用于自动生成代码、优化算法和辅助开发人员提高编程效率。然而，尽管这些技术展现了巨大的潜力，仍然存在一些重要的安全隐患和问题，特别是在生成的代码中可能隐藏着严重的安全漏洞。

在传统的软件开发过程中，安全问题通常是在代码编写完成后，通过静态分析、动态测试等手段进行检测和修复。然而，当前的LLMs技术并未充分考虑到代码的安全性，尤其是在生成复杂程序时，这些模型生成的代码往往缺乏必要的安全审查。例如，某些模型可能会生成包含SQL注入漏洞的代码，导致攻击者能够通过恶意输入操控数据库。又如，生成的代码中可能存在未验证的用户输入，进而引发跨站脚本攻击（XSS）等安全漏洞。2024年11月，一位用户通过ChatGPT生成的代码搭建了一个辅助交易机器人，但是，由于模型生成的代码调用了一个恶意的API，并直接将用户的私钥传递给该API，导致用户钱包被盗取了2.5k美元。这一事件暴露了在生成过程中未审查和过滤恶意代码可能会带来的潜在安全威胁。

更严重的是，现有的LLMs模型在生成代码时并不会主动评估其安全性，导致生成的代码在实际运行时可能引发无法预测的安全漏洞。以SQL注入为例，LLMs可能生成一个包含数据库查询的代码片段，但如果未正确处理用户输入，恶意用户通过输入SQL语句可能导致数据库被攻击和数据泄露。此外，随着越来越多的企业和组织将LLMs应用于实际工作流程，那么，如何确保大语言模型在生成代码和其他内容时不包含潜在的安全风险和漏洞，已经成为一个急需解决的技术问题。

而现有的安全检测手段大多集中在代码生成之后的审查环节，例如使用静态代码分析工具和Fuzz测试等方法，这种现有的做法方法往往存在较高的成本和较长的时间延迟。在快速发展的开发环境中，这种事后进行修复的方法并不能有效地应对大规模自动化编程场景中的安全需求。因此，如何在生成过程中实现对代码的实时安全评估，成为了提升LLMs应用安全性的关键技术挑战之一。

另外，大多数现有方法主要集中在对特定模型的微调和安全对齐，或者侧重于输入输出端的内容过滤机制。这些现有的方案虽然在一定程度上也能解决部分安全问题，但依然存在一些限制和不足。例如，微调和安全对齐的方法需要大量的标注数据和训练资源，而输入输出端的过滤机制则依赖于规则和模板，难以适应复杂多变的生成任务。更为关键的是，这些方法通常是特定于某种模型或任务的，无法在不同模型之间通用，且存在一定的局限性，无法满足大规模、实时以及自动化生成的安全要求。

**发明内容**

本发明所要解决的技术问题是需要提供一种基于大模型自身反思机制的即插式安全增强方法，旨在能够在代码生成过程的每一个阶段，通过自身反思机制进行自动修正、代码优化以及反思验证；在此基础上，还能够在此过程自生成和扩展安全知识库，与自身反思机制互相支持和促进，以满足不同模型之间通用性需求，为满足大规模、实时以及自动化生成的安全要求提供了更好的基础。进一步的，还提供采用了该基于大模型自身反思机制的即插式安全增强方法的即插式安全增强系统。

对此，本发明提供一种基于大模型自身反思机制的即插式安全增强方法，包括以下步骤：

步骤S1，实时接收用户的输入需求，通过大语言模型生成初步代码，并通过大语言模型提取初步代码中的潜在风险，所述潜在风险包括未加密的密码信息、敏感数据以及不符合安全规则或编码规范的操作，所述敏感数据包括API密钥和私密信息；

步骤S2，启动安全检测模块进行反思验证，识别潜在的安全问题，并通过大语言模型自生成的安全知识库对安全问题进行标记；所述安全问题包括SQL注入、XSS攻击以及缓冲区溢出；

步骤S3，调用修正模块进行代码优化，基于RAG检索增强生成模型动态检索所述安全知识库，进行安全问题与安全知识库之间的匹配，在匹配后根据所述安全知识库进行自动修正和代码优化，并返回提取潜在风险和进行反思验证；

步骤S4，输出最终代码，生成安全分析报告，并将反思验证过程中所积累的安全知识进行总结，更新和拓展所述安全知识库；所述安全分析报告的内容包括安全问题、优化代码以及评分。

本发明的进一步改进在于，所述步骤S1包括以下子步骤：

步骤S101，通过大语言模型对用户的输入需求进行解析，根据自然语言描述生成初步代码；

步骤S102，通过大语言模型对每一次生成的初步代码进行分析，提取与安全相关的知识，获取初步代码中的潜在风险；与安全相关的知识包括安全漏洞、攻击方式、安全规则和编码规范。

本发明的进一步改进在于，所述步骤S102中，针对与安全相关的知识，通过大语言模型识别出与代码模式对应的安全漏洞，识别出潜在的攻击方式，并将相应的安全规则和编码规范同步存入至所述安全知识库中。

本发明的进一步改进在于，所述步骤S2包括以下子步骤：

步骤S201，启动安全检测模块，通过静态分析工具对初步代码及其潜在风险进行分析，识别潜在的安全问题；

步骤S202，根据安全知识库中的安全知识，对潜在的安全问题进行分类标记，并结合领域知识以及最佳实践，提供针对性的修复建议或替代方案。

本发明的进一步改进在于，所述步骤S202中，在生成涉及数据库交互的代码时，大语言模型通过参数化查询或ORM框架来规避SQL注入这一安全问题；在生成Web应用代码时，大语言模型将提醒对用户的输入需求进行验证和转义，以避免出现XSS攻击这一安全问题；通过大语言模型根据任务场景所对应的领域，结合领域知识动态调整安全策略，确保生成的内容符合最新的安全规则、编码规范和行业标准。

本发明的进一步改进在于，所述步骤S3包括以下子步骤：

步骤S301，调用修正模块进行代码优化，基于RAG检索增强生成模型动态检索所述安全知识库，进行安全问题与安全知识库之间的匹配，计算安全问题与安全知识库之中各知识条目的向量相似度，获取与安全问题的向量相似度大于预设相似度阈值的知识条目作为匹配的知识条目；

步骤S302，将匹配到的知识条目作为上下文信息，输入至所述修正模块，生成针对安全问题的代码修正建议；

步骤S303，根据生成的代码修正建议，对代码进行自动优化；

步骤S304，返回步骤S2，对每一次生成后的内容进行反思验证，若生成的代码包含不安全的函数调用或硬编码密码，则自动进行替换或删除，以确保生成内容符合安全标准；若生成的代码存在未加密的密码信息，则通过自身反思机制自动加密密码或者删除敏感信息；直到进行反思验证后的内容通过安全评估。

本发明的进一步改进在于，所述步骤S303中，在进行自动优化的过程中，还根据用户的反馈进行调整，如果用户对生成内容中的安全性提出异议或发现新的安全隐患，通过大语言模型再次对内容进行反思验证，以确保输出结果符合安全要求。

本发明的进一步改进在于，所述步骤S303中，对代码进行自动优化之后，还针对优化后的代码自动添加校验和权限控制。

本发明的进一步改进在于，所述步骤S4包括以下子步骤：

步骤S401，在完成代码优化和反思验证之后，输出最终代码；

步骤S402，列出在代码优化和反思验证过程中发现的安全问题，展示经过优化后的代码片段，标识出优化前后的对比，并根据预设的评分标准对代码优化进行评分，以生成安全分析报告；

步骤S403，将每一次代码优化和反思验证过程中所积累的安全知识进行总结，将安全问题、优化方案以及最佳实践进行关联，对所述安全知识库进行更新和拓展，通过大语言模型实现并维护自身反思机制。

本发明还提供一种基于大模型自身反思机制的即插式安全增强系统，采用了如上所述的基于大模型自身反思机制的即插式安全增强方法，并包括：

代码生成模块，实时接收用户的输入需求，通过大语言模型生成初步代码，并通过大语言模型提取初步代码中的潜在风险，所述潜在风险包括未加密的密码信息、敏感数据以及不符合安全规则或编码规范的操作，所述敏感数据包括API密钥和私密信息；

反思验证模块，启动安全检测模块进行反思验证，识别潜在的安全问题，并通过大语言模型自生成的安全知识库对安全问题进行标记；

安全优化模块，调用修正模块进行代码优化，基于RAG检索增强生成模型动态检索所述安全知识库，进行安全问题与安全知识库之间的匹配，在匹配后根据所述安全知识库进行自动修正和代码优化，并返回提取潜在风险和进行反思验证；

输出模块，输出最终代码，生成安全分析报告，并将反思验证过程中所积累的安全知识进行总结，更新和拓展所述安全知识库

与现有技术相比，本发明的有益效果在于：在实时接收用户的输入需求之后，先通过大语言模型生成初步代码，并通过大语言模型提取初步代码中的潜在风险；然后，启动安全检测模块进行反思验证，识别潜在的安全问题，并通过大语言模型自生成的安全知识库对安全问题进行标记；接着，调用修正模块进行代码优化，基于RAG检索增强生成模型动态检索所述安全知识库，进行安全问题与安全知识库之间的匹配，在匹配后根据所述安全知识库进行自动修正和代码优化，在自动修正和优化之后返回进行反思验证，直到通过安全评估；最后，输出最终代码，生成安全分析报告，并将反思验证过程中所积累的安全知识进行总结，更新和拓展所述安全知识库。

因此，本申请能够在代码生成过程的每一个阶段，通过自身反思机制进行自动修正、代码优化以及反思验证，而不是像现有技术在生成任务/输出代码之后，才进行简单的错误修正，以确保每一次的输出结果均符合安全要求；在此基础上，还能够在此过程自生成和扩展安全知识库，使得自生成的安全知识库与自身反思机制形成互相支持和促进的关系，无需引用外部的知识库，也不再局限于某种特定的模型或任务，可以很好地满足不同模型之间通用性需求，为满足大规模、实时以及自动化生成的安全要求提供了更好的基础。

**附图说明**

图1是本发明一种实施例的工作流程示意图；

图2是本发明一种实施例的自身反思机制的原理示意框图。

**具体实施方式**

在本发明的描述中，如果涉及到“若干”，其含义是一个以上；如果涉及到 “多个”，其含义是两个以上；如果涉及到“大于”、“小于”、“超过”，均应理解为不包括本数；如果涉及到“以上”、“以下”、“以内”，均应理解为包括本数。如果涉及到“第一”、“第二”等，应当理解为仅用于相同或是相似技术特征名称的区分，而不能理解为暗示/指明技术特征的相对重要性，不能理解为暗示/指明技术特征的数量，也不能理解为暗示/指明技术特征的先后关系。

在详细描述本发明的较优的实施例之前，先对相关的技术进行说明。其中，涉及的缩略语和关键术语定义包括：LLM，指的是Large Language Model，即大型语言模型，是一种基于深度学习技术的人工智能模型，具有规模庞大的参数量以及强大的自然语言处理能力。Transformer，由论文《Attention is All You Need》提出的基于Self-Attention机制的机器学习模型。Self-reflection，自我反思，是指大型语言模型具有的一种评估并改善自身生成的输出的能力，旨在提升模型可靠性。

现有的安全检测手段大多集中在代码输出之后的审查环节，这种事后修复的方法不能有效地应对大规模自动化编程场景中的安全需求。

针对上述问题，本发明旨在提出了一种基于大模型自身反思机制的即插式安全增强方法与系统，大模型指的是大语言模型，通过在代码生成过程中的每一个阶段，引入安全性自检机制，实时监控和修复生成代码中的潜在安全问题。具体而言，该方法通过引入反思机制，使得LLMs在生成代码时能够根据预设的安全规则，对生成的代码进行自我验证和修正。在生成过程中，本发明的模型不仅会评估代码的功能性，还会评估其安全性，及时发现并修复潜在的安全漏洞，从而减少后期修复的时间和成本。

例如，当用户请求生成一段包含数据库操作的代码时，反思机制会检查代码中是否存在潜在的SQL注入风险。如果检测到潜在的漏洞，模型将自动调整代码，使用参数化查询等安全手段替代不安全的字符串拼接方式，避免SQL注入攻击。又如，当生成包含用户输入处理的代码时，反思机制会检查是否对用户输入进行了严格的验证和消毒，以防止跨站脚本攻击（XSS）等安全问题的发生。

此外，自身反思机制还可以针对模型生成的代码进行自我优化。例如，在生成涉及敏感信息（如API密钥、私钥等）的代码时，模型会自动检测是否存在将敏感信息暴露的风险，并提示用户进行加密或采用环境变量存储等更安全的方式。在此过程中，反思机制不仅提高了生成代码的安全性，还增强了代码的鲁棒性和稳定性。

在此基础上，为了进一步提高生成过程中的安全性，本发明还提出了一种结合安全知识库的安全知识增强方法。通过引入安全知识库，LLMs可以在生成代码时实时调用安全知识，以进一步加强生成代码的安全性。例如，在生成涉及加密操作的代码时，模型可以参考安全知识库中的最佳实践，确保生成的代码符合当前的加密标准，避免常见的加密漏洞。

需要说明的是，如果使用外部知识库，也可能面临噪声和不相关信息的引入问题。为了解决这一技术问题，本发明提出了知识过滤和浓缩方法，通过对外部的安全知识进行过滤和精炼，已形成自生成的安全知识库，确保只将最相关的安全信息引入至生成过程，从而避免引导模型产生错误的或不相关的安全建议等问题。

本发明通过引入自身反思机制与安全知识增强，能够在代码生成过程的每一个结算，实现实时的安全审查和修复，从而降低代码中的安全风险，提升代码生成的可靠性和鲁棒性。这一创新的技术方案，不仅能够有效减少后期安全漏洞修复的成本，还能提升自动化编程的整体安全性，为大语言模型在更多实际应用中的推广和应用提供了技术保障。

在与本发明相关的一种技术方案之中，Rui Zhang等人提出的两方博弈对齐框架在现有方法的基础上进行了显著的扩展和创新。传统的对齐方法，如基于人类反馈的强化学习（RLHF），虽然在实践中被广泛应用，但其局限性在于难以全面覆盖所有任务类型，并且容易因过度依赖预收集的提示而导致模型性能饱和。偏好学习方法和直接从偏好数据中学习的方法试图减少对人类反馈的依赖，但在实际应用中仍面临标注数据质量和动态调整能力的挑战。

相比之下，这种技术方案所提出的两方博弈框架通过引入对抗防御博弈机制，提供了一个更为动态和自适应的对齐环境。该方法的核心在于通过迭代交互来识别和利用潜在弱点，从而在多个方面超越了传统方法。首先，通过对抗方和防御方的互动，模型能够在不断变化的环境中持续学习和改进，显著提高了泛化能力。其次，引入多样性约束确保生成的攻击提示多样化，这不仅增强了模型的鲁棒性，还促进了更广泛的攻击防御策略的发展。

在理论上，这种技术方案通过证明收敛到纳什均衡，提供了算法的稳定性保障。这一理论基础使得该方法在实践中更具可靠性和可预测性。然而，尽管这种技术方案在提高模型安全性和泛化能力方面具有显著优势，但也面临着一些挑战，包括：计算复杂度高，需要多次迭代优化，依赖高性能计算资源，这对硬件设施提出了较高的要求；实现难度较大，依赖高质量的数据集和标注，这在数据获取和处理方面增加了复杂性。

在与本发明相关的另一种技术方案之中，Chen等人提出了SecAlign，这是一种基于偏好优化技术的防御方法，旨在保护大型语言模型（LLMs）免受提示注入攻击。SecAlign通过构建一个包含提示注入输入、安全输出（响应合法指令的输出）和不安全输出（响应注入的输出）的偏好数据集，并在该数据集上进行偏好优化，教导LLM优先选择安全输出而非不安全输出。这种方法首次将各种提示注入的成功率降低到约0%，即使面对比训练期间所见攻击更为复杂的攻击，也显示出良好的泛化能力。此外，经过防御训练的模型在实用性方面与防御训练前相当，具有相似的效用。

SecAlign的主要优点在于其简单性、对模型效用的保持以及对未知攻击的强大安全性，甚至能够抵御基于优化的攻击。通过偏好优化，SecAlign还建立了LLM安全与对齐之间的联系，这两个主题之前是分开研究的。

然而，SecAlign也依然存在一些局限性。首先，它仅适用于指令部分和数据部分有明确分隔（例如通过分隔符）的场景。其次，SecAlign构建的期望输出与其他基于微调的防御方法共享一个缺点：期望输出忽略了数据中的注入指令，而不是将其作为数据的一部分进行处理。这可能导致LLM忽略数据中的一些命令句（这些命令句可能并非注入，而应作为数据处理，例如需要翻译的命令句）。解决此问题需要根据良性指令仔细选择注入，并生成一个特别的期望响应，而不是来自SFT数据集的响应。此外，作为一种AI系统的防御方法，SecAlign还不能实现100%的安全性。对于LLM集成应用中的更强安全性需求，可能需要一个多层级的防御体系，将SecAlign与其他技术（如检测和输入重新格式化）结合起来。最后，SecAlign目前还不能防御提示注入之外的攻击，例如越狱攻击和数据提取。

因此，以上两种相关的技术方案明显均不能解决本发明所需要解决的技术问题。

下面结合附图，对本发明的较优的实施例作进一步的详细说明。

需要说明的是，大多数现有的技术方案主要集中在对特定模型的微调和安全对齐，或者侧重于输入输出端的内容过滤机制。这些现有的方案虽然在一定程度上也能解决部分安全问题，但依然存在一些限制和不足。比如：微调和安全对齐的方法需要大量的标注数据和训练资源，而输入输出端的过滤机制则依赖于规则和模板，难以适应复杂多变的生成任务。更为关键的是，现有的方案，通常是特定于某种模型或任务的，无法在不同模型之间通用，且存在一定的局限性，无法满足大规模、实时以及自动化生成的安全要求。

为此，如图1和图2所示，本实施例提供一种基于大模型自身反思机制的即插式安全增强方法，包括以下步骤：

步骤S1，实时接收用户的输入需求，通过大语言模型生成初步代码，首先基于规则匹配使用正则表达式和静态分析工具（如Semgrep）快速识别敏感信息（如API密钥、密码等），并通过大语言模型提取初步代码中的潜在风险，所述潜在风险包括未加密的密码信息、敏感数据以及不符合安全规则或编码规范的操作等，包括通过大语言模型分析代码上下文，识别不符合安全规则的操作，所述敏感数据包括API密钥和私密信息；

步骤S2，启动安全检测模块进行反思验证，识别潜在的安全问题，并通过大语言模型自生成的安全知识库（安全知识库一个包含常见漏洞模式（如SQL注入、XSS攻击、缓冲区溢出等）的规则库，主要来源包括迭代检索的历史以及以往的对话历史）对安全问题进行标记；所述安全问题包括SQL注入、XSS攻击以及缓冲区溢出等；

步骤S3，动态检索安全知识库并生成修复建议，并调用修正模块进行代码优化，基于RAG检索增强生成模型动态检索所述安全知识库，进行安全问题与安全知识库之间的匹配，在匹配后根据所述安全知识库进行自动修正和代码优化，并返回提取潜在风险和进行反思验证；

步骤S4，输出最终代码，生成安全分析报告，并将反思验证过程中所积累的安全知识进行总结，更新和拓展所述安全知识库；所述安全分析报告的内容包括安全问题、优化代码以及评分。

本实施例旨在提供的是一种通用的、无需微调训练的即插式安全增强方法与系统，旨在通过大语言模型自身的任务生成信息来进行反思并强化其安全性。本实施例在实时接收用户的输入需求之后，先通过大语言模型生成初步代码，并通过大语言模型提取初步代码中的潜在风险；在代码生成阶段，提取初步代码可能存在的风险点。然后，在反思验证阶段，启动安全检测模块进行反思验证，识别潜在的安全问题，并通过大语言模型自生成的安全知识库对安全问题进行标记。接着，在安全优化阶段，调用修正模块进行代码优化，基于RAG检索增强生成模型动态检索所述安全知识库，进行安全问题与安全知识库之间的匹配，在匹配后根据所述安全知识库进行自动修正和代码优化，在自动修正和优化之后返回进行反思验证，直到通过安全评估。最后，在输出阶段，输出最终代码，生成安全分析报告，并将反思验证过程中所积累的安全知识进行总结，更新和拓展所述安全知识库。

需要说明的是，本实施例所述步骤S3中，在根据所述安全知识库进行自动修正和代码优化之后，自动返回提取潜在风险和进行反思验证。在返回提取潜在风险的过程中，实际上返回的是步骤S1对应的代码生产阶段，此时，直接跳过用户输入需求，再次进行潜在风险的识别，并顺序跳转至步骤S2进行反思验证，直到通过步骤S3的代码优化和安全评估之后，才跳转至步骤S4输出最终代码，生成安全分析报告。

因此，本实施例在代码生成过程的每一个阶段，通过自身反思机制进行自动修正、代码优化以及反思验证，而不是像现有技术那样在生成任务/输出代码之后，才进行简单的错误修正，以确保每一次的输出结果均符合安全要求；在此基础上，还能够在此过程自生成和扩展安全知识库，使得自生成的安全知识库与自身反思机制形成互相支持和促进的关系，无需引用外部的知识库，也不再局限于某种特定的模型或任务，可以很好地满足不同模型之间通用性需求，为满足大规模、实时以及自动化生成的安全要求提供了更好的基础。

具体的，本申请通过引入自身反思机制，使得大模型（即大语言模型）能够在生成过程中的每一个阶段，自动识别和修复其中潜在的安全漏洞，包括潜在风险以及安全问题，并通过自我总结和凝练生成安全相关的知识。与传统的依赖外部安全知识库或人工干预的技术方案不同，本实施例的安全机制基于模型内部的生成过程，能够实时识别安全问题，生成安全规则、编码规范以及安全知识等，形成大模型自生成的安全知识库，并在后续生成任务中加以应用。

值得说明的是，本实施例的安全机制不依赖于外部的知识库，而是在生成过程中的每一个阶段，伴随着自身反思机制自生成并维护其安全知识库。因此可以有效地避免外部信息的不相关性或噪声对模型的干扰，同时也减少了对外部安全资源的依赖，具有更高的独立性和灵活性。本实施例通过对模型生成内容的深度反思和知识凝练，能够实现大模型在生成过程中自我优化，提升生成任务的安全性。这一技术方案的核心优势在于其“开箱即用”的能力，也就是即插式安全增强方法与系统，可以适用于各种大模型，无需专门的微调或训练，不再局限于某种特定的模型或任务，从而大大简化了安全防护的实施过程，并提升了大模型应用的普遍适用性和安全性。

在实际应用中，尤其是在政府和企业等对数据安全要求极高的领域，这种通用的即插式安全增强方法与系统显得尤为重要。大模型生成的内容，如文本、代码等，往往涉及到敏感数据和信息，如果缺乏足够的安全防护，可能会引发严重的安全隐患。例如，生成的代码可能包含未加密的敏感信息或存在未验证的用户输入，导致数据泄露或攻击风险。因此，本实施例能够对大模型生成的内容进行实时的安全审查和修复，确保其输出不含有潜在的风险和安全漏洞，并且没有使用的局限性，将对保护数据安全、提升大模型应用的可信度和可靠性具有重要意义。

本实施例的核心在于相互协作并促进的大模型自身反思机制以及自生成的安全知识库，能够在不依赖外部知识库的情况下，通过大模型自身的生成任务和自身反思机制，提高生成内容的安全性。

具体的，本实施例所述步骤S1包括以下子步骤：

步骤S101，通过大语言模型对用户的输入需求进行解析，根据自然语言描述生成初步代码；

步骤S102，通过大语言模型对每一次生成的初步代码进行分析，提取与安全相关的知识，获取初步代码中的潜在风险；与安全相关的知识包括安全漏洞、攻击方式、安全规则和编码规范，简称安全知识。

本实施例所述步骤S102中，针对与安全相关的知识，通过大语言模型识别出与代码模式对应的安全漏洞，安全漏洞对应的安全问题包括且不限于SQL注入、XSS攻击、跨站请求伪造、缓冲区溢出等；识别出潜在的攻击方式（如SQL注入、XSS攻击等），并将相应的安全规则和编码规范同步存入至所述安全知识库中；例如，对于SQL注入漏洞，提取的安全规则可以是“使用参数化查询”，对应的编码规范可以是“在编写SQL语句时，避免直接拼接用户输入，而是使用参数化的方式传递用户输入”等。

在本实施例之中，自身反思机制是核心之一，目的是让大模型能够在生成内容时进行实时的安全审查与优化。在生成任务开始时，大模型根据用户输入生成输出内容，可能是文本、代码等形式。生成后的内容会进入一个安全审查环节，这一环节对生成的内容进行全面评估，识别潜在的安全隐患。

例如，在生成代码时，模型可能会发现生成的代码中包含敏感数据（如API密钥、私密信息等）或不符合安全规范的操作。此时，自身反思机制会启动，分析生成内容中可能的风险源，进而对其进行修正。这一过程能够帮助大模型在生成内容时，自动避免泄露敏感信息、绕过安全控制等潜在的安全漏洞。

如果在第一次生成后，发现内容有问题，大模型会进一步思考并修正输出内容，确保其符合安全标准。例如，若发现生成的代码存在未加密的密码信息，自身反思机制会自动加密密码或者删除敏感信息，修正后的内容会再次提交给安全检测模块进行分析和评估。

具体的，本实施例所述步骤S2用于实现反思验证阶段，包括以下子步骤：

步骤S201，启动安全检测模块，通过静态分析工具对初步代码及其潜在风险进行分析，识别潜在的安全问题；

步骤S202，根据安全知识库中的安全知识，对潜在的安全问题进行分类标记，并结合领域知识以及最佳实践，提供针对性的修复建议或替代方案。

本实施例所述步骤S202中，在生成涉及数据库交互的代码时，大语言模型通过参数化查询或ORM框架来规避SQL注入这一安全问题；在生成Web应用代码时，大语言模型将提醒对用户的输入需求进行验证和转义，以避免出现XSS攻击这一安全问题；在任务生成过中，对生成的数据长度进行限制，使用正则表达式或其他验证机制检查数据格式，在对数组进行访问时确保索引值在数组的有效范围内，在使用动态分配的内存时确保操作不会超出分配的内存范围，并使用strncpy()、strncat()和fgets()等等安全函数进行数据处理和调用；此外，还通过大语言模型根据任务场景所对应的领域，结合领域知识动态调整安全策略，确保生成的内容符合最新的安全规则、编码规范和行业标准。

随着模型不断生成新的内容并进行反思，安全知识库中的知识将不断得到更新和扩展。每次生成任务后，模型会分析生成的内容，识别出新出现的潜在安全问题，并将相应的安全规则和编码规范等信息加入到安全知识库中，从而保持其更新性和时效性。

在未来的生成任务中，本实施例可以调用安全知识库中的内容，快速识别和避免潜在的安全问题。例如，在生成一段API调用代码时，模型可以参考安全知识库中的规范，确保生成的代码没有泄露敏感信息或包含潜在的攻击漏洞。

本实施例所述步骤S3用于实现安全优化阶段，包括以下子步骤：

步骤S301，调用修正模块进行代码优化，基于RAG检索增强生成模型动态检索所述安全知识库，进行安全问题与安全知识库之间的匹配，计算安全问题与安全知识库之中各知识条目的向量相似度，获取与安全问题的向量相似度大于预设相似度阈值的知识条目作为匹配的知识条目；预设相似度阈值指的是预先设置的相似度阈值，可以根据实际情况和需求进行设置与调整；

步骤S302，将匹配到的知识条目作为上下文信息，输入至所述修正模块，生成针对安全问题的代码修正建议；

步骤S303，根据生成的代码修正建议，对代码进行自动优化；

步骤S304，返回步骤S2，对每一次生成后的内容进行反思验证，若生成的代码包含不安全的函数调用或硬编码密码，则自动进行替换或删除，以确保生成内容符合安全标准；若生成的代码存在未加密的密码信息，则通过自身反思机制自动加密密码或者删除敏感信息；直到进行反思验证后的内容通过安全评估（安全评估包括静态评估以及动态评估，静态评估包括SonarQube、Pyroscope代码安全验证工具，动态评估主要是通过大模型对生成的代码进行安全检查，判断代码范式中是否存在常见的漏洞代码）。

本实施例所述步骤S2至步骤S3所实现安全性检测与优化的过程，存在于每一次生成任务中。在生成内容初步完成后，本实施例将通过大模型对其进行安全性评估。这一评估不仅包括检查内容是否存在敏感信息泄露、恶意代码和隐私侵犯等问题，还会对内容是否符合行业安全规范进行全面审查。

一旦检测到潜在的安全问题，系统会启动自动化修正机制，通过自生成的安全知识库进行优化。例如，如果生成的代码包含不安全的函数调用或硬编码的密码，会自动进行替换或删除，确保生成内容符合安全标准。这一过程不仅提高了生成内容的安全性，还增强了模型对潜在风险的敏感度，使其能够在实时生成任务中进行安全优化。

本实施例所述步骤S303中，在进行自动优化的过程中，还根据用户的反馈进行调整，如果用户对生成内容中的安全性提出异议或发现新的安全隐患，通过大语言模型再次对内容进行反思验证，以确保输出结果符合安全要求。通过这种反馈机制，模型的安全性得到了持续改进，保障了最终输出的内容始终处于安全的状态。

本实施例所述步骤S303中，对代码进行自动优化之后，还针对优化后的代码自动添加校验和权限控制，以进一步提高安全优化阶段的可靠性，降低反思验证不通过的可能性。

本实施例所述步骤S4用于实现输出阶段，包括以下子步骤：

步骤S401，在完成代码优化和反思验证之后，输出最终代码；

步骤S402，列出在代码优化和反思验证过程中发现的安全问题，展示经过优化后的代码片段，标识出优化前后的对比，并根据预设的评分标准对代码优化进行评分，以生成安全分析报告；

步骤S403，将每一次代码优化和反思验证过程中所积累的安全知识进行总结，将安全问题、优化方案以及最佳实践进行关联，对所述安全知识库进行更新和拓展，通过大语言模型实现并维护自身反思机制。

也就是说，本实施例在每次生成任务后，会将生成过程中积累的安全知识进行总结，形成自生成的安全知识库。该自生成的安全知识库包含了关于不同类型的安全漏洞、编码规范和安全规则等内容。这一安全知识库并非静态，而是随着模型生成任务的不断进行而扩展和更新，每当模型在生成任务中发现新的潜在风险或新的安全知识，它都会自动将其加入到知识库中。此为其一。

这种自生成的安全知识库具有高度的可扩展性和自适应能力。当模型进行新的生成任务时，它可以从知识库中提取出相关的安全知识，并将其应用到当前任务中。比如，在生成API调用代码时，模型可以参考安全知识库中的规范，确保生成的代码不包含可能的攻击面或不符合安全要求的代码模式。

其二，本实施例的安全知识库的最大优势在于它不依赖外部资源，完全由大模型自身生成和维护。这使得大模型能够在没有外部干预的情况下，独立扩展其安全能力，持续增强生成任务中的安全性。因此，本实施例不仅为大模型的广泛应用提供了安全保障，还为需要高安全性的应用场景（如金融、医疗、政府等）提供了强有力的技术支持。

本实施例还提供一种基于大模型自身反思机制的即插式安全增强系统，采用了如上所述的基于大模型自身反思机制的即插式安全增强方法，并包括：

代码生成模块，实时接收用户的输入需求，通过大语言模型生成初步代码，并通过大语言模型提取初步代码中的潜在风险，所述潜在风险包括未加密的密码信息、敏感数据以及不符合安全规则或编码规范的操作，所述敏感数据包括API密钥和私密信息；

反思验证模块，启动安全检测模块进行反思验证，识别潜在的安全问题，并通过大语言模型自生成的安全知识库对安全问题进行标记；

安全优化模块，调用修正模块进行代码优化，基于RAG检索增强生成模型动态检索所述安全知识库，进行安全问题与安全知识库之间的匹配，在匹配后根据所述安全知识库进行自动修正和代码优化，并返回提取潜在风险和进行反思验证；

输出模块，输出最终代码，生成安全分析报告，并将反思验证过程中所积累的安全知识进行总结，更新和拓展所述安全知识库

综上所述，第一，本实施例能够通过自身反思机制对大模型生成内容进行自我审查和反思，使其能够主动识别生成内容中的潜在安全漏洞及其安全问题，并进行实时修正。需要说明的是，本实施例的自身反思机制并不是像现有技术那样，在生成任务/输出代码之后进行简单的错误修正，而是在生成内容的每一个阶段进行持续的安全评估，确保每一次生成输出都符合安全标准。这种即插式安全增强方法与系统，不仅提高了生成任务中的安全性，还增强了大模型在实际应用中的安全性和可靠性。

第二，大模型通过自生成的安全知识库，持续积累和扩展安全知识。这一安全知识库不仅涵盖了常见的安全漏洞、编码规范和安全规则，还能够随着大模型生成任务的不断进行，自动从生成任务中总结出新的安全知识，形成可持续更新的安全知识体系。尤其是在没有外部知识库的支持下，本实施例依然能够通过自身的反思和学习生成安全知识，适应不同任务中的安全需求。通过这种自生成和自优化的安全机制，能够显著降低大模型在实际应用中出现安全漏洞的风险，特别是在需要高度保障数据安全的行业领域中，具有重要的应用价值。

第三、在生成过程中加入自动化的安全检测与修正机制，确保大模型能够实时识别并修复输出中的安全问题，以便可以在生成内容后对其进行全面检查，及时发现内容中的敏感数据泄露、恶意代码和隐私侵犯等风险点，并采取相应措施进行修正。

以上内容是结合具体的优选实施方式对本发明所作的进一步详细说明，不能认定本发明的具体实施只局限于这些说明。对于本发明所属技术领域的普通技术人员来说，在不脱离本发明构思的前提下，还可以做出若干简单推演或替换，都应当视为属于本发明的保护范围。

图1

说明书附图

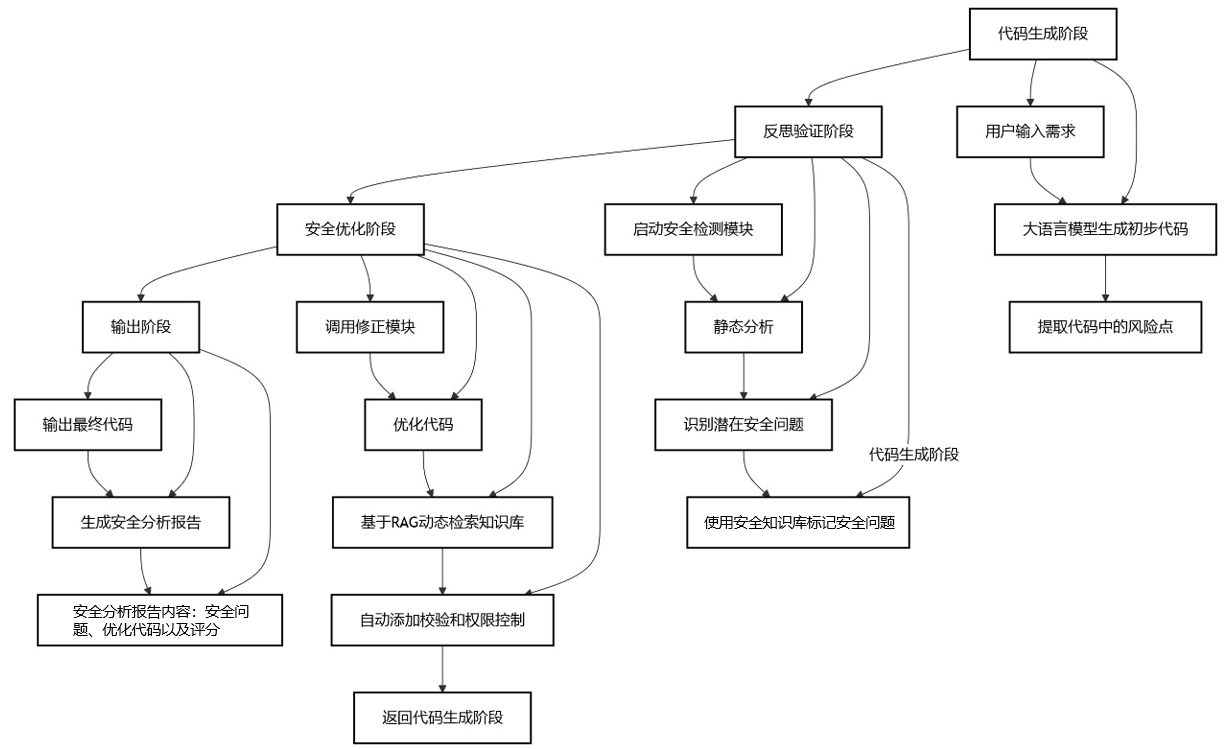


图2