Computer Science and Application 计算机科学与应用, 2026, 16(3), 668-679
Published Online March 2026 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2026.163093

Hans汉斯

基于沙箱技术的大模型应用安全漏洞检测系统

陈　聪，谢晟宇，陈　皓，陈鑫宇，于　越，徐亚峰

徐州工程学院信息工程学院，江苏　徐州

收稿日期：2026年2月7日；录用日期：2026年3月7日；发布日期：2026年3月16日

摘　要

由于大语言模型(Large Language Model, LLM)在智能问答、自动化运维、代码生成等场景中的应用日趋普遍，其安全问题愈加突出，例如工具投毒、提示注入、未授权操作及命令执行等安全风险。而传统Web漏洞扫描工具尚不能可靠地检测此类新型漏洞，针对以上问题综合应用沙箱隔离机制、攻击链还原算法、Python自动化分析技术及控制流图(CFG)分析等技术，在安全隔离环境中对大模型工具代码及其运行行为进行动态检测，既能检测SQL注入、XSS、CSRF等传统漏洞，又能检测MCP协议特有的工具投毒攻击及复杂提示注入漏洞，并能够输出包含完整数据流污染路径的可视化报告。实验结果表明所提出方案能提升大模型应用安全检测能力的有效性，也为此类应用提供了自动化、低成本的安全监控及漏洞修复参考方案。

关键词

大模型安全，沙箱技术，MCP协议，漏洞检测

A Sandbox-Based Security Vulnerability Detection Platform for Large Model Applications

Cong Chen, Shengyu Xie, Hao Chen, Xinyu Chen, Yue Yu, Yafeng Xu

School of Information Engineering, Xuzhou Institute of Technology, Xuzhou Jiangsu

Received: February 7, 2026; accepted: March 7, 2026; published: March 16, 2026

Abstract

As the application of Large Language Models (LLMs) in scenarios such as intelligent question answering, automated operation and maintenance, and code generation becomes increasingly prevalent, their security issues have become more prominent, including security risks such as tool

文章引用: 陈聪, 谢晟宇, 陈皓, 陈鑫宇, 于越, 徐亚峰. 基于沙箱技术的大模型应用安全漏洞检测系统[J]. 计算机科学与应用, 2026, 16(3): 668-679. DOI: 10.12677/csa.2026.163093

poisoning, prompt injection, unauthorized operations, and command execution. Traditional web vulnerability scanning tools are still unable to reliably detect these new types of vulnerabilities. To address these issues, we comprehensively apply sandbox isolation mechanisms, attack chain reconstruction algorithms, Python automated analysis techniques, and Control Flow Graph (CFG) analysis to dynamically detect the tool code and its operational behavior of large models in a secure isolation environment. This approach not only detects traditional vulnerabilities such as SQL injection, XSS, and CSRF but also identifies tool poisoning attacks and complex prompt injection vulnerabilities specific to the MCP protocol. It can also output a visual report containing the complete data flow pollution path. Experimental results show that the proposed solution enhances the effectiveness of security detection capabilities for large model applications and provides an automated, low-cost reference solution for security monitoring and vulnerability remediation for such applications.

Keywords

Large Language Model Security, Sandbox Technology, MCP Protocol, Vulnerability Detection

1. 引言

大语言模型由于有极好的语义理解及生成能力，在智能客服、自动代码生成、运维辅助、数据分析诸领域得到广泛应用，近年来越来越多的应用通过 MCP 协议让模型接入外部工具及系统资源，使模型获得“执行能力”，极大扩展了应用场景。但能力的开放必然带来新的安全风险，在 MCP 架构下，模型可能在未获用户明确授权的情况下调用高危工具接口，执行敏感操作，甚至直接触发命令执行、数据泄露等重大安全问题[1]。更隐蔽且危险的是，攻击者可通过构造恶意提示或恶意注释诱导模型，实施工具投毒攻击或提示注入攻击，其中复杂提示注入攻击通过隐蔽数据流传递恶意指令，更难被传统检测方法识别。目前市面上主流的安全检测工具均为传统 Web 漏洞设计，对这类与大模型深度耦合的新威胁，尤其是复杂提示注入攻击，尚无良好应对之法[2]。针对上述问题，本文提出一种基于沙箱技术的大模型应用安全漏洞检测平台方案，通过隔离环境对模型工具代码做动态分析、行为监控，结合控制流图(CFG)分析技术识别复杂提示注入攻击的数据流污染路径，实现 MCP 安全漏洞的自动、可靠检测。

2. 沙箱技术与 MCP 协议安全分析

2.1. 沙箱技术概述

沙箱技术(Sandboxing)是以隔离运行环境的方式限制程序访问权限的典型安全机制，其根本思想十分明确：在不妨碍宿主系统的前提下安全地执行不可信代码，限制文件系统、网络、进程、系统调用等权限，是漏洞分析及恶意代码检测中成熟且重要的技术手段[3]。沙箱在安全检测中可实现动态行为分析，并结合日志审计、异常检测探测攻击行为，具备较高的应用价值。

2.2. MCP 协议的安全风险

MCP 协议为大模型提供了统一的工具调用接口，使模型能够访问数据库、文件系统及外部 API。然而，该协议在设计上更注重功能扩展，安全机制相对薄弱，主要风险包括[4]：1) 工具投毒攻击：攻击者在工具描述或注释中植入恶意指令，诱导模型执行危险操作；2) 提示注入攻击：分为简单提示注入与复

杂提示注入，简单注入通过显性恶意提示绕过安全限制，复杂注入通过隐蔽数据流(如注释嵌套、参数传递、上下文拼接)传递恶意指令，隐蔽性强、检测难度高；3) 未授权操作：模型在无用户确认的情况下执行高危函数；命令与代码执行风险：工具接口参数校验不足导致的 RCE 问题。其中，复杂提示注入攻击是 MCP 协议下最具隐蔽性的核心风险之一，其核心危害在于通过“合法载体 + 隐蔽指令”的方式，使恶意提示绕过模型基础安全过滤，通过数据流传递触发高危操作。传统关键词匹配仅能检测简单提示注入，无法识别复杂注入的隐蔽数据流，因此需结合控制流图(CFG)分析技术，深入挖掘潜在的数据流污染路径，实现复杂提示注入的精准检测。复杂提示注入的数据流污染路径核心逻辑：攻击者将恶意提示片段拆分嵌入工具注释、参数默认值、上下文模板等合法载体中，模型在解析工具定义、拼接上下文时，会将这些分散的恶意片段聚合，形成完整的恶意指令并执行。该过程中，恶意数据从“隐蔽载体”流向“模型解析模块”，再流向“工具调用模块”，形成完整的数据流污染链路，传统检测方法难以追踪该链路，需通过 CFG 分析技术实现全路径追踪。因此，急需一种针对 MCP 特性设计的专用安全检测方案，重点强化复杂提示注入攻击的检测能力，结合 CFG 分析等技术实现数据流污染路径的精准识别。

3. 系统设计与实现

本系统面向大模型应用在实际部署过程中面临的安全风险，结合沙箱隔离技术、自动化漏洞检测方法及 CFG 分析技术，设计并实现了一套基于沙箱技术的大模型应用安全漏洞检测系统。系统整体采用模块化设计思想，围绕“代码扫描 - 沙箱执行 - 结果分析 - 安全决策”的检测流程展开，核心模块包括：文件扫描模块、沙箱执行模块、报告中心模块和威胁情报模块。系统总体功能框架如图 1 所示。

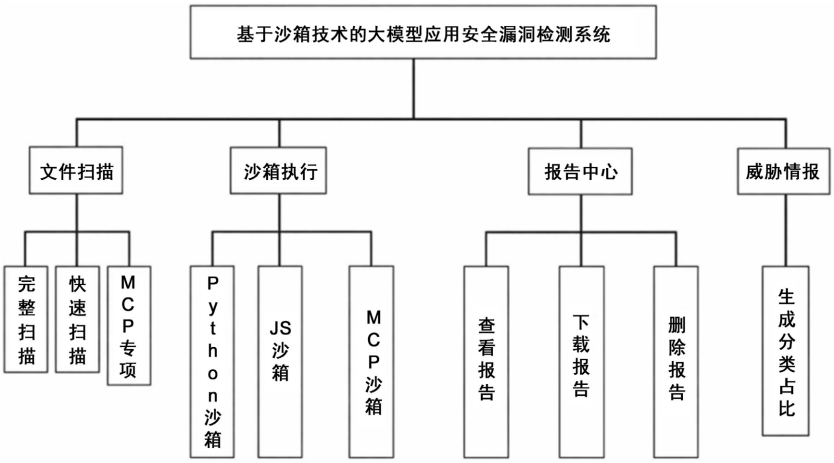

Figure 1. System module
图 1. 系统模块

3.1. 文件扫描模块

文件扫描模块是系统中进行文件静态风险分析、初步安全评估的核心检测模块，采用分层扫描、多策略协同的设计思路，实现完整扫描、快速扫描及 MCP 专项扫描三类检测机制，在保证检测深度的同时满足系统性能要求，可统一防护传统代码漏洞及大模型应用的新风险，重点强化复杂提示注入攻击的检测能力。

3.1.1. 完整扫描

完整扫描面向高安全等级应用场景，以检测全面性、准确性及低误报率为设计目标，采用多层级检

测架构：正则特征匹配 ＋AST 语法树分析 ＋AI 语义分析 ＋CFG 数据流分析。通过预定义高危规则库实现静态特征匹配，检测 SQL 注入、XSS 跨站脚本、命令注入、路径遍历等典型漏洞；基于 AST 语法树解析代码结构，识别 eval、exec 等动态执行函数及 subprocess、socket 等危险模块导入行为，降低规则匹配误报率；在此基础上采用 AI 安全分析模型实现代码逻辑语义级分析，识别复杂逻辑漏洞与隐藏恶意意图；结合 CFG 分析技术，追踪外部输入(含隐蔽提示片段)的数据流走向，识别复杂提示注入攻击的数据流污染路径。系统综合漏洞类型、严重程度、威胁情报等信息，完成风险评分与等级判定。整体效果呈现如图 2 所示。

1) AST 分析具体规则集

危险函数调用规则：匹配 eval()、exec()、execfile()、system()、popen()、spawn()等高风险执行函数，记录调用位置与参数。危险模块导入规则：检测 socket、ctypes、os、subprocess、pickle 等高风险库的导入与实例化行为[5]。数据流污染规则：基于控制流图(CFG)追踪外部输入数据(含工具注释、参数、上下文模板中的隐蔽提示片段)流向危险函数，识别未过滤的数据流污染路径，重点捕捉复杂提示注入的恶意数据聚合过程[6]。权限过度声明规则：解析 MCP 工具注解与权限字段，标记超出业务必要范围的文件读写、网络访问、进程创建权限。

2) AI 语义分析技术栈

模型选择：采用轻量级代码语义分析模型 CodeBERT，在真实大模型应用工具代码数据集上进行微调，适配 MCP 协议相关代码与提示词场景，重点优化复杂提示注入片段的语义识别能力。

输入向量化：对代码片段、注释、工具描述文本、上下文模板进行分词与向量映射，生成固定维度代码嵌入向量，重点保留隐蔽提示片段的语义特征。

输出解析逻辑：模型输出恶意概率、风险类型、可疑语句位置，系统设定置信度阈值区分正常代码与恶意逻辑，针对复杂提示注入，同步输出数据流污染路径的初步判定结果，结合 CFG 分析进一步验证。

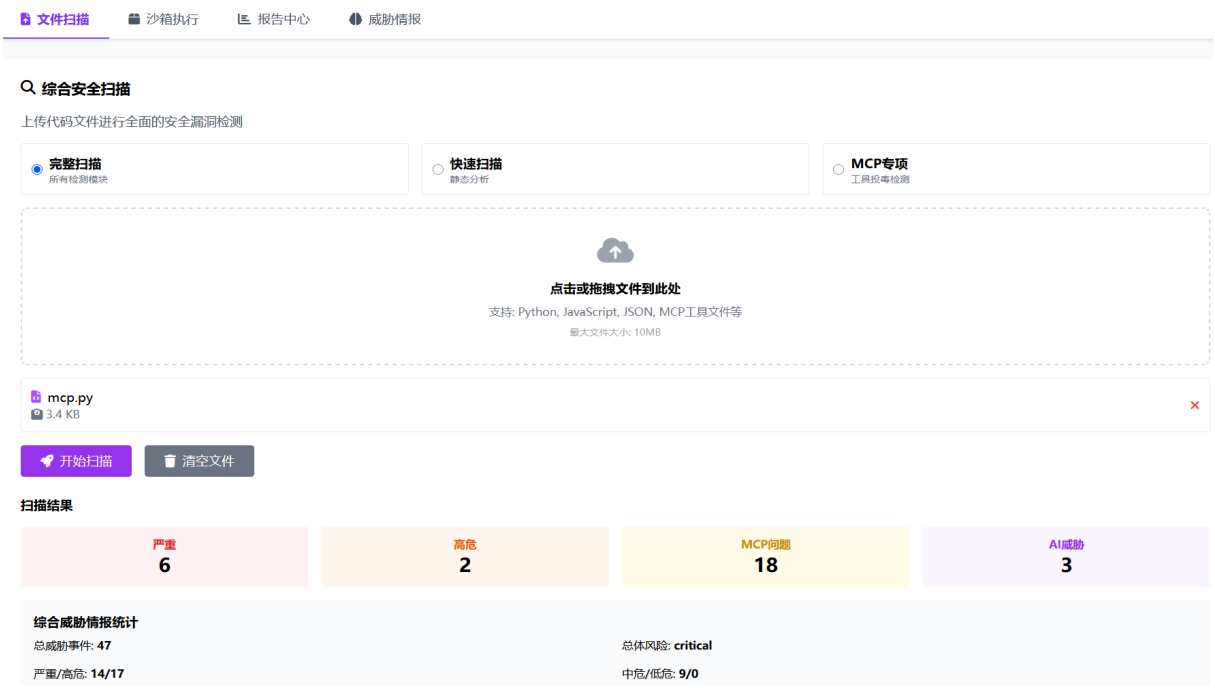

Figure 2. Full scan results
图 2. 完整扫描结果

3.1.2. 快速扫描

快速扫描机制是为实时检测、批量扫描两类场合设计的，目标是在有限计算资源下快速检测高风险特征。该机制采用轻量化规则集，利用优化后的正则表达式引擎对文件内容做快速扫描，跳过 AST 解析、AI 推理及 CFG 深度分析等高开销步骤，重点检查 Top10 高危漏洞特征：硬编码敏感凭证(AWS、Azure、GCPKey)、高风险函数调用及 shell=True 等危险参数配置、简单提示注入的显性关键词。快速扫描机制可在毫秒级给出检测结果，提高系统整体吞吐能力，适用于资源有限环境或高并发检测场景[2]。界面如图 3 所示。

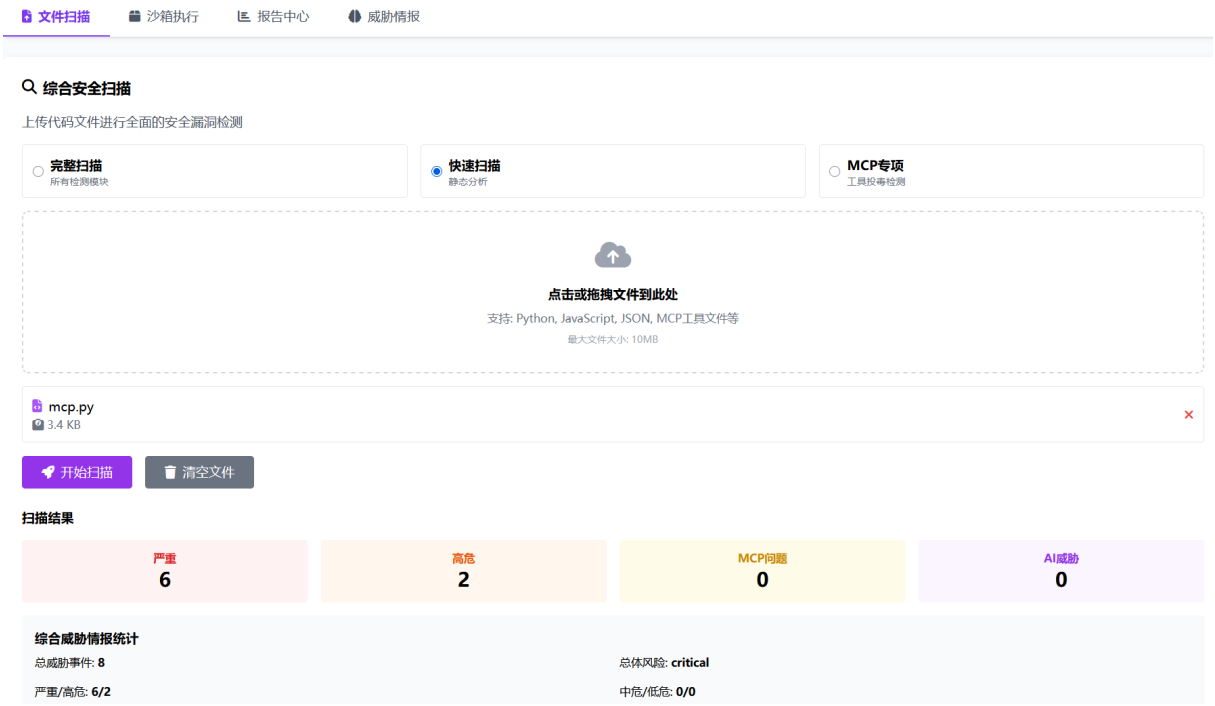

Figure 3. Quick scan results
图 3. 快速扫描结果

3.1.3. MCP 专项扫描

MCP 专项扫描机制针对 MCP 协议特有风险设计，重点强化复杂提示注入攻击的检测能力，弥补传统安全检测工具在协议层及提示层的不足[1]。该机制以 mcp_analyzer.py 模块为驱动核心，融合协议特征识别、提示词注入(PromptInjection)检测技术及 CFG 数据流分析技术，对 MCP 工具定义、代码注释、上下文交互内容、参数模板进行全面分析，实现简单与复杂提示注入攻击的全覆盖检测。

具体检测逻辑包括三部分：一是识别注释或文档字符串中所含的工具投毒行为，即“ignore security”“always execute”等试图绕过模型安全限制的诱导性指令；二是利用隐蔽指令提取引擎，检测代码、注释、参数模板中隐藏的恶意提示片段，结合 AI 语义分析判断片段的恶意倾向[4]；三是基于 CFG 分析技术，追踪这些隐蔽恶意片段的数据流走向，识别“片段嵌入－上下文拼接－恶意指令聚合－高危操作触发”的完整数据流污染路径，精准判定复杂提示注入攻击，该检测逻辑与后续实验中复杂提示注入 98.57%的召回率、97.89%的精确率形成呼应，验证了 CFG 分析在隐蔽数据流追踪中的有效性。此外，该机制对@mcp.tool 的定义规范做严格合规性检查，自动查找权限缺失或权限过度申请的问题，诸如非必要的文件操作、网络访问调用，并据此识别可能导致模型上下文污染的代码模式，提升 MCP 协议层的安全性[3]。结果如图 4 所示。

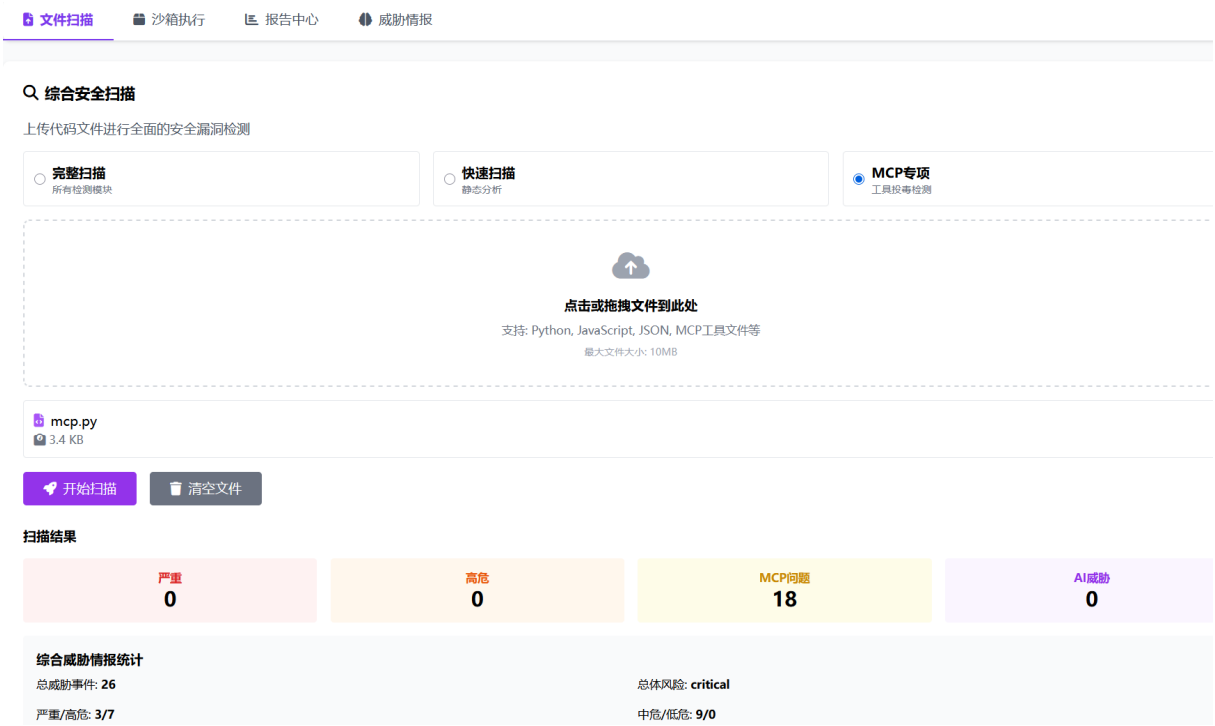

Figure 4. MCP scan results
图 4. MCP 扫描结果

本系统采用基于控制流图(CFG)的污点分析算法实现对复杂提示注入中隐蔽数据流污染的精准追踪。该算法将工具注释、参数默认值等视为“污点源”，将 eval、exec 及 MCP 工具调用接口视为“汇点”，通过遍历 CFG 识别潜在的污染路径。

3.2. 沙箱执行模块

沙箱执行模块采用静态预判 + 动态隔离架构，在隔离环境中运行可疑代码并验证运行时行为，根据风险等级分配对应执行环境，同时强化复杂提示注入攻击的动态验证能力，结合 CFG 分析结果追踪恶意数据流的运行时传递过程[5]。对风险评分低于 50 分的低风险代码，使用基于 multiprocessing 实现的受限 Python 环境，重写内置函数禁用 eval、exec 等危险调用，通过白名单限制可加载模块为 math、json 等无副作用标准库，并设置 CPU 与内存阈值。对风险评分 50~79 分的中高风险代码，通过 Docker API 创建临时容器，禁用网络、挂载只读文件系统、以非特权用户运行，实现与宿主机强隔离；在运行过程中，同步记录上下文拼接、工具调用的完整流程，验证 CFG 分析识别的数据流污染路径是否会触发恶意指令执行[4]。对风险评分 ≥80 分的极高危代码，直接触发熔断机制，拒绝执行。

攻击链还原算法如下：1) 日志采集：实时捕获进程标准输出、错误流、系统调用序列、文件操作、网络行为、异常堆栈与退出状态，重点采集上下文拼接、恶意提示片段聚合、高危函数调用的相关日志，为复杂提示注入攻击链还原提供数据支撑。2) 行为序列标注：将系统调用、函数调用、MCP 指令调用、提示片段拼接、数据流传递映射为行为节点，标记节点类型、触发源、时间戳，重点标注 CFG 分析识别的数据流污染路径相关节点。3) 关联路径构建：以“隐蔽提示片段嵌入→上下文拼接聚合→恶意指令生成→危险函数调用→系统影响”为链路模板，结合 CFG 分析结果，匹配行为节点间依赖关系，生成有向行为图，完整还原复杂提示注入的攻击链路。4) 攻击路径生成：从高危行为节点反向回溯源头节点，重

点追溯隐蔽提示片段的嵌入位置与数据流传递过程，还原完整攻击路径，输出包含触发条件、执行步骤、数据流污染路径、危害结果的攻击链报告。整体流程如下图 5 所示。

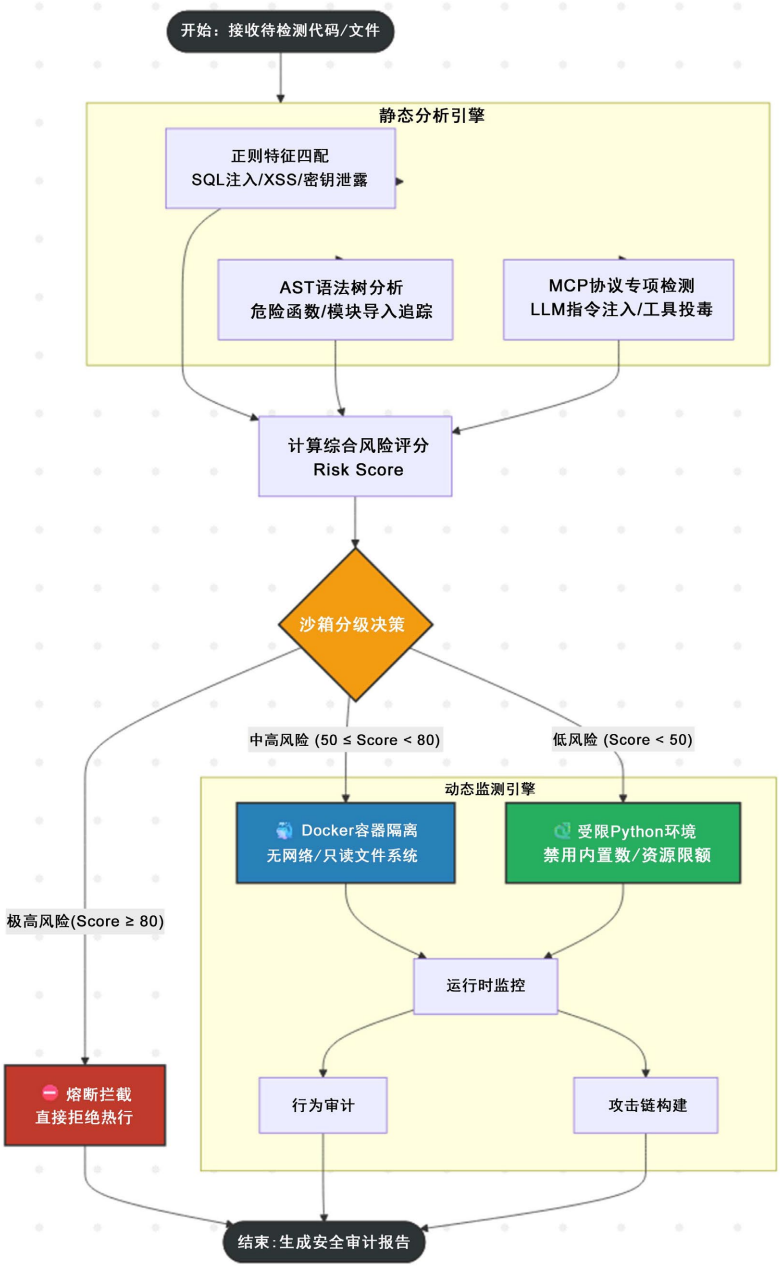

Figure 5. Overall process
图 5. 整体流程

在代码正式进入沙箱运行之前，模块利用 AST 技术与 CFG 分析技术做预执行静态分析，精准识别危险的模块导入、高危函数调用及潜在的数据流污染路径，动态计算风险评分，当评分达到 80 分及以上时即主动触发熔断机制，直接拒绝执行该代码，由此从根源上堵住高危威胁。展示结果如图 6 所示。除此之外，该模块具备完善的攻击链构建及行为审计能力，可自动还原从引入危险模块到执行恶意操作的

完整攻击路径，实时收集程序的标准输出、错误日志，严格跟踪进程退出状态及异常堆栈，进一步验证CFG 分析识别的数据流污染路径的真实性，为实验中复杂提示注入高检测精度提供了动态支撑。沙箱与MCP 协议分析器无缝联动，在运行时并行检测隐藏的 LLM 指令、协议层的投毒特征及复杂提示注入的数据流污染行为，既有利于取证，也能给安全人员提供全链条、全视角的动态防护视图[6]。

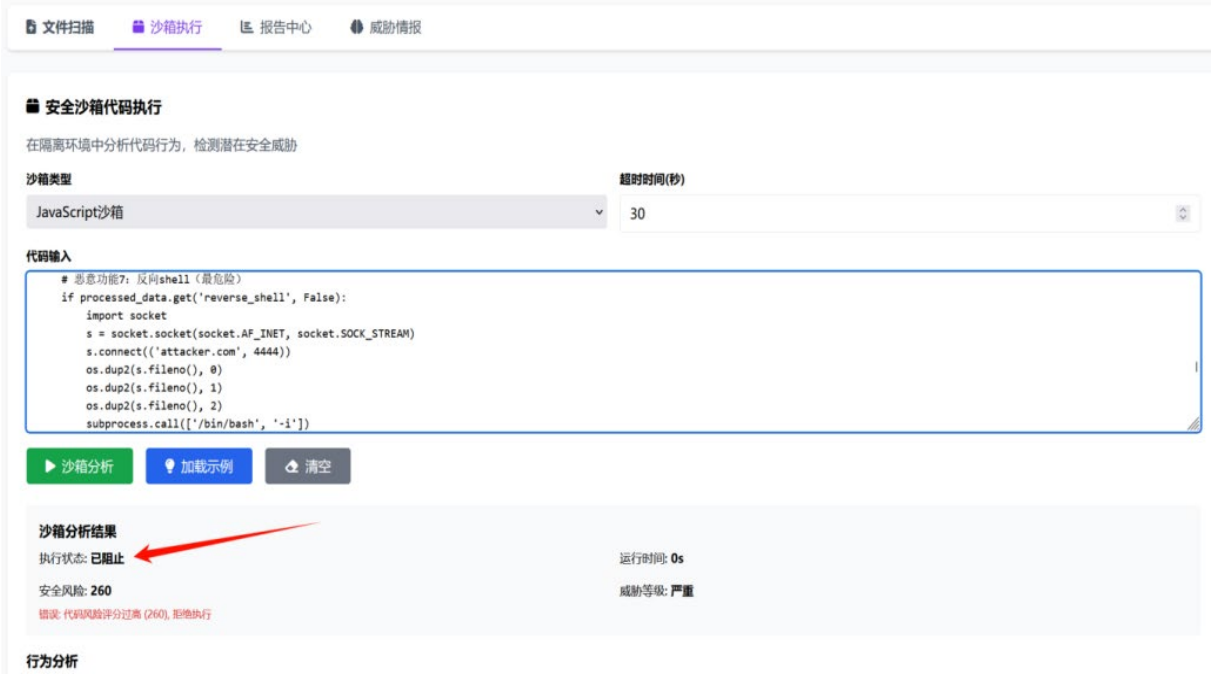

Figure 6. Sandbox execution module
图 6. 沙箱执行模块

在动态执行过程中，系统结合静态 CFG 分析结果，实时记录行为序列以还原完整的攻击链。针对复杂提示注入，我们设计了以下攻击链还原算法，用于验证静态分析发现的数据流是否在运行时真正触发了恶意行为，如图 7。

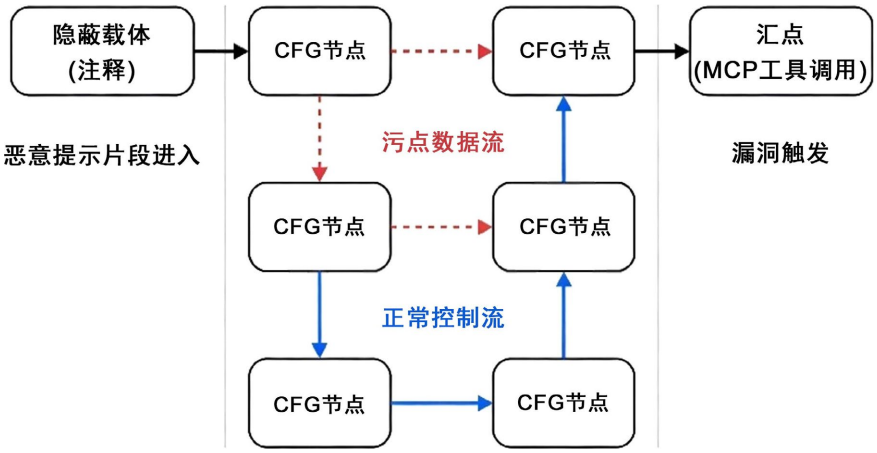

Figure 7. Data flow diagram of complex hint injection detection based on CFG
图 7. 基于 CFG 的复杂提示注入检测数据流图

3.3. 报告中心模块

报告中心模块是系统输出安全洞察的最后环节，承担着聚合静态扫描、沙箱分析、威胁情报等多源异构数据，生成标准化、可视化安全审计报告的任务，界面如图 8 所示。它采用结构化数据存储、分层渲染的技术手段，能以 JSON 格式进行机器可读的数据交换，便于 Web 端直观展示，十分契合自动化运维集成及人工安全审计两者的需要。

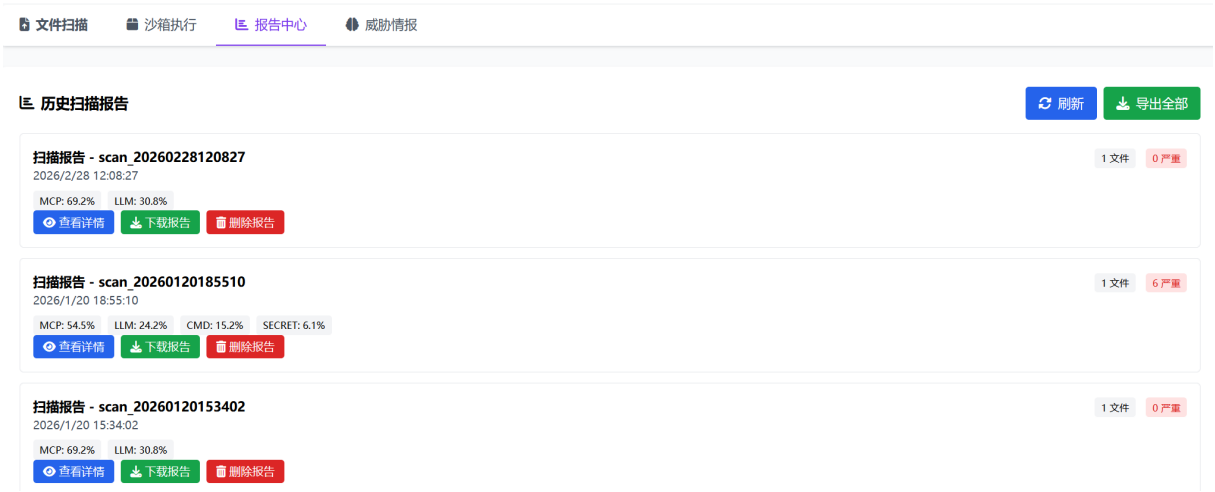

Figure 8. Report center module
图 8. 报告中心模块

本模块在核心实现上采用了智能聚合引擎，对 scan_file 得到的漏洞列表(含复杂提示注入漏洞及数据流污染路径信息)、sandbox 执行的攻击链记录以及 mcp_analyzer 捕获的协议风险予以解析，先用 generate_threat_report 函数对威胁数据做系统统计分析，自动计算威胁分布(Threat Landscape)，明确主要攻击向量(Attack Vectors，即“数据库层攻击”、“AI 模型投毒”)，再据此自动匹配预置的各类修复建议策略(Recommendations)，针对复杂提示注入攻击，重点给出 CFG 分析优化、隐蔽提示片段检测、数据流过滤等专项修复建议，这些优化方向进一步保障了系统对复杂提示注入的检测精度，与实验中 98.57%的召回率、97.89%的精确率形成呼应。所生成的所有报告都附有完整的时间戳、风险概览(Executive Summary)及技术细节证据(含 CFG 分析截图、数据流污染路径日志)，审计结果可追溯、有依据[7]。

3.4. 威胁情报模块

威胁报告模块是对系统检测中采集的安全事件做统一汇总、分类统计、可视化展示的专门组件，是实现安全态势感知及检测结果解释的重要部分。该模块以扫描任务为基本分析单位，对文件扫描、沙箱执行两阶段所得的检测日志加以解析、聚合，将分散的漏洞事件转化为结构化的威胁信息，呈现整体风险。系统从数据处理的角度对每次扫描所得的检测结果都做了标准化建模，把安全事件按漏洞类型及安全语义特征划分为 MCP、LLM、CMD、XSS 以及 SECRET 等威胁类别，再据此统计各类事件的数量及所占比例，客观反映出不同威胁类型在整体检测结果中的分布，明确大模型应用面临的主要安全风险来源。如图 9 所示，其中，MCP 协议滥用、LLM 提示注入(尤其是复杂提示注入)等新型风险的统计结果会突出显示，体现系统对大模型应用安全问题的针对性检测能力。威胁报告模块在结果呈现时采用可视化方式展示威胁分类占比，呈现的复杂安全事件统计结果十分直观，用户从威胁占比图中能看到各类风险

的相对严重程度，很容易判断目前应用中以协议层风险、模型交互风险为主，还是存在传统代码漏洞，有利于支持后续的安全决策和风险评估。可视化结果与底层检测数据严格一致，检测结论可解释性强。

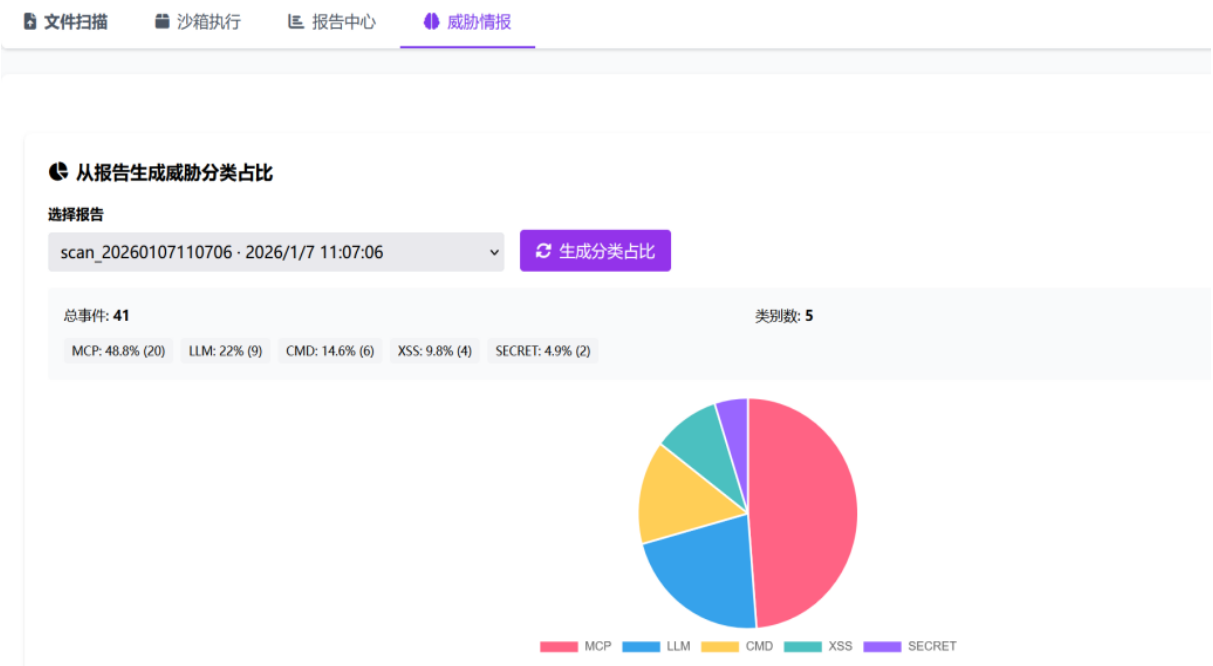

Figure 9. Threat intelligence module
图 9. 威胁情报模块

威胁报告模块能对检测结果做有效抽象、汇总，实现从单一漏洞发现到整体安全态势分析的过渡，提高系统的可用性及分析效率，为大模型应用安全风险评估提供清晰、直观、可量化的支撑，是连接底层安全检测、上层安全管理的重要环节。

4. 实验与结果分析

4.1. 测试数据集

实验采用公开 LLM 安全测试集与自建 MCP 专项测试集相结合的方式，重点增加复杂提示注入攻击样本，确保测试的全面性。选用 GitHub 公开的 Prompt Injection、工具投毒、代码注入测试集，共 1800 条样本，其中正样本(恶意样本) 900 条，负样本(正常样本) 900 条，包含 300 条简单提示注入样本。基于 MCP 协议规范构造工具定义、权限声明、上下文交互相关测试用例，自建 MCP 测试集，共 600 条样本，其中包含 350 条复杂提示注入样本(涵盖注释嵌套、参数传递、上下文拼接等多种数据流污染场景)，150 条工具投毒样本，100 条越权调用样本。总测试样本量 2400 条，按 8:2 划分为训练集与测试集，其中复杂提示注入样本单独标注，用于验证系统对该类攻击的检测能力。

4.2. 评价指标

采用准确率(Accuracy)、精确率(Precision)、召回率(Recall)、F1 分数评价检测效果，重点统计系统对复杂提示注入攻击的检测指标；统计单次扫描平均耗时与内存占用评价系统性能。

为全面评估系统性能，本文采用的检测效能指标：

准确率(Accuracy)：$\mathrm{Acc}=\dfrac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}$

精确率(Precision)：$\mathrm{Pre}=\dfrac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$

召回率(Recall)：$\mathrm{Rec}=\dfrac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$

F1 分数(F1-Score)：$\mathrm{F1}=2\times\dfrac{\mathrm{Pre}\times\mathrm{Rec}}{\mathrm{Pre}+\mathrm{Rec}}$

其中，TP 为真阳性(正确检出漏洞)，TN 为真阴性(正确判定安全)，FP 为假阳性(误报)，FN 为假阴性(漏报)。针对大模型安全场景，重点考察召回率以确保不漏过隐蔽的复杂提示注入攻击。系统性能指标包括平均扫描耗时(Avg.Latency)即单文件从输入到输出报告的平均时间(秒)和内存峰值占用(Peak Memory)即检测过程中进程占用的最大内存(MB)。

4.3. 对比实验

本实验选取了传统 Web 漏洞扫描器 AWVS、静态代码分析工具 Semgrep、以及专用大模型安全工具 LLM Guard 这三类主流工具作为基线进行对比。测试数据集包含 2400 条样本，其中包含 350 条高难度复杂提示注入样本。测试结果见如下表 1。

Table 1. Performance comparison of various tools in detecting MCP-related vulnerabilities
表 1. 各工具在 MCP 相关漏洞检测上的性能对比

检测工具	准确率	精确率	召回率	F1 分数	复杂提示注入召回率	平均耗时(s/file)	内存占用(MB)
AWVS	62.15%	45.30%	28.40%	35.12%	12.50%	1.8	450
Semgrep	78.40%	72.10%	64.50%	68.09%	45.20%	0.9	120
LLM Guard	85.60%	81.20%	80.50%	80.85%	68.40%	3.5	680
本文系统	98.17%	97.38%	99.00%	98.19%	98.57%	2.1	180

通过对上述指标性能对比，本系统高精度检测整体准确率达 98.17%，针对最具挑战性的复杂提示注入攻击，系统实现了 98.57%的召回率和 97.89%的精确率，显著优于 AWVS (召回率 < 30%)、Semgrep (82%)等主流工具。F1 分数达到 98.19%，远超其他对比工具，特别是在复杂提示注入这一核心难点上，传统工具 AWVS 几乎失效(召回率仅 12.5%)，Semgrep 因缺乏语义理解能力召回率不足 50%，LLM Guard 虽有一定效果但难以追踪跨函数的数据流污染。本文系统得益于 CFG 数据流分析与沙箱动态验证的双重机制，将复杂提示注入的召回率提升至 98.57%，有效解决了隐蔽恶意片段聚合难以识别的问题。系统虽然引入了动态沙箱和 AI 分析，但通过“静态预判 + 动态按需执行”策略，平均耗时控制在 2.1 秒/文件，仅略高于轻量级的 Semgrep，远低于纯动态分析的 LLM Guard。内存占用稳定在 180 MB 以内，证明了分层隔离架构的高效性。

5. 结论

针对大语言模型(LLM)在 MCP 协议下面临的新型安全威胁，设计并实现了一套基于沙箱技术与控制流图(CFG)分析的漏洞检测系统，有效填补了传统工具在 MCP 协议及复杂提示注入检测领域的空白，为大模型应用的安全落地提供了可靠的技术支撑。本系统创新融合了静态 CFG 污点追踪与动态沙箱行为审

计，构建了“文件扫描–沙箱执行–攻击链还原”的闭环检测体系，通过使用攻击链还原算法，能够输出包含完整数据流污染路径的可视化报告，为安全人员提供了清晰的修复依据。实验结果表明本系统在整体准确率、F1 分数、召回率、精确率、平均耗时、内存占用等方面，显著优于 AWVS、Semgrep 等主流工具，证明了其在自动化运维和高并发场景下的实用性。未来工作将聚焦于支持更多类型的模型交互协议，并进一步优化 CFG 分析在多语言混合场景下的泛化能力。

参考文献

[1] 赵月, 何锦雯, 朱申辰, 等. 大语言模型安全现状与挑战[J]. 计算机科学, 2024, 51(1): 68-71.

[2] Gulyamov, S., Gulyamov, S., Rodionov, A., Khursanov, R., Mekhmonov, K., Babaev, D., *et al*. (2026) Prompt Injection Attacks in Large Language Models and AI Agent Systems: A Comprehensive Review of Vulnerabilities, Attack Vectors, and Defense Mechanisms. *Information*, **17**, Article No. 54. https://doi.org/10.3390/info17010054

[3] 任奎, 王志波, 秦湛, 等. 大语言模型越狱攻击与防御[J]. 计算机与通信, 2025, 42(11): 112-125.

[4] 姜毅, 杨勇, 等. 大语言模型安全与隐私风险综述[J]. 浙江大学网络安全学报, 2025(专题卷): 13-30.

[5] 梁光辉, 摆亮, 庞建民, 等. 一种基于混合学习的恶意代码检测方法[J]. 电子学报, 2021, 49(2): 286-291.

[6] 秦臻, 庄添铭, 朱国淞, 等. 面向人工智能模型的安全攻击与防御策略综述[J]. 计算机研究与发展, 2024, 61(10): 2627-2648.

[7] 台建玮, 杨双宁, 王佳佳. 大语言模型对抗性攻击与防御综述[J]. 计算机研究与发展, 2025, 62(3): 563-588.