基于大模型的深层 Web 越权漏洞检测方法

覃锦端[1]，尉雯雯[2]，王月兵[1]，柳遵梁[1]，刘　聪[1]
（1. 杭州美创科技股份有限公司，浙江　杭州　310015；2. 陆军军医大学第一附属医院，重庆　400038）

摘　要：越权漏洞检测和挖掘是传统 Web 应用安全的一项重要课题，越权漏洞以其覆盖面广、隐藏深、没有固定流量特征等特点，一直是 Web 应用漏洞治理中的难点。目前业内常见的 Web 越权漏洞挖掘方法以被动式检测插件配合人工手工挖掘为主，被动式检测插件的原理多为替换为高权限/同权限的账号凭据，然后以返回流量包长度大小为漏洞是否存在的判断依据。此种方式虽然可以节省部分人工测试成本，但是只能检测浅层的越权漏洞，对于参数级的越权漏洞还是需要依赖人工手工测试。基于大模型技术，提出一种被动式深层 Web 越权漏洞检测方法，旨在通过大模型自动识别参数名含义和参数值特征，动态生成测试参数发包，并以返回流量包长度大小、具体内容等多个维度为漏洞判断依据。经测试，该方法可以挖掘出更深层次的越权漏洞，并有效节约人工手动挖掘成本。

关键词：Web 漏洞；漏洞检测；越权漏洞；大模型
中图分类号：TP309. 2　　**文献标志码：**A　　**DOI：**10. 19358/j. issn. 2097 - 1788. 2026. 02. 003

中文引用格式：覃锦端，尉雯雯，王月兵，等. 基于大模型的深层 Web 越权漏洞检测方法［J］. 网络安全与数据治理，2026，45（2）：20 - 27.
英文引用格式：Qin Jinduan，Wei Wenwen，Wang Yuebing，et al. A deep Web privilege escalation vulnerability detection method based on large models［J］. Cyber Security and Data Governance，2026，45（2）：20 - 27.

A deep Web privilege escalation vulnerability detection method based on large models

Qin Jinduan[1]，Wei Wenwen[2]，Wang Yuebing[1]，Liu Zunliang[1]，Liu Cong[1]
（1. Hangzhou Meichuang Technology Co.，Ltd.，Hangzhou 310015，China；2. The First Affiliated Hospital（Southwest Hospital）of Third Military Medical University（Army Medical University），Chongqing 400038，China）

Abstract：Privilege escalation vulnerability detection and mining is an important topic in traditional Web application security. Due to its wide coverage，deep concealment，and lack of fixed traffic characteristics，privilege escalation vulnerabilities have always been a difficult point in the governance of Web application vulnerabilities. Currently，the common Web privilege escalation vulnerability mining methods in the industry mainly rely on passive detection plugins combined with manual mining. The principle of the current passive detection plugins is mostly to replace with high-privilege or same-privilege account credentials，and then use the length of the returned traffic packet as the basis for determining whether there is a vulnerability. Although this method can save some manual testing costs，it can only detect shallow privilege escalation vulnerabilities，and for parameter-level privilege escalation vulnerabilities，it still relies on manual testing. Based on large model technology，this paper proposes a passive deep Web privilege escalation vulnerability detection method，aiming to automatically identify the meaning of parameter names and the characteristics of parameter values through large models，dynamically generate test parameters for sending，and use multiple dimensions such as the length and specific content of the returned traffic packet as the basis for vulnerability judgment. After testing，this method can detect deeper privilege escalation vulnerabilities and effectively save manual mining costs.

Key words：Web vulnerability；vulnerability detection；privilege escalation vulnerability；large model

0　引言

越权漏洞，顾名思义就是超越了自身所具备的权限所带来的漏洞。越权漏洞可以让同等身份的账号具备获取其他账号信息、操作其他账号数据的能力，也可以让低权限身份的账号具备高权限账号的能力，一旦应用系统中出现了越权漏洞，就等于安置了一个

“隐藏后门”，将导致网络功能损伤或者隐私泄露等问题，从而造成重大损失[1]。因此，越权漏洞也是 Web 应用安全领域中被尤为关注的一个漏洞门类。但是近年来越权漏洞的检测和治理还是比较低效，人工检测漏洞和利用漏洞费时且易出错[2]，尤其是那些与业务逻辑强相关的深层次越权漏洞，其准确识别与验证成为众多 Web 应用安全人员面临的主要挑战。

越权漏洞在 Web 应用漏洞类别中通常归为逻辑漏洞一类，逻辑漏洞就是与操作系统、编程语言、中间件、数据库等组件无关，与后端函数实现、前端页面表达也无关，而是与业务逻辑相关的一种漏洞。越权漏洞攻击方法通过利用特定的参数或用户身份提升权限访问到非授权内容[3]，例如，账号 zhangsan 查询接口“/getUserInfo？uName = zhangsan”获取自己的个人信息，若 zhangsan 为黑客，那么其发起的查询可能为“/getUserInfo？uName = lisi”，“lisi”这个字符串并不带有任何恶意特征但却完成了越权漏洞的利用。目前业内对于深层越权漏洞的检测主要以人工为主，使用多账号认证凭据替换插件为辅助手段，效率低下，尚需一种高效、可以自动覆盖参数的测试方法。

1 相关工作

1.1 Web 越权漏洞检测研究现状

Web 应用越权漏洞是随着 Web 应用的诞生而出现的，进入 Web2.0 时代后，动态页面、前后端分离、API 接口等技术的兴起更是将越权漏洞的影响力推向新高，为攻击者提供更多的潜在入侵点[4]。受网站业务逻辑设计异构性的影响，人工黑盒测试是目前 Web 越权漏洞黑盒测试的主要方式[5]，近几年来业界主要采用静态代码扫描、动态流量差异检测以及智能特征分析三种方式辅助人工检测越权漏洞。

（1）静态代码扫描：依赖代码树、属性分型等技术，从源代码层面进行扫描，找出可能存在越权漏洞的函数方法和对应的参数。具备代表性的为各类 SAST（静态应用程序安全测试）扫描工具，如 Fortify 等。

（2）动态流量差异检测：被动式捕获 HTTP 流量后根据固定策略对请求参数进行修改，通过比对响应包的长度特征进行漏洞判定。具备代表性的如 Burp-Suite 工具的 Autorize 插件、ztosec 团队的 secscan-auth-check 平台等。

（3）智能特征分析：引入机器学习方法如 Transformer 模型从历史流量中学习正常的访问控制模式和请求特征，从而识别异常的越权请求。

目前所使用的上述三种技术中，静态代码扫描通常会带来较高的误报率且复现困难；动态流量差异检测效果较好但需要人工进行策略的设置以及检测结果的复验；智能特征分析语义分析能力有限且需要海量的业务数据进行训练，无法通用。因此当前的技术与实现 Web 越权漏洞自动化检测之间还存在较大的落差，本文所提方法旨在在动态流量差异检测的被动式检测基础上引入大语言模型（Large Language Model，LLM）能力来解决准确率和自动化程度的问题。

1.2 大语言模型

大语言模型可以说是最近十年最具影响力的科技之一，成为人工智能领域的焦点[6]，其在最近短短的几年间深刻改变了计算机领域的生产方式，并极大地提高了许多领域的生产效率，成为推动技术革新的关键力量[7]。大模型顾名思义就是具有大参数量、大数据规模以及需要庞大计算资源的由人工神经网络构建的一类人工智能模型。在大模型出现之前的人工智能技术，通常只能作为特定领域的技术解决手段，如 AI 图片识别、AI 客服等，而且往往只能在学习过的知识中才能具备较好的效果，未曾学习过的知识效果非常差，无法具备像人一样的通用能力和学习能力。而在大模型出现后，得益于其“大”，使得大模型具备了出色的泛化能力，并拥有一定的类人思考和学习过程，可以遵循人类指令进行复杂推理，从而解决通用任务。即使在从未学习过的问题中，大模型也能够做出较为合理准确的解决方案。目前，人工智能大模型的参数量已经从百亿、千亿上升至万亿规模，也衍生出了如视觉大模型、多模态大模型等多种大模型来解决更多领域的问题，比较具备代表性的有国外的大模型如 OpenAI 的 ChatGPT、Google 的 Gemini 等，国内的大模型如阿里巴巴的 Qwen、深度求索的 DeepSeek 等。

人工智能大模型的发展为网络安全领域带来了新的机遇，大模型强大的泛化和学习能力在流量监测、恶意代码识别、代码审计等领域发挥了积极的作用，提升了网络安全软件的智能水平[8]。本文提出的 Web 越权漏洞检测方法正是利用大模型对 HTTP 流量的检测识别能力，通过构造提示词（Prompt）将大模型的数据包分析能力转化为对越权漏洞的检测能力。

1.3 MCP 协议

自 2022 年 ChatGPT 引发关注以来，人工智能大模型迅速崛起，在逻辑推理和生成质量上取得了快速进展，能够处理复杂任务并提供精准预测[9]。然而，即使是最复杂最先进的模型，也受限于与数据的隔离，大模型如同一个信息孤岛，其信息数据和功能只能在

学习时添加，无法与外界的现有系统和数据进行互联互通。这一局限性严重阻碍了大模型的大规模应用。MCP 协议应运而生。MCP（Model Context Protocol，模型上下文协议）是一个由 Anthropic 公司在 2024 年 11 月开源的新开放标准，它的出现解决了人工智能大模型的“信息孤岛”，帮助大模型更好地解决人类问题。MCP 协议工作时需要三个关键组件：MCP 主机（Host）、MCP 客户端（Client）以及 MCP 服务器（Server），MCP 主机即可以调用 AI 大模型服务的应用软件，如 Trae、Claude Code、Cursor 等；MCP 客户端在 MCP 主机中运行，目前绝大多数的 AI 应用工具都集成了 MCP 客户端功能，用户只需维护一个配置文件即可；MCP 服务器负责暴露现有的工具和数据资源，创建可以重用的提示模板工作流供 AI 大模型调用。本文提出的 Web 越权漏洞检测方法利用了 MCP 协议，使得 AI 大模型可以调用系统上的 Python 等已有工具实现 HTTP 数据包请求重放、前置正则匹配过滤等流程，通过工具或功能模块的协同完成相关任务[10]。

2 成果论述与实现

2.1 整体架构模块设计

本文设计了一种基于大模型的深层越权漏洞检测方法。区别于传统的被动式越权漏洞检测技术，本文设计的深层越权漏洞检测系统，不再局限于传统的认证凭据替换来进行越权漏洞检测，而是利用 AI 大模型强大的流量识别能力，理解 HTTP 请求头、HTTP 请求参数的含义后对相应的参数值进行报文修改[11]，然后重放替换参数后的请求包并自动识别 HTTP 响应包差异来判断是否存在越权漏洞。基于大模型的深层越权漏洞检测方法由前置过滤模块、AI 动态参数修改模块、MCP 数据包重放模块、AI 越权漏洞检测模块四大模块组成，前置过滤模块通过被动式流量代理过滤掉无用的 HTTP 流量如静态文件请求、多媒体文件请求等，减少需要大模型进行识别和操作的 HTTP 流量包数量，提高效率并降低成本；AI 动态参数修改模块通过提示词设计，引导大模型对请求包中的关键参数进行有效识别和修改；MCP 数据包重放模块则利用 MCP 协议使得大模型具备调用 Python 等工具进行流量包重放的能力；AI 越权漏洞检测模块同样利用提示词设计，使得大模型可以比对响应包的差异来判断是否存在越权漏洞。本文提出的基于大模型的深层越权漏洞检测方法基本模块如图 1 所示。

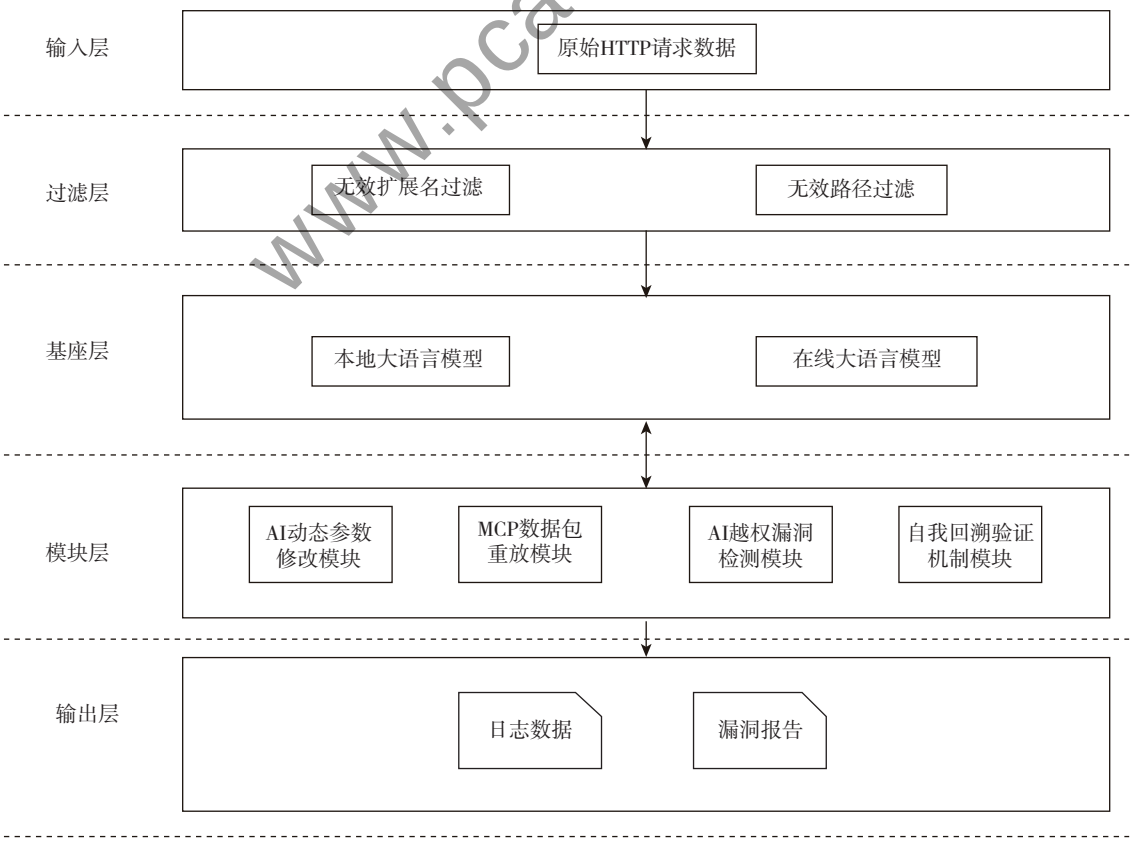

图 1　基于大模型的深层越权漏洞检测方法基本模块示意图

2.2 各模块具体设计

AI 动态参数修改模块、AI 越权漏洞检测模块、自我回溯验证机制模块是本文所述越权漏洞检测架构的核心部分，前置过滤层、MCP 数据包重放模块则分别负责冗杂流量过滤和系统工具调用的任务，辅助核心模块更好地完成检测工作。

2.2.1 前置过滤层设计

在 Web 应用系统运行过程中，越权漏洞绝大部分只会出现在 API 接口中，但是 Web 应用系统的功能除了 API 接口之外还会请求其他的一些资源如静态资源、多媒体资源等。当处于白盒测试条件下时，可以根据开发人员提供的 API 接口文档针对性地进行越权漏洞测试，但是当处于黑盒测试条件下时，由于无法获得完整的接口列表，只能通过对 Web 应用系统进行完整的功能测试才能进行越权漏洞检测，此时这些静态资源、多媒体资源的请求流量就会对测试过程产生干扰，出现误报并增加系统开销。当引入 AI 大模型进行 HTTP 流量分析时，这些多余的请求更是会消耗大量的大模型推理资源、token 窗口长度。因此，在 HTTP 流量接入到 AI 大模型之前，必须对流量进行过滤，丢弃掉无用的请求数据包，才能提高整个系统架构的运行效率。本文提出的越权漏洞检测架构的前置过滤模块通过文件后缀名模式、路径模式两种过滤模式配合，可以过滤掉静态资源、多媒体资源等无用请求，为后续任务提供了高质量的特征输入[12]。两种过滤模式具体详情如表 1 所示。

表 1　过滤模式表

过滤模式	过滤内容	过滤示例
文件扩展名模式	.js，.css，.png，.jpg，.jpeg，.gif，.ico，.svg，.woff，.woff2，.ttf 等文件后缀请求	https：//api. imdemo. com/index. js
路径模式	/static/，/assets/，/public/，/images/，/img/，/css/，/js/，/fonts/等路径的请求	https：//api. imdemo. com/static/ *

2.2.2 AI 动态参数修改模块设计

本文设计了一种 AI 动态参数修改机制，基于 AI 大模型对 HTTP 请求流量包中的关键位置进行智能修改，从而测试在不同值下接口的响应。AI 动态参数修改模块遵循的工作机制如下：

（1）阶段 1：AI 大模型接入配置

根据实际情况选用国内外的 AI 大模型，如 Gemini、ChatGPT、DeepSeek、Qwen 等，接入过程选择兼容 OpenAI 的方式进行接入，接入方式为本地/在线 API 方式。

（2）阶段 2：AI 大模型初始提示词（Prompt）设计

人为为大模型赋予一个 Web 安全测试专家的角色，使其在后续的上下文中更能专注于 Web 安全测试领域，从而获得更好的表现。

（3）阶段 3：AI 大模型核心工作提示词设计

在此阶段中，需要对大模型的 HTTP 请求修改工作进行具体的指导，包括需要修改的参数类型、参数规则示例以及修改完成后的返回格式，核心的参数变化规则提示词如下：

1. user_id 类型：尝试减 1、加 1、设为 1（管理员）、设为 0、随机数；
2. phone 类型：替换为其他手机号格式；
3. email 类型：替换为其他邮箱；
4. role/permission 类型：尝试 admin、root、1、0 等权限值；
5. token/session 类型：尝试空值、其他用户 token；
6. 数字 ID 类型：尝试 1、0、－1、随机数；
7. 状态标志：尝试 0、1、true、false 等。

2.2.3 MCP 数据包重放模块设计

本文设计了一种 MCP 数据包重放模块，AI 大模型修改后的 HTTP 请求流量数据通过 JSON 格式发送给 MCP 数据包重放模块，该模块会对数据进行组装，调用 Python 中的 requests 等模块进行流量重放，并获取响应供后续分析使用。MCP 数据包重放模块工作流程如图 2 所示。

2.2.4 AI 越权漏洞检测模块设计

本文设计了一种 AI 越权漏洞检测模块，基于 AI 大模型本身的数据识别分析优势，通过提示词指导大模型对修改后的 HTTP 响应数据进行对比分析，识别接口是否存在平行越权漏洞、垂直越权漏洞。漏洞判定标准核心提示词如下。

判断标准：

1. 如果 URL 包含 login、signin、auth 等关键词，这是登录接口，应该跳过越权检测以避免误报；
2. 如果修改参数后能访问到其他用户的数据→水平越权；
3. 如果修改参数后能访问到更高权限的功能→垂直越权；

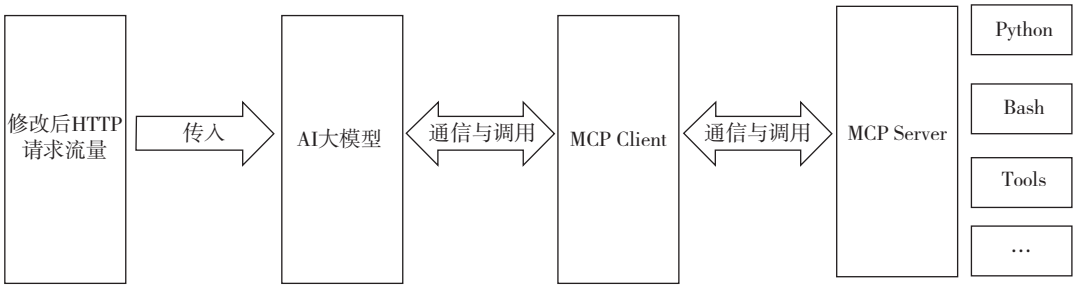

图2　MCP 数据包重放模块工作流程示意图

4. 如果响应内容包含敏感信息→信息泄露；

5. 如果响应状态码、内容长度、关键字段有显著差异→可能存在漏洞；

6. 如果响应完全相同或返回错误→可能不存在漏洞。

特殊处理：

－登录接口（URL 包含 login/signin/auth）：直接返回 is_ vulnerable = false，避免误报；

－认证失败的响应差异是正常行为，不应视为越权漏洞。

2.2.5　自我回溯验证机制模块

本文设计了一种自我回溯验证机制模块，基于一种多级推理算法的回溯机制，降低大模型幻觉，实现“反思”和评估，得出最终的越权漏洞检测置信度评分和风险等级评级。其中置信度是一个百分比值，百分数越大信任程度越高，风险等级为固定三个级别。其“反思”核心提示词如下：

1. 响应中是否包含具体的个人身份信息（PII），如身份证号、手机号、地址；

2. 响应中是否包含具体的业务资产，如订单金额、账号余额；

3. 如果只是包含通用的错误代码、配置信息或公开的列表数据，请认定为无风险；

4. 根据上述特征输出风险等级；

5. 根据越权检测流程重要程度生成权重；

6. 根据传入的各阶段状态和权重，输出置信度分数（0% ~100%），分数越高表示越有可能存在越权漏洞。

自我回溯验证机制模块示意图如图 3 所示。

2.3　大模型驱动的越权检测算法流程

本文设计了一种多级推理算法流程，将各模块输入输出进行联动，通过状态计算和标志传递完成检测流程。

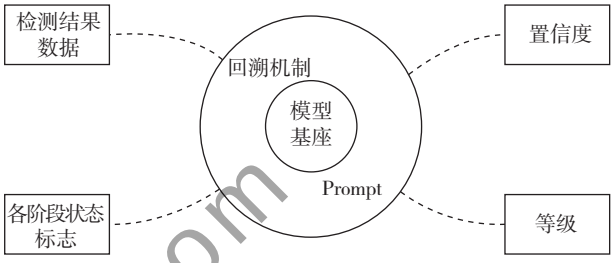

图3　自我回溯验证机制模块示意图

（1）前置过滤层存在一种过滤算法 filChain( )

输入：原始 HTTP 请求序列 $sData_x$，过滤模式 fMo_x（文件扩展名模式 $extMo_x$/路径模式 $dirMo_x$）。

输出：过滤后请求序列 $fData_x$，过滤标志 $fStatus_x$。

过程：按照过滤模式对请求数据流进行过滤的同时，对前后数据序列进行比对，当序列发生变动时证明存在过滤行为，filterStatus 置为 1；若未变动则此次数据流均为有效数据序列无需过滤，filterStatus 置为 0。该算法表达式如式（1）所示：

$$\text{filChain}(sData_x,\ fMo_x) = \begin{cases} fData_x \\ fStatus_x(0,\ 1) \end{cases} \tag{1}$$

（2）AI 动态参数修改模块存在一种修改算法 mChange( )

输入：过滤后请求序列 $fData_x$，修改提示词 mPrompt。

输出：修改后请求数据集 $mData_x$，修改标志 $mStatus_x$。

过程：根据 mPrompt，大语言模型会完成对 $fData_x$ 数据的自动化修改，并生成 $mData_x$。修改过程中存在三种修改状态，即参数级修改、请求头修改以及混合修改。$mStatus_x$ 会分别置为 1、2、3，来标记当前请求数据包的修改方式。该算法表达式如式（2）所示：

$$\text{mChange}(fData_x,\ mPrompt_x) = \begin{cases} mData_x \\ mStatus_x(1,\ 2,\ 3) \end{cases} \tag{2}$$

（3）MCP 数据包重放模块存在一种重放算法 rPeter()

输入：修改后请求数据集 $mData_x$，MCP 重放配置参数 rTool。

输出：修改后响应数据集 $rPoData_x$。

过程：大语言模型通过 MCP 协议调用重放工具 rTool，对 $mData_x$ 进行重放，输出 $rPoData_x$。该算法表达式如式（3）所示：

$$rPeter(mData_x, rTool) = rPoData_x \tag{3}$$

（4）AI 越权漏洞检测模块存在一种检测算法 deteRisk()

输入：原始请求响应数据集 $sPoDta_x$，修改后响应数据集 $rPoData_x$，检测提示词 dPrompt。

输出：检测结果数据 $deteData_x$，检测标志 $dStatus_x$。

过程：根据 dPrompt，大模型会比对 $sPoDta_x$ 和 $rPoData_x$ 的差异，从而判断是否存在越权漏洞，并输出 $deteData_x$。当存在越权漏洞时，$dStatus_x$ 置为 1；当不存在越权漏洞时，$dStatus_x$ 置为 0。该算法表达式如式（4）所示：

$$deteRisk(sPoDta_x, rPoData_x, dPrompt) = \begin{cases} deteData_x \\ dStatus_x(0, 1) \end{cases} \tag{4}$$

（5）自我回溯验证机制模块存在一种评估算法 assRisk()

输入：自检提示词 aPrompt，检测结果数据 $deteData_x$，检测标志 $dStatus_x$，过滤标志 $fStatus_x$，修改标志 $mStatus_x$。

输出：漏洞置信度 $conLevel_x$，漏洞等级 $rLevel_x$。

过程：首先大模型会根据 aPrompt 输出各检测阶段的权重值集合 W_x，根据权重对各个阶段的状态标志进行加权平均，以此回溯得到漏洞置信度 conLevel，该阶段算法表达式如式（5）所示：

$$conLevel_x = assRisk(aPrompt, W_x, dStatus_x, fStatus_x, mStatus_x) = \sum_{i=1}^{x} W_x dStatus_x fStatus_x mStatus_x = \begin{cases} 0\% \sim 50\%, & \text{认为误报} \\ 50\% \sim 75\%, & \text{可能存在漏洞} \\ 75\% \sim 100\%, & \text{确定存在漏洞} \end{cases} \tag{5}$$

然后使用自检提示词 aPrompt 根据检测结果数据 $deteData_x$ 和漏洞置信度 $conLevel_x$ 进行再检，得到漏洞等级 $rLevel_x$，该阶段算法表达式如式（6）所示：

$$rLevel_x = assRisk(aPrompt, deteData_x, conLevel_x) = \begin{cases} 低风险 \\ 中风险 \\ 高风险 \end{cases} \tag{6}$$

在上述算法中，x 是一个标识符，用于区分不同数据实例。

3 实验与分析

3.1 实验环境

构建集成 AI 大模型以及 MCP 调用模块的 Web 越权漏洞检测系统，通过一个存在越权漏洞的 DEMO 系统实验验证本文所述的越权漏洞检测方法的可行性、先进性。涉及的系统逻辑以及架构均使用 Python 语言实现，涉及的数据使用 SQLite 数据库进行存取，AI 大模型是使用智谱 AI 的 glm-4.5-air 模型。系统开发环境为安装了 Windows 11 操作系统、内存大小为 16 GB、处理器为 Intel i5 以及磁盘大小为 500 GB 的 PC 主机。

基于上文中所设计的架构，在本实验中开发了一款 AiForAccess 越权漏洞检测工具，基于该工具实现如下的实验过程：

（1）用户启动一个存在垂直越权、水平越权漏洞的 DEMO 应用系统，并设置应用代理为指定端口 A；

（2）用户启动 AiForAccess 越权漏洞检测工具，并监听端口 A，接收来自端口 A 的 HTTP 请求流量；

（3）用户正常使用 DEMO 应用系统的功能，访问相应的接口；

（4）AiForAccess 越权漏洞检测工具对 HTTP 请求流量进行自动修改，并对返回数据进行识别分析，将越权漏洞判断结果返回至可视化界面中。

同时，本实验通过对比法、控制变量法设计了测试用例，在相同条件下分别使用纯人工检测、插件辅助检测、AI 自动化检测三种方法对 DEMO 应用系统中的越权漏洞检测结果、耗时进行对比，得出各方案的优劣性。

3.2 实验实现

表 2 所示为本节所述实验的实验用例。通过设计如下的实验用例，可以验证本文提出的基于大模型的深层 Web 越权漏洞检测方法在检测深层水平越权漏洞、深层垂直越权漏洞方面对比目前现有的检测方法的优势。

表 2　实验测试用例表

实验模式	实验用例	用例编号
纯人工检测模式	访问高权限接口，能否检出垂直越权漏洞	1 – 1
	访问同权限接口（带请求参数），能否检出水平越权漏洞	1 – 2
	访问同权限接口（不带请求参数），能否检出水平越权漏洞	1 – 3
BurpSuite 插件（Autorize）辅助检测模式	访问高权限接口，能否检出垂直越权漏洞	2 – 1
	访问同权限接口（带请求参数），能否检出水平越权漏洞	2 – 2
	访问同权限接口（不带请求参数），能否检出水平越权漏洞	2 – 3
基于大模型的深层 Web 越权漏洞检测模式（AiForAccess）	访问高权限接口，能否检出垂直越权漏洞	3 – 1
	访问同权限接口（带请求参数），能否检出水平越权漏洞	3 – 2
	访问同权限接口（不带请求参数），能否检出水平越权漏洞	3 – 3

3.3　实验结果与分析

图 4 所示为本实验所设计的 AiForAccess 越权漏洞检测工具，该工具会自动修改请求，通过 MCP 调用重放后分析并对比响应，自动检测出是否存在越权漏洞，并输出相应的检测详情。

通过上述设计的 AiForAccess 越权漏洞检测工具以及对照组运行实验用例，得到的实验结果如表 3 所示。

图 4　AiForAccess 越权漏洞检测工具示意图

表 3　实验测试结果表

用例编号	预期结果	实验结果	耗时/s
1 – 1	检出	检出	35
1 – 2	检出	检出	31
1 – 3	检出	检出	41
2 – 1	检出	检出	13
2 – 2	检出	未检出	16
2 – 3	检出	未检出	11
3 – 1	检出	检出	17
3 – 2	检出	检出	24
3 – 3	检出	检出	24

综上实验结果可以看出，AiForAccess 越权漏洞检测工具在检测准确率、耗时方面对比传统检测手段存在明显优势。

4　结束语

本文通过将 AI 大模型技术、MCP 技术融入传统被动式越权漏洞检测方法，提出了一种更为自动化、准确率更高的 Web 越权漏洞检测方法，并借助 AI 大模型的数据识别与分析能力，实现了对深层越权漏洞的挖掘。实验验证了该方法的正确性与可行性，表明其在提升检测效率与覆盖范围方面具有实用价值。

同时，本研究也存在一定的局限性。例如，实验所采用的数据集和测试环境规模有限，未能涵盖所有可能的业务场景与复杂权限模型；此外，方法在实时性与系统资源消耗方面的优化仍有进一步提升空间。

未来工作将围绕扩展实验验证范围、优化算法响应效率以及探索在多阶段渗透测试中的自动化集成展开，以增强该方法的适用性与稳定性。

参考文献

[1] 张文潇，苏新，顾依凌. 基于深度强化学习模型融合的海洋气象传感器网络入侵检测方法［J］. 物联网学报，2025，9（1）：89－102.

[2] 王海波，王其文，郭燕，等. 面向攻防对抗的自动化漏洞利用研究综述［J］. 网络空间安全科学学报，2025，3（3）：38－56.

[3] 冯景瑜，潘濛，王佳林，等. 基于深度语义解析的 API 越权漏洞攻击主动防御方法［J］. 信息网络安全，2025，25（6）：933－942.

[4] 毕超. 金融业 AI 大模型的安全风险及应对［C］//2024 世界智能产业博览会人工智能安全治理主题论坛，2024：279－281.

[5] 宋虹，马俊龙，王伟平，等. 基于响应相似性判定的 Web 越权漏洞测试方法［J］. 信息安全学报，2025，10（2）：17－29.

[6] 李守伟，张嘉政，何海波，等. 基于区块链的大模型数据监管体系设计［J］. 信息安全研究，2025，11（8）：682－693.

[7] 刘井强，田星，舒钰淇，等. 大模型赋能软件供应链开发环节安全研究综述［J］. 信息安全学报，2024，9（5）：87－109.

[8] 马勇，罗森林，吴云坤，等. 提升网络安全软件中预训练大模型推理速度的研究［C］// 第 38 次全国计算机安全学术交流会，2023：207－210.

[9] 苏艳芳，袁静，薛俊民. 大模型安全评估体系框架研究［C］//第 39 次全国计算机安全学术交流会，2024：107－111.

[10] 吴佩泽，李光辉，吴津宇. 基于大语言模型的电力监控系统资产脆弱性管理技术研究［C］//2024 世界智能产业博览会人工智能安全治理主题论坛，2024：243－247.

[11] 赵川 徐雁飞. 一种越权漏洞攻击方法实例研究［J］. 信息安全研究，2019，5（3）：248－252.

[12] 曹骏，向尕，任亚唯，等. 基于大模型的少样本 APT 攻击事件抽取方法［J］. 信息网络安全，2025，25（9）：1338－1347.

（收稿日期：2025－12－25）

作者简介：

覃锦端（1997－），男，本科，工程师，主要研究方向：网络安全、数据安全、漏洞挖掘。

尉雯雯（1984－），女，本科，高级工程师，主要研究方向：数据安全、医院信息系统。

王月兵（1995－），通信作者，男，本科，高级工程师，主要研究方向：网络安全、数据安全、黑客攻防。E-mai：wangyuebing@ mchz. com. cn。

版权声明

凡《网络安全与数据治理》录用的文章，如作者没有关于汇编权、翻译权、印刷权及电子版的复制权、信息网络传播权与发行权等版权的特殊声明，即视作该文章署名作者同意将该文章的汇编权、翻译权、印刷权及电子版的复制权、信息网络传播权与发行权授予本刊，本刊有权授权本刊合作数据库、合作媒体等合作伙伴使用。同时，本刊支付的稿酬已包含上述使用的费用，特此声明。

《网络安全与数据治理》编辑部