Behavioral Detection Methods for Automated MCP Server Vulnerability Assessment

Christian Coleman
Old Dominion University

Behavioral Detection Methods for Automated MCP Server Vulnerability Assessment

Christian Coleman

Old Dominion University COVA CCI Cybersecurity Undergraduate Research Program

9/18/2025

Abstract

The Model Context Protocol (MCP) has emerged as a critical standard for connecting AI agents to external data sources and tools. Still, its adoption has introduced significant security vulnerabilities across multiple attack surfaces. While recent research has catalogued extensive vulnerability taxonomies and attack implementations, automated detection methodologies remain limited. Current detection tools primarily employ static code analysis, which fails to identify behavioral vulnerabilities that only manifest during runtime server interactions. This study explores behavioral detection approaches for identifying MCP server vulnerabilities through systematic query-based testing, with particular emphasis on context manipulation techniques. Preliminary analysis of existing vulnerability research reveals 48 distinct attack types across four primary attack surfaces. The proposed methodology employs context-varied querying where identical requests are framed differently to expose inconsistent security behaviors. Using existing MCP security frameworks (MCPLIB and MCPSECBENCH), this research will evaluate the types of vulnerabilities that can be detected through behavioral analysis and assess the feasibility of automating these detection methods. This research addresses a critical gap in MCP security by developing systematic approaches to proactive vulnerability identification in deployed MCP server environments.

Introduction

The Model Context Protocol (MCP) represents a paradigm shift in AI agent architecture, enabling seamless integration between large language models and external tools and data sources. Developed by Anthropic as an open standard, MCP has rapidly gained adoption across major AI platforms, including Claude Desktop, OpenAI, and Cursor, facilitating more capable and versatile AI assistants (Anthropic, 2025). However, this increased connectivity and functionality have simultaneously expanded the attack surface for AI systems, introducing novel security challenges that traditional cybersecurity approaches struggle to address.

Recent comprehensive studies have identified extensive vulnerability landscapes within MCP ecosystems. Guo et al. (2025) catalogued 31 distinct attack types through their MCP Attack Library (MCPLIB), demonstrating practical implementations across four major categories: direct tool injection, indirect tool injection, malicious user attacks, and LLM inherent attacks. Concurrently, Yang et al. (2025) developed MCPSECBENCH, a systematic security benchmark that identified 17 attack types across four primary attack surfaces, revealing that over 85% of attacks successfully compromise at least one major MCP platform.

Despite these advances in understanding MCP vulnerabilities, current detection methodologies remain predominantly reactive and manual. Existing security tools focus

primarily on static code analysis or signature-based detection, approaches that prove insufficient for identifying behavioral vulnerabilities that only manifest during runtime interactions with MCP servers. The dynamic and context-dependent nature of many MCP attacks—particularly those involving tool poisoning, indirect prompt injection, and cross-platform behavioral inconsistencies—necessitates detection approaches that can assess server behavior through systematic interaction rather than code inspection alone.

This research addresses the critical gap between vulnerability identification and automated detection by proposing behavioral detection methods specifically designed for MCP server environments. The framework employs systematic querying techniques, with particular focus on context manipulation approaches, to probe server responses and identify anomalous behaviors indicative of security vulnerabilities. This provides a proactive approach to MCP security assessment that complements existing static analysis tools.

Existing Research and Detection Methods

The security implications of MCP adoption have become increasingly apparent as deployment scales across enterprise and consumer environments. The protocol's architecture, while enabling powerful AI agent capabilities, introduces multiple attack vectors that traditional security frameworks were not designed to address (Narajala & Habler, 2025). Recent research has comprehensively documented these vulnerabilities while revealing significant limitations in current detection methodologies.

Guo et al. (2025) provided the most comprehensive analysis of MCP attack methodologies to date, implementing 31 distinct attack types within their MCPLIB framework. Their research revealed critical insights into MCP agent behavior, including agents' blind reliance on tool descriptions, sensitivity to file-based attacks, and vulnerability to chain attacks exploiting shared context. The study demonstrated that file-based operations, which typically execute without user confirmation, present particularly high-risk attack vectors. Their quantitative analysis showed that file-based injection attacks achieved nearly 100% success rates across all tested platforms, while prompt-based attacks showed significant variation depending on platform implementation.

Yang et al. (2025) complemented this work through MCPSECBENCH, systematically evaluating attack effectiveness across three major MCP platforms. Their findings revealed significant variations in platform vulnerability, with Claude Desktop showing superior resistance to prompt injection attacks (0% attack success rate) compared to OpenAI (66.7%) and Cursor (100%). This platform-specific vulnerability variation suggests that detection approaches must account for implementation differences across MCP hosts. Their research also demonstrated that core protocol and implementation vulnerabilities affect all platforms universally, while prompt-based and tool-centric attacks exhibit considerable variability.

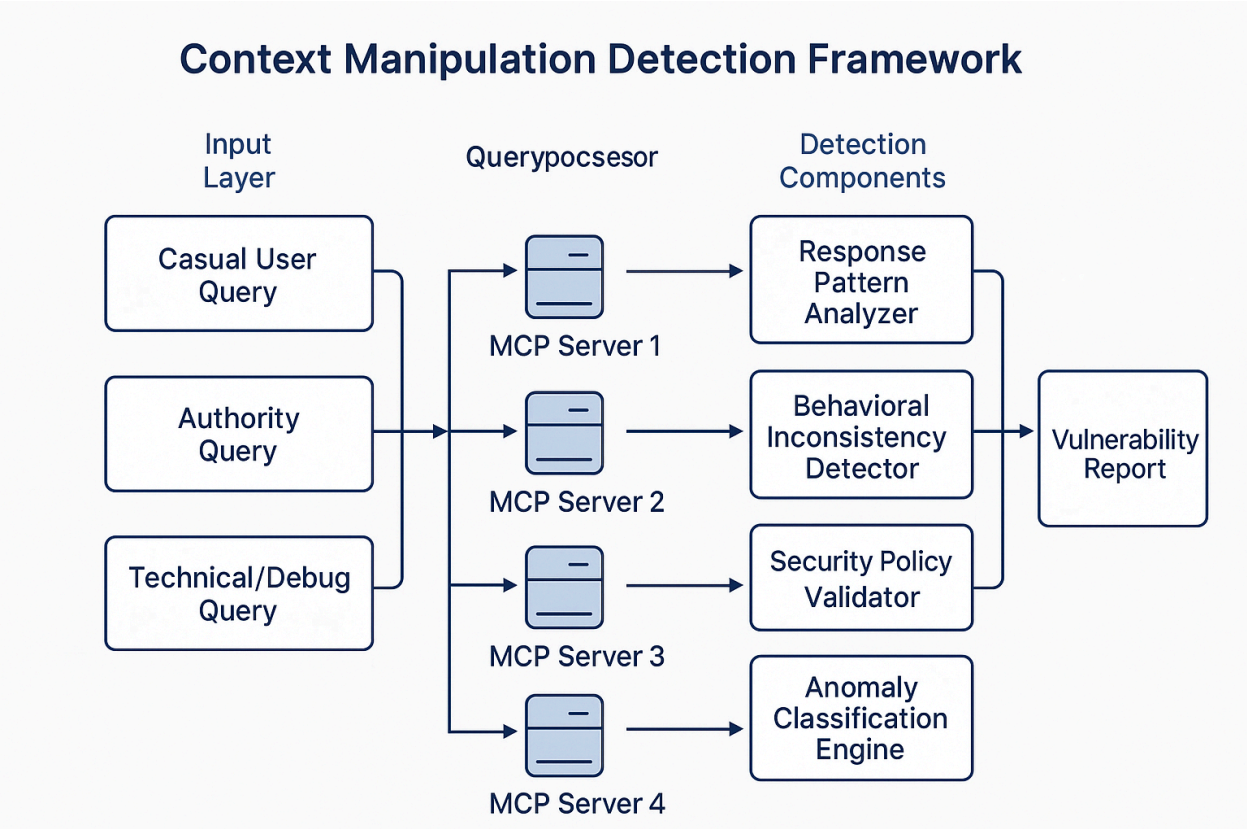
Despite comprehensive vulnerability identification, existing MCP security tools primarily employ static analysis or signature-based detection methods. Invariant Labs' MCP-Scan focuses on detecting tool poisoning attack features through predefined patterns, scanning tool descriptions for malicious instructions (Invariant Labs, 2025). While effective for known attack signatures, this approach cannot identify novel attack variations or behavioral inconsistencies

that emerge during runtime. Tencent's AI-Infra-Guard employs reasoning-based frameworks to analyze correlations between MCP services and security risks, using a ReAct-like framework to generate security analysis reports (Tencent, 2025). However, this approach remains largely theoretical without practical validation of detection accuracy across diverse vulnerability types.

Kumar et al. (2025) proposed MCP Guardian as a security-first middleware layer, implementing authentication, rate limiting, and Web Application Firewall (WAF) protections. While valuable for prevention, such approaches do not address the fundamental challenge of identifying vulnerabilities in existing MCP server deployments. The middleware approach assumes proper configuration and cannot detect vulnerabilities in the servers themselves. Radosevich and Halloran (2025) explored broader attack spectrums, including command execution and credential theft through their MCP Safety Audit, but their work focused on demonstrating attack feasibility rather than developing systematic detection methodologies.

Behavioral analysis has proven effective in other cybersecurity domains for identifying threats that evade signature-based detection. Anomaly detection systems in network security analyze traffic patterns to identify deviations from normal behavior, successfully detecting zero-day attacks and insider threats. Similarly, behavioral authentication systems identify users based on interaction patterns rather than static credentials. The concept of context-based security testing has precedent in web application security, where fuzzing techniques vary input formats to identify parsing vulnerabilities. However, the application of systematic context manipulation to AI agent security represents a novel approach not yet explored in existing literature for MCP systems. This gap between comprehensive vulnerability identification and practical automated detection methodologies motivates the proposed behavioral detection framework.

Methodology



The detection framework operates through a multi-component architecture that processes queries across multiple MCP servers simultaneously to identify behavioral vulnerabilities. As illustrated in Figure 1, the system begins with a Query Processor that transforms identical functional requests into three contextual variations—casual user, authority, and technical/debug formats. These contextually varied queries are then distributed to multiple MCP servers under assessment, enabling comparative analysis across different server implementations and configurations.

The detection process relies on four core components that analyze server behaviors rather than code structure. The Response Pattern Analyzer captures and compares server outputs across all three query contexts, documenting variations in disclosed information, tool selections, and permission requests. The Behavioral Inconsistency Detector identifies deviations in security enforcement, flagging instances where servers respond differently based on perceived query authority rather than actual permissions. The Security Policy Validator examines whether servers maintain consistent access control boundaries regardless of contextual framing, revealing configuration vulnerabilities and insufficient enforcement mechanisms. Finally, the Anomaly Classification Engine categorizes detected inconsistencies according to vulnerability type and severity, enabling prioritized remediation efforts.

This component-based approach enables the detection of several critical vulnerability categories. Tool poisoning vulnerabilities manifest when servers select different tools for functionally identical requests across contexts, indicating manipulation susceptibility. Data exfiltration risks appear when servers inappropriately disclose sensitive configuration details, system information, or credentials based on query framing rather than authentication status. Permission boundary failures occur when servers bypass or inconsistently enforce access controls depending on contextual authority signals. Configuration drift vulnerabilities emerge through inconsistent security policy application across identical functional requests, revealing implementation weaknesses.

The framework's strength lies in its focus on observable server behaviors during runtime interaction rather than static code analysis. Unlike existing detection approaches that examine server implementations for known vulnerability patterns, this methodology identifies security flaws through systematic behavioral testing. By querying multiple servers simultaneously with contextually-varied requests, the framework can detect both server-specific vulnerabilities and systemic weaknesses across MCP implementations, providing comprehensive security assessment capabilities that complement traditional static analysis tools.

Conclusion

This research has examined the critical gap between MCP vulnerability identification and automated detection methodologies. Through a comprehensive analysis of 48 documented vulnerability types across recent MCP security research, this study has demonstrated that current detection approaches—primarily based on static code analysis and signature matching—prove insufficient for identifying behavioral vulnerabilities that only manifest during runtime server interactions.

The proposed context manipulation detection framework addresses this limitation by introducing systematic behavioral testing as a complementary detection methodology. By presenting functionally identical queries in varying contextual frames and analyzing response consistency, this approach can identify security vulnerabilities that evade traditional detection methods. The framework's foundation in behavioral analysis principles from other cybersecurity domains, combined with its specific adaptation to MCP architecture characteristics, provides a theoretically sound basis for practical implementation.

In conclusion, while static analysis and signature-based detection remain valuable tools in the MCP security toolkit, they are insufficient on their own to address the full spectrum of runtime behavioral vulnerabilities. The context manipulation detection framework proposed in this research offers a complementary approach that enhances security assessment capabilities through systematic behavioral testing. By bridging the gap between vulnerability identification and automated detection, this methodology supports the development of more secure and resilient AI agent ecosystems. As MCP adoption continues to expand, the integration of behavioral detection methods into standard security practices will become increasingly critical for maintaining the integrity and trustworthiness of AI-powered systems.

References

Anthropic. (2025). Introducing the Model Context Protocol. Retrieved from https://www.anthropic.com/news/model-context-protocol

Guo, Y., Liu, P., Ma, W., Deng, Z., Zhu, X., Di, P., ... & Wen, S. (2025). Systematic Analysis of MCP Security. arXiv preprint arXiv:2508.12538.

Invariant Labs. (2025). MCP security notification: Tool poisoning attacks. Retrieved from https://invariantlabs.ai/blog/mcp-security-notification-tool-poisoning-attacks

Kumar, S., Girdhar, A., Patil, R., & Tripathi, D. (2025). MCP Guardian: A Security-First Layer for Safeguarding MCP-Based AI System. arXiv preprint arXiv:2504.12757.

Narajala, V. S., & Habler, I. (2025). Enterprise-Grade Security for the Model Context Protocol (MCP): Frameworks and Mitigation Strategies. arXiv preprint arXiv:2504.08623.

Radosevich, B., & Halloran, J. (2025). MCP Safety Audit: LLMs with the Model Context Protocol Allow Major Security Exploits. arXiv preprint arXiv:2504.03767.

Tencent. (2025). AI-Infra-Guard: AI infrastructure security framework. Retrieved from https://github.com/Tencent/AI-Infra-Guard

Yang, Y., Wu, D., & Chen, Y. (2025). MCPSECBENCH: A Systematic Security Benchmark and Playground for Testing Model Context Protocols. arXiv preprint arXiv:2508.13220.