A Critical Security and Architectural Review of the Model Context Protocol (MCP) Ecosystem

Yusuf Talha ARABACI
M.Sc. Student in Software Engineering
Karabuk University
Karabuk, Turkey

***Abstract*—The transition of Large Language Models (LLMs) from passive text generators to active, tool-using agents has been hindered by a persistent interoperability crisis: the "$N \times M$" integration barrier. The Model Context Protocol (MCP), introduced in late 2024, attempts to resolve this by standardizing the interface between AI agents and external data. This paper provides a critical architectural synthesis of the MCP ecosystem, grounded in an empirical analysis of 1,899 deployed servers. We argue that while decoupling intelligence from data sources is architecturally sound, it exposes a "Lethal Trifecta" of security risks—unrestricted data access, internet connectivity, and autonomous action—that current governance models fail to address. Our review of benchmarks, including MCPGAUGE and LiveMCP-101, identifies a significant "Protocol-Behavior Mismatch," where frontier models frequently fail to comply with tool-use directives despite correct protocol implementation. We conclude by analyzing the industry's shift toward a "Code Execution Paradigm" to mitigate context bloat and propose a tiered defense strategy utilizing mandatory sandboxing and information flow control.**

***Keywords*—Model Context Protocol, MCP, context bloat, code execution paradigm, agentic security, lethal trifecta, multi-agent systems.**

I. Introduction: The Interoperability Crisis

The scalability of "agentic AI"—systems capable of planning, reasoning, and acting—has long been constrained by a fundamental infrastructure deficit. While LLMs have achieved remarkable linguistic fluency, their ability to interact with external environments has remained brittle. The core challenge was not a lack of intelligence, but the absence of a pervasive communication standard. Connecting any given model (N) to distinctive tools (M) historically required bespoke integrations, trapping developers in an unmanageable "$N \times M$" maintenance cycle.

Introduced in late 2024, the Model Context Protocol (MCP) emerged as the industry's answer to this fragmentation. Analogous to a "USB-C port" for the AI layer, MCP provides a universal standard for connectivity, backed by Anthropic, OpenAI, and Google. This standardization has rapidly shifted the development paradigm from stateless, ephemeral API calls to stateful, persistent agent sessions.

Yet, this solution precipitates a new class of vulnerabilities. Empowering AI agents to query databases, manipulate files, and execute code introduces a "Lethal Trifecta" of risks: access to sensitive Data, connection to the public Internet, and the authority to take Action. Recent audits reveal that 66% of open-source MCP implementations exhibit structural "code smells" and 5.5% contain high-severity tool poisoning vectors. This paper contends that while MCP is an architectural necessity, it currently lacks the requisite security maturity. We must move beyond the novelty of connection to the rigor of governance.

II. Methodology: Architectural Design

MCP creates a modular ecosystem by strictly decoupling the "Client/Host" (the reasoning engine) from the "Server" (the execution context). This architecture relies on specific primitives and negotiation protocols.

A. Connection and Negotiation

Unlike stateless REST APIs, an MCP session relies on a persistent handshake. The Host application (e.g., an IDE or Chat Interface) initializes an **MCP Client**, which establishes a connection to a standalone **MCP Server**. Upon connection, a *Capability Negotiation* phase occurs. The Client and Server exchange supported feature sets—such as "sampling" (the server querying the LLM) or "prompts" (standardized templates)—ensuring that agents only discover tools they are architecturally capable of wielding.

B. Protocol Primitives

The protocol organizes interaction into three distinct abstractions:

1) **Tools (Execution):** Functions that alter system state (e.g., `INSERT INTO users...`). These represent the highest risk surface.
2) **Resources (Context):** Read-only data streams (e.g., logs, file contents) that provide context without side effects.
3) **Prompts (Guidance):** Pre-defined templates that structure agent behavior for specific workflows, such as code review or data analysis.

C. Transport Layer Implications

Security posture is heavily influenced by the transport selection:

- **Stdio (Local):** The server operates as a subprocess of the host. Data remains within the local memory space, offering implicit isolation and high throughput.
- **HTTP/SSE (Remote):** Essential for distributed architectures. The client sends commands via HTTP POST, receiving asynchronous updates via Server-Sent Events (SSE). While scalable, this introduces

significant latency overhead and expands the attack surface to network-based vectors.

III. Analysis: Performance & Efficiency

The integration of external tools creates a secondary challenge: efficiency. The ecosystem is currently grappling with “Context Bloat,” where the comprehensive description of available tools overwhelms the model’s cognitive window.

A. The Cost of Context

Providing an agent with granular control requires feeding it detailed JSON schemas for every available tool. At enterprise scale, this approach floods the context window with definitions, diluting the signal of the user’s actual query. Empirical data suggests this can increase input token costs by a factor of $236\times$. More critically, it creates a "Lost in the Middle" effect, where models struggle to retrieve the correct tool definition from a saturated context.

B. The Code Execution Shift

To mitigate this, the industry is pivoting toward a “Code Execution Paradigm.” Rather than making serial API calls (the “Direct Tool Calling” method), models are increasingly trained to write and execute scripts (e.g., Python). This shift offers dramatic efficiency gains. Instead of multiple round-trips to filter a dataset, the agent writes a single script to fetch and process the data in-memory. Benchmarks indicate this approach can reduce token consumption by **98%** and significantly lower latency, though it necessitates a robust sandboxing infrastructure.

IV. Security Assessment: The Lethal Trifecta

The convergence of capabilities in MCP agents creates a threat landscape distinct from traditional web applications. We categorize these risks as the “Lethal Trifecta”: **(1) Private Data Access**, **(2) Internet Connectivity**, and **(3) Action Authority**.

A. Threat Vectors

1) **Indirect Prompt Injection (IPI):** Attackers exploit the agent’s ability to read external content. By embedding malicious, invisible instructions in a webpage, an attacker can hijack the agent’s control flow. The agent, treating the webpage as context, executes the attacker’s commands—potentially exfiltrating data or corrupting files.
2) **Tool Poisoning:** This represents a supply chain attack on the agent’s capabilities. If an attacker publishes a compromised MCP server, it acts as a trojan horse. Research identified a **5.5%** poisoning risk in the current open-source ecosystem.
3) **Sampling Manipulation:** A compromised server can return falsified data or conversation history during a sampling request, effectively gaslighting the model into making incorrect decisions.

B. Defense Strategy

Perimeter defense is insufficient for agentic systems. We propose a layered "Defense-in-Depth" model:

- **Tiered Sandboxing:** Servers must be isolated based on risk. Basic tools may run in containers, but Code Execution endpoints require microVM isolation (e.g., Firecracker) to ensure kernel-level separation.
- **Taint Tracking:** Information Flow Control (IFC) systems must tag data originating from untrusted sources (e.g., the web) as "tainted," preventing its use in sensitive sinks (e.g., file deletion) without explicit sanitization.

V. Empirical Evaluation & Gaps

Synthesizing data from recent benchmarks clarifies the current limitations of the ecosystem.

A. The Reality Gap

Frameworks such as **MCPGAUGE** and **LiveMCP-101** reveal a discrepancy between capability and reliability.

- **Performance Reality:** In real-world scenarios, even frontier models (e.g., GPT-4o, Claude 3.5 Sonnet) often achieve success rates **below 60%** for complex, multi-step tasks.
- **Initiative Deficit:** Models frequently exhibit passivity, failing to utilize available tools unless explicitly prompted.
- **Compliance Failures:** There is a persistent issue where models hallucinate tool parameters or ignore schema constraints, despite a valid protocol handshake.

B. Research Priorities

Four areas require urgent academic attention:

1) Protocol-Behavior Mismatch: The protocol is syntactically robust, but the agents are semantically fragile. Future work must bridge the gap between providing a tool and ensuring the model understands *when* to use it.

2) Secure Runtimes: As Code Execution becomes standard, the burden of security shifts to the runtime environment. Standardized, verified sandboxing protocols are required.

3) Trust Registry: The ecosystem lacks a mechanism for trust. A centralized "Registry of Trust," similar to SSL certificate authorities, is needed to validate MCP servers.

4) Domain Extensions: Generic protocols are insufficient for regulated industries. Extensions enforcing HIPAA or GDPR compliance at the protocol level are necessary for adoption in healthcare and finance.

VI. Future Outlook

The trajectory of MCP points toward Multi-Agent Systems (MAS). The **AgentX** pattern illustrates this evolution, where MCP serves as the connective tissue for specialized agent teams—Planners, Researchers, and Executors—working in concert. We envision a hybrid future where high-performance backends remain on gRPC/REST, while the dynamic, adaptive layer of agent interaction standardizes on MCP.

VII. Conclusion

The Model Context Protocol has successfully transformed the architectural landscape, enabling agents to act as genuine digital colleagues rather than isolated processors. However, the empirical evidence—highlighting low reliability rates and substantial security risks—demonstrates that the ecosystem is still in a volatile infancy. The path forward lies not in expanding features, but in deepening governance. Adopting standards like ISO 42001 and enforcing secure-by-default architectures are not merely best practices; they are existential requirements for the safe deployment of agentic AI.

Kaynakça

[1] X. Hou, Y. Zhao, S. Wang, and H. Wang, "Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions," *arXiv preprint arXiv:2503.23278*, 2025.

[2] N. Krishnan, "Advancing Multi-Agent Systems Through Model Context Protocol: Architecture, Implementation, and Applications," *arXiv preprint arXiv:2504.21030*, 2025.

[3] A. Ehtesham, A. Singh, G. K. Gupta, and S. Kumar, "A Survey of agent interoperability protocols: MCP, ACP, A2A, ANP," *arXiv preprint arXiv:2505.02279*, 2025.

[4] M. M. Hasan et al., "Model Context Protocol (MCP) at First Glance: Studying the Security and Maintainability of MCP Servers," *arXiv preprint arXiv:2506.13538*, 2025.

[5] M. Flotho et al., "MCPmed: A Call for MCP-Enabled Bioinformatics Web Services for LLM-Driven Discovery," *arXiv preprint arXiv:2507.08055*, 2025.

[6] M. Mastouri, E. Ksontini, and W. Kessentini, "Making REST APIs Agent-Ready: From OpenAPI to MCP Servers for Tool-Augmented LLMs," *arXiv preprint arXiv:2507.16044*, 2025.

[7] S. Fan et al., "MCPToolBench++: A Large Scale AI Agent Model Context Protocol MCP Tool Use Benchmark," *arXiv preprint arXiv:2508.07575*, 2025.

[8] W. Xing et al., "MCP-Guard: A Defense Framework for Model Context Protocol Integrity in Large Language Model Applications," *arXiv preprint arXiv:2508.10991*, 2025.

[9] W. Song et al., "Help or Hurdle? Rethinking Model Context Protocol-Augmented Large Language Models," *arXiv preprint arXiv:2508.12566*, 2025.

[10] Z. Luo et al., "MCP-Universe: Benchmarking Large Language Models with Real-World Model Context Protocol Servers," *arXiv preprint arXiv:2508.14704*, 2025.

[11] M. Yin et al., "LiveMCP-101: Stress Testing and Diagnosing MCP-enabled Agents on Challenging Queries," *arXiv preprint arXiv:2508.15760*, 2025.

[12] G. Chhetri et al., "Model Context Protocols in Adaptive Transport Systems: A Survey," *arXiv preprint arXiv:2508.19239*, 2025.

[13] S. S. K. A. Tokal et al., "AgentX: Towards Orchestrating Robust Agentic Workflow Patterns with FaaS-hosted MCP Services," *arXiv preprint arXiv:2509.07595*, 2025.

[14] P. He et al., "Automatic Red Teaming LLM-based Agents with Model Context Protocol Tools," *arXiv preprint arXiv:2509.21011*, 2025.

[15] L. Coppolino et al., "Asset Discovery in Critical Infrastructures: An LLM-Based Approach," *Electronics 14(16):3267*, 2025.

[16] N. Bhandarwar, "Integrating Generative AI and Model Context Protocol (MCP) with Applied Machine Learning for Advanced Agentic AI Systems," *Int. J. of Computer Trends and Technology*, 2025.

[17] A. Singh et al., "A Survey of the Model Context Protocol (MCP): Standardizing Context to Enhance Large Language Models (LLMs)," *Preprints 202504.0245*, 2025.

[18] A. Korinek, "AI Agents for Economic Research," *NBER Working Paper 34202*, 2025.