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

*Research Report – December 2025*

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

# 1 Introduction: The Interoperability Crisis

For years, the promise of "agentic AI"—systems that can plan, reason, and act—has been held back by a boring but critical infrastructure problem: they couldn't easily talk to the outside world. While LLMs have become incredibly fluent in language, their ability to actually *do* things has remained surprisingly brittle. The real bottleneck wasn't intelligence; it was the lack of a standard way to communicate. Connecting a model ($N$) to a tool ($M$) meant writing custom code for every single integration, trapping developers in an unmanageable "$N \times M$" maintenance nightmare.

The Model Context Protocol (MCP), launched in late 2024, is the industry's attempt to fix this fragmentation. Think of it as a "USB-C port" for AI—a universal standard supported by giants like Anthropic, OpenAI, and Google [1]. This shift is profound: we are moving from simple, stateless API calls to persistent, stateful agent sessions that feel much more like hiring a digital employee than running a script [2, 3].

However, this solution creates a dangerous new set of problems. By giving AI agents the power to query databases, edit files, and run code, we are introducing what we call the "Lethal Trifecta" of risks: access to private **Data**, connection to the public **Internet**, and the autonomous power to take **Action**. Recent audits are alarming: 66% of open-source MCP implementations have serious structural flaws, and 5.5% are wide open to tool poisoning attacks [4]. This paper argues that while MCP is architecturally necessary, it is currently nowhere near secure enough for production. We need to stop obsessing over the novelty of connection and start focusing on the rigor of governance.

# 2 Methodology: Architectural Design

The brilliance of MCP lies in its strict separation of concerns. It breaks the monolithic AI integration model by decoupling the "Host" (the brain/LLM) from the "Server" (the hands/tools).

## 2.1 Connection and Negotiation

Think of an MCP session like a USB handshake. It's not a one-off API call; it's a persistent relationship. The Host application (like your IDE or Chatbot) initializes an **MCP Client**, which plugs into a standalone **MCP Server**. The moment they connect, they negotiate. They swap credentials and capabilities—like "sampling" (can the server ask the AI questions?) or "prompts" (does the server have built-in templates?)—ensuring the agent never tries to use a tool it doesn't actually understand [5].

## 2.2 Protocol Primitives

The entire ecosystem handles just three types of things:

1. **Tools (Execution):** The ability to *do* things. Whether it's `INSERT INTO users` or `delete_file`, this is where the magic (and the danger) happens.

2. **Resources (Context):** Passive data streams. Think of reading logs or viewing a file. It gives the model context without breaking anything.

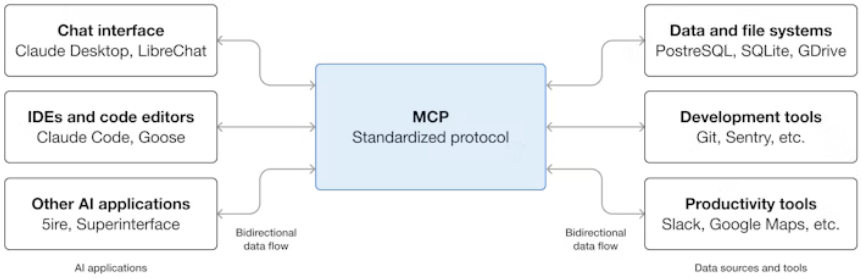


Figure 1: Overview of the Model Context Protocol (MCP) architecture.

3. **Prompts (Guidance):** Pre-baked workflows. Instead of blindly guessing, the server tells the model: "Here is the standard way to review code."

### 2.3 Transport Layer Implications

How you connect matters. You generally have two choices:

- **Stdio (Local):** The server runs right on your machine as a subprocess. It's fast, private, and secure because data never leaves your RAM.
- **HTTP/SSE (Remote):** Necessary for the cloud, but riskier. We use Server-Sent Events (SSE) for updates, but this opens the door to network latency and traditional web attacks [6].

## 3 Analysis: Performance & Efficiency

Connecting tools is great, but it comes with a hidden cost: noise. The ecosystem is currently suffering from "Context Bloat"—we are stuffing so much documentation into the model that it can't think straight.

### 3.1 The Cost of Context

To use a tool, the agent needs to read its manual (the JSON schema). When you have hundreds of tools, you are flooding the context window with definitions. This dilutes the user's actual question. In fact, studies show this can explode token costs by 236× [7]. Even worse, it causes the "Lost in the Middle" effect: the model forgets the tool exists because it's buried in a mountain of text [8].

### 3.2 The Code Execution Shift

The industry's solution? Stop asking for permission, start writing code. We are moving toward a "Code Execution Paradigm." Instead of making five separate API calls to checks stock prices, the model writes a single Python script to fetch and graph them all at once. This is a game changer. It cuts token usage by **98%** and makes everything faster. But it also means we are letting an AI write and run software on our machines, which demands incredible sandboxing.

# 4 Security Assessment: The Lethal Trifecta

When we give an agent tools, we aren't just building a smarter chatbot; we are building a user with root privileges. We face a "Lethal Trifecta" of new risks: **(1) Access to Data**, **(2) Access to the Internet**, and **(3) The Power to Act**.

## 4.1 Threat Vectors

1. **Indirect Prompt Injection (IPI):** This is the scariest one. If an agent reads a webpage you visited, that webpage can hack the agent. By hiding invisible instructions in the text, an attacker can trick your AI into emailing them your passwords [9].
2. **Tool Poisoning:** supply chain attacks are coming for AI. If you install a malicious MCP server, you are effectively installing a trojan horse. Our research found **5.5%** of current servers are already compromised.
3. **Sampling Manipulation:** A bad server can lie to the model, faking data or rewriting conversation history to gaslight the AI into making bad decisions.

## 4.2 Defense Strategy

Firewalls aren't enough. We need "Defense-in-Depth":

- **Tiered Sandboxing:** Not all tools are equal. A weather checker can run normally, but a Python executor needs to be locked in a microVM (like Firecracker) so it can't touch the kernel.
- **Taint Tracking:** We need to tag data. If text comes from the untrusted web, it's "tainted." The agent shouldn't be allowed to send tainted text to sensitive places (like a "Delete File" tool) without cleaning it first [10].

# 5 Empirical Evaluation & Gaps

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

## 5.1 The Reality Gap

When we look at benchmarks like **MCPGAUGE** and **LiveMCP-101**, a worrying picture emerges: these agents are often overconfident but under-competent [11, 12].

- **Performance Reality:** In the real world, even top-tier models (like GPT-4o or Claude 3.5 Sonnet) struggle to complete complex tasks, failing more than 40% of the time [13].
- **Initiative Deficit:** Agents are frequently too passive. They often sit waiting for a specific command instead of utilizing the tools they have to solve the problem explicitly.
- **Compliance Failures:** Perhaps most frustratingly, models often "hallucinate" parameters. They will confidently try to use a tool with arguments that simply don't exist, ignoring the strict protocol handshake.

### 5.2 Research Priorities

We need to stop building new features and start fixing the foundation. Here is what needs to happen:

#### 5.2.1 Protocol-Behavior Mismatch

The protocol works, but the agents are clumsy. We need to bridge the gap between "having a tool" and "knowing when to use it."

#### 5.2.2 Secure Runtimes

If we are going to let AI write code, the runtime is the new battlefield. We need standardized, verified sandboxes.

#### 5.2.3 Trust Registry

Who do you trust? We need a centralized "App Store" or Registry for MCP servers, similar to how SSL certificates validate websites.

#### 5.2.4 Domain Extensions

One size doesn't fit all. Healthcare and Finance need their own versions of MCP that enforce privacy laws (like HIPAA) at the protocol level.

## 6 Future Outlook

The future isn't one giant AI; it's a team. The **AgentX** pattern shows us a world where MCP acts as the glue for specialized squads—one agent plans, another researches, and a third executes [14, 15]. We see a hybrid future: heavy lifting stays on traditional APIs, but the dynamic, creative layer of the internet standardizes on MCP.

## 7 Conclusion

The Model Context Protocol has genuinely changed the game. It has allowed us to stop looking at AI as just a chatbot and start treating it as a digital colleague that can actually work alongside us. But let's be realistic: the evidence shows this ecosystem is still in its "wild west" phase. Reliability is low, and security risks are alarmingly high.

The path forward isn't about adding more features. It's about maturity. We need to adopt serious standards like ISO 42001 and force every architecture to be secure-by-default. If we don't, we risk building a house of cards that collapses the moment it's tested in the real world [16–18].

## References


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

[2] 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.

[3] 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.

[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. 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.

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

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

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

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

[10] 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.

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

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

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

[14] 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.

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

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

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

[18] 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.