ChatGPT

Shortfalls of the Model Context Protocol (MCP) in Production

Overview

The Model Context Protocol (MCP) is a proposed standard for connecting large-language-model (LLM) assistants to external tools and data sources. It advertises an **"agentic"** workflow where an LLM can plan tasks and call tools via HTTP. Proponents portray MCP as a way to build autonomous agents, but industry analyses and early experiments reveal significant shortcomings that limit its suitability for real-world, enterprise-grade systems. The issues include lack of observability and auditability, insecure design, poor performance and scalability, complexity for developers, non-determinism in LLM use, and challenges to enterprise adoption and governance. The following sections synthesize criticisms from technical blogs, security analyses, industry experts and peer-reviewed research.

1. Limited observability and auditability

- **Black-box interactions** – MCP uses a streaming protocol based on server-sent events (SSE). The SSE channel makes interactions stateful, keeping the socket open until an LLM task completes. Analysts note that this breaks the stateless nature of REST APIs and prevents observability of each step in the LLM's reasoning process; external auditors cannot reliably replay or inspect the prompts, tool calls and responses, which are essential for compliance and debugging [1] [2].
- **No transaction logs** – MCP does not provide a transaction log similar to a database's log. There is no standardized way to record the prompts, intermediate reasoning steps, and tool calls during a single inference. The absence of logs means that actions taken by the agent cannot be reconstructed, making it impossible to audit or roll back unintended behaviour. The multi-agent TRiSM review emphasizes that LLM-based agents lack "trust-and-audit modules" that trace the sequence of tool uses and decisions [3].

2. Security and privacy risks

2.1 Prompt injection and tool poisoning

- **Prompt injection** – Attackers can embed malicious instructions in messages that the LLM interprets. Security research shows that an MCP server compromised by **prompt injection** can cause the agent to send confidential files or perform unauthorized actions; because the tool definitions are included in the system prompt, any injection becomes part of the agent's internal context [4]. The TRiSM review notes that "prompt infection" can spread from one agent to another like a computer virus, causing cascading misbehavior [5].
- **Tool poisoning** – Malicious servers can alter tool definitions after user approval. An experiment demonstrated that changing a seemingly harmless "get_fact_of_the_day" tool allowed exfiltration of chat histories [6]. Tools can also be shadowed by similarly named malicious services, causing the LLM to invoke the wrong tool [7].

2.2 Token theft and identity spoofing

- **OAuth token theft** – MCP servers store OAuth tokens for services like email or cloud storage. If a server is compromised, an attacker can steal tokens and impersonate the user across multiple services, performing privileged operations that appear legitimate [8].
- **Spoofing and impersonation** – The TRiSM paper lists **spoofing** and **impersonation** as distinct threats in multi-agent systems: adversaries may fake credentials or mimic trusted agents to issue unauthorized commands [9]. Because MCP lacks integrated authentication and relies on external implementations, verifying the identity of the calling agent is difficult [10].

2.3 Privacy leakage and data aggregation

- **Breaking data silos** – MCP encourages agents to connect disparate services. This can inadvertently aggregate sensitive information across silos (e.g., HR data, private emails, financial records). Analysts warn that LLM agents can infer confidential company events by cross-referencing Slack conversations and internal documents [11]. The TRiSM paper likewise notes that multi-agent systems may share sensitive information via shared memory, increasing the risk of unauthorized disclosure [12].
- **Unscoped permissions** – Many MCP servers request broad scopes to access multiple services, making them prime targets for attackers. Pillar Security points out that stolen tokens can be used to send emails or delete cloud resources [8], and OWASP recommends implementing least-privilege scopes for each tool [13].

3. Performance and cost inefficiencies

- **Context window bloat** – MCP requires embedding tool definitions and metadata into the LLM's prompt. For each tool, the agent includes JSON schemas and descriptions. Analysts note that when many tools are loaded, the context can consume thousands of tokens, slowing responses and increasing cost [14] [15]. Shrivu Shankar's analysis found that state-of-the-art LLMs succeeded in only 16 % of tool-use tasks, partly because the context is too crowded for the model to pick the right tool [16].
- **High latency for large datasets** – Business analytics teams tested MCP connectors on datasets larger than 100 MB. The "deep research" mode took 10–30 minutes to return results, turning analytics into "archaeology" instead of real-time insights [17].
- **Token and infrastructure costs** – Multi-agent systems often require repeated reasoning cycles and iterative tool calls. Galileo Labs notes that medium-complexity agents consume 10–50× more tokens than a standard chat because they loop to refine responses, leading to runaway costs [18]. Additional evaluation and debugging steps (e.g., using large LLMs to judge outputs) also inflate compute expenses [19].

4. Architectural and design limitations

4.1 Stateful connections and scalability challenges

- **SSE vs. REST** – MCP's reliance on server-sent events means the server must maintain an open connection per request. CData's engineers warn that this stateful design complicates load balancing and consumes server resources; a single long-running call can tie up threads, hampering scalability

[1] . The design also conflicts with serverless environments: tests on AWS Lambda showed cold-start delays, complex logging, and infrastructure overhead [20] .

- **No native error handling or versioning** – MCP does not define how tools should report errors, how they are versioned, or how to update tool definitions safely. Developers must implement ad-hoc error handling and lifecycle management. Dmitry Degtyarev notes that different MCP servers handle errors inconsistently and there is no mechanism for tool versioning or deprecation [21] .

4.2 Lack of state and memory in single requests

- **Stateless interactions** – The MCP specification treats each call as independent; there is no mechanism for persistent state across calls. This means the agent cannot maintain long-term memory or context other than the information encoded in each prompt. Consequently, complex tasks requiring multi-step planning, like iterative document editing or multi-page reasoning, are difficult to implement and error-prone.

4.3 Reactive rather than proactive intelligence

- **No proactive analytics** – MCP agents only answer questions when asked. Narrative BI's evaluation found that because LLM agents cannot automatically monitor data or detect anomalies, they remain reactive; they cannot proactively alert users to trends or issues [22] .
- **Lack of ETL and data cleaning** – MCP assumes that data fed into the tool is already clean. Real business datasets contain inconsistent formats, duplicates and missing values; MCP lacks built-in extraction, transformation and loading (ETL) capabilities, forcing teams to build custom pipelines [23] .

5. Developer and organizational barriers

5.1 Technical complexity and immature tooling

- **Custom server requirement** – MCP does not provide a turnkey server implementation. Developers must build and host their own server with OAuth support, SSE streaming and tool registration. This requires significant expertise in networking, security and LLM prompting. Medium posts note that the early spec and limited documentation make development challenging [24] .
- **Lack of ecosystem maturity** – Because MCP is nascent, few tools exist, and the developer community is small [25] . Integrations with major enterprise platforms (like SAP or Salesforce) are mostly absent, forcing teams to build connectors themselves [26] .

5.2 Authorization and identity management gaps

- **Enterprise SSO friction** – The initial MCP authorization flow assumed that the MCP server acts as both resource and authorization server; enterprise identity providers rarely support the required dynamic client registration, making the flow incompatible with corporate SSO [27] . Later revisions decoupled the resource and authorization servers, but still rely on features many identity providers do not support, leaving implementers to build workarounds [28] .
- **Manual consent flows** – Enterprise administrators cannot centrally review or restrict what tools are being authorized. Users must individually grant third-party permissions, resulting in poor oversight [29] . This fosters "consent fatigue," where users blindly approve requests, increasing the risk of malicious tool approvals [30] .

5.3 Vendor lock-in and future uncertainty

- Early adopters risk being tied to a rapidly evolving spec. Experts caution that the MCP protocol could change significantly or be replaced by other standards, causing current implementations to become obsolete [31]. Without broad adoption, enterprises may avoid investing in MCP due to fear of being locked into a proprietary ecosystem. [25].

6. Real-world examples and case studies

6.1 Business data analysis

Narrative BI evaluated LLM agents on real company data. When connecting to large CSV or SQL datasets, the agents often failed to parse the data or took so long that the insights were irrelevant. Many common business platforms (e.g., ERP systems) lacked MCP connectors, requiring custom extraction and cleaning pipelines [26] [32].

6.2 Autonomous WhatsApp agent (tool poisoning)

Invariant Labs demonstrated a tool-poisoning attack against a chat application that used an autonomous agent. By subtly altering a tool's API implementation, the agent was tricked into retrieving and exfiltrating private message history [6]. This illustrates how even a benign-looking tool can be used for malicious data extraction when the protocol lacks integrity checks or robust sandboxing.

6.3 Multi-agent research assistants

Anthropic engineers built a multi-agent system where different agents acted as planner, researcher and writer. They reported that multi-agent architectures consumed roughly **15 ×** more tokens than single-agent chats and only offered benefits for tasks that can be parallelized [33]. For tasks requiring shared context, like code generation, the multi-agent setup underperformed due to coordination overhead. This shows that the cost and complexity of multi-agent architectures can outweigh the benefits.

7. Criticisms of agentic AI and multi-agent systems

7.1 Reliability and failure modes

Academic researchers analysed why multi-agent LLM systems fail. The **MAST** taxonomy identifies three classes of failure: **specification issues** (vague instructions, ambiguous role definitions, poor task decomposition), **inter-agent misalignment** (communication ambiguity, agents ignoring or withholding information, redundant or circular work), and **task verification gaps** (no overseer or judge agent, inadequate termination logic) [34] [35].

A summary article elaborates these points, noting that ambiguous instructions cause agents to diverge, improper role delineation leads to loops, and poor task decomposition results in incoherent outputs [36]. Miscommunication leads agents to misinterpret outputs or re-fetch identical data [37], while absence of oversight causes runaway cycles and high costs [38]. Errors compound when agents pass flawed outputs to one another [39], making the system brittle.

7.2 Lack of trust and transparency

- **Hallucinations, bias and non-determinism** – Large-language models occasionally fabricate information or produce biased outputs. A reliability study found that 61 % of companies experienced accuracy problems when using LLM agents [40]. When agents chain their reasoning, hallucinations can propagate through the system. The TRiSM paper states that the stochastic nature of LLM reasoning hinders repeatability and traceability [41].
- **Observability void** – RX-M highlights that agentic systems lack comprehensive audit trails; it is difficult to trace which agent made what decision and why, which undermines trust [42]. Edstellar similarly notes that black-box models reduce transparency and hamper compliance auditing [40].
- **Compound errors and emergent behaviour** – The TRiSM paper warns that failures can cascade when one compromised agent influences others; emergent misbehaviour and collusion become real threats [43]. Traditional single-model defenses do not account for these multi-agent interactions.

7.3 Cost and infrastructure challenges

- **Token and evaluation costs** – Multi-agent systems often require iterative reasoning, resulting in far more token consumption. Galileo Labs calculates that projects may consume 10–50× more tokens than simple chat interactions [18]. Evaluating multi-agent outputs (e.g., using a large model as a judge) further increases costs [19].
- **Infrastructure complexity** – Building agentic systems requires specialized hardware, scalable orchestration, tool servers, vector databases and messaging layers. RX-M calls these systems "Gordian knots," noting that integration with legacy systems and cloud services is challenging [44]. The lack of skilled engineers with expertise in both AI and software architecture exacerbates this bottleneck [45].
- **Dynamic LLM landscape** – The rapid pace of LLM upgrades makes it difficult for enterprises to plan. RX-M notes that constant changes require teams to continually update prompts, connectors and safety filters [46].

7.4 Trust, risk and security management (TRiSM)

The TRiSM survey provides a formal risk taxonomy for agentic AI, identifying threats such as **prompt injection**, **memory poisoning**, **collusive failure**, **emergent misbehavior** and **prompt leakage** [43]. It warns that multi-agent systems amplify security risks because agents share memory and use external tools [3]. The authors argue that standard AI safety frameworks are inadequate; a dedicated trust, risk and security framework is needed to manage agent interactions, control tool use and ensure compliance [47].

8. Implications for enterprise adoption

8.1 Governance and policy

- **Lack of regulatory alignment** – Enterprises must comply with regulations like GDPR, HIPAA and ISO/IEC 42001. The TRiSM paper notes that existing frameworks provide only general guidance and do not address inter-agent data sharing or emergent behaviours [47] [48]. Without clear standards, companies risk non-compliance.
- **Need for human oversight** – Security researchers recommend a human-in-the-loop for critical operations. When agents propose actions (e.g., spending money or sending sensitive data), a human

should review and approve. Invariant Labs' demonstration of tool poisoning reinforced the importance of transparent UIs that show which tools are being called and why [6].

8.2 Multi-tenancy and scalability

- **Single-user design** – Most available MCP servers are single-tenant. Enterprises require multi-tenant deployments with strict data isolation, concurrency controls and rate limiting. Xenoss notes that multi-tenancy remains unsolved; companies must build their own gateways to manage policies and tool discovery [49].

8.3 Cultural and talent challenges

- **Digital immaturity** – Many organizations lack the infrastructure and DevOps maturity to expose secure, reliable APIs to external agents. RX-M remarks that digital immaturity and legacy systems hinder agentic AI adoption [50].
- **Talent scarcity** – There is a shortage of engineers with both AI and traditional software expertise. Building and maintaining agentic systems requires rare skillsets (prompt engineering, security, distributed systems); this becomes a bottleneck for adoption [45].

Conclusion

While the Model Context Protocol and agentic AI promise a future where autonomous LLM agents seamlessly integrate with business processes, **current research exposes serious limitations**. MCP suffers from poor observability, significant security risks (prompt injection, token theft, tool poisoning), high latency and cost, a lack of memory and proactive capabilities, and fragile developer experience. Multi-agent systems amplify these issues, introducing coordination failures, emergent misbehaviour, and exponential costs. Industry experts, academic researchers and security analysts consistently warn that **responsible deployment requires robust governance, risk management, human oversight, and careful alignment with regulatory frameworks** [47] [40]. Organizations considering MCP or agentic AI should weigh these risks against potential benefits, invest in rigorous testing and security, and explore whether simpler, deterministic approaches might better meet their needs.

---

[1] [14] [25] Shortcomings of Model Context Protocol (MCP) Explained
https://www.cdata.com/blog/navigating-the-hurdles-mcp-limitations

[2] [10] [15] [21] [24] [31] Everything That Is Wrong with Model Context Protocol | by Dmitry Degtyarev | Medium
https://mitek99.medium.com/mcps-overengineered-transport-and-protocol-design-f2e70bbbca62

[3] [5] [9] [12] [41] [43] [47] [48] [2506.04133] TRiSM for Agentic AI: A Review of Trust, Risk, and Security Management in LLM-based Agentic Multi-Agent Systems
https://ar5iv.labs.arxiv.org/html/2506.04133v3

[4] [8] The Security Risks of Model Context Protocol (MCP)
https://www.pillar.security/blog/the-security-risks-of-model-context-protocol-mcp

[6] [20] [27] [29] [49] MCP in enterprise: real-world applications and challenges
https://xenoss.io/blog/mcp-model-context-protocol-enterprise-use-cases-implementation-challenges

7 13 30 Understanding MCP security: Common risks to watch for | Datadog
https://www.datadoghq.com/blog/monitor-mcp-servers/

11 16 Everything Wrong with MCP - by Shrivu Shankar
https://blog.sshh.io/p/everything-wrong-with-mcp

17 22 23 26 32 The Limitations of MCPs: Why They Fail in Business Data Analysis
https://www.narrative.bi/analytics/mcp-limitations

18 19 The Hidden Costs of Agentic AI: Why 40% of Projects Fail Before Production
https://galileo.ai/blog/hidden-cost-of-agentic-ai

28 Enterprise Challenges With MCP Adoption | Solo.io
https://www.solo.io/blog/enterprise-challenges-with-mcp-adoption

33 How we built our multi-agent research system \ Anthropic
https://www.anthropic.com/engineering/multi-agent-research-system

34 35 Why Do Multi-Agent LLM Systems Fail?
https://arxiv.org/pdf/2503.13657

36 37 38 39 Why Multi-Agent LLM Systems Fail: Key Issues Explained | Generative AI Collaboration Platform
https://orq.ai/blog/why-do-multi-agent-llm-systems-fail

40 AI Agents: Reliability Challenges & Proven Solutions [2025]
https://www.edstellar.com/blog/ai-agent-reliability-challenges

42 44 45 46 50 Agentic AI Part 3: Challenges in Agentic AI | RX-M Blog
https://rx-m.com/agentic-ai-part-3-challenges-in-agentic-ai/