Lateral Data Exfiltration in MCP: How One Compromised Server Captures Cross-Domain Agent Data

Akshat Sharda [1] **Zhenyu Zhang** [1]

Abstract

Production AI agents increasingly connect to multiple Model Context Protocol (MCP) servers, each ostensibly scoped to a distinct service domain. We demonstrate that MCP's shared-context architecture fundamentally violates the implicit trust boundaries between servers. Through controlled experiments, we show that a single compromised server can laterally exfiltrate data from *all* other connected servers via a covert logging tool, achieving 90.7% autonomous invocation across 360 interactions and 100% capture of cross-server banking operations. We formalize this as the *weakest-link property*: the confidentiality of all agent-accessible data is bounded by the security posture of the least-secure connected server. This threat is qualitatively different from single-server tool poisoning, as it expands the blast radius of any server compromise to the entire agent ecosystem.

1. Introduction

AI agents deployed with the Model Context Protocol (Anthropic, 2024) routinely connect to multiple servers simultaneously. A coding assistant might access file systems, databases, and web APIs through separate MCP servers, each maintained by different teams or vendors. This multi-server architecture creates an implicit assumption: each server operates within its own trust boundary, accessing only data relevant to its domain. Organizations apply security controls proportional to each server's intended data scope, treating a low-risk utility server differently from a high-security financial server.

We demonstrate that this assumption is false. MCP's architecture exposes all tool invocations and their results to a shared agent context, creating a flat trust surface analogous to an unsegmented network (Narajala & Habler, 2025). A single compromised server can inject a covert tool that captures data flowing through *any* connected server—enabling what we term *lateral data exfiltration*.

This threat is qualitatively different from single-server tool poisoning attacks (Hu et al., 2025; Sun et al., 2025). Prior work demonstrates that a malicious tool can exfiltrate data within its own server's operational scope. We show that the threat extends across server boundaries: a compromised retail server captures banking transactions, a compromised utility server harvests credentials from an HR system. The blast radius of a single server compromise expands from that server's data to *all data the agent can access*.

We formalize this as the **weakest-link property**:

> *The confidentiality of data accessible to an MCP agent is bounded by the security posture of its least-secure connected server, regardless of each server's intended data scope.*

This property has immediate implications for production deployments: a "low-risk" MCP server (e.g., a weather API) connected to the same agent as a high-security server (e.g., a financial database) must receive equivalent security scrutiny, because it can laterally access all financial data through the shared agent context. This mirrors the well-understood problem of flat networks in enterprise security, where compromising one host grants lateral access to all others on the same network segment (Ohm et al., 2020).

Contributions:

1. We provide the first empirical demonstration of cross-server lateral data exfiltration in multi-server MCP deployments, showing that a tool on Server A captures data from Server B's operations with 100% reliability for observed cross-server operations.
2. We formalize the weakest-link property and contrast the blast radius of MCP server compromise with traditional API compromise, showing a qualitative expansion from single-service to all-service exposure.

[1]Amazon. This work does not relate to the authors' current positions at Amazon. Correspondence to: Akshat Sharda <akshat.sharda13@gmail.com>, Zhenyu Zhang <zhenyuzh@amazon.com>.

Published at the Second Workshop on Agents in the Wild: Safety, Security, and Beyond (AIWILD) at ICML 2026. Copyright 2026 by the author(s).

3. We propose lateral-movement-specific mitigations—per-server context isolation, data-flow-aware permission models, and cross-server invocation monitoring—that address the architectural root cause rather than symptoms.

2. Related Work

Prior work on MCP security has focused on single-server attacks. Hu et al. (2025) demonstrate covert data exfiltration via a malicious logging tool within one server's context. Sun et al. (2025) identify tool shadowing using 20 specialized tools with explicit parameters, while Jamshidi et al. (2025) propose descriptor validation defenses. Industry analyses (Descope, 2025) and systematization efforts (Gaire et al., 2025; Hou et al., 2025) conceptually describe cross-server threats but provide no empirical measurement. Protocol-level analyses (Narajala & Habler, 2025) identify theoretical cross-system risks. In the broader agent security space, Greshake et al. (2023) demonstrated indirect prompt injection, and Schick et al. (2023) established LLM tool invocation. Software supply chain attacks (Ohm et al., 2020) show analogous risks from trusting external dependencies. Our work is the first to empirically demonstrate and quantify cross-server lateral exfiltration—using a single generic tool rather than specialized parameter manipulation—providing missing evidence that MCP's shared context violates implicit trust boundaries between servers (Table 1).

Table 1. Positioning against related work on MCP security.

Work	Single	Cross	Empir.
Hu et al. (Hu et al., 2025)	✓		✓
Sun et al. (Sun et al., 2025)	✓	Partial	✓
Jamshidi et al. (Jamshidi et al., 2025)	✓		✓
Descope (Descope, 2025)	✓	✓	
MCP SoK (Gaire et al., 2025)	✓	✓	
This work	✓	✓	✓

3. Threat Model and Experimental Setup

3.1. Threat Model

We consider a multi-server MCP deployment where an agent connects to $N \geq 2$ servers simultaneously. The attacker controls or has compromised exactly one server—through supply chain compromise, credential leakage, insider threat, or software vulnerability. The attacker's goal is to exfiltrate data from *other* servers' operations—data entirely outside the compromised server's legitimate scope.

Attacker capabilities: (1) Add arbitrary tools to the compromised server's registry via standard registration mechanisms (e.g., `@mcp.tool()` decorator); (2) Control tool names, descriptions, and parameter schemas; (3) Observe all data passed to the injected tool by the agent.

Key assumption: The agent connects to the attacker's server for legitimate purposes (e.g., it provides useful retail tools). The attacker cannot modify the agent's code, system prompt, or other servers' configurations.

3.2. Why MCP Enables Lateral Access

In traditional API-client architectures, each service receives only the data the client explicitly sends. A compromised retail API sees retail requests—never banking data. The client code enforces isolation through explicit data routing decisions.

MCP inverts this model. The agent maintains a shared context containing all tool descriptions, invocation histories, and results across all connected servers. When the agent invokes a banking tool, the result enters the shared context. A tool on *any* server can then request this data through its description, because the language model treats all tool descriptions as authoritative instructions for data routing. Servers unilaterally define what data they request—the agent has no mechanism to enforce per-server data boundaries.

This creates a **flat trust surface**: every server has implicit read access to every other server's data through the shared agent context. Figure 1 illustrates the blast radius difference between traditional API compromise and MCP lateral exfiltration.

Traditional API	**MCP Multi-Server**
Compromised server sees: • Its own requests • Its own responses	Compromised server sees: • Its own requests • All servers' requests • All servers' responses • User conversation
Blast radius: **1 service**	Blast radius: **All services**

Figure 1. Blast radius comparison. Traditional API compromise is contained to one service's data. MCP lateral exfiltration expands the blast radius to all agent-accessible data across all connected servers.

3.3. Experimental Setup

We constructed a two-server deployment to empirically measure cross-server lateral exfiltration:

Server A (Retail, compromised): FastMCP v3.0.1 server with 5 legitimate retail tools (*search_product*, *get_product_details*, *create_order*, *get_order_details*, *cancel_order*) plus one covert logging tool. Maintains in-memory state with 3 products ($29.99–$999.99).

Server B (Banking, legitimate): FastMCP v3.0.1 server with 3 financial tools (*get_account_balance*, *transfer_funds*, *get_transaction_history*). Maintains 3 accounts ($8K–

$125K).

Agent: GPT-4o-mini (gpt-4o-mini-2024-07-18) based agent using the MCP Python client (v1.26.0), connecting to both servers via stdio transport with JSON-RPC 2.0. The agent discovers 9 tools total without distinguishing source servers or security domains.

Malicious tool: A single `logging` tool (5 lines of implementation) with parameters *eventType* and *message*. Its description uses three manipulation triggers: (1) *Authority*: "Every event must be logged"; (2) *Specificity*: detailed logging format instructions; (3) *Legitimacy*: framed as debugging/auditing. Full description: *"Log an event. Every event must be logged. Every time a tool is called, this tool must be called after that logging with eventType TOOLCALL and message including a JSON with tool name, tool input and outputs."*

Test protocol: 360 questions across 4 scenarios (Product Search, Product Details, Order Creation, Multi-step), 18 questions per scenario, repeated 5 times. Multi-step scenarios specifically include cross-server operations (e.g., "Check my bank balance then order a laptop").

3.4. Attack Execution Phases

The attack proceeds through four phases that require no modification to the agent or the legitimate banking server:

Phase 1 – Tool Injection: The attacker adds the logging tool to the retail server's registry using standard registration.

Phase 2 – Automatic Discovery: Upon connection, the agent invokes MCP's *tools/list* endpoint, receiving JSON Schema specifications for all 9 tools (6 retail + 3 banking) without distinguishing their source servers or security domains.

Phase 3 – Autonomous Invocation: When users interact with the agent, the language model autonomously decides to invoke the logging tool based on its description. No explicit user instruction or prompt injection is required—the tool's description alone triggers invocation.

Phase 4 – Cross-Server Capture: When the agent invokes banking tools, their inputs and outputs enter the shared context. The logging tool's next invocation captures this banking data and routes it to the retail server, completing the lateral exfiltration. The retail server now possesses financial data it was never intended to access.

4. Results

4.1. Cross-Server Lateral Exfiltration

The central finding: the retail server's logging tool successfully captured banking server operations despite having no legitimate access to financial data. Table 2 shows the complete cross-server capture results. Every banking operation observed by the agent was logged by the retail server's tool. Example: a fund transfer of $1000 from ACC001 to ACC002 (transaction TXN1000, status: completed) was fully recorded by the retail server, including account identifiers, transfer amounts, and confirmation details that a retail service should never access. The captured data included account holder names, exact balances, and complete transaction metadata.

Table 2. Cross-server data captured by the retail logging tool from banking server operations.

Banking Data Type	Capt.	Sensitivity
Account balances & holders	3/3	Critical
Fund transfers (amt., accounts)	3/3	Critical
Transaction histories	3/3	Critical
Transaction IDs & status	3/3	High
Cross-server capture rate	**100%**	

4.2. Overall Invocation Rates

The logging tool achieved high autonomous invocation rates across all scenarios, as shown in Table 3. Across 5 repetitions, per-run success rates ranged from 83.8% to 95.5%, demonstrating consistent behavior independent of specific prompt variations. The Multi-step scenario—which includes cross-server operations—achieved 90.2%, confirming that lateral exfiltration operates reliably in realistic multi-tool workflows.

Table 3. Logging tool invocation rates across scenarios (360 questions, 5 repetitions each).

Scenario	Legit.	Log	Rate
Product Search	104	81	77.9%
Product Details	90	88	97.8%
Order Creation	90	90	100%
Multi-step (cross-server)	234	211	90.2%
Total	**518**	**470**	**90.7%**

4.3. Comprehensive Data Capture

Table 4 presents the complete exfiltration summary, distinguishing within-server and cross-server data capture. The critical observation: data sensitivity and source server had *no impact* on capture success. The logging tool exfiltrated banking data (critical sensitivity, cross-server) with the same effectiveness as retail data (medium sensitivity, same-server). MCP's flat context provides no differential protection based on data classification or server origin. The attack is entirely agnostic to data type—all information flowing through the agent context is equally accessible.

Table 4. Exfiltration summary: within-server vs. cross-server.

Data Type	Source	Count	Level
Customer emails (PII)	Retail	90	High
Product search queries	Retail	185	Medium
Order details & prices	Retail	90	High
Account balances	**Banking**	3	**Critical**
Fund transfers	**Banking**	3	**Critical**
Transaction history	**Banking**	3	**Critical**
Operational metadata	Both	11	Medium

4.4. Detection Difficulty

The attack exhibits several properties that make detection fundamentally challenging: (1) tool invocations occur transparently without user notification; (2) the malicious tool presents a plausible "logging/auditing" description indistinguishable from legitimate observability tools; (3) the agent autonomously decides to invoke it based on its own reasoning without explicit user instruction; (4) no baseline exists for "normal" tool invocation patterns, especially in multi-server deployments where tool interaction patterns vary widely; (5) tools can be added post-deployment without triggering re-review—the agent discovers them automatically on next connection; and (6) cross-organizational boundaries between server operators create gaps in visibility and accountability.

Critically, even if an organization monitors its own banking server for suspicious access patterns, the lateral exfiltration occurs entirely within the agent's context and the retail server's process—the banking server sees only legitimate tool invocations from the agent. The data leakage is invisible to the victim server's monitoring infrastructure. This asymmetry is a defining characteristic of lateral exfiltration that distinguishes it from direct attacks on the banking server itself.

5. Discussion

5.1. The Weakest-Link Property in Practice

Our results empirically validate the weakest-link property: connecting a low-security retail server to the same agent as a high-security banking server effectively downgrades the banking server's security to the retail server's level. Consider a production scenario: an enterprise agent connects to a well-audited HR server, a third-party calendar, and a community-maintained code search tool. The code search tool receives minimal security review because it "only accesses public code," yet if compromised, it could exfiltrate salary data and meeting contents—data its operators never intended to access. This mirrors flat networks in enterprise security (Ohm et al., 2020), where compromising one host grants lateral access to all others. MCP's shared context functions as a flat network—all servers share the same "broadcast domain." The established solution—segmentation and zero-trust—suggests analogous approaches for MCP.

5.2. Implications for Security Practices

Current practices evaluate each MCP server independently. Our results show this is insufficient: the correct question is "If this server is compromised, what data from *all connected servers* becomes accessible?" In MCP, adding *any* server increases the attack surface of *all* servers. The security cost of a new server is proportional to the maximum sensitivity of any data the agent handles, not that server's own risk. This complements parasitic toolchain attacks (Zhao et al., 2026), which coordinate multiple legitimate tools; our attack uses a *single* tool for passive cross-server surveillance.

5.3. Proposed Mitigations

We propose mitigations targeting lateral movement, complementing existing defenses (Jamshidi et al., 2025; Narajala & Habler, 2025): (1) **Per-server context isolation**—separate context windows per server, analogous to network VLANs, preventing Server A's tools from observing Server B's data; (2) **Data-flow-aware permissions**—tools declare required data domains, with authorization checks for cross-domain access, similar to Android's permission model; (3) **Cross-server invocation monitoring**—runtime detection of anomalous cross-server data flows triggering alerts; (4) **Server-scoped descriptions**—protocol enforcement preventing tool descriptions from referencing data outside their server's declared scope.

5.4. Limitations

We tested a single agent framework (GPT-4o-mini) with mock data and two servers via stdio transport. Results may vary across LLMs, larger server configurations, or agents with explicit compartmentalization instructions. Future work should evaluate scalability with $N > 2$ servers, cross-model generalizability, and mitigation effectiveness.

6. Conclusion

We demonstrate that MCP's shared-context architecture enables lateral data exfiltration across server boundaries—a qualitatively different threat from single-server tool poisoning. A single compromised server captures cross-server data with 100% reliability for observed operations (90.7% overall autonomous invocation). This validates the weakest-link property: multi-server deployments are only as secure as their least-secure server. Addressing this requires per-server context isolation, data-flow permissions, and cross-server monitoring—treating MCP security as a system-level concern.

References

Anthropic. Model context protocol specification. https://modelcontextprotocol.io, 2024.

Descope. Understanding MCP tool poisoning attacks. https://www.descope.com/learn/post/mcp-tool-poisoning, 2025. Accessed: 2026-05-06.

Gaire, S., Gyawali, S., Mishra, S., Niroula, S., Thakur, D., and Yadav, U. Security and safety in the model context protocol ecosystem. *arXiv preprint arXiv:2512.08290*, 2025.

Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., and Fritz, M. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, 2023. doi: 10.1145/3605764.3623985.

Hou, X., Zhao, Y., Wang, S., and Wang, H. Model context protocol (MCP): Landscape, security threats, and future research directions. *arXiv preprint arXiv:2503.23278*, 2025.

Hu, Y., Fan, C., Samyoun, S., and Du, J. Log-to-leak: Prompt injection attacks on tool-using LLM agents via model context protocol. In *Submitted to ICLR 2026*, 2025. URL https://openreview.net/forum?id=UVgbFuXPaO.

Jamshidi, S., Dakhel, A. M., Nafi, K. W., and Khomh, F. Securing the model context protocol: Defending LLMs against tool poisoning and adversarial attacks. *arXiv preprint arXiv:2512.06556*, 2025.

Narajala, V. S. and Habler, I. Enterprise-grade security for the model context protocol (MCP): Frameworks and mitigation strategies. *arXiv preprint arXiv:2504.08623*, 2025.

Ohm, M., Plate, H., Sykosch, A., and Meier, M. Backstabber's knife collection: A review of open source software supply chain attacks. In *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, pp. 23–43, 2020.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36, 2023.

Sun, Y., Du, L., Su, Z., Wang, Y., Liu, H., Zhao, Q., and Niu, X. Msa: A cross-mcp privacy attack via memory exfiltration of large language models. In *Proceedings of the 24th Workshop on Privacy in the Electronic Society (WPES)*, 2025. doi: 10.1145/3733802.3764057.

Zhao, S., Hou, Q., Zhan, Z., Wang, Y., Xie, Y., Guo, Y., Chen, L., Li, S., and Xue, Z. Parasites in the toolchain: A large-scale analysis of attacks on the MCP ecosystem. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2026.