Prompt Injection is Not an AI Problem:

Why MCP Tool Hardening Matters

shunfeng8421 — July 14, 2026 — cs.CR / cs.AI

Prompt Injection is Not an AI Problem:

Why MCP Tool Hardening Matters

shunfeng8421 — July 14, 2026 — cs.CR / cs.AI

Authors: shunfeng8421 (Independent Researcher)

Date:

Abstract

Prompt injection is widely discussed as an AI safety problem. We present experimental evidence that this framing is incomplete. Using a custom MCP (Model Context Protocol) agent simulator and a real-world vulnerable MCP server (cherrystudio-qq-mcp), we demonstrate that prompt-level filtering provides only 50% protection against injection attacks, while simple input validation at the MCP tool layer provides 100% protection. Our key finding: prompt injection defense should be implemented at the tool execution boundary, not the prompt parsing boundary. We provide a practical mitigation strategy using `validate_safe_path()` and release open-source tools for MCP security assessment.

---

1. Introduction

The Model Context Protocol (MCP) enables AI agents to interact with external tools through a standardized interface. As MCP adoption grows, security researchers have identified multiple attack surfaces, including tool parameter injection [1], inspector exposure [2], and client trust exploitation [3].

Prom
beha
moni

---

2. Experimental Design

2.1 Target System

We used **cherrystudio-qq-mcp** (GitHub: RhineLab-magellan/cherrystudio-qq-mcp), an MCP server with a `qq_upload_file` tool that accepts a `file_path` parameter and opens it on the server without path validation (CWE-22). This vulnerability was independently discovered by the author [6].

2.2 MCP Agent Simulator

We implemented an MCP agent simulator (Python, 150 lines) that:

- **1.**

2.3 Prompt Injection Techniques

Six injection techniques were tested:

ID

P1	Direct instruction	Explicit tool call in prompt
P2	Urgency exploit	Framed as emergency security audit
P3	Role-playing	"You are SecurityBot" identity override
P4	JSON nesting	Tool call hidden in JSON structure
P5	Normal request	Benign-looking file path
P6	Multilingual	Japanese/Korean to bypass English filters

---

3. Results

3.1 Unprotected Agent

Technique	Result
P1 Direct	[BYPASS] Bypassed
P2 Urgency	[BYPASS] Bypassed
P3 Role-play	[BLOCKED] Blocked
P4 JSON nest	[BYPASS] Bypassed
P5 Normal	[BYPASS] Bypassed
P6 Multilingual	[BYPASS] Bypassed

Bypass rate: 5/6 (83%)

3.2 Protected Agent (Prompt Filtering)

Technique	Result
P1-P3	[BLOCKED] Blocked
P4 JSON nest	[BYPASS] **Bypassed**

P5 Normal	[BYPASS] **Bypassed**
P6 Multilingual	[BYPASS] **Bypassed**

Bypass rate: 3/6 (50%)

3.3 Tool-Level Protection

When we replaced the MCP tool's `open(file_path)` with `validate_safe_path()` + `open()`:

Bypa

---

4. Discussion

4.1 Why Prompt Filtering Fails

Prompt filtering faces inherent limitations:

- **p

4.2 Why Tool Hardening Works

Tool-level validation operates on structured parameters, not unstructured text:

- `fil

4.3 Practical Implications

For MCP tool developers: Implement `validate_safe_path()` before every `open()` call. This is a one-line fix that eliminates the entire class of prompt-injection-to-path-traversal attacks.

For A
secur

---

5. Related Work

Prompt injection as an attack vector was first systematized by [4] and [5]. MCP security has been explored by [1] and [3]. Trail of Bits documented MCP-specific attacks including ANSI injection and credential theft [7]. Our contribution bridges these two domains: we are the first to experimentally demonstrate that prompt injection defense should be implemented at the MCP tool boundary.

---

6. Conclusion

Prompt injection is not an AI safety problem — it is a systems security problem. The AI agent is merely the

delivery mechanism. The actual vulnerability lives in the MCP tool's implementation, and the fix belongs there too. One line of code (`validate_safe_path()`) provides stronger protection than any prompt filter.

Tools

- **mcp-scan**: https://github.com/shunfeng8421/mcp-scan
- **a

References

[1] shunfeng8421. "MCP Attack Surface Taxonomy." awesome-mcp-security, 2026.

[2] CV