Network Working Group
Internet-Draft
Intended status: Informational
Expires: 3 December 2026

A. Syed

June 2026

Security Considerations for Model Context Protocol (MCP) Implementations in AI Agent Systems

draft-mohiuddin-mcp-security-considerations-00

Abstract

The Model Context Protocol (MCP) connects large language models (LLMs) to external tools, data sources, and services. The MCP specification does not define normative security requirements. This document analyzes recurring vulnerability classes that have been publicly reported in MCP server implementations, describes an automated detection approach implemented in the open-source tool mcp-safeguard, and proposes security considerations and mitigations for implementors and operators. The document also describes Protocol Pivoting, a cross-protocol lateral-movement pattern in systems that compose MCP with other agent protocols.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 3 December 2026.

Copyright Notice

Table of Contents

1. Introduction

The Model Context Protocol (MCP) is an interface for connecting LLMs to external capabilities. The protocol enables AI agents to invoke tools exposed by MCP servers, which are software components that accept tool calls from an AI host and execute operations against external systems.

The MCP specification does not include normative security requirements. This document analyzes vulnerability classes that arise from that gap, references public reports of those vulnerabilities in specific implementations, describes an automated detection approach, and offers guidance to implementors and operators.

The detection approach described in this document is implemented in mcp-safeguard, an open-source automated MCP security scanner developed by the author that performs black-box evaluation of running

MCP servers. Detection results referenced in this document were obtained against controlled test fixtures that model the vulnerability classes described in Section 4. Independent research, such as MCPTox, reports high attack-success rates against production MCP servers and corroborates that these classes are exploitable outside controlled environments.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. MCP Architecture and Trust Model

An MCP deployment consists of three components:

* MCP Host: the AI application that manages connections to MCP servers.

* MCP Client: a component within the host that implements the MCP protocol and routes tool calls to servers.

* MCP Server: an external process or service that exposes tools through the MCP protocol.

The trust relationship is asymmetric. The host trusts the server to correctly scope its tool effects, and the server trusts tool call parameters to originate from an authorized host with vetted inputs. Both assumptions are frequently violated in practice.

Tool call parameters originate from LLM reasoning, which is susceptible to prompt injection. An adversary who can influence the content the LLM processes can influence tool call parameters. MCP servers MUST treat all tool call parameters as untrusted.

4. Vulnerability Classes

4.1. Server-Side Request Forgery (SSRF)

MCP servers that accept URL parameters and pass them to HTTP clients without validating the resolved IP address are vulnerable to SSRF. An adversarial prompt can cause such a tool to request:

* Cloud instance metadata services (for example 169.254.169.254), which can expose IAM credentials on affected cloud hosts.

* Internal network services not exposed to the public internet.

* Loopback services bound only to 127.0.0.1.

Naive mitigations that block a literal metadata IP can be bypassed by DNS rebinding, where a hostname resolves to a public address at validation time and to a restricted address at connection time, and by redirect chains, where an allowed URL returns an HTTP 3xx redirect to a restricted destination.

4.2. Excessive Tool Permissions

MCP filesystem tools frequently expose paths beyond what their documented function requires, allowing an adversarial prompt to read arbitrary files.

4.3. Prompt Injection Surface Exposure

Some MCP tools return content from external sources to the LLM without sanitization, allowing an adversary who controls that content to inject instructions that alter the agent’s subsequent behavior.

4.4. MCP Lifecycle Bypass

Some implementations answer tools/list before the initialize handshake completes, which can skip access-control logic applied during initialization.

4.5. Information Leakage

Some servers return verbose errors including stack traces, internal paths, and dependency versions, which aid fingerprinting.

4.6. Authentication Enforcement Gaps

Some servers answer tool calls without verifying authentication, which is especially serious for servers exposing code execution or filesystem access.

5. Publicly Reported Instances and Automated Detection

This section references public reports of the SSRF class. These reports were filed by their respective reporters in public issue trackers and are cited here as evidence that the class occurs in deployed implementations. This document does not claim authorship of those reports.

5.1. SSRF Reports in HTTP-Fetching MCP Servers

SSRF affecting an HTTP-fetching MCP server in the modelcontextprotocol/servers repository has been discussed in public issues, including issues 4116, 4143, and 4205. The reported pattern matches the SSRF class above: a URL parameter passed to an HTTP client without IP validation, reachable through DNS rebinding and redirect chains.

5.2. SSRF Reports in Browser-Automation MCP Servers

SSRF affecting a browser-automation MCP server has been discussed in public issue 1626 of the microsoft/playwright-mcp repository. Because such servers may run with access to a local browser profile, the reported impact includes access to internal services and to authenticated session state.

5.3. Automated Detection with mcp-safeguard

mcp-safeguard implements black-box probes for the classes above. For SSRF, it submits a benign request to a controlled external host, a request that redirects to a restricted address, and a direct metadata-endpoint request, and it classifies a server as affected when a restricted destination is reachable. Against test fixtures modeling the SSRF class, mcp-safeguard detects the class and emits machine-readable findings suitable for use in continuous integration. Validation against production deployments is future work and is not claimed here.

6. Protocol Pivoting: Cross-Protocol Lateral Movement

This document describes Protocol Pivoting, a lateral-movement pattern in systems that compose multiple agent protocols, for example MCP for tool invocation together with Agent-to-Agent (A2A) delegation and an autonomous negotiation protocol. Related work has described pivoting within a shared MCP context; this document focuses on movement that crosses protocol boundaries.

An adversary who gains influence over one protocol layer can use that influence to act across other layers:

1. Initial access: an injected instruction enters content processed by an MCP-using agent.

2. MCP step: the instruction causes the agent to invoke an MCP tool against an internal service.

3. A2A step: the agent delegates a task to a downstream agent, which inherits the originating agent’s trust.

4. Negotiation step: the downstream agent negotiates with an external service under the originating agent’s identity.

Mitigations that consider each protocol in isolation do not address movement that crosses protocol boundaries. This is identified here as an area for further analysis.

7. Security Considerations for MCP Implementors

URL validation: servers that accept URL parameters MUST validate the resolved IP address before issuing requests, and MUST perform this check at connection time rather than at parse time to resist DNS rebinding. Servers SHOULD block RFC 1918 ranges, loopback, link-local (169.254.0.0/16, fe80::/10), and other non-routable ranges.

Network egress: server processes SHOULD run with egress restrictions enforced at the OS or container level, independent of application validation.

Redirects: servers SHOULD NOT follow HTTP redirects by default. When redirects are followed, each destination MUST undergo the same IP validation as the original URL.

Tool permission scoping: filesystem tools MUST scope path access to the minimum set of directories required, and MUST NOT expose root filesystem access absent explicit operator configuration.

Lifecycle enforcement: servers MUST NOT process tool calls before the initialize handshake completes.

Logging: servers SHOULD emit structured, attributable logs of tool invocations to support detection and incident response.

8. Security Considerations for MCP Operators

Operators deploying MCP-based systems SHOULD apply network egress controls, run servers with least privilege, and monitor tool invocations. Operators SHOULD treat any MCP server that processes externally controlled content as exposed to prompt injection and SHOULD constrain the effects available to such servers.

9. IANA Considerations

This document has no IANA actions.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

Author’s Address

Anas Mohiuddin Syed
Chicago, IL
United States of America
Email: anasmohiuddinsyed@gmail.com