



NSA CSI: MCP Security Design Considerations (1/3)

NSA Cybersecurity Information Sheet U/OO/6030316-26, PP-26-1834, May 2026 v1.0 | What MCP is, why its design inverts the usual client-server pattern, and the nine security concerns the NSA observed in design, implementation and practice

#NSA #MCP #AgenticAI #CSI

✓ The advisory at a glance

Document	Model Context Protocol (MCP): Security Design Considerations for AI-Driven Automation , NSA Cybersecurity Information Sheet (CSI) U/OO/6030316-26, PP-26-1834, May 2026, version 1.0, 17 pages, 25 cited works
Audience	organisations adopting MCP in high-stakes or production environments; developers of MCP clients and servers; National Security Systems, Department of War and Defense Industrial Base (DIB) operators
Companion	ASD ACSC (Australian Signals Directorate's Australian Cyber Security Centre) Careful adoption of agentic AI services (2026) for agentic AI in general; this CSI covers the MCP layer specifically
Structure	executive summary → introduction (travel-assistant example, Figure 1) → nine security concerns → six real-world issue classes → nine recommendations → conclusion
Stance	MCP is promising but immature : security depends on implementation discipline, not on protocol guarantees; adopt with caution and controls that go beyond what the specification suggests

Executive summary in four points

01

MCP is the de facto standard for AI service communication

used in business, finance, legal and software development; AutoGen Studio, Harvey AI, Agentverse and Copilot ship it

02

Adoption outpaced the security model

like early web protocols: flexible, underspecified, ambiguous for safe use

03

The interaction pattern is inverted

servers query and sometimes execute actions for clients, creating new, poorly traced attack paths

04

Risks are demonstrable, not hypothetical

public labs ship deliberately vulnerable MCP servers; arbitrary code execution arises easily where user logic is executed

📄 Where this sheet fits

Sheet 1/3 (this one): architecture and the nine concerns. Sheet 2/3: the six real-world issue classes with their attack mechanics. Sheet 3/3: the nine recommendations as a hardening programme. The protocol itself (JSON-RPC, transports, OAuth 2.1 authorisation, OWASP agentic threat catalogue) is covered in the emerging track sheet *Agentic AI and MCP Security*.

💡 What MCP is

≡ Model Context Protocol (MCP)

An **application-level protocol** with a simple, agreed messaging pattern and transport format used by AI-enabled systems to manage interactions between services. Open-sourced by **Anthropic in November 2024**, now embedded in most AI orchestration and agent frameworks. It passes **structured messages** (a role, message content, an optional tool or context) between the components of a multi-step workflow: model evaluation, enrichment, pre-processing, task automation.



The CSI's running example: one short prompt, five automated data-gathering steps orchestrated through MCP

MYTH

✗ MCP is just another client-server API: the client asks, the server answers

✗ Standard web controls (authn, authz, input validation) are enough

FACT

✓ MCP often expects the server to query and sometimes execute actions for the connected client

The component you connect to can drive work on your side; trust therefore flows in the unusual direction.

Source: CSI executive summary

✓ They remain necessary but do not address the novel, systemic risks of agentic systems

Dynamic tool invocation, implicit trust between agents and shared context need lifecycle-wide controls.

Source: CSI, Security concerns

The inversion the NSA insists on: it is what makes MCP attack paths unfamiliar

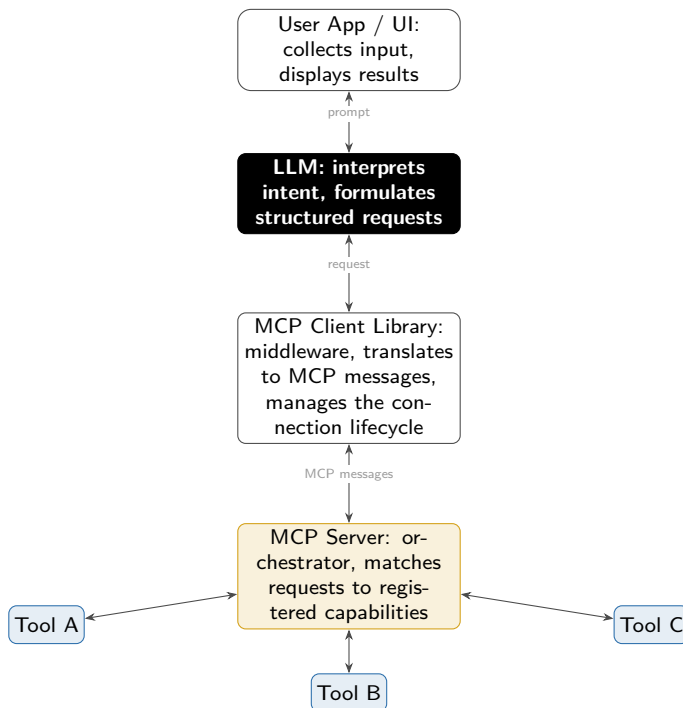


Figure 1 of the CSI: how the MCP components interact (the server orchestrates, tools execute, results flow back up the chain)

CWE	Name	How it shows up in an MCP server
CWE-77	Command injection	a tool builds a command line from a parameter the model supplied
CWE-78	OS command injection	a tool shells out (<code>subprocess</code> , <code>os.system</code>) with unsanitised arguments
CWE-94	Code injection	a tool executes code that a parameter or a resource contained
CWE-95	Eval injection	a tool calls <code>eval()</code> / <code>exec()</code> on model- or user-provided text

Arbitrary Code Execution (ACE), the high-severity class the CSI singles out, and the CWE ids under which it is tracked

EXAM TIP

Expected one-liner: **secure-by-default behaviour must be enforced through implementation rigour, proper coding practices, clearer protocol specifications and robust validation tools**, because the protocol does not enforce it. Remember the four CWEs as the ACE family: 77, 78, 94, 95.

⚠ Three systemic risks of agentic systems

⚡ Dynamic tool invocation

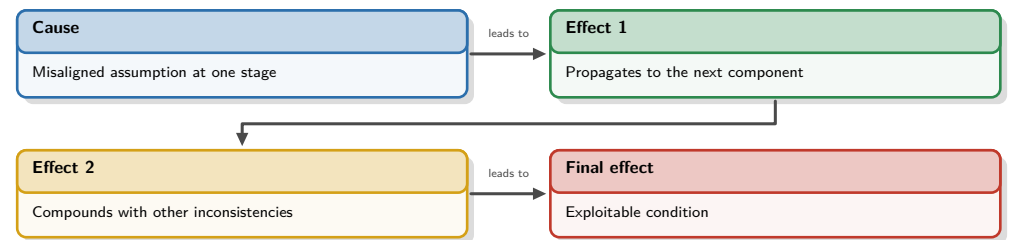
Agents autonomously call **new tools at runtime**: the set of code that can run on your behalf is not fixed at deployment time.

🔗 Implicit trust relationships

One agent's output is **assumed valid by another** without explicit verification, so a poisoned result propagates.

📁 Context sharing

Long-lived or overlapping **context windows leak, blend or misalign** across tasks, so data from one job reaches another.



Why the CSI refuses point fixes: the agentic environment is a continuum

⚠ Scope of the fix

These risks cannot be patched at the interface or endpoint level. Securing MCP means reworking the **entire lifecycle**: protocol design and agent behaviours, runtime scheduling and coordination, integration with external services, and long-term operational monitoring.

⚠ The nine security concerns

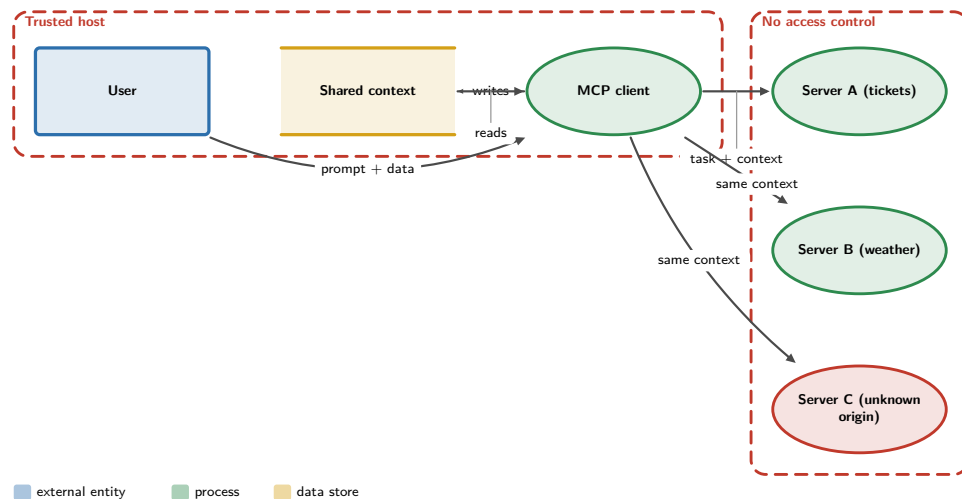
Concern	Gap is in	Severity	Consequence
Access control	protocol	critical	untraceable access to data and context; no RBAC exchanged at instantiation
Insecure context serialisation	protocol + implementation	high	injection through serialised comments, prompts or embedded model calls
Poor approval work-flows	implementation	high	silent capability changes after an initial consent
Token or session security	protocol	high	replay, session hijacking, client impersonation
Misconfigurations	operations	high	trivial servers repurposed onto sensitive data; no task isolation
Inconsistent behaviours	implementation	medium	non-deterministic outcomes an attacker can precondition
Poor or missing audit logs	implementation	high	no traceable sequence of actions; IR and accountability fail

Denial of service, fatigue	operations	medium	prompt storms, recursive tasks, lethargy techniques
Implicit trust / propagation	design	critical	unverified task propagation across servers

Concern catalogue: where the gap sits and what it leads to

1 Access control

Binding a session to an identity is **not defined by the protocol**; it is optional and left to the implementer. Clients, servers, proxies and hosts can all process data with **no access control at all**. Many implementations omit authentication; those that have it rarely enforce roles, not even the Create / Read / Update / Delete (**CRUD**) split that every database interface has. MCP has no way to exchange **Role-Based Access Control (RBAC)** permissions when a connection is instantiated, so access boundaries between tasks or services cannot be enforced or verified.



The sharing problem: one client fans context out to several unrelated servers, so a message meant for one leaks to the others (unverified task propagation)

Unverified task propagation

Tasks passed between MCP servers or agentic components **without validation of their origin, scope or intent** (CSI footnote 1). Leads to overreach, leakage of sensitive context, or unintended activation of unrelated downstream tools.

2 Insecure context or data serialisation

- What is **serialised** context, configuration and payload objects, with **no strict validation or schema enforcement** required; each implementer invents its own
- Why it is **dangerous** serialised **comments or prompts can carry executable code or embedded model calls**; permissive deserialisation without isolation between code and text (data, comments) opens an injection path

- Known pattern** echoes **OWASP A08:2017 Insecure Deserialization**, with a new twist: the content may be parsed as natural language by a model

3 Poor approval workflows

MCP itself cannot enforce these security principles at the protocol level.

— MCP specification, security guidance quoted by the CSI

- Consent at install** user grants account access (e.g. the GitHub MCP server to a GitHub account) and clicks allow on an action
- Trust is cached** the server is now connected; some implementations cannot even codify approvals
- Capability changes silently** a change in tools or data access of an already-trusted server needs **no new approval**
- Nobody notices** continual small improvements in the background; the user is never prompted for informed consent
- Sensitive access on demand** the previously benign service reads sensitive resources without triggering any review

How a benign, approved server turns into an unreviewed sensitive-data reader

4 Token or session security

What the MCP specification mandates for sessions

- Authorisation is optional: not every implementation uses it
- Servers rely on OAuth 2.1 bearer tokens without protocol-level lifecycle management (refresh, revocation, reuse control)
- Expiration and rotation are recommended in some drafts, never mandated by the core spec
- Token passthrough and lifecycle weaknesses are acknowledged in the spec's Security Best Practices and Lifecycle pages
- Idempotency is not enforced by MCP; left to JSON-RPC and message-queue layers, so the client must implement it

Idempotency

Executing an operation several times changes nothing beyond the first application. Without it, a **replayed or duplicated message re-executes a side effect** (send, pay, delete). Consequences of the session gaps: message replay, unauthorised reuse of valid sessions, and after hijacking, impersonation of a legitimate client, injection of malicious prompts, undetected interaction with servers.

5 Misconfigurations and poor implementation

The weather server

A trivial server built to return weather for a location is deployed without restrictions, then **re-purposed** to access user or asset information at that location. Same interface, same message structure, no task or data isolation: MCP has no built-in way to isolate services for privacy or security. Open-source frameworks let developers deploy before they understand the implications.

6 Inconsistent behaviours

What diverges	What it causes
interpretation of serialised context	unpredictable, exploitable gaps between components
handling of messages and server tasks	violated assumptions
context shaping	security blind spots
timing of responses	denial of safeguards

Preconditioned travel agent

Two agents receive the same prompt. One links it to an **earlier e-mail about vacation plans**, the other treats it as new. This probabilistic reading of prior context can be exploited: a malicious actor **preconditions** the agent (plants context) so that it arrives at a specific, unsafe outcome. Non-deterministic behaviour is a new class of issue, outside traditional information security practice.

7 Poor or missing audit logs

What the CSI says logs must provide

☒ baseline / mandatory ☐ recommended – verify in your context

- ✓ A traceable sequence of actions across sessions
- ✓ Source information from HTTP headers where available
- ✓ Native detection of protocol anomalies: repeated or malformed requests
- ✓ Authorisation failures and RBAC violations as alertable events
- ☐ **MUST** Alerting, not just recording, in multi-tenant environments

8 Denial of service and fatigue-based techniques

Prompt storm — A burst of prompts or malformed inputs that saturates an agent-serving MCP server.

Recursive task request — A task that spawns further tasks (agent calls tool calls agent) until resources are exhausted.

Lethargy — Exhausting a server with legitimate but overly complex tasks, causing delays and interruptions across the workflow; hard to distinguish from valid high load.

⚠ DANGER

None of these are unique to MCP, but MCP **opens the door** to them when resource limits, rate limits and recursion depth are not managed. Sheet 3/3 covers the limits to set.

✓ Self-test

- Q1 Which organisation published the CSI, and when?**

A The US National Security Agency (NSA), May 2026, version 1.0, reference U/OO/6030316-26.
- Q2 Who originally released MCP, and when?**

A Anthropic, as an open-source framework, in November 2024.
- Q3 What does the CSI mean by the inverted interaction pattern?**

A Instead of clients requesting data from servers, MCP often expects servers to query and sometimes execute actions for the connected clients.
- Q4 Name the four CWEs the CSI lists for Arbitrary Code Execution.**

A CWE-77 command injection, CWE-78 OS command injection, CWE-94 code injection, CWE-95 eval injection.
- Q5 What three systemic agentic risks does the CSI name?**

A Dynamic tool invocation, implicit trust relationships, context sharing.
- Q6 What is unverified task propagation?**

A Tasks passed between MCP servers or agentic components without validation of their origin, scope or intent.
- Q7 Does MCP mandate token lifecycle management?**

A No. Authorisation is optional, bearer tokens follow OAuth 2.1 with no protocol-level refresh, revocation or reuse control; expiry and rotation are only recommended.
- Q8 What is a lethargy technique?**

A Exhausting server resources with legitimate but overly complex tasks, causing delays that are hard to tell from normal high load.
- Q9 Why can an approved MCP server become dangerous later?**

A A change in its capabilities or data access after the initial consent usually needs no new approval, so users are never re-prompted.