The MCP trust-boundary audit checklist

Twelve questions, each answered in writing with primary-source evidence · Alex Schose · alexschose.com

A trust-boundary review of an MCP server, agent client, or autonomous-agent product should be able to answer the twelve questions below — each with evidence drawn from the product itself, not from an interview or a policy document. None of it is framework-specific.

If a product can answer all twelve in writing, it has a real threat model and an audit is mostly about probing the edges. If it can answer fewer than half, the audit is about getting the rest answered before anything ships. The examples of what failure looks like are drawn, de-identified, from real reviews.

01 Where is the trust boundary drawn?

If the answer is “between the user and the server,” that is a protocol-layer answer, and prompt-injection in tool output crosses it. The right answer names a substrate the model cannot influence: a process namespace, a container, a VM, a hypervisor.

BAD A server bound its HTTP transport to every network interface with no authentication, fronting a live signing key — a single unauthenticated request could move funds. The boundary existed only in the README.

VERIFY There is a named enforcement layer below the protocol, and the transport is authenticated and scoped by default — not by operator opt-in.

02 What can the model name?

Anything the model can name, it can be talked into acting on. List every resource the tool surface can reference — files, endpoints, services, credentials, accounts.

BAD A tool schema exposed raw host, path, and header parameters directly to the model, so the reachable set was whatever the host happened to have, not what the operator chose.

VERIFY The set of nameable resources equals the set the operator explicitly selected.

03 What does “denied” look like versus “not found”?

If forbidden and absent produce different responses, the model has an existence oracle it can enumerate.

BAD Read-shaped tools that answered “permission denied” for a real-but-forbidden resource and “not found” for an absent one — the difference maps the hidden inventory one probe at a time.

VERIFY Every read-shaped tool returns the same response shape for absent and forbidden. A correct isolation layer returns a uniform “not found” for both.

04 What is the egress surface?

Default-deny with an explicit allow-list is auditable. Default-allow with an attempt to block known-bad is not.

BAD An open-proxy tool fetched any URL the model named, failed open when its robots check errored, and followed redirects — a one-call request-forgery into internal services.

VERIFY Network egress is default-deny with an explicit allow-list the model cannot extend.

05 Where do credentials originate?

If the model holds long-lived credentials, a prompt-injection holds them too.

BAD A tool attached the operator’s API token to whatever host the caller named — naming an attacker-controlled host exfiltrated the token in a single call.

VERIFY Credentials are issued per task by a component the model cannot influence, scoped to the one resource the task needs, and never attached to a destination the model chose.

06 What happens to disposable state?

Ephemeral compute — a fresh container or VM per task, destroyed on completion — should be the default. Persistence should be an operator’s affirmative choice, not the model’s.

BAD Tools wrote to unconfined paths that survived across tasks, letting one task plant state the next one would read.

VERIFY Compute is fresh per task and destroyed on completion by default; persistence requires an explicit operator action.

07 What is loggable, and by whom?

Logging inside a trust domain the model controls is not logging — it is the model writing fiction.

BAD The action log lived in a store the agent could itself read and rewrite, so the record of what it did was only as honest as the agent.

VERIFY The audit log lives in a substrate the agent cannot reach, is append-only and tamper-evident, and records the action without leaking the secret values it acted on.

08 What happens when the model is wrong?

The failure mode is the whole game: a compromised or hallucinating model's blast radius is bounded, or it is not.

BAD A tool with arbitrary local-file read and no path confinement exfiltrated whatever it read to a configured endpoint. One hallucinated path, unbounded disclosure.

VERIFY Blast radius is bounded to one disposable surface by construction, regardless of how the protocol layer is configured.

09 Is every resource scoped to the caller's tenant? added

A multi-tenant product carries a second boundary the single-operator threat model ignores — and it is the one that leaks.

BAD An unscoped lookup let any caller read another tenant's application configuration, workflow graph, and embedded secrets — no exploit required, just a different identifier.

VERIFY Every lookup is filtered by the authenticated tenant at the data layer, not by an interface that merely hides the other tenants.

10 What leaves through telemetry? added

The exfiltration path nobody audits is the observability pipeline.

BAD A product shipped full prompts and complete model outputs to a third-party error-reporting endpoint by default, on every call — the operator's data left the building before anyone read a log line.

VERIFY Telemetry is off by default or scrubbed end-to-end; you can name exactly which fields leave and who receives them.

11 Is authority graduated, or all-or-nothing? added

A single trust boundary means one injection inside it holds every right the product grants.

BAD A binary trusted/untrusted model where any in-scope action implied full read, write, and lifecycle authority over everything in scope.

VERIFY Authority graduates per action and per resource — read below execute below write below lifecycle — and the model cannot widen its own tier. A trading agent I probed in a red-team exercise pinned the user's identity and trade limits to server-side values re-checked on every call; no prompt could argue, inject, or template the agent into another account or past the limit. That is the property to look for: authority the conversation cannot move.

12 Is the data the agent parses safe against hostile input? added

An agent ingests files and query results. The parser is an attack surface the trust-boundary questions miss — the model does not have to be wrong if the data is.

BAD A metadata filter interpolated untrusted keys and values straight into SQL; a model-file parser crashed or read out of bounds on a malformed sub-hundred-byte file.

VERIFY Every parser the agent feeds on validates file-declared sizes and counts before allocating, uses parameterised queries for anything reaching a database, and treats model files and tool results as hostile input.

Twelve answers, in writing, with evidence — that is a real threat model. I run this as a fixed-scope review of MCP servers, agent clients, and autonomous-agent products.

alexschose.com · independent security review for AI-agent products.