ILLUSTRATIVE SAMPLE · NOT A CLIENT ENGAGEMENT · NOT A 0-DAY DISCLOSURE

AI Agent & MCP Security Audit — sample report

Classification: Illustrative sample. Not a client engagement. Not a vulnerability disclosure.

Prepared by: Kevin Fairley (MCPSentry 0.1.0, with a human review and an AgentGate policy overlay)

Date: 22 November 2026, America/Los_Angeles

Targets (public open-source reference servers, not production deployments):

Target	Package	Version in tree	Git revision
Official Filesystem MCP Server	`@modelcontextprotocol/server-filesystem`	2.4.0	`modelcontextprotocol/servers@c799c8552b0f27b7a33a7b9233773da144dddcf5`
Official Memory MCP Server	`@modelcontextprotocol/server-memory`	2.2.1	same revision

The official repository states that these are **reference implementations**, written to demonstrate MCP features, not as production-ready products. This sample is meant to show what a Quick Scan looks like on well-known public code. It is not a claim about every deployment of these packages, and it is not a 0-day report.

A third well-known server, `@playwright/mcp` 0.0.86 (`microsoft/playwright-mcp@2337aff0213d28db41d3e3daf48cee19b3d0bd82`), was opened for a scan. MCPSentry found **0 tools**, because the tool source lives in the Playwright monorepo (`packages/playwright-core/src/tools/mcp`) rather than in the `playwright-mcp` repository. That target is **not scored** below.

Raw scanner output is in `raw-scans/`.

---

1. Executive summary

MCPSentry was run as a static scan (no tools were invoked) against the two official reference servers above, on 22 November 2026 at 5:27 p.m. PT.

Target	Tools found	Scanner findings	After human review
Filesystem MCP	16	1 high (MCP004 `copy_file`), 1 medium (MCP007 missing rate limit)	The MCP004 flag is a **heuristic lead, not a confirmed path-traversal bug**. MCP007 stands, with context. MCP006 (missing HTTP auth) was **not emitted**, for a scanner reason explained below; human review treats the HTTP posture as the same class of issue as Memory.
Memory MCP	9	1 high (MCP006 missing auth), 1 medium (MCP007 missing rate limit)	Both findings stand as pattern-based leads, with important context: the HTTP transport defaults to loopback and checks `Origin`.

Headline, in plain language:

1. Both reference servers can be served over Streamable HTTP (`--transport streamable-http`). In the code we scanned, that endpoint does not implement a bearer or API-key check, and it does not implement a rate limiter. The default bind is loopback (`127.0.0.1:3001`) and browser requests are limited to a localhost `Origin`. The residual risk is mainly what happens if `HOST` is widened or the port is otherwise reachable.
2. Filesystem tools that write (`write_file`, `append_file`, `edit_file`, `move_file`, `copy_file`, and others) are annotated, and `copy_file` / `move_file` call `validatePath` against the process's allowed directories before touching the disk. MCPSentry still flagged `copy_file` as an unsafe sink because the handler eventually calls `copyFile()`. That is how the rule is written — syntactic, not a proof of missing sandboxing. A human should not report it as a confirmed bypass.

3. AgentGate was **not** installed in front of these servers (they are not AgentGate clients). A policy overlay is included so the sample shows that part of the paid offer: write and delete tools would require an allow-list, and several would require a human approval hook, if this were a production agent.

Nothing in this report is a guarantee of safety, and nothing here is a new exploit write-up. Where a public advisory already exists in the project's own changelog, it is named only at the advisory-ID level.

---

2. Method

- **Scanner:** MCPSentry 0.1.0, `scan <path> --exclude __tests__ --format all --fail-on none`.
- **Rules in play:** MCP001 hardcoded secrets, MCP002 dishonest annotations, MCP003 over-broad tools, MCP004 unsafe sinks, MCP005 prompt-injection descriptions, MCP006 missing HTTP auth, MCP007 missing HTTP rate limit, MCP008 npm lockfile CVEs (OSV). The filesystem package was scanned as its own directory, so the monorepo `package-lock.json` was **not** in scope for MCP008 on that run.
- **Live probe:** not used on these targets for the sample (a probe only runs `initialize` and `tools/list`, but it still starts the server). A paid Quick Scan would use a probe when the client provides a safe staging command.
- **AgentGate overlay:** desk review against AgentGate's least-privilege model (default deny, approval hooks on high-risk tools, argument constraints, rate limits). No runtime intercept was installed, so this is not a red-team score against these servers.
- **Human review:** every scanner finding was read back against the handler source.

---

3. Tool inventory (condensed)

3.1 Filesystem MCP — 16 tools, all annotated

Read-only (annotations say `readOnlyHint=true`, `destructiveHint=false`): `read_file`, `read_text_file`, `read_media_file`, `read_multiple_files`, `list_directory`, `list_directory_with_sizes`, `directory_tree`, `search_files`, `get_file_info`, `list_allowed_directories`.

Write / change (annotations present; several marked `destructiveHint=true`): `write_file`, `append_file`, `edit_file`, `create_directory`, `move_file`, `copy_file`.

The allowed-directory model, `--read-only`, and `--read-only-dir` are documented by the maintainers and are visible in the source (`validatePath`, `read-only-directories.ts`). That is relevant context for every write-tool finding.

3.2 Memory MCP — 9 tools, all annotated

Reads: `read_graph`, `search_nodes`, `open_nodes`.

Writes: `create_entities`, `create_relations`, `add_observations`.

Destructive annotations present: `delete_entities`, `delete_observations`, `delete_relations`.

Memory is a persistent graph. In an agent setting that graph is also **context** (OWASP ASI06). This sample did not attempt to poison it.

---

4. Findings

Severity here is the scanner's severity, followed by the human disposition.

F-1. HTTP MCP transport: no detected authentication (Memory: MCP006, high)

Where: `src/memory/http.ts` (Streamable HTTP at `/mcp`).

Scanner: MCPSentry emitted MCP006 for Memory because it found an HTTP MCP transport and did not find a bearer token, API key, or `Authorization` check.

Human review: The finding is a fair lead, with context. The transport:

- Binds **loopback by default** (`DEFAULT_HOST = "127.0.0.1"`, port 3001)
- Can be widened with the `HOST` environment variable
- Applies `localhostOriginValidation()` so a browser page is only accepted from a localhost `Origin`
- Does not, in the code we read, require a bearer token or API key

A process listening only on loopback is a weaker exposure than a public bind. If `HOST` is set to a reachable interface, the missing application-level credential becomes a real access-control gap (OWASP LLM03 / ASI03; NIST Measure 2.7-style security evaluation).

Disposition: Confirmed as a **design limitation of this HTTP mode**, not as a proof that a default stdio install is remotely reachable. Recommendation: if HTTP is used beyond loopback, require a bearer or API key and reject missing credentials with 401. Stdio-only deployments are out of scope for MCP006.

F-2. HTTP MCP transport: no detected rate limit (Filesystem and Memory: MCP007, medium)

Where: `src/filesystem/http.ts`, `src/memory/http.ts`.

Scanner: MCP007 on both targets.

Human review: Confirmed as a pattern-based lead. There is no rate limiter in those HTTP listeners. Combined with write or delete tools, an unbounded client could create cost or integrity pressure (OWASP LLM06). The same loopback-default context as F-1 applies.

Disposition: Accepted, with the loopback / `HOST` caveat. Recommendation: per-client request caps and a 429 / `Retry-After` response if HTTP is enabled.

F-3. Filesystem `copy_file` flagged as an unsafe sink (MCP004, high) — not confirmed as a bypass

Where: `src/filesystem/server.ts` (tool `copy_file`, around line 987).

Scanner: MCP004 reports that `copy_file` passes tool arguments into a filesystem write sink (`copyFile()`), and recommends rejecting path traversal and never writing to an absolute path supplied by the model.

Human review: The handler does **not** pass raw arguments straight to `copyFile()`. It first calls `validatePath` on the source (mode `"read"`) and on the destination (mode `"write"`, including read-only directory checks), then copies the validated paths. MCPSentry's sink rule is syntactic; it does not model that sanitization. Reporting this as a confirmed directory-escape would be dishonest.

Disposition: Lead, downgraded after review. Residual risk remains the residual risk of any path sandbox (symlink races, logic bugs). This report does not claim a new bypass and does not include a reproduction. The project's own changelog records recent path-handling fixes, including work described against public GitHub Security Advisories (`GHSA-rq2p-q7c4-6g27` and related path-pinning changes). Those advisories are public; this sample does not re-describe them.

F-4. Scanner limitation: Filesystem did not get an MCP006 finding

Where: MCPSentry HTTP-auth heuristic, not a new bug in the filesystem server's HTTP stack.

Human review: Filesystem's HTTP listener is structurally the same class as Memory's (loopback default, `HOST` override, localhost `Origin` check, no bearer or API key). MCP006 still did not fire on Filesystem. The auth heuristic treats a match for the word `unauthorized` (among other patterns, including `401` and `bearer`) as evidence that authentication exists. In `src/filesystem/lib.ts` there is a comment about not creating files in "unauthorized locations." That comment is about the **directory allow-list**, not HTTP authentication, and it is enough for the heuristic to stay quiet.

Disposition: This is recorded as a **scanner limitation** (already consistent with MCPSentry's own threat model: MCP006/MCP007 are pattern-based). A paid audit always includes this human pass. Memory's MCP006 finding should be read as covering the shared HTTP pattern on both servers.

---

5. AgentGate policy overlay (desk review)

These servers are not running behind AgentGate. The notes below show what a Standard-tier policy review would say if an agent were given these tools.

Filesystem, if used by an agent:

- Default deny. Allow only the directories the user already passed on the command line; never let the model add a new root.
- Read tools may be allow-listed at low risk, still with path constraints.
- `write_file`, `append_file`, `edit_file`, `create_directory`, `copy_file`: allow only with path constraints; consider an approval hook when the destination is sensitive.
- `move_file` (and any delete-equivalent behavior): **approval required**, high risk.
- If HTTP mode is enabled beyond loopback: AgentGate does not replace transport authentication; add that separately, plus a rate limit.

Memory, if used by an agent:

- Reads (`read_graph`, `search_nodes`, `open_nodes`) may be allow-listed at low risk for a dedicated memory agent.
- Creates and observation writes: allow with size limits; treat stored text as untrusted in later prompts (ASI06).
- All `delete_*` tools: **approval required**.
- Do not let a low-privilege agent ask a higher-privilege agent to run a delete (AgentGate's privilege-escalation check).

AgentGate must sit on the tool-call path to enforce any of this. A runtime that calls the filesystem or memory server without going through `authorize()` is unprotected. AgentGate's included red-team suite is 18/20 cases blocked, 0 false positives on 8 benign cases, with two documented residual misses; those numbers describe the included suite only.

---

6. Framework mapping (this sample)

Finding	OWASP LLM 2026	OWASP Agentic 2026	NIST AI RMF
F-1, F-4 HTTP auth posture	LLM03 Excessive Agency	ASI03 Identity and privilege abuse	Map (system boundary / network exposure); Measure (auth control tested and documented)
F-2 Missing rate limit	LLM06 Unbounded Consumption	ASI08 Cascading failures (resource pressure)	Measure (resource / abuse controls)
F-3 `copy_file` sink flag	LLM03 / LLM10 (output leading to a write)	ASI02 Tool misuse (if a model is over-privileged)	Measure (control exists: `validatePath`; residual risk documented)
Policy overlay	LLM03 Excessive Agency	ASI02, ASI03, ASI05, ASI06, ASI07	Govern / Manage recommendations only

Entries not assessed in this sample (and marked as such): LLM05, LLM07, LLM08, LLM09; ASI01 (no live injection test against a model); ASI09; ASI10; MCP008 lockfile CVEs for the monorepo; live probe drift.

---

7. Recommendations (illustrative)

1. Treat Streamable HTTP as a different threat model from stdio. Keep the loopback default unless there is a reason not to; if `HOST` is widened, add an application-level credential and a rate limit.
2. Do not disable path validation, and do not treat a clean MCPSentry report — or a downgraded MCP004 — as proof the sandbox is complete.
3. If an agent is going to call these tools, put a default-deny policy in front (AgentGate or an equivalent), with approval hooks on move/delete and on any tool that can exfiltrate.
4. In a client engagement, re-run MCPSentry on the **deployed** revision, include the lockfile in scope, and add a `tools/list` probe.

---

8. Limitations

- Static analysis is heuristic, not sound taint analysis. Wrappers and generated code can hide or invent a sink. MCPSentry says this in its own threat model.
- A clean report is not a guarantee of safety. This sample is the opposite case: it has findings, and some of them changed after a human read the handler.
- No tools were invoked. No exploit, proof-of-concept, or live bypass was attempted.
- OSV / MCP008 was not applied to the monorepo lockfile in the filesystem-only path.
- Playwright MCP was not scored, for the source-layout reason above.

Appendix A — scanner summary tables

Copied from the raw Markdown reports, unchanged.

Filesystem

- Scanned: `/tmp/mcp-targets/mcp-servers/src/filesystem`
- When: 2026-11-23T01:27:22.753Z (22 November 2026, 5:27 p.m. PT)

- Tools found: 16
- Findings: 2 (MCP004 high, MCP007 medium)

Memory

- Scanned: `/tmp/mcp-targets/mcp-servers/src/memory`
- When: 2026-11-23T01:28:08.872Z (22 November 2026, 5:28 p.m. PT)
- Tools found: 9
- Findings: 2 (MCP006 high, MCP007 medium)

Prepared 22 November 2026, America/Los_Angeles, by Kevin Fairley. Scanner: MCPSentry 0.1.0. This PDF is a rendering of sample-audit-report.md.