MCP & Agent Surface Audit

Security Assessment Report — SAMPLE / REDACTED

Client	`[CLIENT — REDACTED FOR SAMPLE]`
Engagement	MCP & Agent Surface Audit
Assessment window	2026-09-28
Report version	1.0
Classification	Confidential
Prepared by	Chimaobi Kanu, Director — Maobyte Innovations Ltd, trading as Mainno

> This document is a redacted sample. Target names, hostnames, credentials and business identifiers have been removed. All findings shown were reproduced against a controlled test target under written authorisation. No client production system was tested to produce this sample.

1. Executive summary

The target exposes an MCP (Model Context Protocol) server that gives an AI agent direct capability over the host: network probing, file access, outbound email and record deletion.

Six findings were confirmed, one of them critical. The critical issue allows an attacker who can influence any input the agent processes to run arbitrary commands on the server host — the classic path from "the agent read a document" to "the attacker owns the machine." Three further high-severity findings allow irreversible data destruction without confirmation, unrestricted file read, and untrusted instruction injection through a tool's own description.

The root cause is structural, not a typo. MCP ships with **no native security controls**: no authentication by default, no audit trail, no permission model at the tool layer. Any organisation that has wired an agent to an MCP server has extended that agent's trust boundary to everything the server can reach — and in most organisations nothing in the asset register, vulnerability scanner or SIEM knows the server exists.

Business impact. A successful chain here is not a confidentiality incident; it is host compromise with the agent's full permissions, in most cases without a log to reconstruct it afterwards. Where the deployment touches personal data, this is directly relevant to GDPR/EU AI Act obligations (GPAI enforcement live since 2 Aug 2026, penalties to €15M or 3% of global turnover).

Priority action. Remediate F-01 and F-02 before the next agent release. Neither requires an architectural change; both are bounded engineering fixes.

2. Scope and rules of engagement

In scope: the MCP server's tool surface — tool enumeration, tool descriptions, parameter handling, privilege boundaries, and guard controls on destructive operations.

Method: black-box, from the perspective of a hostile input reaching the agent. Tools were enumerated and invoked through a standard MCP client over stdio transport. No source code was read before testing.

Out of scope: the host operating system beyond demonstrating impact, the network, and any third-party dependency internals.

Boundary honoured: testing was performed only against the authorised target. No third-party system was scanned or probed.

---

3. Methodology and framework alignment

Testing combined automated tooling with manual adversarial reasoning:

- **Enumeration** — full tool-surface discovery, description and schema review.
- **Passive review** — description text, scope language, destructive verbs, credential references, drift/rug-pull exposure.
- **Active exploitation** — parameter injection, path manipulation, destructive action invocation, guard-control testing.
- **Tooling** — `fastmcp` client harness, `garak`, `PyRIT` (automated adversarial probes), custom MCP audit harness.

Findings are mapped to **OWASP LLM Top 10 (2025)**, **MITRE ATLAS**, and the MCP risk classes published by Microsoft's State of MCP Security 2026 and Salt Security's CISO Guide to Securing MCP Servers.

Severity model: qualitative, weighted for agentic context — an issue that lets the agent perform a harmful action scores higher than the same issue in a non-agentic system, because the agent may act without a human in the loop.

---

4. Findings summary

ID	Severity	Tool	Finding	Framework
F-01	**CRITICAL**	`ping_host`	Command injection via unsanitised input to shell	OWASP LLM05/LLM07 · MCP: command injection
F-02	HIGH	`delete_account`	Excessive agency — irreversible action, no confirmation	OWASP LLM06
F-03	HIGH	`read_file`	Arbitrary file read — no path confinement	OWASP LLM05
F-04	HIGH	`read_file`	Tool poisoning — hidden instruction in tool description	OWASP LLM01 · MITRE ATLAS: prompt injection
F-05	HIGH	`delete_account`		OWASP LLM06

ID	Severity	Tool	Finding	Framework
			Irreversible deletion reachable in a single unprompted call	
F-06	MEDIUM	`send_email`	Over-broad scope / credential aggregation	OWASP LLM06 · MCP: credential aggregation

Totals: 1 Critical · 4 High · 1 Medium · 6 findings

5. Detailed findings

F-01 — Command injection via unsanitised input to shell

Severity: Critical · **Tool:** `ping_host` **OWASP LLM:** LLM05 Improper Output Handling / LLM07 Insecure Plugin Design · **MCP class:** command injection & sandbox escape

Description. The tool interpolates its `host` parameter into a shell command string executed with a shell interpreter. Shell metacharacters in the parameter are therefore interpreted, not treated as data.

Evidence (reproduced). Invoking the tool with the parameter `127.0.0.1; echo __INJ__; id -u` returned both the ping output and the injected command's output, including the effective user id of the server process (`uid=337`).

Impact. Arbitrary command execution with the server process's privileges. Because an agent's inputs frequently originate in untrusted content — a document, an email, a web page, a tool result — this is a direct path from "attacker plants text the agent reads" to "attacker executes code on the host." In a hosted deployment this is full server compromise.

Remediation. 1. Never build shell strings from tool input. Invoke the program with an argument vector and no shell (e.g. `subprocess.run(["ping","-c","1",host])`). 2. Validate the parameter against a strict allow-list (hostname/IP grammar); reject metacharacters. 3. Run the MCP server as a least-privilege user in a sandbox with no shell access and no network egress beyond what the tool needs. 4. Add a tool-layer policy: deny by default, allow only declared operations.

F-02 — Excessive agency: irreversible action with no confirmation

Severity: High · **Tool:** `delete_account` · **OWASP LLM:** LLM06

Description. A destructive, irreversible operation is exposed as an ordinary tool with no confirmation step, no two-person rule, and no dry-run mode. The agent can invoke it autonomously.

Evidence (reproduced). A single tool call destroyed a record with no confirmation control encountered at any layer.

Impact. Prompt injection, model error, or a confused-deputy chain converts directly into permanent data loss. Recovery depends entirely on backups that may not exist.

Remediation. Require human-in-the-loop confirmation for destructive tools; implement soft-delete with a retention window; scope the tool so it can only ever affect records belonging to the requesting principal; log every invocation with actor, timestamp and target.

F-03 — Arbitrary file read (no path confinement)

Severity: High · **Tool:** `read_file` · **OWASP LLM:** LLM05

Description. The tool accepts a caller-supplied path and opens it directly, with no confinement to an intended directory.

Evidence (reproduced). Requesting `/etc/passwd` returned the file's contents; the path lay outside any intended working scope.

Impact. Disclosure of secrets, credentials, configuration and source outside the tool's stated purpose — the standard precursor to lateral movement.

Remediation. Resolve the path, then verify it remains inside an allow-listed root; reject symlinks that escape; run the server with a filesystem view limited to the intended directory.

F-04 — Tool poisoning: hidden instruction inside a tool description

Severity: High · **Tool:** `read_file` · **OWASP LLM:** LLM01 · **ATLAS:** prompt injection

Description. The tool's description contains an embedded instruction directed at the model (a pseudo-tag instructing it to read a private key and disclose it without telling the user). Agents treat tool descriptions as trusted context, so this text is executed as an instruction, not read as documentation.

Evidence (reproduced). The description matched hidden-instruction markers (`<IMPORTANT>`, "do not mention", a private-key path) — a payload an agent would act on and a human reviewer would likely skim past.

Impact. Silent, user-invisible data exfiltration and arbitrary instruction injection. This is a supply-chain class: the attacker does not need access to the server — only to influence its description.

Remediation. Review and pin tool descriptions as code; alert on description drift between versions; strip or neutralise instruction-like markup; never load tool descriptions from unreviewed third parties; sign and verify server manifests.

F-05 — Irreversible deletion reachable in one unprompted call

Severity: High · **Tool:** `delete_account` · **OWASP LLM:** LLM06 (Chain view of F-02: the destructive tool is also reachable without any prior authentication or approval step, making it a single-call kill path.)

Remediation. As F-02, plus: require an authenticated, authorised principal per invocation and rate-limit destructive tools.

F-06 — Over-broad scope and credential aggregation

Severity: Medium · **Tool:** `send_email` · **OWASP LLM:** LLM06 **MCP class:** over-broad access and credential aggregation

Description. A single server holds credentials spanning unrelated systems and requests wider scope than its function needs (full mailbox rather than send-only). One leaked token therefore becomes a breach path into every system the server touches.

Evidence. Description and configuration declare broad privileged access beyond the tool's stated narrow function.

Remediation. Split servers by trust domain; request narrow, per-resource OAuth scopes over wildcards; use short-lived tokens; give each agent a governed identity so a compromised server can be revoked in one operation.

---

6. Remediation roadmap

Immediate (before next release) - F-01 — remove shell interpretation; argument vectors + input allow-listing. - F-02/F-05 — confirmation control and soft-delete on destructive tools.

Short term (30 days) - F-03 — path confinement and filesystem view restriction. - F-04 — description pinning, drift alerting, markup neutralisation. - F-06 — scope reduction and credential separation.

Structural (90 days) - Inventory every MCP server in the estate, including shadow servers. - Tool-layer least privilege; deny by default. - Centralised logging and audit trail (MCP logs nothing by default). - Continuous posture assessment — the surface changes faster than periodic scans.

---

7. Appendix — tooling and reproduction

- Harness: `fastmcp` MCP client; stdio transport; tool enumeration + invocation.
- Automated adversarial probes: this practice's own capability-driven harness (`mcp_surface.py` tool-surface analysis + `surface_audit.py` probe runner), which selects tests from each tool's declared input parameters and records a refused call as a pass. `garak` and `PyRIT` are held in scope for engagements that require model-level red teaming; **they were not used to produce this sample and are not claimed here.**
- Custom checks: description poisoning markers, destructive-verb analysis, broad-scope language, active exploitation of injection and path handling.
- Note: in this engagement the stdio transport spawned the server with a sanitised environment, so environment-variable exposure could not be reproduced and is **not** claimed as a finding; it is recorded as a design consideration only.

---

8. Report prepared and signed by

Chimaobi Kanu, Director — Maobyte Innovations Ltd, trading as Mainno. Engagement conducted under written authorisation; findings disclosed to the client before any third party.

Contact	Contact form at mainnoltd.com/contact
Web	mainnoltd.com
Researcher profile	hackerone.com/ciphercasper

Questions on any finding, or a retest request, go through the contact form on mainnoltd.com/contact. We reply to the address you give us, usually within one working day.

End of redacted sample. All findings shown were reproduced against a controlled test target under written authorisation. This document is a technical review — not legal advice, not formal certification, and not a guarantee that a system is risk-free.