NyquistAI

Penetration Testing Report

Execution Date: 09/28/2026 - 10/01/2026
Report Date: 10/01/2026

1. Assessment Details

1.1 Scope

The scope of the assessment is shown below. Systems and applications not listed have not been pentested.

Asset	Type
https://mcp-test.nyquistai.com/agent/mcp	MCP server

1.2 Purpose

This document defines the methodology used to conduct a penetration test of the Model Context Protocol (MCP) server operated by NyquistAI. It records the standards followed, the scope and boundaries of testing, the control areas assessed, the tooling used, and the way findings are rated and evidenced.

It is written to serve as compliance evidence: it demonstrates to the requesting client, and to any auditor acting on their behalf, that the assessment was systematic, repeatable, and aligned to recognised security-testing standards rather than performed ad hoc.

1.3 Standards and frameworks

The assessment methodology is aligned to the following references:

- **NIST SP 800-115**, Technical Guide to Information Security Testing and Assessment — overall test process (planning, discovery, attack, reporting).
- **Penetration Testing Execution Standard (PTES)** — engagement structure and phase definitions.
- **OWASP Application Security Verification Standard (ASVS)** — control verification baseline for authentication, access control, and input handling.
- **OWASP API Security Top 10 (2023)** — API-layer risk taxonomy (notably broken object-level and function-level authorization).
- **OWASP Top 10 for LLM / Agentic Applications** — agentic-specific risks (prompt injection, tool poisoning, excessive agency) relevant to MCP tooling.

- **MCP Authorization Specification (version 2025-11-25)** — protocol-level authorization requirements, including OAuth 2.1 resource-server behaviour, RFC 9728 Protected Resource Metadata, and RFC 8707 Resource Indicators.
- Supporting RFCs: **OAuth 2.1** (draft), **RFC 9728** (Protected Resource Metadata), **RFC 8707** (Resource Indicators), **RFC 8414** (Authorization Server Metadata), **RFC 7636** (PKCE).

1.4 Test environment and fidelity

Testing was performed against **a staging environment configured to mirror production on all security-relevant parameters** (framework and dependency versions, TLS termination and any fronting proxy, Auth0 configuration model including audience/resource identifiers, and the full tool set with identical input schemas). Production was not subjected to intrusive testing.

A minimal, non-intrusive production smoke test was performed to confirm that the production TLS certificate/chain, the production Protected Resource Metadata document, and production audience-binding behaviour match what was validated in staging.

1.5 Limitations

- The assessment is a point-in-time evaluation; subsequent code or configuration changes may alter the security posture;
- Testing was conducted against staging; conclusions transfer to production to the extent the environments match;
- Blind vulnerabilities (e.g. blind SSRF) are only detectable where an out-of-band interaction was observed; absence of an OOB signal does not prove absence of the class;
- Auth0 is treated as a trusted identity provider; its internal security is assumed and not tested.

2. Approach

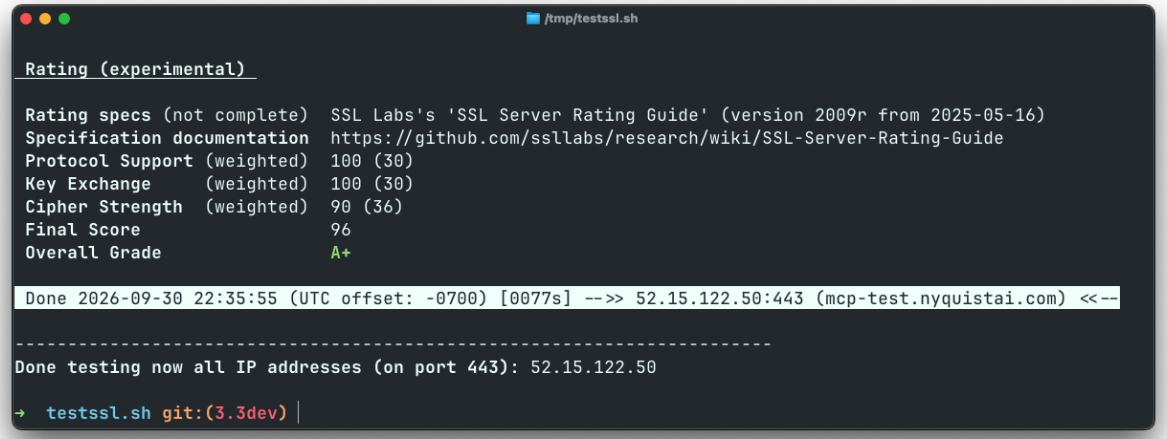
The assessment was conducted as a grey-box, authenticated test: the testing team operated with valid credentials and knowledge of the system architecture, enabling thorough coverage of authenticated attack surfaces within the test window.

3. Results

3.1 Transport security

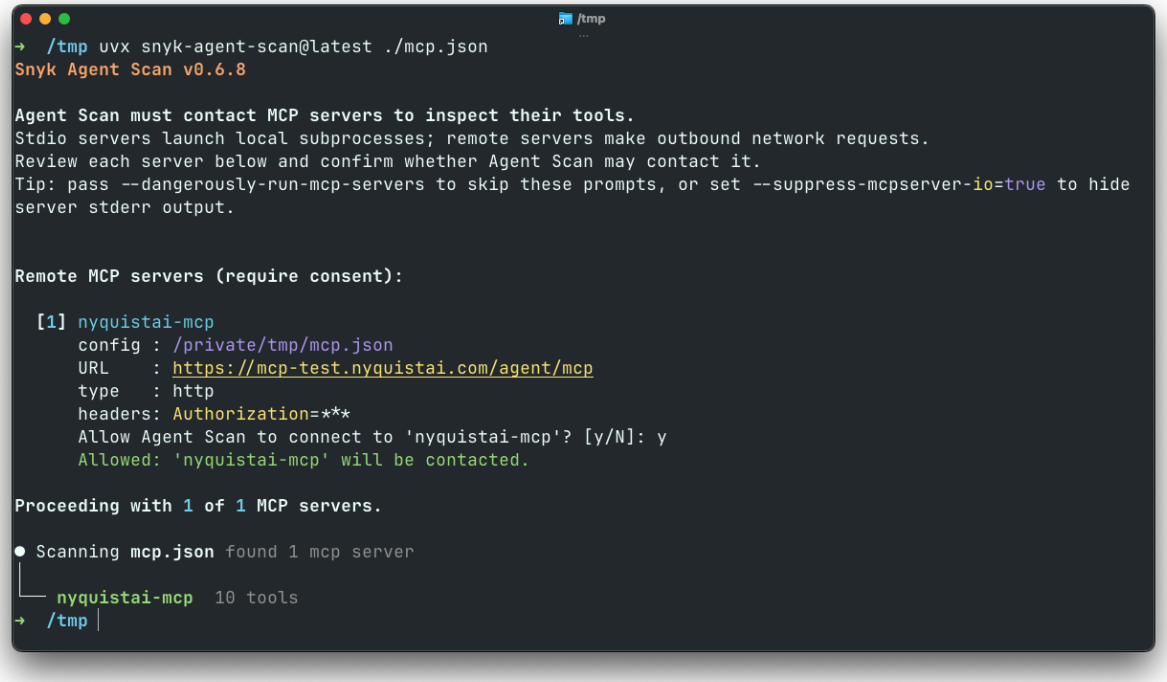
- Techniques / test cases: TLS version/cipher/cert-chain validation; plaintext-fallback and HSTS checks; CORS / Origin handling

- Tool: testssl.sh
- Results: A+



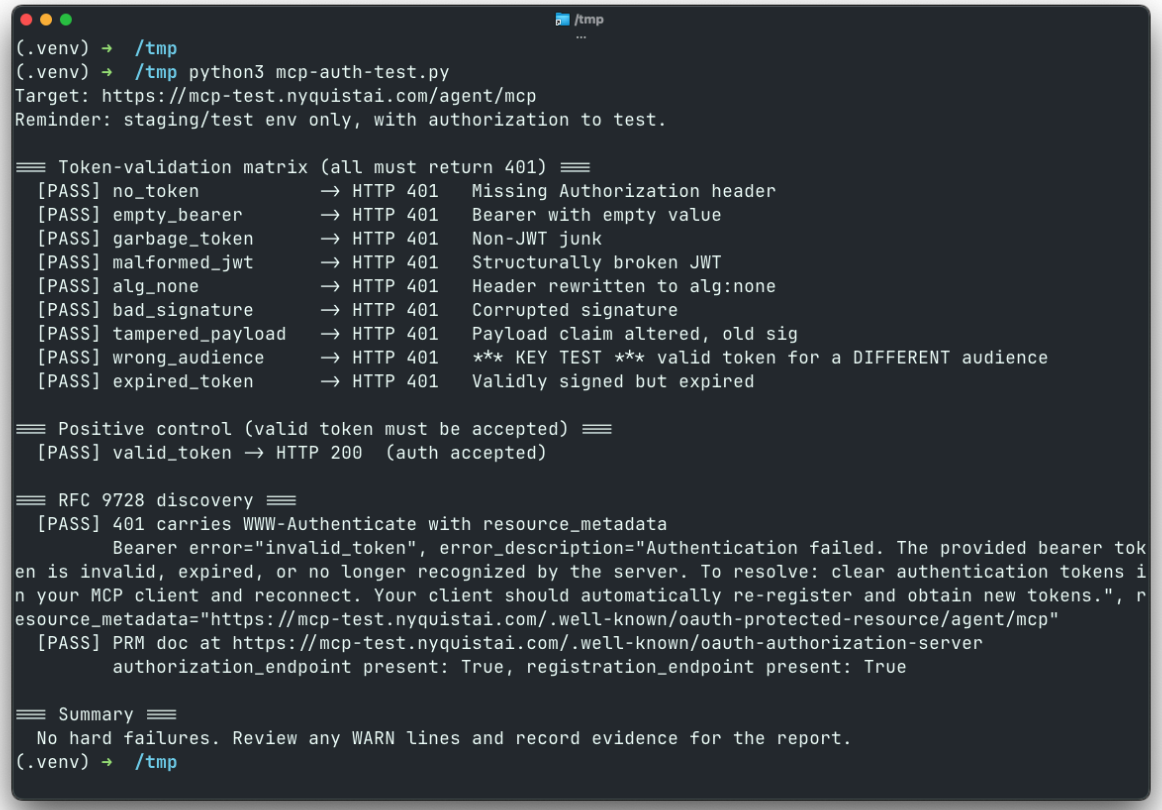
3.2 MCP protocol-specific

- Techniques / test cases: Tool-description scanning for prompt injection, tool poisoning, tool shadowing, toxic flows, rug-pull; Streamable HTTP session handling
- Tool: agent-scan (https://github.com/snyk/agent-scan)
- Results: cleared



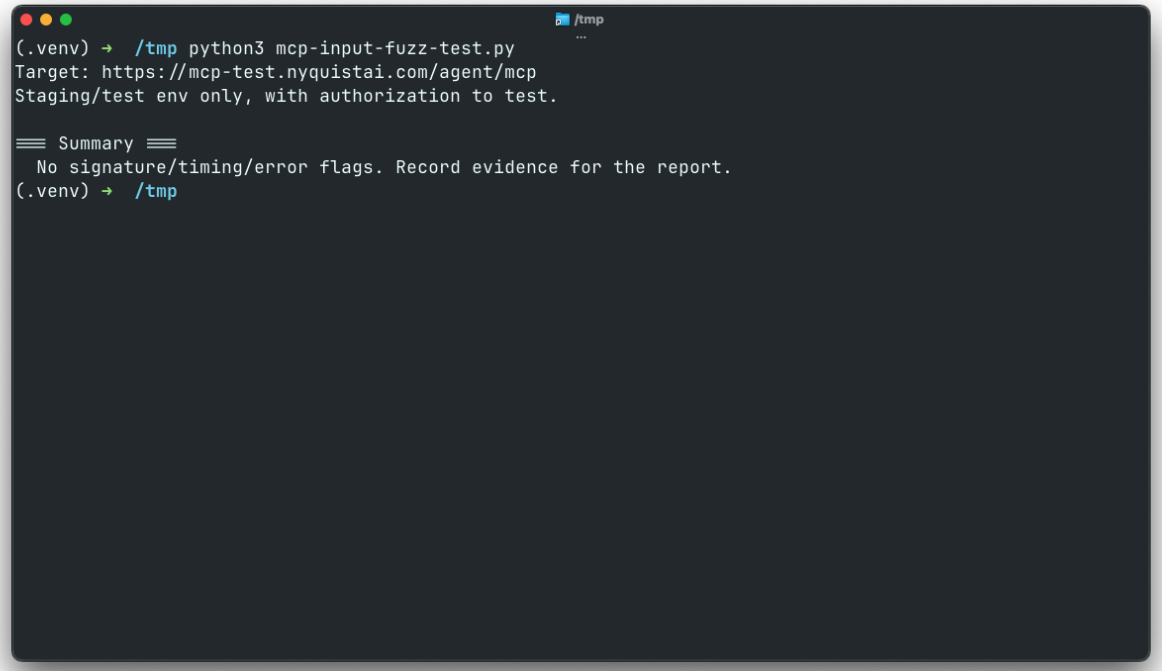
3.3 Authentication

- Techniques / test cases: Token-validation matrix (missing, malformed, alg:none, tampered signature, tampered payload); audience/resource binding & token-passthrough rejection; expiry and issuer enforcement; RFC 9728 discovery correctness; PKCE / flow hygiene
- Tool: MCP Inspector (https://github.com/modelcontextprotocol/inspector) and custom harness code (mcp-auth-test.py)
- Results: passed



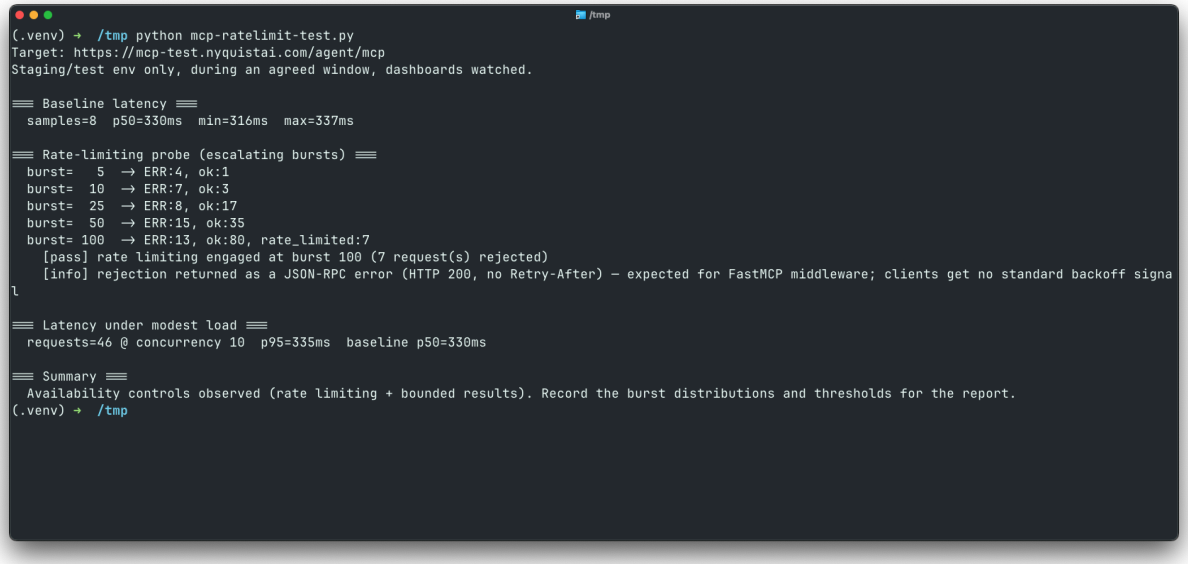
3.4 Tool input validation

- Techniques / test cases: SQL/NoSQL injection; command injection; template injection; schema/type abuse; schema-enforcement check
- Tool: MCP Inspector (https://github.com/modelcontextprotocol/inspector) and custom harness code (mcp-input-fuzz-test.py)
- Results: passed



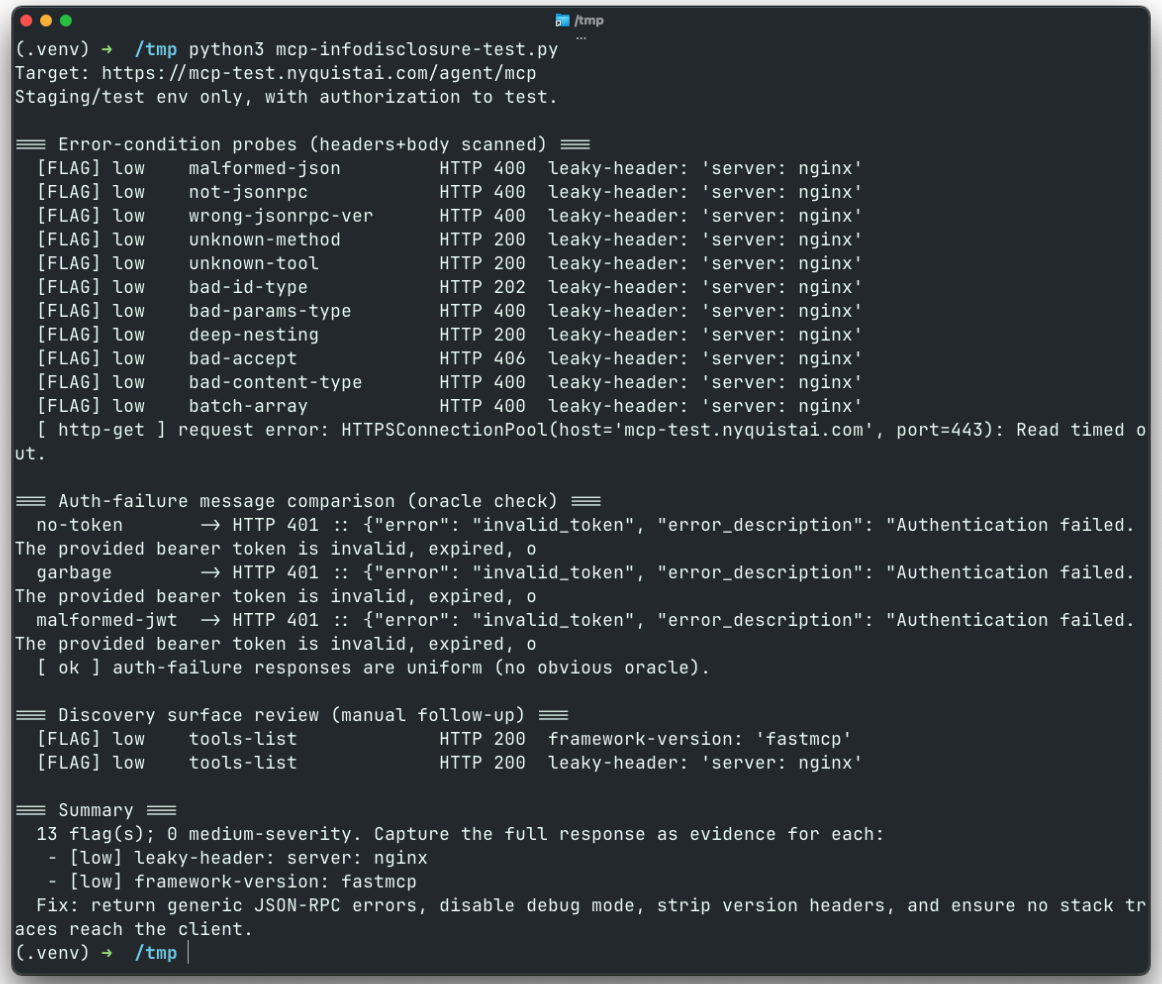
3.5 Resource exhaustion / availability

- Techniques / test cases: Rate-limit threshold measurement (escalating bursts, Retry-After, per-token vs per-IP); latency-under-load
- Tool: MCP Inspector (https://github.com/modelcontextprotocol/inspector) and custom harness code (mcp-ratelimit-test.py)
- Results: passed



3.6 Information disclosure / error handling

- Techniques / test cases: Error-condition probes (malformed JSON, bad JSON-RPC, unknown method/tool, type abuse, odd headers); stack-trace / version / path / DB-error / debug-mode detection; auth-failure oracle check; discovery-surface review
- Tool: MCP Inspector (https://github.com/modelcontextprotocol/inspector) and custom harness code (mcp-infodisclosure-test.py)
- Results: two minor issues identified and marked as accepted



The two minor issues identified are:

- [low] leaky-header: server: nginx
- [low] framework-version: fastmcp

The “**Server: nginx**” are response headers returned by the MCP server. And each tool object returned by tools/list includes a FastMCP-specific metadata namespace under the standard MCP _meta field, in the form **"_meta": {"fastmcp": {"tags": []}}**.

The only information disclosed is the identity of the server framework. Framework identification provides marginal reconnaissance value and does not, by itself, grant access or enable a specific attack. No confidentiality impact on protected assets. Access additionally requires a valid authenticated token.