Shadow AI Discovery

SENSOR DETECTION ACCURACY BENCHMARK

How reliably the sensor spots shadow AI on an endpoint.

The live endpoint collectors were run against a controlled host seeded with known AI artifacts — API keys, MCP configs, downloaded skills, LLM SDK manifests and agent CLIs — alongside clean files that must stay silent.

COLLECTORS	PLANTED ARTIFACTS	METHOD
5 filesystem / PATH	**27**	**Live collect(), fixture host**

CONFIDENTIAL · PREPARED 13 AUGUST 2026 · v2 (measured)

EXECUTIVE SUMMARY

Full recall across every collector, with no false positives.

100%
MEAN RECALL AFTER TUNING

96%
MEAN RECALL AT BASELINE

0
FALSE POSITIVES (CLEAN FILES)

27
PLANTED ARTIFACTS MEASURED

A throwaway home directory was populated with known shadow-AI artifacts plus clean control files. Each collector's actual detection code was run against it in isolation — the numbers below are exactly what the sensor reported.

RECALL BY COLLECTOR — BASELINE VS. TUNED

COLLECTOR	CASES	BASELINE	TUNED	GAIN	STATUS
API Key Discovery	7	100%	100%	—	PASS
MCP & Agent Config Discovery	9	78%	100%	▲ +22	PASS
Downloaded Skill Discovery	3	100%	100%	—	PASS
LLM SDK / Dependency Discovery	3	100%	100%	—	PASS
AI CLI Tool Discovery	5	100%	100%	—	PASS

Recall = planted artifacts detected ÷ planted. False positives = findings raised against the clean control files. Four collectors were already exact at baseline; MCP config discovery had one real gap, fixed overleaf.

DEFECT FOUND & FIXED

The benchmark surfaced a real detection gap.

Before 77.8%

An MCP config file sitting directly at the home root — the common `~/.mcp.json` convention used by agent tooling — was never detected. The collector walked known sub-directories and a fixed list of relative paths, but never checked the home root itself by filename. Both the config and the MCP server it declared were missed.

After 100%

Added an explicit home-root filename check for every known MCP config name (`.mcp.json`, `claude_desktop_config.json`, and the rest). Bounded — a direct name lookup, not a filesystem walk — so it adds coverage without cost. Re-measured at full recall, clean files still silent.

This is exactly what the benchmark is for: a plausible-looking collector that quietly missed a mainstream config location. Caught by ground truth, fixed with a scoped change, and verified by re-measurement — the baseline figure in this report was regenerated with the fix reverted so the improvement is shown honestly.

WHAT EACH COLLECTOR PROVES

- **API Key Discovery** — provider keys in shell profiles and .env files, matched by both value-shape and variable name; Anthropic keys are not mislabeled as OpenAI.
- **MCP & Agent Config Discovery** — Claude / Cursor / root MCP configs and the individual servers they declare, plus downloaded SKILL.md files.
- **Downloaded Skill Discovery** — SKILL.md packs in fixed skill roots and in Downloads / Documents.
- **LLM SDK / Dependency Discovery** — openai / anthropic / langchain / litellm declared in requirements.txt, package.json and pyproject.toml.
- **AI CLI Tool Discovery** — agent and LLM CLIs on PATH, without flagging ordinary developer tools.

METHODOLOGY

How this benchmark works — real measurement, not estimates.

Fixture host	A throwaway home directory is populated with known planted shadow-AI artifacts plus clean control files that must stay silent.
Live collectors	Each collector's actual detection routine runs in an isolated subprocess pointed at the fixture — no finding is hand-written.
Recall	Planted artifacts detected divided by planted — the headline detection figure per collector.
False positives	Findings raised against the clean control files, which must produce nothing.
Scope	The five filesystem / PATH collectors have clean ground truth and are benchmarked here. Runtime collectors — live network egress, DNS and installed desktop apps — are validated separately against a controlled live host.
Tuning loop	Run → measure → fix the misses in the collector or signature set → re-run → record a new iteration. Goal: ≥99% recall with minimal false positives.

> The companion workbook ShadowAI-Sensor-Benchmark.xlsx lists every planted test case, per iteration.

Numbers in this document are generated directly from the benchmark result files; regenerating the report re-reads the measured output, so the PDF cannot drift from what the harness actually recorded.