The SecureSkill Capability Report: Edition 1

Edition 1 · Measurement: 2026-05-17

SecureSkill is the pre-install security layer for agent skills. Before a skill runs in your agent, the scanner inspects its files, traces what it would do, and assigns a verdict: SAFE, CAUTION, or BLOCK. The scanner runs a three-phase, ten-layer architecture: six deterministic-evidence layers strip obfuscation and extract verifiable findings from the skill's files; two correlation layers cross-reference those findings to surface dangerous combinations no single check would catch; two AI-reasoning layers then interpret the assembled evidence and assign the final verdict. Evidence is established before interpretation, and verdicts map to recognized vocabularies (OWASP ASI Top 10, OWASP LLM Top 10, NIST AI RMF, MITRE ATLAS, EU AI Act) that enterprise security teams already use. This brief reports the first end-to-end measurement of that scanner against 250 fixtures across five agentic frameworks, each constructed independently of the scanner using attack techniques drawn from MITRE ATLAS, OWASP LLM and ASI Top 10, CVE corpora, and public red-team research.

THE SETUP

Five frameworks. Fifty fixtures each. Two hundred and fifty in total.

OpenClaw, Claude (Anthropic), Hermes, OpenAI Codex, and Cursor each have a distinct skill file format and a distinct attack surface. A skill that compromises a Claude agent has different file structures and different injection points than a skill that compromises Codex or Cursor. The benchmark covers all five.

The 250 fixtures are functioning skill packages, not stubs or contrived blobs. Each one has the file shape its host framework actually loads: `_meta.json` plus `hooks/handler.js` files for OpenClaw, YAML frontmatter in `SKILL.md` for Claude, the `metadata.hermes:` block for Hermes, MCP yaml references and `agents/openai.yaml` for Codex, and the `.cursor/` directory structure for Cursor. A skill in the corpus could be uploaded to its target framework and would execute. That matters: scoring a security tool against fake or simplified samples is the easiest way to inflate a number.

METHODOLOGY IN BRIEF

Fixtures were constructed under attacker-perspective discipline. Scanner internals (YARA rules, extraction logic, scoring anchors) were not consulted during construction. Attack techniques were drawn from external sources: MITRE ATLAS, OWASP LLM Top 10, OWASP ASI Top 10, CVE corpora, public red-team writeups, academic incident reports. The corpus was not built backward from "what the scanner can already detect." This is the core credibility lever of the benchmark: the scanner was not graded against its own training signal.

Each fixture is one of three classes:

- **SAFE.** Legitimate skill, zero network exfil, scoped behavior. Examples: file format validators, local cryptographic utilities, configuration auditors.
- **CAUTION.** A real concern that requires user awareness but is not necessarily malicious. Examples: skills that read credentials with a defensible purpose, skills that send data to known network destinations.
- **BLOCK.** Actual attacks: deceptive `SKILL.md` instructions paired with malicious payloads in supporting files (yaml configs, hook handlers, scripts, references, lockfiles). Each framework's attack surface is distinct, so coverage is framework-specific.

Per-framework distribution: 50 fixtures each, split 10 SAFE / 20 CAUTION / 20 BLOCK. Each set spans multiple attack families (configuration hijack, supply-chain abuse, credential exfiltration, cross-agent contamination, persistence, prompt injection) rather than concentrating on a single technique. Representative examples in the corpus:

Framework	Representative attack classes covered
OpenClaw	Cross-file payload splitting, hook-handler exfiltration, container-escape attempts, resource abuse (keyloggers, worm propagation), DAN-style evasion framing
Claude	`settings.json` injection (CVE-2025-59536 class), MCP tool shadowing, subagent spawning via `context: fork`, `PostToolUse` hook exfiltration, memory poisoning via persistent `SKILL.md` state
Hermes	`metadata.hermes:` frontmatter abuse, gateway exfiltration, DNS tunnel exfiltration, cross-agent contamination via the message bus, worm propagation across Hermes agents
Codex	`agents/openai.yaml` extra-tool injection beyond declared scope, `dependencies.tools` MCP server hijack, `display_name` brand-spoofing with mismatched MCP domain, cross-ecosystem poison (writes to `.cursorrules` or `CLAUDE.md` from Codex context)
Cursor	`.cursor/hooks.json` inline-hook abuse, cross-primitive chains (hook plants rule, rule references hook), MCP stdio command spawn from relative paths, prompt-injection evasion (ignore-previous, system-redefine, Cyrillic), brand-spoofing fixtures (fake Sentry / Vercel / GitHub)

A scanner run produces three outputs that get compared against the fixture's expectations: a verdict (SAFE / CAUTION / BLOCK); a numeric score from 1 to 10 mapped to those verdicts in fixed bands (SAFE: 1-3, CAUTION: 4-6, BLOCK: 7-10), with each fixture specifying a tighter expected range within its band like `[7, 9]` for a top-tier BLOCK; and a list of finding-category labels like `["credential_harvesting", "scope_mismatch"]`. The three pass tiers in this brief check those outputs at progressively stricter levels, and they are not equal in weight.

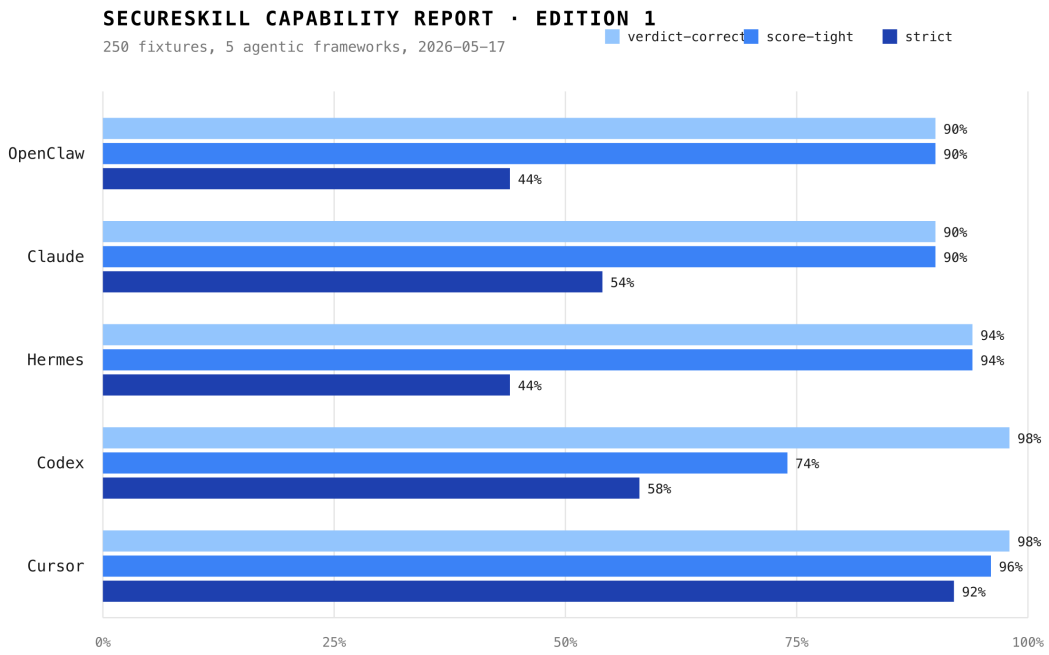
Tier	Rank	What must match	What this measures
Verdict-correct	Primary	Verdict only	Capability. Did the scanner identify the risk level correctly?
Score-tight	Secondary	Verdict ' AND score lands inside the fixture's narrower expected range	Precision. Did the score land where the corpus expected, not just in the correct verdict bucket?
Strict	Tertiary	Verdict ' AND score ' AND every expected finding-category label appears	Bookkeeping. Did the scanner describe the finding using the same category words the corpus authored?

Verdict-correct is the real capability measure. Score-tight adds a precision constraint that depends on how narrowly the corpus authored its expected score ranges. Strict adds a vocabulary constraint that depends on whether the scanner's category labels match the exact words the corpus expected, regardless of whether both describe the same attack accurately. A reader looking for "is the scanner catching attacks?" should look at verdict-correct. Strict is useful as a corpus-drift signal but does not measure scanner capability.

Limitations stated up front. These 250 fixtures are clean-context Claude-authored skills, not real-world adversarial samples gathered from production traffic. A determined attacker drafts payloads designed to evade the specific scanner they are targeting; clean-context fixtures, no matter how varied, cannot measure that. This brief reports capability against documented attack classes, not robustness against novel, scanner-aware adversaries. That workstream is separate.

THE FINDINGS

The scanner produced the right verdict on 235 of 250 fixtures (94.0%). That is the headline measurement.



Aggregate results across all 250 fixtures:

- Verdict-correct: **94.0%** (235 / 250)
- Score-tight: **88.8%** (222 / 250)
- Strict: **58.4%** (146 / 250)

Two false negatives, thirteen false positives. The scanner skews cautious: it is much more likely to over-flag a benign skill than under-flag a malicious one. For a pre-install audit tool, that is the right side to err on, though the false-positive volume affects user experience.

The most interesting number in the data is the gap between tiers, not the headline itself. Verdict-correct sits at 94.0%. Strict sits at 58.4%. Of the 35.6-point gap, almost none is capability failure.

The 13-fixture drop from verdict-correct to score-tight breaks down as: verdict ' ,score landed in the right bucket, but the model's score sat at the edge of the bucket rather than the middle of the fixture's narrower expected range. Twelve of those thirteen are Codex, whose fixtures were constructed with tighter expected ranges than the other four frameworks. The verdict was always correct.

The 76-fixture drop from score-tight to strict is label drift. One fixture in the corpus flagged a skill that exfiltrates project state to a configurable endpoint, and the fixture metadata expected the scanner to emit the finding-category labels `["scope_mismatch", "data_exfiltration"]`. The scanner emitted `["data_exfiltration", "credential_harvesting"]`. The verdict matched, the score matched, both lists describe the same attack accurately. The fixture wanted "the skill is acting outside its declared scope"; the scanner said "the skill is stealing credentials." Both descriptions are correct, both terms appear in MITRE ATT&CK and the OWASP taxonomies, and a security analyst reading either output would correctly identify the threat. Per the adjudication policy documented in `final-prelimtesting.md` §2.1, this is fixture metadata that needs updating against current model vocabulary, not a scanner failure.

Seventy-six of the 250 fixtures show this pattern. They look like 31 percentage points of "scanner is wrong." They are 31 percentage points of bookkeeping debt between a corpus constructed months ago and a model whose category vocabulary has shifted since.

FRAMEWORK COMPARISON

Framework	Verdict-correct	Verdict miss	Score-tight	Strict
OpenClaw	90% (45/50)	5	90% (45/50)	44% (22/50)
Claude	90% (45/50)	5	90% (45/50)	54% (27/50)
Hermes	94% (47/50)	3	94% (47/50)	44% (22/50)
Codex	98% (49/50)	1	74% (37/50)	58% (29/50)
Cursor	98% (49/50)	1	96% (48/50)	92% (46/50)
Total	**94.0% (235/250)**	**15**	**88.8% (222/250)**	**58.4% (146/250)**

Three observations from the per-framework breakdown.

Cross-framework parity holds. All five frameworks land within 8 points of each other on verdict-correct (90 to 98). The scanner does not have framework-specific blind spots; it identifies risk at comparable rates across distinct skill formats.

The single missed attack in the corpus is `block-lockfile-inject-001` on OpenClaw, a BLOCK-class lockfile-injection skill that scored CAUTION. It is named directly because it appears in the open scorecard at `phase1benchmark.md`.

Codex shows the widest gap between verdict-correct (98%) and score-tight (74%). Codex fixtures were specified with narrower expected score ranges than the other frameworks; the model's scores landed correctly inside the BLOCK and CAUTION buckets but at the edges of the fixture ranges rather than the middle. Every Codex BLOCK fixture was correctly blocked.

WHAT THIS MEANS

The scanner makes the right call on 235 of 250 fixtures with one genuine missed attack. That is a defensible safety floor for a pre-install audit tool. The thirteen false positives are the cost of skewing cautious, which is the correct skew for a tool whose job is to be the last check before a third-party skill executes in someone's agent context.

What 250 clean-context fixtures cannot measure is adversarial robustness. That is a separate workstream, separately measured, and this brief does not address it. Edition 2 will include adversarial-robustness probes alongside an expanded corpus and multi-sample stability checks.

The opinion this brief stakes plainly: **agentic-skill scorecards should separate "did the scanner make the right call" from "did the scanner describe the call in the words the corpus expected."** Conflating the two inflates capability numbers with vocabulary-match credit. The three-tier reporting in this brief is the operational fix.

The numbers here reflect sustained prompt iteration, per-framework calibration, regression sweeps run against every scanner change, and an adjudication policy that records every divergence rather than papering over it. The accuracy is built, not claimed.