StackOne Defender 0.8.2 in Practice: An Apache-2.0 Prompt-Injection Filter for Tool Calls — What Works, What Does Not

A bundled ONNX classifier now sits between your agent and its tools, catching a vendor-reported 88.7% of prompt injections. The interesting number is the other 11.3%.

What it is

StackOne Defender is an Apache-2.0 npm library that scans tool-call responses for indirect prompt injection before those responses reach the model context. Latest release is v0.8.2, dated 19 August 2026. Install is one command — `npm install @stackone/defender` — and the ONNX model ships inside the package, so there is no separate model download and no model host to configure. Architecture is two tiers: pattern and heuristic detection first, then an ONNX ML classifier, with the ML

tier enabled by default. That second tier requires two optional peer dependencies, `onnxruntime-node` and `@huggingface/transformers`. Package metadata shows a relicensing from SSPL-1.0 to Apache-2.0, which matters more than it sounds like it does — SSPL would have made this unusable for anyone embedding it in a hosted product.

The problem it solves

Indirect prompt injection through tool output is the live attack surface for agent deployments, and almost nothing sits in the right place to catch it. Your agent calls a tool. The tool returns a GitHub issue body, a Jira comment, a scraped page, an email thread. That text goes straight into the model's context window, where it is indistinguishable from instructions you wrote.

> Input guardrails inspect what the user typed. Model-level safety training inspects what the model intends. Neither of them looks at the untrusted text your tool just handed the model — which is exactly where the attack lives.

Defender's interception point is between the tool result and the model context. Per the StackOne docs, it protects agents using tool calls via MCP, CLI, or direct function calling, and scans responses on the way back. That is an output filter on tool responses, not a model-level or input-level guardrail. The distinction is the whole product — and the narrowness is correct, because the trust boundary in an agent system is not the user prompt, it is every byte of text that enters context from a system you do not control.

What it does well

It fits inline without infrastructure. The model is bundled and local: no network hop, no per-call API cost, nothing to provision. Agents make a lot of tool calls, and a hosted safety-model round trip on each one is a cost line and a latency ticket waiting to happen.

The two-tier design is sensible cost engineering. Cheap pattern matching catches the obvious cases; the classifier handles what patterns miss. Having Tier 2 on by default is the right call — an ML tier that ships off by default is an ML tier that

nobody enables.

Apache-2.0 removes the adoption blocker. Relicensing away from SSPL-1.0 means you can vendor this into a commercial agent product without legal review turning into a three-week project.

The transport-agnostic surface matters. MCP, CLI, and direct function calling all funnel through the same filter. StackOne's own integration layer exposes things like 98 GitHub actions to agents via MCP, A2A and SDK, with agent authentication and tool-calling execution — so the company is building this against its own attack surface, not as a detached research artifact. Dogfooding is weak evidence but it is not zero evidence.

Rough edges

The 88.7% figure is vendor-reported with no named dataset. Reviewing the public GitHub repo and StackOne docs turns up no published independent benchmark, no named evaluation dataset behind that number, and no third-party latency measurement. I am not calling the number wrong — I am saying you cannot check it, and neither can I.

88.7% is a failure rate, not a success rate, if you read it as a control. Roughly one in nine injections gets through. For a filter that is one layer among several, that is useful. For a filter you treat as a boundary, it is a false sense of security. I have written before about guardrails that fail when the payload is encoded rather than written in plain language; a classifier trained on injection text has an obvious blind spot for injections that do not look like text until the model decodes them.

> A detector at 88.7% is a filter. Treated as a boundary, it is a liability — because the 11.3% that passes now arrives pre-blessed.

Optional peer dependencies mean two install paths. If `onnxruntime-node` and `@huggingface/transformers` are not present, you are running on Tier 1 patterns alone. Verify in CI that the ML tier is actually loaded in production, because a silent fallback to heuristics is exactly the failure mode you would not notice until an incident review.

Scope is deliberately narrow. Defender is designed for tool-call output filtering, not general model safety or user-input jailbreak defense. If you are shopping for a single guardrail product, this is not it. If you already have input-side and policy-side controls and the tool-output gap is open, this fills that specific gap.

v0.8.2 is pre-1.0. Expect API changes.

How it relates to garak, and to permission design

The closest open-source counterpart people will reach for is NVIDIA's garak LLM vulnerability scanner, at v0.13.1 since 1 October 2025, installed with `python -m pip install -U garak` and requiring Python >=3.10 and <=3.12. But garak is a CLI scanner that probes models rather than an inline runtime guardrail — it tells you what your system is vulnerable to. Defender blocks traffic in the request path. They are complementary: garak in CI, Defender inline. The deeper alternative is not a filter at all. Cloudflare's approach in its open-sourced internal AI workspace — agents start with zero access and gain only task-specific permissions — attacks the same problem from the blast-radius side. My take: detection and least-privilege are additive, and if you can only do one, do least-privilege first. A successful injection that can only read one document is a nuisance; a successful injection with broad tool access is an incident.

On the vendor

StackOne is a UK company, London-headquartered with a registered office at Camburgh House, 27 New Dover Road, Canterbury CT1 3DN, England. It raised a $20M Series A that closed 6 May 2025, led by GV, with Workday Ventures, XTX Ventures, Episode 1 and Playfair participating, plus angels from OpenAI, DeepMind, Microsoft and MuleSoft. Total funding is reported between $23.6M and $23.91M across three to four rounds, including a $312K early-stage round on 2 June 2023 and a seed round on 29 November 2023. Defender is the open-source edge of a commercial integration platform — a stable-enough position, and Apache-2.0 means a fork survives any strategy change.

Verdict

My take: try it if you are running agents against untrusted tool output today and currently have nothing in that position. The install cost is one npm

command and two peer deps, and the model is bundled. Going from zero coverage to imperfect coverage on the tool-output path is a real improvement, and there are very few options that fit inline with no separate model host.

Skip it if you are looking for a general-purpose guardrail, if your agents only touch data you control end to end, or if your compliance posture requires independently verified detection rates — because those do not exist for this yet.

If you deploy it: assert in CI that Tier 2 is loaded, log every block with the triggering content so you can build your own evaluation set, and run garak against your assembled system separately. Do not let the presence of a filter justify widening tool permissions. That trade — more access because we have a detector — is how an 88.7% number turns into a breach.

I expect this deserves a revisit once someone publishes a third-party benchmark. Until then, treat 88.7% as a directional claim from a vendor with an obvious incentive, not as a measured property of your deployment.