EDAMAME

Runtime Security for Coding Agents

Technical brief — EDAMAME · PR-2026-002 · 2026-05-26

A path to securing Cursor, Claude Desktop, Claude Code, Codex, and OpenClaw: discover unmanaged agent footprints, anchor the host, verify divergence, and detect runtime vulnerabilities.

Frank Lyonnet, PhD — founder & CEO, edamame.tech (former INRIA researcher) · Minh Anh — founding engineer, edamame.tech (Stanford CS)

---

1 · Executive summary

Coding agents are becoming part of everyday software delivery. They read code, run shell commands, access tokens, install packages, and interact with external services. Workstations, runners, and self-hosted coding hosts now deserve the same serious security treatment as classical CI/CD choke points.

Hardening stays the precondition, and runtime verification closes the gap by aligning declared coding-agent intent with host-observable behavior: process ancestry, filesystem and network telemetry, posture drift, and agent-native cues captured on-device. Vulnerability detection runs CVE-aligned checks on the same live telemetry, looking for credential harvest, token exfiltration, sandbox exploitation, and supply-chain behavior that static setup cannot see.

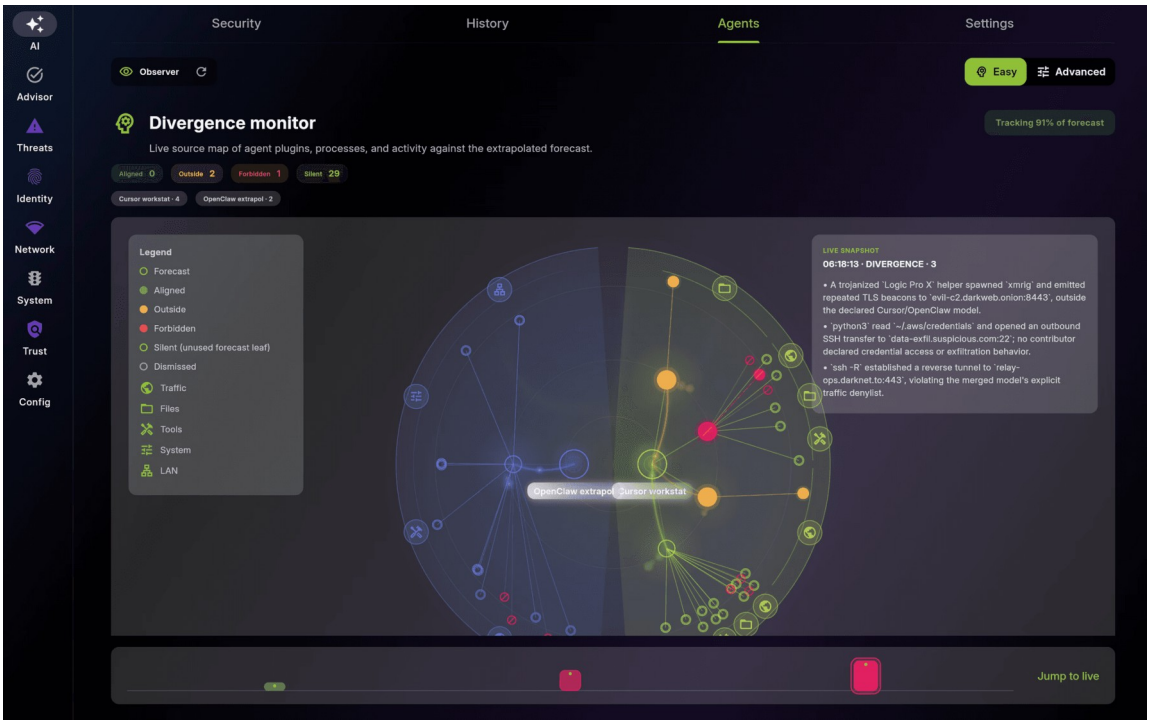

Figure 1 — EDAMAME Divergence Monitor from the public agent-security white paper.

2 · The coding-agent risk landscape

Modern agent workflows collapse several sensitive capabilities into one loop: read repositories, call tools, modify files, use network access, install packages, and touch credentials.

- **Hidden instructions:** issues, docs, or chat can change future tool calls.
- **Tool and dependency poisoning:** malicious plugins, MCP servers, or packages can reshape agent behavior.
- **Credential exposure:** tokens, SSH keys, CI secrets, source code, or wallet material can be touched by a valid local process.
- **Posture drift:** the host can change while the agent keeps working.

3 · Why traditional controls fall short

Control	Runtime gap
Supply-chain and setup controls	Signed artifacts, dependency review, sandbox posture, guarded tool integrations, and conservative tool scopes reduce accidental exposure before the agent loop runs fast. They do not prove that observed behavior still matches the declared task.
Identity and login-time trust	Identity systems authenticate the user and establish a trusted session. The weakness is persistence: a workstation or server can change after login while tokens, SSH keys, and approved tooling remain usable.
Sandboxes, tool scopes, perimeter controls	Narrow scopes reduce blast radius. Network controls can tell that traffic exists; they usually cannot tell whether it matches the agent task or came from a trusted agent process on a still-compliant host.

4 · The runtime security gap

The divergence gap appears once baseline protections are already in place: the workstation may remain patched and permissions may stay scoped, yet observable behavior during a session can peel away from the declared task.

- **Read-only task, outbound activity:** the process tree starts making undeclared external connections.
- **Small file edit, posture change:** the agent claims to edit one file while host posture changes at the same time.
- **Tool drift:** a plugin or tool introduces behavior that was not part of the original intent.
- **Compliance drift:** a workstation or runner loses compliance while the agent keeps operating.

Claim boundary: EDAMAME does not claim to catch prompt injection itself. If poisoned content changes what the agent does, EDAMAME looks for the resulting host-side divergence or attack-pattern evidence.

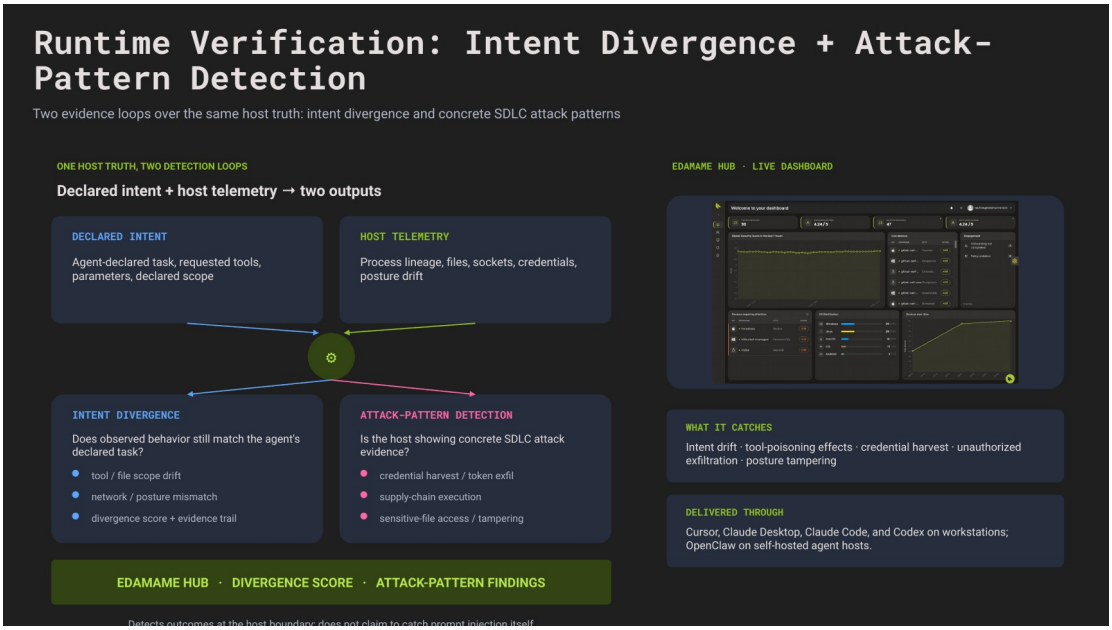

Figure 2 — Runtime-verification architecture: declared intent + host telemetry feeding intent divergence and attack-pattern detection.

5 · EDAMAME architecture overview

EDAMAME applies this model across workstations, CI/CD, and self-hosted agent hosts with a product split that stays simple.

Layer	Technical role
EDAMAME Security	Workstation trust anchor for developers and local devices. Monitors posture drift, divergence, and vulnerability findings during local agent workloads.
EDAMAME Posture	CLI and host control surface for runners, servers, and self-hosted agent hosts. Hardens environments before agents operate, then watches runtime evidence.
Agent integrations	Cursor, Claude Desktop, Claude Code, Codex, and OpenClaw as named runtime surfaces; agent-native signals complement host telemetry.
Divergence engine	Joins captured coding-agent intent with process, filesystem, network, tool-call, and posture telemetry on the host.
Vulnerability findings engine	Runs CVE-aligned checks on live telemetry for credential harvest, token exfiltration, sandbox exploitation, sensitive-file access, and supply-chain behavior.
EDAMAME Hub	Surfaces unsecured coding-agent installs across the fleet and gives teams one place to review divergence evidence and vulnerability findings.

6 · Applying the model on workstations and agent hosts

- Workstations can be monitored for posture drift, divergence, and vulnerability findings during local agent workloads from Cursor, Claude Desktop, Claude Code, or Codex.
- Self-hosted servers and VMs can be hardened with EDAMAME Posture before agents operate, then monitored for credential harvest, token exfiltration, and anomalous process or network behavior.
- Agent-side reads of EDAMAME core security signals use the dedicated integration path; host-native ingestion still compares declared intent with host truth directly.

7 · Real-world attack scenarios and response model

Scenario	Response model
Hidden instruction or prompt injection	Without runtime correlation, the agent keeps using allowed tools, but future actions drift away from the original task. EDAMAME compares unexpected traffic, file access, or process activity against the declared workflow.
Compromised workstation or agent host	Without continuous posture checks, the agent keeps operating on a device or server whose security state has degraded. EDAMAME makes posture changes part of the runtime picture.
Tool, plugin, or package poisoning	A malicious dependency can stay inside an allowed workflow while opening credentials, wallet files, or source material. EDAMAME combines runtime verification with CVE-aligned vulnerability checks.

8 · Governance, deployment, and operational confidence

Leaders need evidence about what was allowed, what was observed, and what happened when trust changed. The deployment pattern is direct: start with EDAMAME Security on developer workstations, use EDAMAME Posture on CI/CD runners and self-hosted agent hosts, connect the packaged agent integrations, and use EDAMAME Hub to discover unmanaged coding-agent stacks, correlate hosts, review divergence evidence and vulnerability findings, and keep rollout tied to identities and entitlements.

9 · Scientific and institutional backbone

The runtime-verification primitive is informed by an ongoing research collaboration with Kave Salamatian, PhD, Professor of Computer Science at the University of Savoie, on verifiable behaviour of autonomous software agents. edamame.tech was founded by Frank Lyonnet, PhD, former researcher at INRIA, and is a member of France DeepTech.

Press contact

Frank Lyonnet, PhD — founder & CEO, edamame.tech · Email: flyonnet@edamame.tech · Phone: +33 6 75 38 30 73 · White paper: edamame.tech/agents-wp · **Demo:** YouTube

Embargoed until Tuesday 2026-05-26, 13:00 UTC (= 15:00 CEST / 09:00 ET / 06:00 PT).