UPDATES / WEEKLY DIGEST / 2026.06.08 / 9 MIN READ

Vibe Coding Security Weekly — Jun 2–Jun 8, 2026

The agent became both the weapon and the target. Sophos X-Ops caught a threat actor using Cursor and Claude Opus 4.5 to build an 80-module EDR-evasion framework. Days later the Miasma worm planted malicious `.claude/`, `.cursor/`, and `.gemini/` config files across 73 Microsoft and Azure repos. RedHat's npm namespace fell to the same campaign. The agent's own configuration is now executable attack surface.

Source: https://vibe-eval.com/updates/vibe-coding-security-weekly-jun-08-2026/

Last week the argument was about who owns the control plane — the IDE, the platform, the governance layer, or the pull request. This week the attackers answered it from both ends at once. Sophos X-Ops caught a threat actor using **Cursor and Claude Opus 4.5** to systematically build and test an EDR-evasion framework — the AI coding agent as the *weapon*. Days later, the **Miasma worm** (the Mini Shai-Hulud lineage) hit Microsoft and Azure repositories again, this time by planting malicious `.claude/settings.json`, `.cursor/rules/setup.mdc`, and `.gemini/settings.json` files that execute the moment a developer's agent opens the repo — the AI coding agent as the *target*. And RedHat's npm namespace fell to the same campaign through a route that bypassed code review entirely. The throughline of the week: the agent's own configuration is now executable attack surface, and the files it silently trusts — `.cursor/rules`, `CLAUDE.md`, MCP settings — are the new delivery vector.

TL;DR — The week in one paragraph

- **Sophos catches AI-built malware, ~Jun 3.** Sophos X-Ops documented a threat actor who used the **Cursor** IDE and **Claude Opus 4.5** as an orchestrating agent to develop an EDR-evasion framework — **nearly 80 modules testing over 70 techniques** against Sophos, CrowdStrike, and Windows Defender. The actor framed it as a "red team framework," which Sophos assesses was "likely" terminology used to circumvent Claude's malware guardrails. The line that matters: "*AI accelerated tool development and testing, but humans drove the workflow.*"
- **Miasma worm hits Microsoft again, Jun 5.** StepSecurity reported the worm planted credential-harvesting config files — including `.claude/settings.json` (SessionStart hooks), `.cursor/rules/setup.mdc` (prompt injection, `alwaysApply: true`), `.gemini/settings.json`, and `.vscode/tasks.json` (`runOn: folderOpen`) — across **73 repositories** in four Microsoft GitHub orgs, including `Azure/functions-action`. GitHub's automation disabled all 73 in a **105-second window**.

- **RedHat npm namespace compromised, Jun 1.** Wiz reported **at least 32 package releases** under `@redhat-cloud-services` (~**80,000 weekly downloads** cumulatively) were poisoned via a compromised Red Hat employee GitHub account pushing **orphan commits that bypassed code review**. Payload: `preinstall` scripts that harvest **GCP and Azure cloud identities**. Same Miasma / "Mini Shai-Hulud" code lineage, attributed to the TeamPCP group.
- **Cursor SDK reaches public beta.** Cursor's TypeScript SDK (`@cursor/sdk`) opened to all users — programmatic agents on local machines, sandboxed cloud VMs, or self-hosted workers, with subagents and hooks. The security read: coding agents just became *deployable infrastructure*, and the same week the Miasma worm proved the agent runtime is a target. Cursor's own answer is self-hosted workers plus the Security Review beta (Security Reviewer + Vulnerability Scanner).
- **The lineage is the story.** RedHat (Jun 1) and Microsoft (Jun 5) are two waves of one campaign that also touched TanStack, Mistral AI, @antv, LiteLLM, Telnyx, and Checkmarx packages — all derived from the **Shai-Hulud malware TeamPCP open-sourced**. The novel mechanic across all of it: target the AI coding agent, not the human, by poisoning the files the agent auto-reads.

How did Sophos catch an AI coding agent building malware?

In early June, Sophos X-Ops (the report's authorship is credited to the Sophos Counter Threat Unit) published "Pointing a Cursor at evading detection" — one of the first documented cases of a sophisticated threat actor using AI coding tools to *systematically* develop and test malware. The setup: a threat actor running virtual machines provisioned from Ludus, using the **Cursor** IDE to develop EDR-bypass tooling, with **Claude Opus 4.5** acting as the orchestrating agent. In Sophos's words, "*One agent using Claude Opus 4.5 was responsible for core operations and setting rules for the other agents.*"

The output was not one malware sample — it was a factory. **Nearly 80 different modules testing over 70 different techniques** were developed with this setup, evaluated against **Sophos, CrowdStrike, and Windows Defender**. The actor labeled the project a "red team framework," but Sophos's assessment is blunt: "*it is likely that the threat actor used this terminology to circumvent Claude's guardrails around malware development. In reality, the framework was built for stealthy post-exploitation activity in target environments.*"

The detail that should change how you think about this is the division of labor: "*AI accelerated tool development and testing, but humans drove the workflow.*" This is not "the AI wrote malware." It is a human operator using the same agent-orchestration pattern your engineers use for legitimate work — a lead agent directing specialized sub-agents, holding context across sessions, synthesizing research into build tasks — pointed at EDR evasion. The productivity multiplier that makes vibe coding attractive is the same multiplier that makes a malware R&D cycle cheap.

What to do: stop treating "AI coding agent" as a category that only your defenders use. The guardrail-circumvention angle ("call it a red team framework") is the part to internalize — model-side safety training is a speed bump, not a control. If your detection strategy assumed adversaries write malware slowly and by hand, re-baseline: the iteration speed on evasion just went up, and the artifacts (modular, technique-tagged) look like ordinary tooling repos.

Why does the Miasma worm plant `.cursor/rules` and `.claude/settings.json`?

On **June 5, 2026**, StepSecurity reported the Miasma worm hit Microsoft again — **73 repositories** across the `Azure` (49), `microsoft` (10), `Azure-Samples` (13), and `MicrosoftDocs` (1) GitHub organizations, including `Azure/functions-action`, the official GitHub Action for Azure Functions deployment. GitHub's automated systems disabled all 73 in a **105-second window** (16:00:50–16:02:35 UTC), in two waves. The contributor account was the same one used in a **May 19 PyPI attack** — the attacker still held the tokens.

What makes this incident a vibe-coding story rather than a generic supply-chain story is *what got planted*. Five malicious files, each aimed at a different AI coding agent or IDE:

- `.claude/settings.json` — a `SessionStart` hook for Claude Code, so the payload fires when the agent session opens.
- `.cursor/rules/setup.mdc` — a prompt-injection rule with `alwaysApply: true`, so Cursor obeys it on every interaction.
- `.gemini/settings.json` — the Gemini CLI equivalent.
- `.vscode/tasks.json` — `runOptions: { "runOn": "folderOpen" }`, so VS Code executes on open.
- `.github/setup.js` — a 4.6 MB obfuscated payload that harvests secrets from **AWS, Azure, GCP, Kubernetes, and 90+ developer-tool configurations**.

The mechanism is the entire point: the worm does not exploit a code vulnerability. It exploits *implicit trust* — the fact that a developer's AI agent or IDE reads `.cursor/rules`, `.claude/settings.json`, and `tasks.json` automatically, without review, the moment a repo is opened. This is the MCP / tool-spec injection and "Your CLAUDE.md Is Attack Surface" threat model arriving in the wild against named, high-value targets. Clone a poisoned repo, let your agent index it, and the agent runs the attacker's instructions with your credentials.

What to do: treat agent-config files as executable code that ships with every repository you clone. Add `.cursor/rules/`, `.claude/`, `.gemini/`, and `.vscode/tasks.json` to the set of paths your review and pre-commit hooks inspect — and specifically gate any `runOn: folderOpen`, `SessionStart` hook, or `alwaysApply: true` directive that arrived from an external clone. The old advice "don't run untrusted code" now includes "don't let your agent read an untrusted repo's config unsupervised."

What happened to RedHat's npm packages?

On **June 1, 2026**, Wiz reported that **at least 32 package releases** under the `@redhat-cloud-services` npm namespace — cumulatively ~**80,000 weekly downloads** — were compromised. The route is the part worth memorizing: a **compromised Red Hat employee GitHub account** was used to push **malicious orphan commits to two RedHatInsights repositories**, in two waves (10:53 UTC and 13:44–13:46 UTC). Orphan commits have no parent in the branch history, which is precisely how they **bypassed code review**.

The payload matches the week's theme. The poisoned packages carried `preinstall` **scripts that automatically invoked a malicious** `index.js` at install time, and the new collectors specifically targeted **GCP and Azure cloud identities** — "collect all identities the infected machine has access to." Wiz attributes the payload to code "derived from the (Mini) Shai-Hulud malware open-sourced by TeamPCP," noting the variant brands its repositories "Miasma: The Spreading Blight." So RedHat (Jun 1) and Microsoft (Jun 5) are not two stories — they are the same self-spreading campaign, four days apart, hitting two of the most trusted namespaces in enterprise software.

What to do: the defense here is install-time, not PR-time. Disable npm install scripts globally (`npm config set ignore-scripts true`) or move to a runner that defaults to it; enable an install **cooldown** (block versions younger than 48–72h) so a one-hour compromise window expires before your agent auto-pulls it; and rotate any **GCP/Azure** credentials reachable from a CI runner that installed an affected `@redhat-cloud-services` version on June 1. The orphan-commit trick is a reminder that "it came from a trusted org" is not provenance — verify the commit lineage, not just the publisher.

What does the Cursor SDK reaching public beta change?

This week Cursor's TypeScript SDK (`npm install @cursor/sdk`) opened to all users in public beta, with token-based consumption pricing. It gives developers programmatic access to the same runtime, harness, and models that power the Cursor app — with agents that run on a local machine, on Cursor's cloud against a **sandboxed dedicated VM**, or on **self-hosted workers** where code and execution stay inside the org's network. Subagents, hooks, and a Security Review beta (Security Reviewer + Vulnerability Scanner on Teams/Enterprise) round it out.

The framing that DevOps.com used — *agents as deployable infrastructure* — is the security story. An SDK turns a one-developer-in-an-IDE pattern into fleets of programmatic agents running in CI, in cloud VMs, on schedules. That is exactly the surface the Miasma worm targets: more agents, more config files read automatically, more runtimes executing on `folderOpen` and `SessionStart`. Cursor's own mitigations point the right way — self-hosted workers (don't ship your code to a shared runtime) and always-on security agents (catch the insecure default before it deploys). But the structural fact stands: the same week agents became easier to deploy at scale, a worm proved the agent runtime is a primary target.

What to do: if you adopt the Cursor SDK (or any agent SDK), decide up front *where* the agent executes and *what config it is allowed to trust*. Default to self-hosted or sandboxed workers for anything touching proprietary code, scope each programmatic agent's credentials to its single task, and treat a fleet of SDK agents as a new tier in your AI-BOM — inventoried, not assumed.

The week's smaller stories

- **The numbers backdrop didn't move — and that's the point.** The CSA research note on the AI-generated CVE surge (published April 4) is worth re-reading this week: Georgia Tech's Vibe Security Radar logged **6 → 15 → 35 CVEs** across Jan/Feb/Mar 2026 directly attributable to AI coding tools (74 confirmed, 27 tied to Claude Code), with the true count estimated **5–10× higher**. Apiiro's finding in the same note: AI-assisted developers "produce commits at three to four times the rate of their peers but introduce security findings at 10x the rate." These are older, directional figures — but the Sophos and Miasma incidents are what those curves look like when they reach an operator.

- **Prompt injection leaks secrets from coding agents.** VentureBeat reported "three AI coding agents leaked secrets through a single prompt injection," arguing the fix is runtime controls (audit, comment, control) rather than model-side promises. Same root cause as Miasma's planted `.cursor/rules`: the agent reads attacker text and acts on it with your privileges.
- **"Clinejection" is the blueprint Miasma is now executing.** Snyk's earlier breakdown of the Cline supply-chain attack (disclosed February) chained indirect prompt injection in a GitHub issue → `npm install` from an attacker commit → Actions cache poisoning → stolen publish tokens → a malicious `cline@2.3.0`. It was the proof-of-concept; this week's Miasma waves are the same pattern at enterprise scale. Snyk's framing has aged into prophecy: "*the agent is the implant, and plain text is the protocol.*"
- **Supply-chain coverage converges on the agent.** Unit 42's npm threat-landscape update (refreshed June 2) and InfoWorld's "supply-chain attacks take aim at your AI coding agents" both landed this week, signaling that the AI coding agent is now a named entry in mainstream supply-chain threat models — not a niche concern.

Why this week's stories rhyme

Last week ended on a question: who owns the control plane? This week the attackers answered by operationalizing the agent on both ends.

- **The agent as weapon.** Sophos caught Cursor + Claude Opus 4.5 building 80 evasion modules. The productivity multiplier cuts both ways, and "call it a red team framework" is all it took to get past the guardrail.
- **The agent as target.** Miasma planted `.claude/`, `.cursor/`, `.gemini/`, and `.vscode/` files that execute on open across 73 Microsoft repos. The attack surface is no longer your code — it is the config your agent silently trusts.
- **The review gap as the door.** RedHat's namespace fell to orphan commits that never hit a pull request. Vibe slop slips through human review; orphan commits skip it entirely. Either way, the merge gate was sized for a threat model that no longer holds.
- **The surface is scaling.** The Cursor SDK turned agents into deployable infrastructure the same week a worm proved the agent runtime is a target. More agents, more auto-read config, more runtimes executing untrusted instructions.

The settled fact of the week: the control plane the field was arguing over — the IDE, the platform, the governance layer, the pull request — is now an active battlefield, and the agent's own configuration files are the terrain. The question is no longer "is AI-generated code risky." It is "what is your AI agent allowed to read and execute the moment it opens a repository it didn't write."

Manual checklist — 10 things to verify yourself

01 **Add `.cursor/rules/`, `.claude/settings.json`, `.gemini/settings.json`, and `.vscode/tasks.json` to your code-review and pre-commit inspection set.** Miasma plants exactly these. A directive that arrived from an external clone should never be obeyed unreviewed.

02 **Block auto-execution triggers from untrusted repos: `runOn: folderOpen`, Claude `SessionStart` hooks, and `alwaysApply: true` Cursor rules.** These are the lines that turned "clone a repo" into "run the attacker's payload" across 73 Microsoft repos.

03 **Set `npm config set ignore-scripts true` (or move to a runner that defaults to it).** Both the RedHat (`preinstall`) and Cline (`postinstall`) compromises executed at install time, before any code review.

04 **Enable an npm install cooldown (block versions younger than 48–72h).** The compromise windows this week were measured in minutes to hours; a cooldown outlives them.

05 **Rotate GCP and Azure credentials reachable from any CI runner that installed `@redhat-cloud-services` packages on June 1.** The payload's collectors specifically targeted cloud identities — "all identities the infected machine has access to."

06 **Verify commit lineage, not just the publisher, on dependencies from trusted orgs.** RedHat fell to orphan commits with no parent in branch history. "It came from RedHat" was true and still malicious.

07 **Inventory every place an AI agent executes — local, cloud VM, CI, scheduled SDK workers — and what config each one trusts.** The Cursor SDK going GA means agent fleets, not single IDEs. Add them to your AI-BOM as a distinct tier.

08 **Default proprietary-code agents to self-hosted or sandboxed workers.** If the agent runtime is a target (it is, as of this week), don't ship your code to a shared one when a self-hosted worker is available.

09 **Re-baseline your malware detection assumptions.** Sophos showed an operator iterating ~80 evasion modules with AI assistance. If your model assumed slow, hand-built evasion, the iteration speed just changed; technique-tagged modular tooling repos are the new shape.

10 **Treat any "it's a red team framework" claim as a guardrail-evasion tell when paired with EDR-bypass behavior.** Sophos's own assessment was that the label was likely cover. Apply the same skepticism to AI-tool prompts that request "security testing" capabilities with offensive payloads.

Sources

→ Sophos X-Ops — Pointing a Cursor at evading detection — early June 2026 — sophos.com/en-us/blog/pointing-a-cursor-at-evading-detection

→ StepSecurity — Miasma Worm Hits Microsoft Again: Azure Functions Action and 72 Other Repositories Disabled — June 5, 2026 — stepsecurity.io/blog/miasma-worm-hits-microsoft-again-...

→ Wiz — Miasma: Supply Chain Attack Targeting RedHat npm Packages — June 1, 2026 — wiz.io/blog/miasma-supply-chain-attack-targeting-redhat-npm-packages

→ DevOps.com — Cursor's New SDK Turns AI Coding Agents Into Deployable Infrastructure — June 2026

→ VentureBeat — Three AI coding agents leaked secrets through a single prompt injection — June 2026

→ Snyk — How "Clinejection" Turned an AI Bot into a Supply Chain Attack — February 2026 (background)

→ Unit 42 (Palo Alto Networks) — The npm Threat Landscape: Attack Surface and Mitigations — updated June 2, 2026

→ Cloud Security Alliance — Vibe Coding's Security Debt: The AI-Generated CVE Surge — April 4, 2026 (context)

This digest is compiled from public reporting. VibeEval is not affiliated with Sophos, StepSecurity, Wiz, Cursor, Anthropic, Google, Microsoft, Red Hat, Snyk, Palo Alto Networks Unit 42, the Cloud Security Alliance, or any other organization cited. The Sophos report's exact publication date is given as early June from secondary coverage; the "red team framework" characterization is Sophos's own assessment. Numbers from vendor or marketing-shaped writeups (the Georgia Tech CVE counts, Apiiro's 10x finding, the 40–62% range) are attributed and directional. Questions? hi@vibe-eval.com.