boostsecurity

CISO FIELD GUIDE — SUPPLY CHAIN RISK

What CISOs Need to Know About Developer Endpoint Risk

The bottom line: your EDR can't protect you from a new class of supply chain risk.

Your developers run AI coding agents at user privilege on credential-loaded laptops. In the age of the "citizen developer," so do people in product, ops, and finance — and your security team can't see any of them.

EDR watches processes; AppSec covers source control. So attackers strike where you can't see: **at install time, on the laptop.**

If your team gets hit by a supply chain attack, can you answer:

- Which machines installed it?
- What credentials were present?
- What was the time window?

Most teams take days to answer. **You don't have days.**

What senior leadership is about to ask

If you're like most CISOs, you get 15 to 30 minutes of board time per quarter — and you're about to start spending a significant share of those minutes on supply chain attacks.

15–30
minutes of board time per quarter

29%
of directors call their cyber updates "very effective"

50%+
say reporting on evolving threats needs improvement

Attackers hit production through **five named incidents in the past year.** Gartner predicted 45% of organizations would face one by 2025 — organizations exceeded that number in 2024.

`Axios` `NX` `tj-actions/changed-files` `Shai-Hulud` `TeamPCP`

Consider this guide a cheat sheet. It's far better to bring this conversation to your board **proactively, with solutions in hand** — than in the aftermath of a compromise.

Why your existing stack doesn't see this

Your security team watches processes, code, and hardware. Attackers exploit the one layer none of those tools reach: **the agent runtime.** When an attacker poisons a package or hides an injection in a README, the agent carries out those instructions using its own authorized shell access, credentials, and network connections.

The pattern — `living off the agent` — mirrors the older "living off the land" technique, except the agent replaces the OS utility. Attackers bring nothing new to the laptop, because the agent already has everything they need.

	TRADITIONAL EDR	TRADITIONAL APPSEC	DEV ENDPOINT SECURITY
Malicious package install	✗ Authorized process	✗ Pre-SCM, invisible	✓ Flags at install time
Rogue MCP server connection	✗ Application-layer, can't see MCP use	✗ Never registers at CI	✓ Inventories and blocks
Agent reads credentials	✗ Normal file read	✗ Happens on laptop	✓ Detects and masks
Untrusted IDE extension	✗ Part of IDE process	✗ Outside code pipeline	✓ Enforces allowlist
Indirect prompt injection	✗ Normal text file read	✗ Executes pre-SCM	✓ Scans context files
Agent shell commands	✗ Authorized user activity	✗ Happens on laptop	✓ Logs and governs

Your AppSec team owns SCM and CI; your IT team owns laptop hardware and OS. Right in the middle, the agent runtime falls into a black hole with zero visibility. **Attackers have noticed.**

Who counts as a ”developer” now?

How many people at your company run coding agents? Your engineering headcount won't tell you.

People in product, ops, finance, and marketing all run them — often building scripts they'd have asked engineering for. They expense "Cursor," "Claude Code," or "Copilot" as productivity tools, and their managers approve without involving security.

Do they even need it? They don't ship to production, so they don't create production risk — except the agent runs `npm install` and any malicious payload executes instantly. Hours before anyone opens a PR; weeks before anyone discovers the package was compromised. **By the time anyone considers shipping the code, every event that matters has already happened on the laptop.**

Where do your AI coders fall — and where is their risk?

	INSIDE SDLC	OUTSIDE SDLC
REACHES PRODUCTION	**Engineering production work** Your AppSec team covers this — incompletely.	**Ops & RevOps internal tools** Your AppSec team has never inventoried this.
STAYS ON LAPTOP	**"Just prototyping"** No coverage.	⚠ **Vibe coding** AppSec, IT, and security all miss it.

Does that last quadrant actually matter?

Probably more than you think. The non-engineer's laptop inverts your assumptions about where the valuable credentials live.

ENGINEER'S LAPTOP

GitHub PATs

AWS / GCP / Azure keys

npm publish tokens

Your security team monitors for misuse of these — imperfectly.

NON-ENGINEER'S LAPTOP

Active Salesforce session

Okta cookie → every SSO app

Workday session

Slack token

Nobody monitors for agent-driven misuse of these.

An attacker can monetize active SaaS sessions **in minutes**, while developer credentials take more work. Your security team monitors the first group — but nobody monitors the second.

Your employees in non-engineering roles are sitting ducks.

How an attacker compromises a vibe coder

A marketer opens **Cursor** on her company laptop and asks it to automate a Salesforce-to-Slack workflow. The agent writes a script and installs a popular open-source package to handle the API calls. An attacker compromised that package last week — but the compromise isn't public yet. Neither she nor the agent can tell.

The install triggers a hidden post-install script. It drops an infostealer that extracts her active **Okta and Salesforce session tokens** from the browser. Within seconds, the attacker logs into every SaaS app behind her company's single sign-on.

Her laptop never touched source control, so AppSec couldn't touch it. Her EDR logged an authorized Cursor process. Security saw nothing — but the credentials were stolen.

Ask your EDR vendor three questions

1. When an agent reads `~/.aws/credentials`, can your EDR tell whether a developer asked for it or a prompt injection buried in a dependency triggered it?
2. If a `pip install` pulls a transitive dependency the developer never specified, can your EDR evaluate that package before the agent executes it?
3. If a compromised package exfiltrates secrets by encoding them into the agent's normal LLM API callback traffic, does your EDR generate any signal at all?

They'll say no to all three. EDR monitors process behavior at the kernel level. It has no model of agent intent, no visibility into prompt-driven execution chains, and no way to tell legitimate agent-to-API traffic from exfiltration through that same channel.

CASE STUDY — TEAMPCP, MARCH 2026

TeamPCP published two malicious versions of `litellm` to PyPI containing a `.pth` infostealer that runs on every Python startup — no import required. Every developer or agent that ran `pip install litellm` during that window handed over their credentials automatically.

70 repos exposed

15 defaced

182 repos wiped

...in spite of a full stack of EDR and AppSec tools operating normally. How fast could your team tell you which machines installed it, and what credentials were present?

What "defensible" looks like

With Boost, your team answers the board in **hours, not days.**

Query the fleet

Search `chalk@5.6.1` and see every machine that installed it, in seconds.

Retroactive flags

When a package is reclassified weeks later, every affected machine flags automatically.

Enforce at install

Allowlists on the laptop, on the agent's clock — the attacker never gets a window.

How it deploys

A lightweight observer on the IDE and CLI layer. No kernel hooks, zero latency on prompts, zero disruption to builds.

What you bring to the board

`Fleet coverage` `Policy compliance` `Exposure by severity` `MTTD / MTTR`

The question to take upstairs

✗ LOSES

"I need budget for a new tool." Directors will ask why your existing tools aren't enough.

✓ LANDS

"I have a category of risk our controls can't see, and here's my plan to fix it." The board engages with the gap first.

> ”We took on an unknown number of new AI agents and AI-assisted coders last year — agents run by our developers and LLMs used by people outside engineering. Our existing controls don’t see them. Here is the plan to fix that, and the metrics by which we’ll measure progress.”

THE OPENER

Ask your security team this week

- How many people run coding agents on company hardware right now — and how would we find out?
- If `chalk@5.6.1` were reclassified malicious tomorrow, how long until we know who installed it?
- When someone discloses a malicious npm package, what does our team do first?
- Our cyber insurance renewal could cover this surface next year — how will we answer?

Talking points for leadership

- Our coding-agent headcount and engineering headcount are different numbers — an unmanaged attack surface.
- Payloads execute the instant someone types install. The credential theft happens before any code review.
- EDR watches processes, AppSec watches source — attackers exploit the blind spot neither was designed to see.
- One policy engine across endpoint, source, and CI costs less than three disconnected ones.

Developer endpoints concentrate a risk your tools can’t see.

You've seen where your tools fail and where attackers operate. You have five questions for your team this week, and an opener for your board.

Now find out what's actually running on your endpoints.

Request your complimentary AI-BOM check & diagnosis

We'll show you your unmanaged agents, unverified MCP connections, and untrusted extensions — before you decide anything.

VISIT
boostsecurity.io

EMAIL
hello@boostsecurity.io