Developer Supply Chain Security Guide

James Barnett, WebTechKitchen

June 2026

The Developer's Security Checklist: Defending Against Supply Chain Attacks

By James Barnett, WebTechKitchen *June 2026*

The Short Version

Supply chain attacks have shifted their target from production systems to development environments. In the last week alone, attackers compromised 144+ npm packages, infected a JetBrains Marketplace namespace, and deployed worms capable of harvesting every cloud credential on a developer's machine — including GitHub tokens, AWS keys, and SSH keys. If you install npm packages or use IDE plugins without a systematic review process, your secrets are at risk before your code ever ships.

The Attack Has Already Shifted to Your Laptop

Production systems get attention. Security teams monitor them. Patches get applied. Incident response plans exist.

Your laptop? Not so much.

Attackers have figured this out. Three separate supply chain campaigns in June 2026 targeted the development environment specifically:

- **Shai-Hulud** compromised 32 packages in the @redhat-cloud-services npm namespace. 117,000 weekly downloads. The attack used GitHub Actions OIDC credentials to bypass code review entirely — meaning even if you reviewed the code, you'd never see the malicious commit.

- **Miasma** hit 7 Red Hat npm packages via preinstall hooks. Before your `npm install` even finished, it had already harvested your GitHub Actions secrets, AWS access keys, GCP credentials, Azure tokens, SSH keys, and Docker credentials.

- **15 malicious JetBrains plugins** sat in the Marketplace from October 2025 until June 2026 — eight months — with two exceeding 25,000 downloads each. They functioned normally while quietly exfiltrating AI API keys from the developer's environment.

The common thread: these attacks didn't break anything visible. No crashes, no unusual behavior, no alerts. They ran once, quietly, and sent your credentials somewhere else.

One compromise of a developer's machine can hand attackers direct access to your production AWS account, your GitHub repos, your CI/CD pipeline, and every secret in your .env files. The blast radius is enormous.

What Most Teams Get Wrong

Most developers think of supply chain security as something the security team handles at the CI/CD level — running SAST scans, checking for known CVEs in package-lock.json, maybe using Snyk or Dependabot.

That's necessary, but it's not enough. Here's what they're missing:

Preinstall hooks bypass most scanning. The Miasma attack executed malicious code via `preinstall` scripts — code that runs the moment you type `npm install`, before any static analysis tool sees it. Most vulnerability scanners check installed packages for known CVEs. They don't execute the code in a sandbox to watch what it does. They can't catch this.

Namespace squatting is invisible until it's too late. The Mastra namespace attack pushed 140+ malicious packages in an 88-minute window. The packages had the same names, similar version numbers, and identical functionality — plus one new dependency that stole cryptocurrency wallet data. Your lockfile didn't protect you if you ran `npm install` that day without pinning to exact content hashes.

IDE plugins get zero scrutiny. Most developers apply the same level of rigor to installing a JetBrains plugin as they do to installing a browser extension — which is to say, none. Eight months. 25,000+ downloads. Functioning as advertised. No one noticed.

The uncomfortable truth: the supply chain attacks that will actually hit you aren't in a CVE database. They're running right now, doing exactly what the package advertises, plus one extra thing you'll never see.

A Practical Defense Framework

You can't review every line of every package you install. But targeted changes dramatically reduce your exposure. Here's what actually works:

1. Audit your npm preinstall and postinstall hooks immediately. Run `npm install --ignore-scripts` when installing new dependencies. For existing projects, look for preinstall, install, and postinstall entries in dependencies you didn't write. Any hook in a transitive dependency you don't recognize is worth investigating.

2. Pin dependencies to content hashes, not versions. Use `npm ci` (not `npm install`) in CI — it uses package-lock.json to install exact versions. Go further: consider switching to pnpm or Yarn Berry, which pin content hashes. A version number can be re-published with different content. A content hash cannot be faked.

3. Rotate your secrets now, not after an incident. The Miasma and Shai-Hulud worms targeted: GitHub tokens, AWS access keys, GCP credentials, Azure tokens, SSH keys, Docker credentials, and AI API keys. If any of these live in your shell environment as env vars — which they usually do for developer convenience — rotate them. Costs you 20 minutes. Not rotating costs potentially everything.

4. Isolate development credentials from production. Your local dev machine should never have production AWS credentials. Use separate IAM roles with minimal permissions for dev. If your laptop is compromised, the blast radius stops at dev — not prod. This isn't a new principle, but after these attacks it's worth checking whether you've actually implemented it or just nodded at the idea.

5. Audit installed IDE plugins today. Go to your IDE's plugin list right now. For every AI-related plugin: when was it last updated? Who published it? Does the publisher have a track record beyond this single plugin? If you can't answer yes to all three — uninstall and reinstall from the vendor's official site directly. The JetBrains attack specifically targeted developers who installed "AI assistant" plugins.

6. Run npm audit and Dependabot — but know their limits. `npm audit` catches known CVEs. Dependabot opens PRs for outdated packages with known vulnerabilities. Run both. But understand that neither caught Miasma, Shai-Hulud, or the Mastra attack — because novel attacks aren't in the CVE database yet. These tools are table stakes, not a complete defense.

7. Monitor network traffic from your dev machine. Tools like Little Snitch (Mac) or similar let you see what your IDE and build tools are phoning home to. The malicious JetBrains plugins contacted attacker-controlled servers. This is detectable — but only if you're looking.

Where This Is Heading

Supply chain attacks targeting development environments will get more sophisticated, not less.

AI-assisted attacks. The same tools that help you write code also help attackers write more convincing malicious packages with better documentation, cleaner code, and lower detection rates. The JetBrains plugins passed human review for eight months. That bar is going to get lower.

The IDE becomes the primary attack surface. As AI coding assistants become essential workflow tools — Cursor, GitHub Copilot, Codeium — they create a persistent, trusted, highly-privileged process on your machine with access to your files, terminal, and environment variables. The $60 billion SpaceX/Cursor acquisition signals this is central infrastructure now. Attackers will treat it as such.

Platform responses are coming, but they'll lag. JetBrains and npm will add stricter signing requirements, automated behavioral sandboxing for install hooks, and faster takedown processes. These will help — but they'll lag behind attacker innovation. The near-term answer is defense in depth at the developer level.

The developers who come out of this cleanly are the ones who applied systematic discipline: rotate credentials regularly, isolate dev from prod, know what preinstall hooks are running on their machines.

Quick Reference Checklist

- ☐ Run `npm install --ignore-scripts` for new dependencies
- ☐ Audit preinstall/postinstall hooks in your current package-lock.json
- ☐ Switch CI to `npm ci` with a committed lockfile
- ☐ Rotate GitHub tokens, AWS access keys, and AI API keys today
- ☐ Separate dev and production credentials / IAM roles
- ☐ Audit every AI-related IDE plugin — verify publisher legitimacy
- ☐ Enable npm audit and Dependabot (baseline coverage, not complete)
- ☐ Install a local network monitor (Little Snitch or equivalent)

Questions? Need help implementing any of this? Reach out at https://webtechkitchen.com/contact