How to Defend Against Dependency Supply Chain Attacks

Registry hardening, pinning, and monitoring

Step 1: Configure Private Registry

Proxy public registries through a private registry to control what enters your environment:

- Set up Artifactory, Nexus, or cloud-native registry (AWS CodeArtifact, GCP Artifact Registry)
- Configure as proxy for npm, PyPI, Maven Central, etc.
- Enable vulnerability scanning on ingestion
- Block packages that fail policy checks (license, vulnerability, age)

Step 2: Pin and Verify Dependencies

Lock dependency versions and verify integrity:

- Use lock files (package-lock.json, poetry.lock, go.sum) — commit them
- Enable hash verification (npm --integrity, pip --require-hashes)
- Pin exact versions in production (no ^ or ~ ranges)
- Review all dependency updates before merging (Renovate/Dependabot PRs)

Step 3: Detect Slopsquatting and Typosquatting

AI tools may hallucinate package names that attackers then register:

- Verify package existence and authorship before installing
- Check package age (new packages targeting existing names = suspicious)
- Use namespace reservation where available (npm scopes, PyPI namespaces)
- Monitor for dependency confusion attacks (internal vs. public name conflicts)