

npm-age-pin

May 22, 2026 • spencerburleigh.com/npm-age-pin

Supply-chain attacks are a growing problem for anyone installing dependencies. An attacker compromises a maintainer account or publishes a malicious package, and an ordinary install brings their code onto your machine. In the [Shai-Hulud npm attack](#) ^[a], compromised packages used malicious post-install scripts to spread the attack.

The observed activity has grown sharply: [Sonatype recorded 3,430 malicious-package advisories in 2025](#) ^[b], compared with an annual average of 931 in 2021–2023. That’s one research team’s detection data, but it gives a sense of the change in scale.

I rarely need a dependency update right when it comes out, and researchers, registries, and other users usually spot compromises quite quickly, so waiting to install updates feels like an easy way to prevent most of these attacks.

[npm-age-pin](#) ^[c] enforces this rule: select a release at least **72 hours old**, pin its exact version, and install with installation scripts disabled.

Give new code a little time

Installing a package is a decision to trust someone else’s code. Depending on the package and your settings, arbitrary code can run during installation, before you’ve even used the library. A familiar package name doesn’t protect you if its latest release came from a compromised account.

Three days is a practical default, not a point at which code becomes safe. Malware can go undiscovered longer, and an urgent security fix may justify a reviewed exception. I also use [Socket](#) ^[d], whose [npm integration](#) ^[e] checks packages before installation and can block them based on security alerts.

Make agents follow the waiting period

Coding agents make this more important because they can choose and install dependencies without much human attention. An agent trying to fix a build might grab the latest version, use `npx`, or switch installation commands.

Putting “wait three days” in an instructions file is a start but agents forget and do unpredictable things. We need to enforce the rule deterministically here and broadly: if you really need an agent to do something, you must use a deterministic check to enforce the behavior.

Use it

Requires Node.js 22+ and npm on macOS or Linux. Preview a pin without installing anything:

```
git clone https://github.com/sburl/npm-age-pin.git
cd npm-age-pin
node bin/npm-age-pin.mjs lodash --dry-run
```

After [adding the command to your PATH](#) ^[f], run it from your project:

```
npm-age-pin lodash --dry-run
npm-age-pin lodash
npm-age-pin lodash@latest
```

A bare name selects the most recently published eligible stable release and saves an exact version. That can be an older major or a maintenance release, so preview the choice. A tag checks its current target and refuses if that version is too new; it doesn’t guess the tag’s previous value. Exact versions must pass the age check too.

The standalone tool checks **direct packages named on the command line**, not the full transitive dependency graph, and doesn’t bundle Socket. Its `--ignore-scripts` ^[g] setting suppresses install scripts, but cannot protect you from malicious code you later import or execute. Review the lockfile and account for packages that need build scripts.

For agents, the remaining step is to enforce which installation paths are available. Plain `npm install` bypasses this standalone command. A wrapper that the agent can bypass or modify is a convenience, not a complete boundary. Route installs through the checks and restrict alternate paths outside the agent’s control, so everything stays on track.

I shared the first version of this in my first agentic coding setup [CrossCheck PR #143 — enforce safe-install policy via PreToolUse and Git hooks](#) ^[h].

Links

- a. github.blog/security/supply-chain-security/our-plan-for-a-more-secure-npm-supply-chain
- b. sonatype.com/resources/research/attacking-the-assembly-line
- c. github.com/sburl/npm-age-pin
- d. socket.dev

- e. docs.socket.dev/docs/socket-npm-socket-npx
- f. github.com/sburl/npm-age-pin#use
- g. docs.npmjs.com/cli/v11/using-npm/config/#ignore-scripts
- h. github.com/sburl/CrossCheck/pull/143