PHPVERSE 2026

Composer & Packagist Supply Chain Security in 2026

What we shipped, what is coming, and who pays for it.

Nils Adermann

@naderman · Private Packagist · packagist.com

PHP packages are now a direct target

Recent hits on PHP packages

intercom/intercom-php

Apr 30, 2026

GitHub repo access, then malicious releases published, Composer plugin.

laravel-lang

May 22, 2026

GitHub access via a dev machine, then a credential stealer in new tags.

What changed

- **Attackers target GitHub access directly**

  Via JS dependencies PHP devs use too, compromised IDE plugins, Python tools AI agents install, or misconfigured GitHub Actions.

- **GitHub takeover used to be the hard part**

  Used to be harder than stealing an npm or PyPI push token.

- **OIDC & build attestations are closing our one structural advantage**

  As npm and PyPI adopt OIDC and SLSA build attestations, they gain the source-to-consumer tie Composer always had. Number of attacks keeps rising, AI accelerating it.

Takeaway *One compromised GitHub account, or one poorly secured GitHub Action, is now enough to hit a widely used PHP package.*

How an attack reaches Composer users

GitHub repo

A version is a git tag pointing at a commit.

→

Packagist.org

Records a URL pointing at the GitHub API endpoint that builds a zip from that commit.

→

Composer

Downloads the zip from the GitHub API URL in the package metadata.

⚠️ **Re-tag a version (move the tag to a new commit) and a backdoor reaches everyone who updates.**

- Git tag model is a strength: installs trace to a source commit, not a separate upload.
- But the zip is generated dynamically, no stable stored artifact to checksum or sign yet.
- An existing composer.lock pins a commit hash, so safe from re-tag. Exposure is composer update / require / remove.

What we've shipped, and what's next

In place today

- Aikido malware feed
- Rapid manual response
- Public transparency log
- zizmor, allow listed third party GitHub Actions, workflows default to read-only

Shipped, last 2 weeks

- Composer 2.10 dependency policies
- Stable version immutability
- Source fallback deprecation

Coming

- Minimum release age (cooldown)
- MFA in the log and on profiles
- Organizational package ownership
- Bulk package management for orgs
- Organizational security controls

Longer-term

- Staged release publishing with MFA approvals
- MFA support for FIDO2
- mandatory MFA
- Immutable artifacts on Packagist.org
- Build provenance and Trusted Publishing
- Client-side provenance policies

Detection and response, today

- **Aikido malware feed, built into Packagist.org**
  Available to Composer under a CC-BY 4.0 licence. Flagged the malicious versions in every recent incident.

- **Manual reports from Socket**
  Through private channels, for now. Open to further data providers with a suitable free licence.

- **Packagist team pulls bad versions quickly**
  Widely used packages cleaned within minutes, sometimes in the middle of the night.

- **Focusing on prevention rather than reaction**
  Manual takedowns do not scale.

Takeaway *So far, the worst has been avoided through fast reaction and some luck.*

Transparency log

- Public record of events with a security focus:
  - ownership and maintainer changes
  - new & deleted versions
  - version reference changes
  - account events (email reset, password reset)
- Funded by the German Sovereign Tech Agency / Fund, via the PHP Foundation.
- Recorded the git re-tagging in every recent attack, enabling exact timelines.
- API on the way
  - Want to build monitoring on it? Talk to us and help shape it.

Why a record matters

Takeover visibility
A new maintainer on a popular library is visible to anyone.

Account review
What happened to an account or package, and when.

Incident timelines
A confident postmortem instead of a guess.

Consumer observability
Anything bad leaves traces that anyone can audit.

Composer 2.10: dependency policies

- One framework, config.policy: malware, security advisories, abandoned packages. Supersedes config.audit.
- Each policy:
  - block (remove from resolution),
  - audit (ignore / report / fail),
  - ignore (exemptions).
- Custom policies:
  - A URL listing versions to block
  - A POST endpoint deciding per package
- A comprehensive place for future policies, not another one-off check.

Defaults match what most projects want

- Malware: blocked on update and install, even from the lock file.
- Advisories: blocked on update, still installable to assess impact.
- Abandoned: audit-only.

```
Installing dependencies from lock file (including require-dev)
Verifying lock file contents can be installed on current
platform.
Your lock file does not contain a compatible set of packages.
Please run
composer update.
  Problem 1
    - Package evil/badpkg v1.0.0 (in the lock file) was not
    loaded, because it was flagged as malware (see
https://packagist.org/packages/evil/badpkg/filter-lists/malwar
e/)
    reason: malware. To ignore filters for this package, add
    the package to the "policy.malware.ignore" config. To turn
    the feature off entirely, you can set
    "policy.malware.block" to false.
```

Private Packagist Apply and enforce these policies organization-wide, including on older Composer versions.

Minimum release age (cooldown)

- **Refuse versions newer than N hours or days**
  Most malicious versions are caught or pulled within hours of publication.

- **Blocker: attackers can control timestamps**
  Packagist.org reads release time from git, so an attacker controlling the repo controls the clock. Not a security input until we assign it.

Takeaway *Easy to specify, impossible to trust until we control the publication timestamp. Staged publishing fixes that.*

Closing the source fallback

- **Failed dist downloads silently fell back to source checkouts**
  Any failed dist/artifact download would try a source checkout, no matter the reason, even an intentional 404 meant to block a malicious download.

- **Why it is a risk**
  It could install a git tag the registry would never serve, including a malicious one.

- **2.10: disabled and deprecated**
  Removed in 2.11. Rely on it? Reach out, usually just a config fix.

  Report at github.com/composer/composer/issues/12889

- **Explicit source installs still work**
  Request via preferred-install, not by accident.

A typical Packagist.org package in composer.lock

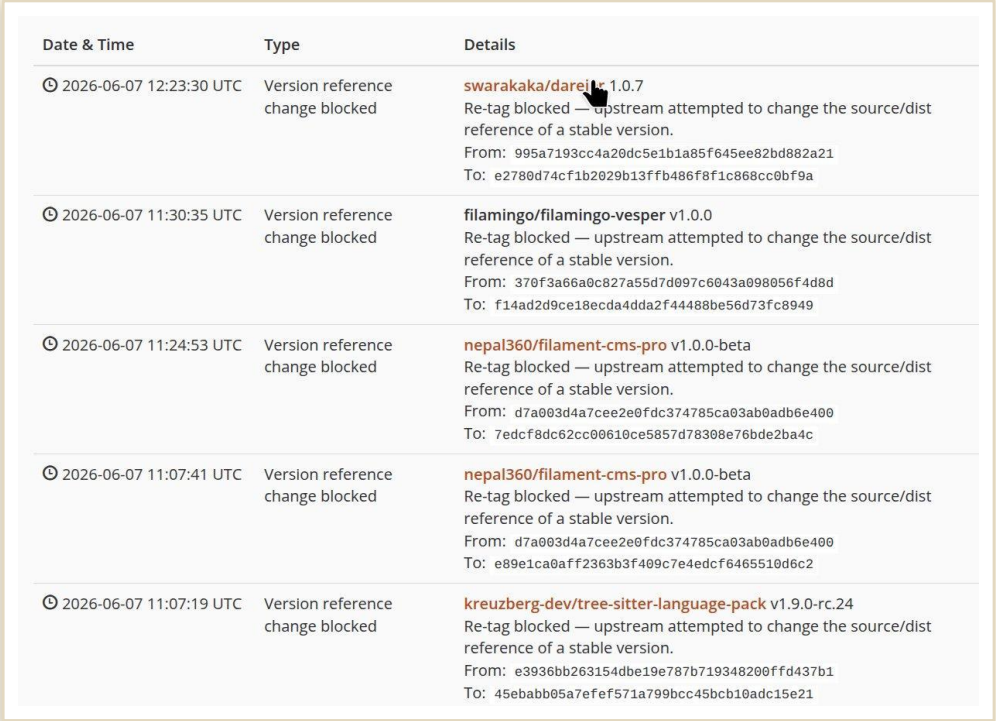
```
{
  "name": "monolog/monolog",
  "version": "3.7.0",
  "source": {
    "type": "git",
    "url": "github.com/.../monolog.git",
    "reference": "8e30c9f1"
  },
  "dist": {
    "type": "zip",
    "url": "api.github.com/.../zipball/8e30c9f1",
    "reference": "8e30c9f1"
  }
}
```

Both dist and source pin the same commit. A locked install is reproducible.

Private Packagist Disable dist fallback and strip source URLs across the whole organization.

Packagist.org: stable versions are now immutable

- Published stable (non-dev) versions are no longer silently rewritten on upstream re-tag. Attempt detected, rejected, maintainer notified.
- dev versions track branches, so they still follow their moving reference by design.
- Each tag's git commit is recorded in the transparency log, so the reference is verifiable.
- Re-tagging a fix minutes later harms users, they can't tell original from replacement: Release a new version!



Date & Time	Type	Details
2026-06-07 12:23:30 UTC	Version reference change blocked	swarakaka/darelik 1.0.7 Re-tag blocked — upstream attempted to change the source/dist reference of a stable version. From: `995a7193cc4a20dc5e1b1a85f645ee82bd882a21` To: `e2780d74cf1b2029b13ffb486f8f1c868cc0bf9a`
2026-06-07 11:30:35 UTC	Version reference change blocked	**filamingo/filamingo-vesper** v1.0.0 Re-tag blocked — upstream attempted to change the source/dist reference of a stable version. From: `370f3a66a0c827a55d7d097c6043a098056f4d8d` To: `f14ad2d9ce18ecda4dda2f44488be56d73fc8949`
2026-06-07 11:24:53 UTC	Version reference change blocked	nepal360/filament-cms-pro v1.0.0-beta Re-tag blocked — upstream attempted to change the source/dist reference of a stable version. From: `d7a003d4a7cee2e0fdc374785ca03ab0adb6e400` To: `7edcf8dc62cc00610ce5857d78308e76bde2ba4c`
2026-06-07 11:07:41 UTC	Version reference change blocked	nepal360/filament-cms-pro v1.0.0-beta Re-tag blocked — upstream attempted to change the source/dist reference of a stable version. From: `d7a003d4a7cee2e0fdc374785ca03ab0adb6e400` To: `e89e1ca0aff2363b3f409c7e4edcf6465510d6c2`
2026-06-07 11:07:19 UTC	Version reference change blocked	kreuzberg-dev/tree-sitter-language-pack v1.9.0-rc.24 Re-tag blocked — upstream attempted to change the source/dist reference of a stable version. From: `e3936bb263154dbe19e787b719348200ffd437b1` To: `45ebabb05a7efef571a799bcc45bcb10adc15e21`

Re-tag attempts, now blocked and logged.

Takeaway *We've recommended never re-tagging for years. Now the repository enforces it.*

Staged publishing for releases

We keep pulling from git. We stop presenting a release immediately, and add optional steps before it is installable.

1

Detect

Pull the new tag from git, as today.
Not yet shown to users.

→

2

Verify

Maintainer confirmation via MFA, ideally FIDO2.
Optionally wait for a scan, e.g. Aikido.

→

3

Publish

Release becomes installable, with a publication time we assign, not read from git.

Unlocks at once: trustworthy cooldown, hardened releases for high-reach packages, a real staged release. npm shipped staged publishing in May 2026.

The hard part: keeping dev versions that track mutable branches working alongside it.

MFA: visible, then mandatory

- **~6 months: MFA events in the public transparency log**
  Enable, disable and recovery-code use, visible alongside existing entries.

- **Then: MFA status on maintainer profiles**
  A private hygiene question becomes a visible package property.

- **Direction: FIDO2, then mandatory MFA**
  All accounts, stricter for orgs and large packages, with lead time. PyPI made 2FA mandatory Jan 2024; npm moving to FIDO 2FA.

- **Public security posture info for users**
  Consumers depend on these packages and deserve to factor the security posture into their choices.

Takeaway *Maintain anything on Packagist.org and MFA is off? Turn it on now.*

Organizational package ownership

- **Organizations as first-class entities**
  Members and permissions, package and vendor-prefix ownership, team transitions without a shared login. Replaces the shared company account.

- **Funded by the Sovereign Tech Agency, via the PHP Foundation**
  Same programme as the transparency log, now in year two.

- **Basis for much of the roadmap**
  Org-level mandatory MFA, bulk management across a vendor prefix. How else would Symfony stage a release across all its packages?

Takeaway *Fixing ownership unblocks MFA for organiziations and staged releases.*

Hosting immutable artifacts on Packagist.org

- **Store the generated zip once, immutably**
  Stop regenerating per request. Keep one artifact, record its hash in the transparency log, checksummable.

- **Verifiable immutability beats a trust model we don't have**
  Residual risk of our infra altering an artifact is smaller than the value of a stable, verifiable release.

- **Deliberately no uploads yet**
  Not until provenance and verification exist. Uploads without them caused the leaked-credential compromises on npm, PyPI and others.

Takeaway *Verifiable immutability we can ship now, without repeating other ecosystems' upload-credential mistakes.*

Build provenance and verification

- **Build provenance and attestations on hosted artifacts**
  SLSA provenance plus Sigstore attestations, verified client-side. What was built, where, when, from which commit.

- **The 2.10 policy framework already aims here**
  Built to let orgs choose, at fine granularity, which artifacts to trust by provenance.

- **Standards we target**
  OpenSSF Principles for Package Repository Security Authorization Track L3 (repo provides build provenance), and SLSA Dependency Track (Working Draft) L3-L4.

- **Long-term**
  Major client and repository work. npm shipped signed provenance in 2023, PyPI attestations in 2024. We want the ergonomics right first.

Trusted Publishing with OIDC

- **Publish with short-lived, scoped OIDC credentials**
  Not long-lived tokens. Temporary, self-expiring, configured once over public info.

- **For consumers and CI: nothing to leak**
  No tokens to lose, rotate, or leak. Transitions stop being about who holds the credential.

- **Live and widely adopted**
  On Private Packagist since Sept 2025. PyPI, RubyGems, crates.io, npm and NuGet have all adopted it.

- **Keeping the strong source-to-artifact tie**
  Keeps the source-to-artifact tie if we let people upload to Packagist.org: phpstan's phar build, or multi-package repos like Symfony or the Google Cloud SDK.

Private Packagist: the supply chain security tool for PHP

PRIVATE PACKAGIST

Everything Packagist.org and Composer ship, plus organization-wide enforcement a single developer or AI agent cannot override. For a business running on PHP packages, this is the way to a safe supply chain.

Enforced org-wide, today

- Disabled dist mirroring fallbacks for third party repos
- Disables dist-to-source fallback by stripping source URLs
- Require a safe and up-to-date Composer
- Block malware with no local overrides
    - even on older Composer versions

Shipping this week

- Plugin allow list
- Org-wide control over dependency policy: security advisories, abandoned packages, malware
    - Applies on older Composer versions too

Takeaway *The recommended way to run a secure PHP supply chain in a business.*

The infrastructure behind Packagist.org

455K

packages

5.6M versions

4B

installs / month

and accelerating

4,800

requests / second

average

50 TiB

transfer / month

mostly 304s + JSON

- Metadata only, no artifacts. Mostly 304s thanks to caching, plus JSON. Growth accelerating since early this year, AI a likely driver.
- Off our home-grown multi-region OVH setup to Bunny CDN about a year ago. Direct artifact distribution is part of the plan.
- AWS core: EC2, RDS, ElastiCache / Redis. Redundant, handles webhooks and pulls data in, mostly from GitHub.
- Want to invest far more in hardening security of infrastructure and processes. Last third-party infrastructure audit: summer 2024, funded by Alpha-Omega at the Linux Foundation.

Live data:

`Statistics` packagist.org/statistics

`Dashboard` p.datadoghq.com/sb/x98w56x71erzshui-4a54c45f82bacc991e83302548934b6a

Who pays for this

~$1M

2026 expenses

to run what exists

~$1.3M

needed

for the ambitious work, with help

~$520K

from Private Packagist

the majority, and a strain

- Private Packagist funds most of it, tough for us. Rest from sponsors: Aikido, AWS, Bunny, Algolia, Tideways, Datadog, plus a Sovereign Tech Agency grant. Looking for more.
- Bigger picture, from the OpenSSF “Open Infrastructure Is Not Free” letters (I and II), co-signed: 10+ trillion downloads in 2026, run by small paid teams and volunteers on donations and credits, AI driving load up.
- The ask: commercial users and stakeholders come to the table as paying customers.

OpenSSF, Open Infrastructure Is Not Free:

`Part I` openssf.org/blog/2025/09/23/open-infrastructure-is-not-free-a-joint-statement-on-sustainable-stewardship

`Part II` openssf.org/blog/2026/05/06/open-infrastructure-is-not-free-part-ii-the-hidden-cost-of-running-package-registries

Takeaway *Open infrastructure is not free. Today it runs on a handful of sponsors and one product subsidizing the rest.*

What to do

If you maintain packages

- Enable MFA now.
- Move off shared accounts.
- Upgrade to Composer 2.10.
- Install zizmor on all GitHub Actions

If you depends on Composer & Packagist

Ship an SDK for a paid SaaS, or run an open-core business on packages we distribute?

Sponsor us: sponsoring@packagist.org

If you consume from Packagist.org

Ease load on the free service through:

- Caching
- Internal private Composer repository (Private Packagist!)
- Update to Composer 2.10

Questions / feedback?

Bluesky @naderman.de Mastodon @naderman@phpc.social X @naderman

contact@packagist.com packagist.com

Nils Adermann · Private Packagist