



DrupalCamp

Berlin 2024



DrupalCamp Berlin 2024

Composer Best Practices and Supply Chain Security

Nils Adermann
@naderman

Private Packagist
<https://packagist.com>





Supply Chain?







18

MSC HOME TERMINAL

17

MSC HOME TERMINAL

16

MSC HOME TERMINAL

15

MSC HOME TERMINAL

14

13

MSC TRIESTE

MSC BELGIUM

2.0





Supply Chain - But for Software?!

Raw materials

Refining, processing, constructing

Product components

Assembly, logistics

Quality assurance

Order fulfillment

Source code

Build process

Dependencies, Hardware, Network

Package management

QA / Tests / CI Service

Deployment process

Take with a grain of salt - this comparison will only take you so far

Software Supply Chain

A software supply chain is composed of the components, libraries, tools, and processes used to develop, build, and publish a software artifact.

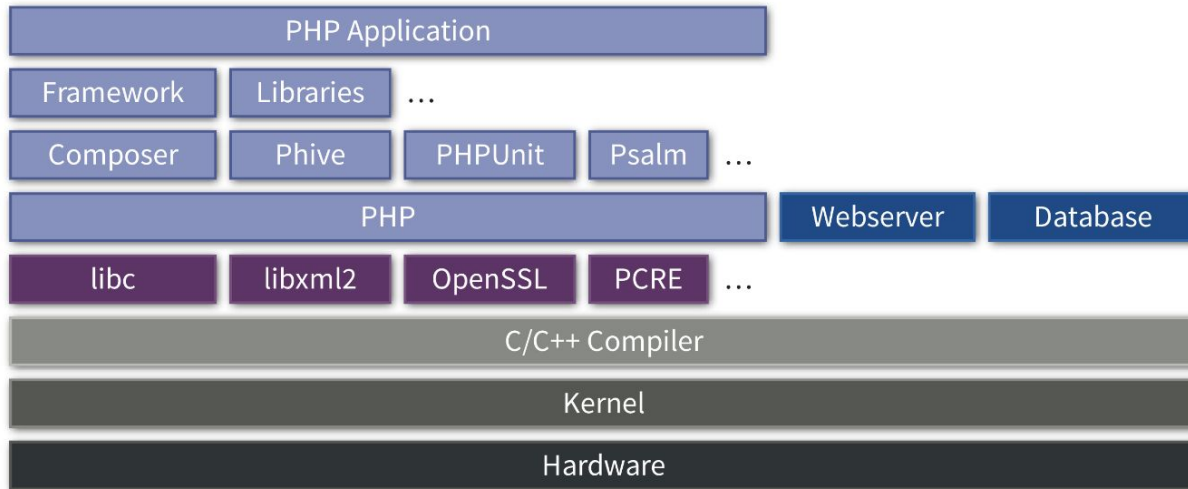
https://en.wikipedia.org/wiki/Software_supply_chain

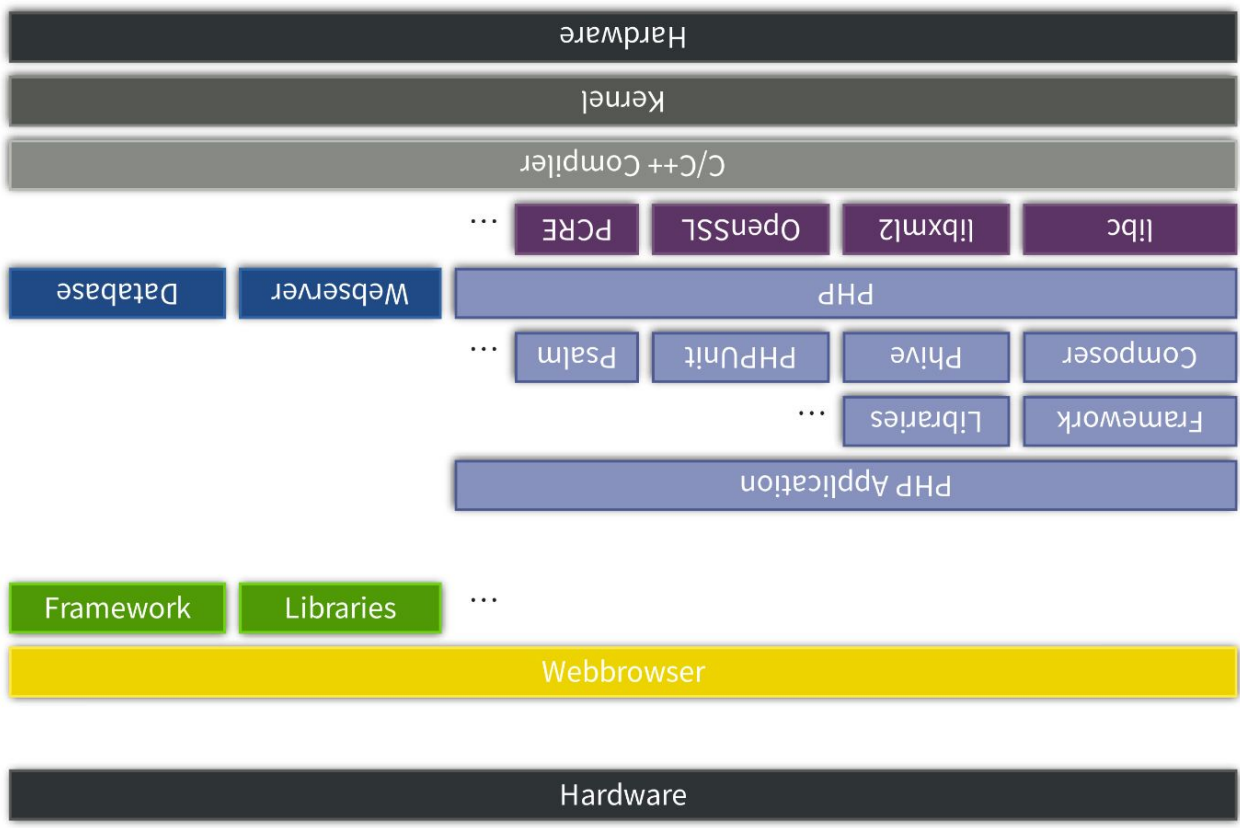
Software Supply Chain

In other words:

The “full-stack” and all processes & tools involved in making and assembling it

Full-stack





Supply Chain Attacks

- Heartbleed - <https://heartbleed.com/> - 2014
 - OpenSSL: System memory accessible externally
- SolarWinds Orion / 2020 United States federal government data breach
 - attackers gained entry to a build system, likely through a compromised Office 365 account
 - modified software updates to include remote access on any machine installing Orion
 - discovered in December '20 after breach Sep '19

Supply Chain Attacks

- Log4Shell
 - <https://en.wikipedia.org/wiki/Log4Shell>
 - Log4j vulnerability, standard Java logging library
 - existed 2013 - November 24, 2021
 - Arbitrary code execution, extremely widely used, CVSS Score 10/10
- XZ Utils / liblzma
 - https://en.wikipedia.org/wiki/XZ_Utils_backdoor
 - Introduced by covert malicious maintainer
 - Backdoor in compression library running in OpenSSH process granting remote access
 - Fortunately detected very early in distribution on March 29th

Supply Chain Attacks

- Depublication of left-pad
 - <https://qz.com/646467/how-one-programmer-broke-the-internet-by-deleting-a-tiny-piece-of-code>
- PyPi Typosquatting with malicious code
 - <https://blog.phylum.io/phylum-discovers-revived-crypto-wallet-address-replacement-attack/>
- Public Travis CI Logs (Still) Expose Users to Cyber Attacks
 - <https://blog.aquasec.com/travis-ci-security>
- Malicious commits made to php-src in the name of Rasmus Lerdorf and Nikita Popov
 - <https://news-web.php.net/php.internals/113838>

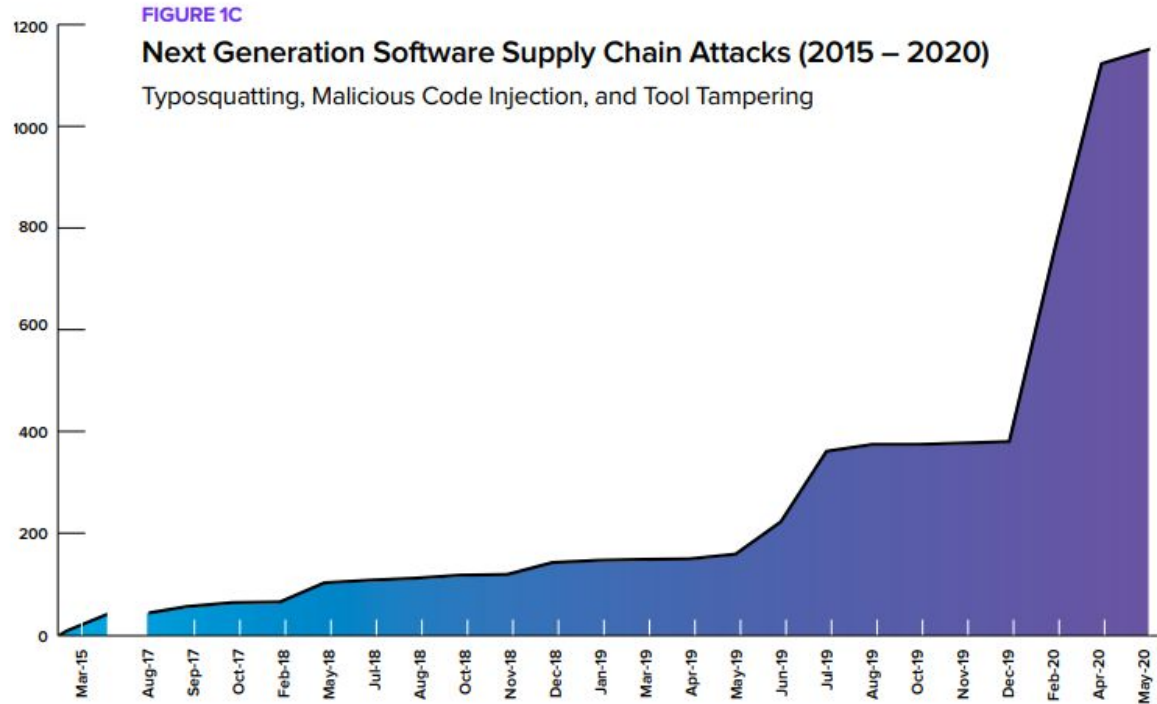
Other Supply Chain Problems

- Jira: Atlassian customers frustrated by weeks-long outage, lack of communication from company
 - <https://www.techrepublic.com/article/atlassian-customers-frustrated-by-weeks-long-outage-lack-of-communication-from-company/>
- Following theft of GitHub OAuth tokens from Heroku, GitHub resets tokens but Salesforce takes weeks to reset passwords and restore functionality
 - <https://www.zdnet.com/article/heroku-to-begin-user-password-reset-almost-a-month-after-github-oauth-token-theft/>

Why should you care?

- Online crime is rampant
- **Criminals may attack your Drupal website** to steal your visitors/users/customers identities, payment info, or other personal data even if it's just for phishing or social engineering
 - Don't think your data isn't valuable!
- Still essentially fighting the same OWASP Top 10 as 20 years ago
 - But also in your dependencies!

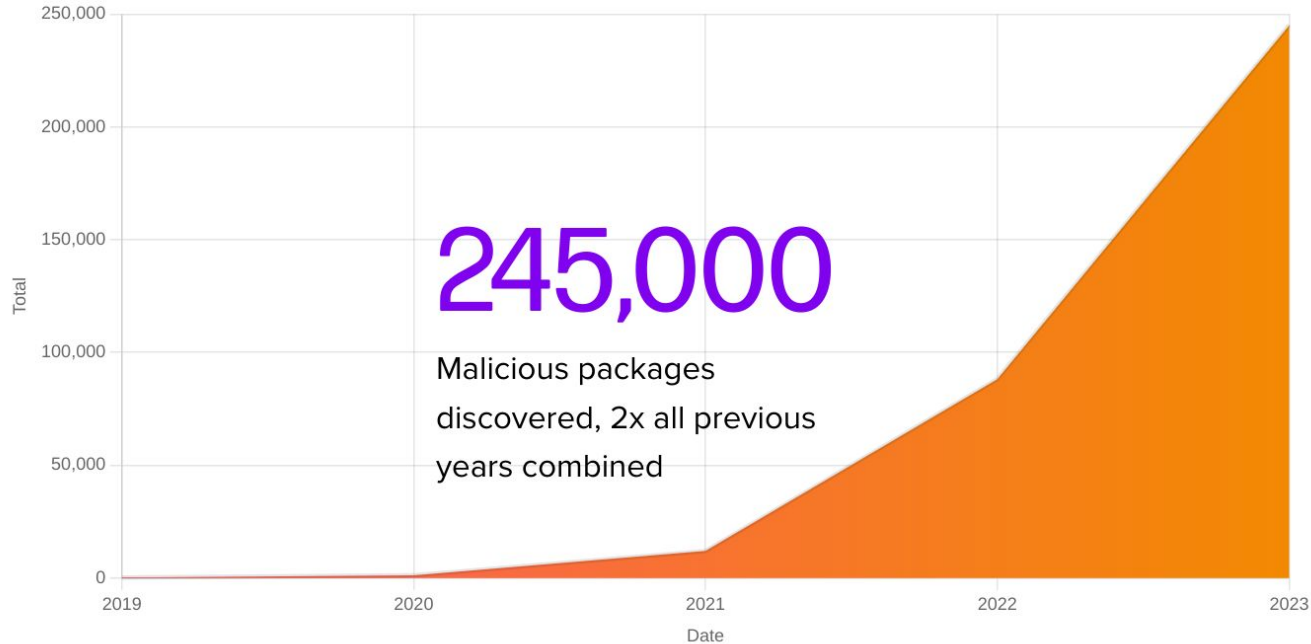
Supply Chain Attacks



“2020 State of the Software Supply Chain” by sonatype

https://www.sonatype.com/hubfs/Corporate/Software%20Supply%20Chain/2020/SON_SSSC-Report-2020_final_aug11.pdf

FIGURE 1.7. NEXT GENERATION SOFTWARE SUPPLY CHAIN ATTACKS (2019-2023)

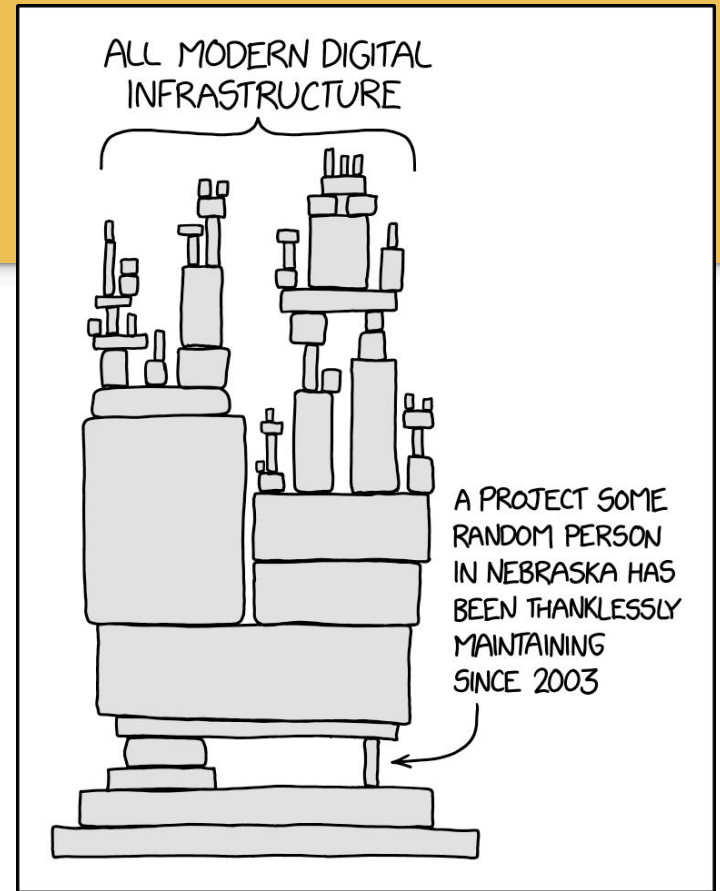


“9th Annual State of the Software Supply Chain” by sonatype

<https://www.sonatype.com/hubfs/2023%20Sonatype-%209th%20Annual%20State%20of%20the%20Software%20Supply%20Chain-%20Update.pdf>

Supply Chain Funding

- \$2,000 donations per year to OpenSSL
- \$841 in 3 days after Heartbleed 🤖
- Creation of Open Software Security Foundation (OpenSSF) at Linux Foundation
 - > \$10M raised by 2021
- German Government: Sovereign Tech Fund/Agency
 - <https://sovereigntechfund.de> since 2022
 - €17M budget in 2024, €11.5M in 2023
- Alpha-Omega
 - <https://alpha-omega.dev/> since 2022
 - \$2.8M granted in 2023



Supply Chain Funding

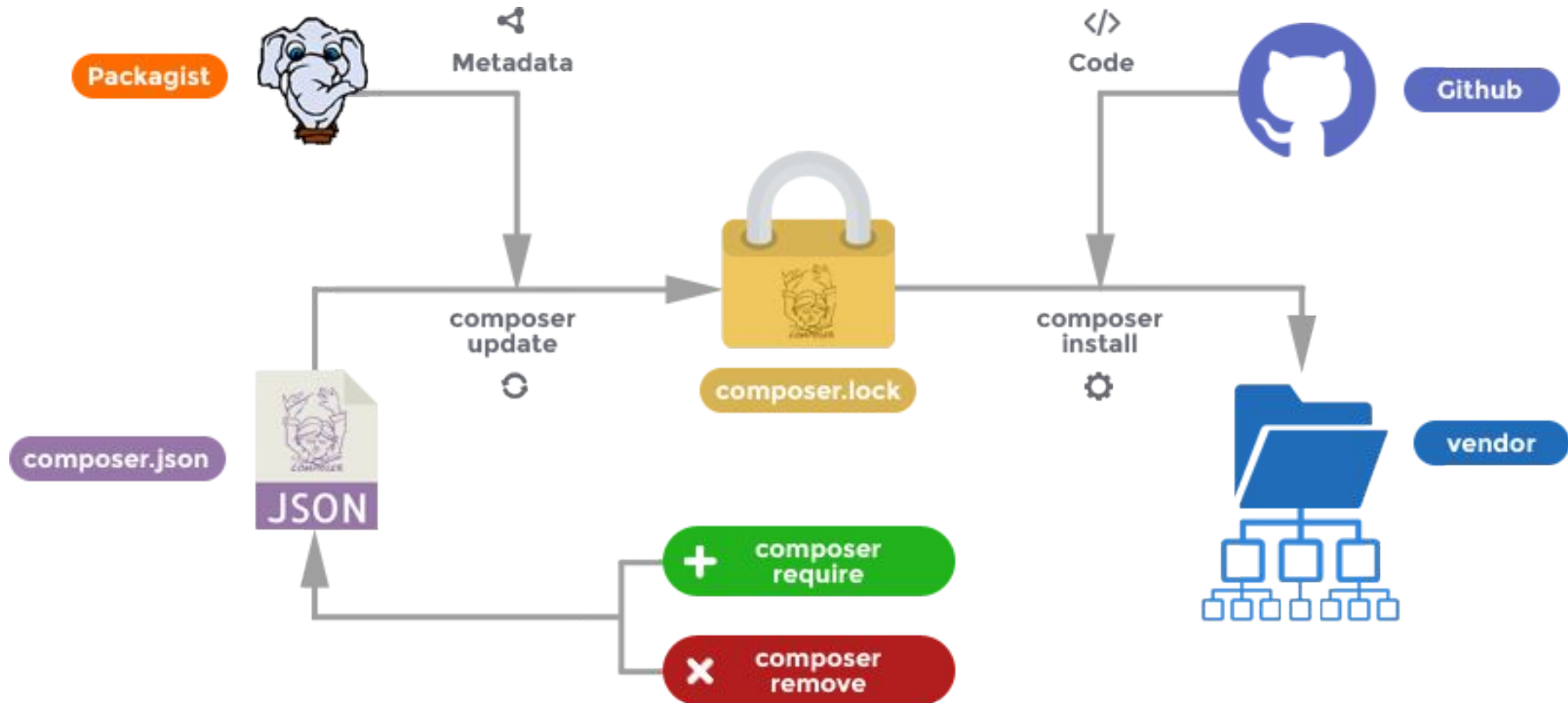
- It's your supply chain, you need to help fund it!
- Drupal credit system is great! Participate!

- composer fund will tell you which of your dependencies need financial help
- Sponsor the PHP Foundation
 - <https://thephp.foundation/sponsor/>
- Buy a Private Packagist subscription to help fund Composer development
 - <https://packagist.com>
- Join the Open Source Pledge
 - Commit to sponsoring open source for at least \$2000/year per FTE-equivalent developer
 - <https://opensourcepledge.com/>



Composer Best Practices and Supply Chain Security

composer update vs. composer install



Packagist.org

- Metadata only
 - No checksums for GitHub stored packages
 - <https://github.com/sansecio/composer-integrity-plugin>
 - No signatures
 - <https://www.drupal.org/project/infrastructure/issues/3325040> - TUF
 - No way to upload code
- Positively
 - Everything over TLS
 - Installation from GitHub source archive URLs improves trust in artifacts
 - Smaller attack surface on packagist.org

Composer Supply Chain Vulnerabilities

- Mar 11, 2021: Git Clone Security Vulnerability
 - <https://blog.packagist.com/git-clone-security-vulnerability/>
 - Git vulnerability on case insensitive filesystems can be exploited through Composer if you clone dependencies
- Apr 27, 2021: Composer Command Injection Vulnerability
 - <https://blog.packagist.com/composer-command-injection-vulnerability/>
 - Code execution through Mercurial repository URL injection
- Apr 13, 2022: Composer Command Injection Vulnerability
 - <https://blog.packagist.com/cve-2022-24828-composer-command-injection-vulnerability/>
 - Code execution through Git or Mercurial branch names

Composer Supply Chain Attacks

- May 19, 2022: GitHub Repo Jacking
 - Attacker registered GitHub username of former maintainer
 - Republished package with malicious code to steal AWS credentials
 - <https://thehackernews.com/2022/05/pypi-package-ctx-and-php-library-phpass.html>
 - <https://github.blog/2024-02-21-how-to-stay-safe-from-repo-jacking/>
 - Problematic with VCS repo URL references in composer.json too
 - Packagist.org uses GitHub repo ids: <https://github.com/composer/packagist/pull/1411>
- May 1, 2023: Packagist.org maintainer account takeover
 - <https://blog.packagist.com/packagist-org-maintainer-account-takeover/>
 - Editing of source URLs no longer allowed beyond 50k installs

Protecting yourself from Composer Supply Chain Attacks

- Common wrong suggestion: “Vendoring”
 - Committing the contents of your vendor directory to source control
- Wrong why?
 - You still need to update your dependencies
 - Either still use the dependency manager to update the vendor’d dependencies
 - Or download everything manually
 - A lot of error prone work
 - Would you notice repo jacking?
 - But there’s more!

Why vendoring doesn't protect you

- Who here knows how to commit changes to the files?

Why vendoring doesn't protect you

- Who here knows how to commit changes to the files?
 - `git add vendor/` will not delete files, can lead to bugs and security issues
 - Must use `git add -A vendor/`
- vendor directory contents can diverge from expected content
 - How do you verify vendor directory contents match the lock file?
 - e.g. are deleted packages really deleted?
- Managing conflicts in larger teams gets even harder than managing lock file contents

Why vendoring doesn't protect you

- Bad Actor scenarios, e.g. disgruntled employee
 - Scenarios
 - Could place code in unmanaged directory in vendor looking like a dependency
 - Could modify code of existing package in vendor/
 - Would your review process catch these as part of a large update commit?
 - If not, do you have tooling to notice the discrepancy?
 - Is building this tooling less work/cheaper than using a private Composer repository?

Generally: **No, don't commit the vendor directory**

TUF on packages.drupal.org

- The Update Framework (TUF) maintains the security of software update systems, providing protection even against attackers that compromise the repository or signing keys.
- Composer plugin
 - necessity for auto update initiative
 - an experiment for Packagist
- Does not help identifying authors of software
- Does not mean delivered software doesn't have security issues
- Does not solve the question of whether you can trust the maintainers

TUF on packages.drupal.org

```
composer config allow-plugins.php-tuf/composer-integration true  
composer require php-tuf/composer-integration:dev-main --dev
```

```
composer config repositories.drupal-core composer https://packagist-signed.drupalcode.org
```

```
mkdir tuf
```

```
curl -o tuf/packages.drupal.org.json https://packages.drupal.org/8/metadata/1.root.json
```

```
curl -o tuf/packagist-signed.drupalcode.org.json
```

```
https://packagist-signed.drupalcode.org/metadata/1.root.json
```

```
composer tuf:protect https://packages.drupal.org/8
```

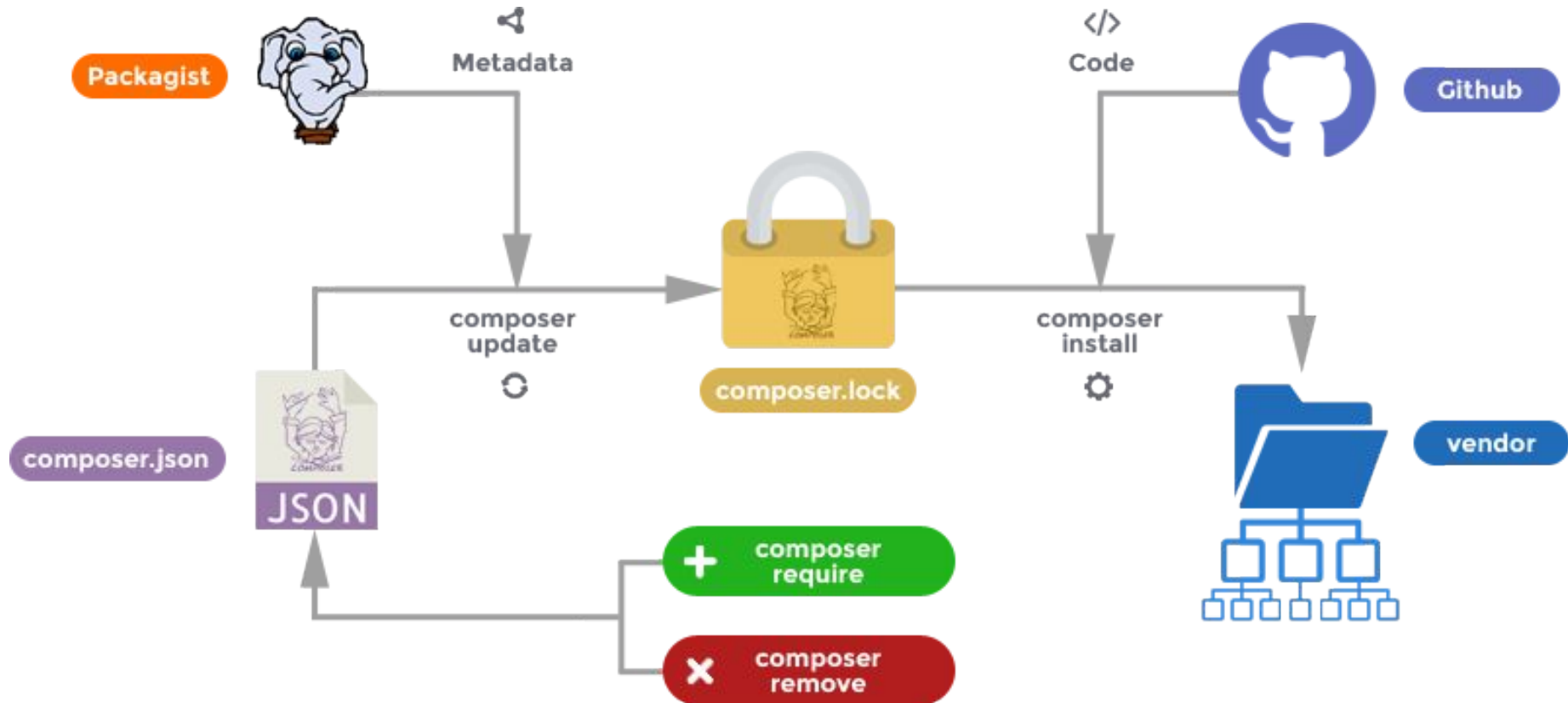
```
composer tuf:protect https://packagist-signed.drupalcode.org
```


Use your own Composer repository

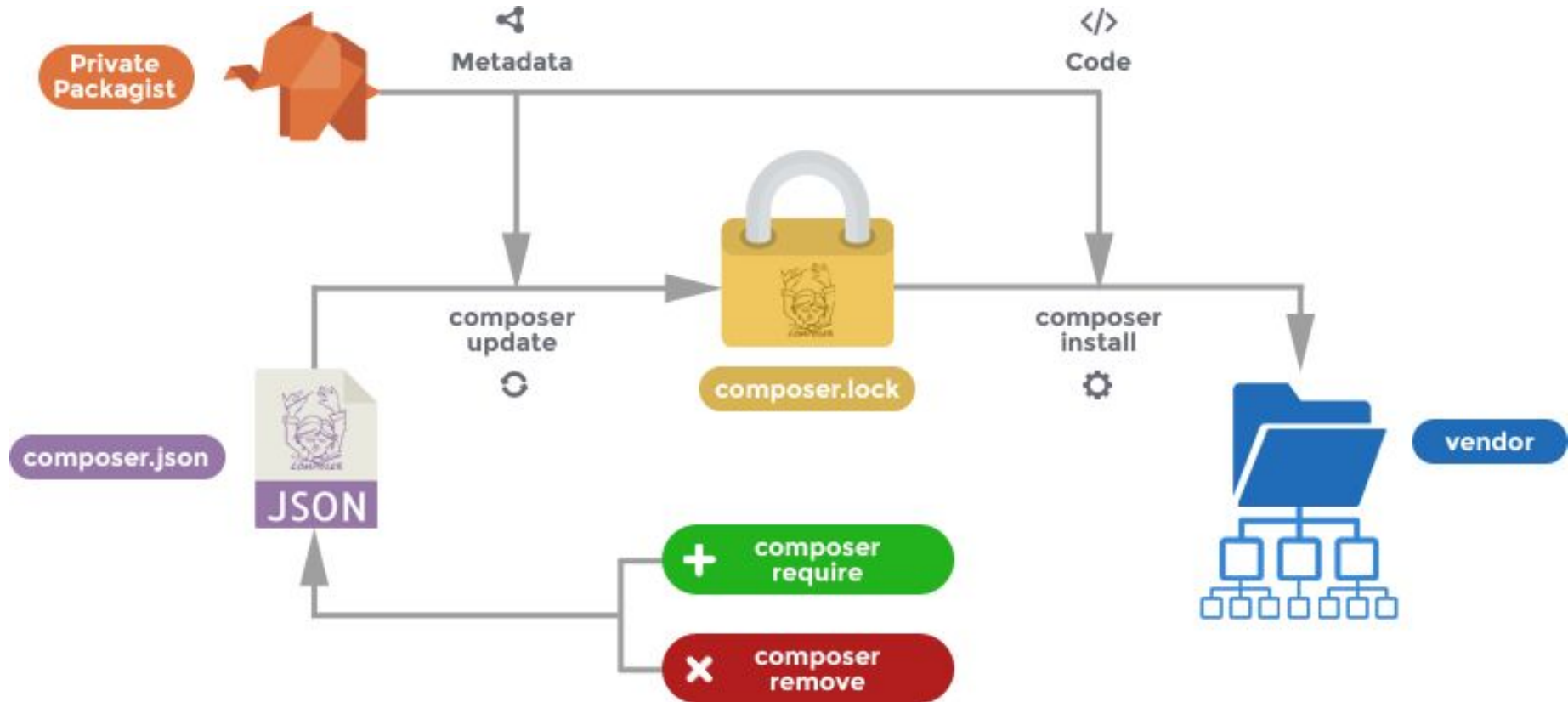
- Satis
- JFrog Artifactory
- Sonatype Nexus Repository
- Cloudsmith
- GitLab Package Registry
- ...
- **Private Packagist**

Private Packagist

- Stores a copy of all used versions of your dependencies
 - Safe from deletion
 - Safe from modification
- Serves package metadata **and** code
- Possible with some alternatives but usually with more effort and less convenience
 - e.g. copy all dependencies into git repositories, how do you keep those updated then?



Private Packagist





Never Deploy without a Lock File

Do not run composer update during deployments

Recommended use of Composer in your Deployment Process

- commit composer.lock
- CI/CD
 - run composer install (not update!)
 - generate any potentially generated code
 - dump an optimized autoloader
 - package everything into an archive
- deployment
 - upload to production servers, move in place
 - run composer check-platform-reqs
 - switch webserver to use new code

Result

- no surprises in production
 - same dependency versions as tested
 - no risk of composer conflicts during deploy
 - code doesn't change at runtime
- deploying to multiple servers
 - exact same state everywhere
 - no unnecessarily repeated work

Composer 2.4: composer audit

- **composer audit** Command

- Lists vulnerable versions in composer.lock
- Uses packagist.org vulnerability db API
 - GitHub advisory database
 - FriendsOfPHP/security-advisories
- Checks packages.drupal.org for Drupal module vulnerabilities
- Returns non-zero if vulnerabilities found -> can check in CI

- `composer update` implies `audit --format=summary`

- `composer require --dev roave/security-advisories:dev-latest`

Update Dependencies Frequently

- Set up a schedule or regular reminder to run dependency updates
- Set up alerting when vulnerabilities are discovered in your dependencies
 - GitHub Dependabot
<https://docs.github.com/en/code-security/dependabot/dependabot-alerts/about-dependabot-alerts>
 - Snyk
<https://snyk.io/product/open-source-security-management/>
 - **Private Packagist** Security Monitoring
<https://packagist.com/features/security-monitoring>

Update Dependencies Frequently

Better yet: Automate your updates


- Mend Renovate <https://www.mend.io/renovate/>
- GitHub Dependabot <https://github.com/dependabot>
- *(WIP: Private Packagist Automated Updates - talk to me after!)*

Get a pull request anytime an update is necessary

Caution!

Private Packagist Update Review


GitHub
BitBucket
GitLab


 **Update php-dev-dependencies #2794**
renovate wants to merge 2 commits into `main` from `renovate/php-dev-depend...`


This PR contains the following updates:

Package	Type	Update	Change
friendsofphp/php-cs-fixer	require-dev	minor	3.13.2 -> 3.16.0
friendsofphp/php-cs-fixer	require-dev	minor	3.14.1 -> 3.16.0
phpstan/phpstan-symfony	require-dev	patch	1.3.1 -> 1.3.2

This PR has been generated by [Mend Renovate](#). View repository job log [here](#).



 Update php-dev-dependencies Verified ✓ e6f84d0

 **private-packagist** bot commented 27 minutes ago • edited ▾

accounting/composer.lock

Dev Package changes

Package	Operation	From	To	Changes
friendsofphp/php-cs-fixer	upgrade	v3.13.2	v3.14.1	diff - changelog
phpstan/phpstan-symfony	upgrade	1.3.1	1.3.2	diff - changelog

core/composer.lock

Package changes **NOT DEV**

Package	Operation	From	To	Changes
psr/cache	upgrade	2.0.0	3.0.0	diff - changelog
symfony/cache-contracts	upgrade	v2.5.2	v3.2.1	diff - changelog

Composer Plugins & Scripts

- Composer 2.2 introduced a requirement to explicitly enable plugins
 - `config.allow-plugins`
 - protects you from unintentionally executing malicious code before reviewing `composer.lock` changes
- Scripts & plugin selection is limited to root `composer.json`
 - Protects from attacks by malicious maintainers, dependency confusion or other accidental dependencies
 - You still have to review your lock file changes!

Composer Best Practices and Supply Chain Security: Key Takeaways

- **composer.lock matters!**
 - Commit composer.lock
 - Review changes
- **Use a private Composer repository**
 - Don't use "Vendoring"
 - Recommendation: Private Packagist
- **Automate Dependency Updates**
 - Or at least set up monitoring for published vulnerabilities in your dependencies
- **Implement a safe deployment process**
 - Don't run composer update in deploys



Questions / Feedback?



Private Packagist
<https://packagist.com>

E-Mail: contact@packagist.com

X: [@naderman](#)

Mastodon: [@naderman@phpc.social](#)