Phantom Dependencies is your requirements.txt *haunted?*

Seth Larson

(yes)

https://sethmlarson.dev

PHANTOM DEPENDENCIES

What to expect?

- No more jumpscares (promise!)
- Phantom dependencies and where to find them
- Manifesting
- “Is your software vulnerable?”

“Is my software vulnerable?”

https://sethmlarson.dev

What is a “Phantom Dependency”?

“Phantom dependencies are
dependencies used by your code
that are **not declared in the *manifest***”

– Endor Labs, 2023

What is a “Manifest”?

Examples of software manifests:

- requirements.txt
- poetry.lock, uv.lock
- pyproject.toml, setup.py (“install_requires”)
- pip freeze

LET'S MANIFEST
(LIVE DEMO)

https://sethmlarson.dev

What is a “Manifest”?

Examples of software manifests:

- requirements.txt
- poetry.lock / uv.lock
- pyproject.toml / setup.py (“install_requires”)
- pip freeze

What is a “Manifest”?

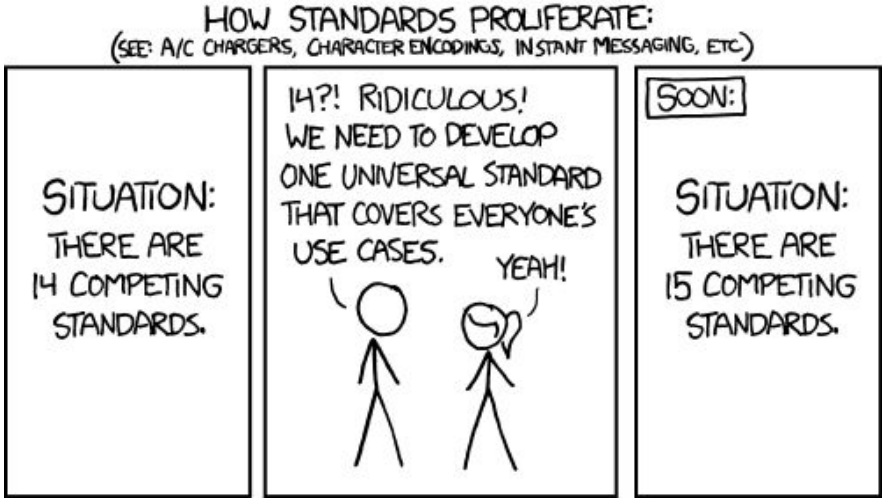
Examples of software manifests:

- requirements.txt
- poetry.lock / uv.lock
- pyproject.toml / setup.py (“install_requires”)
- pip freeze
- **Software Bill-of-Materials**

Software Bill-of-Materials (SBOM “ess-bom”)

“Ecosystem-Agnostic Software Manifest”

CycloneDX, SPDX, ~~SWID~~



Why? **Compliance: EU Cyber Resilience Act (CRA)**

Vulns, licensing, source code analysis, inventory, etc.

Why are Phantom Dependencies a problem?

Software can’t see ~~ghosts~~ software

Software only sees **metadata**

Why are Phantom Dependencies a problem?

Software can’t see ~~ghosts~~ software

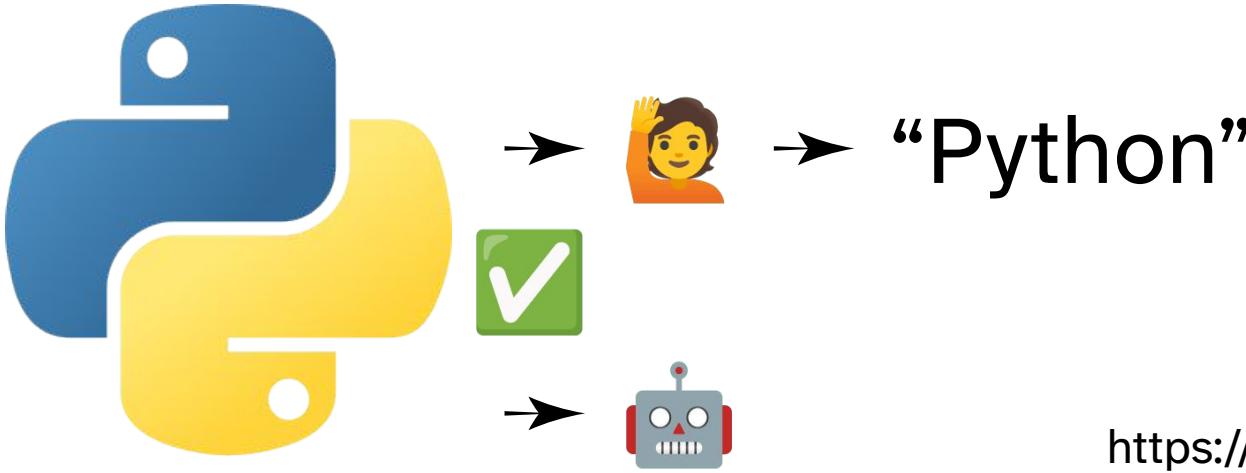
Software only sees **metadata**

https://sethmlarson.dev

Why are Phantom Dependencies a problem?

Software can’t see ~~ghosts~~ software

Software only sees **metadata**

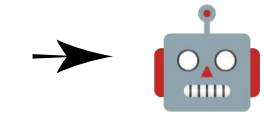


https://sethmlarson.dev

Why are Phantom Dependencies a problem?

Software can't see ~~ghosts~~ software

Software only sees **metadata**



"Python"

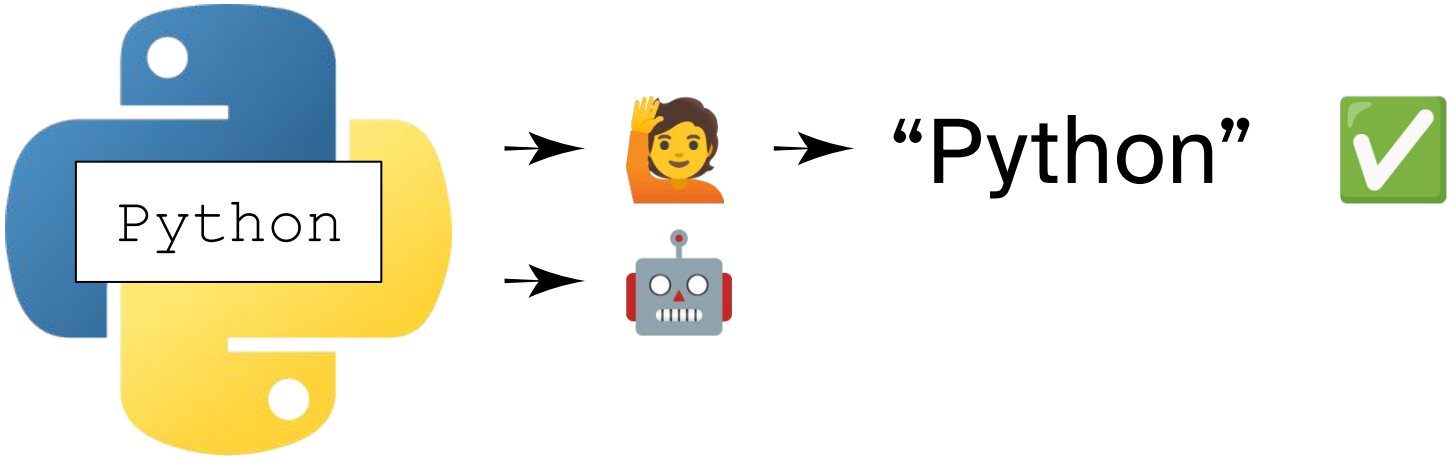
...?

https://sethmlarson.dev

Why are Phantom Dependencies a problem?

Software can’t see ~~ghosts~~ software

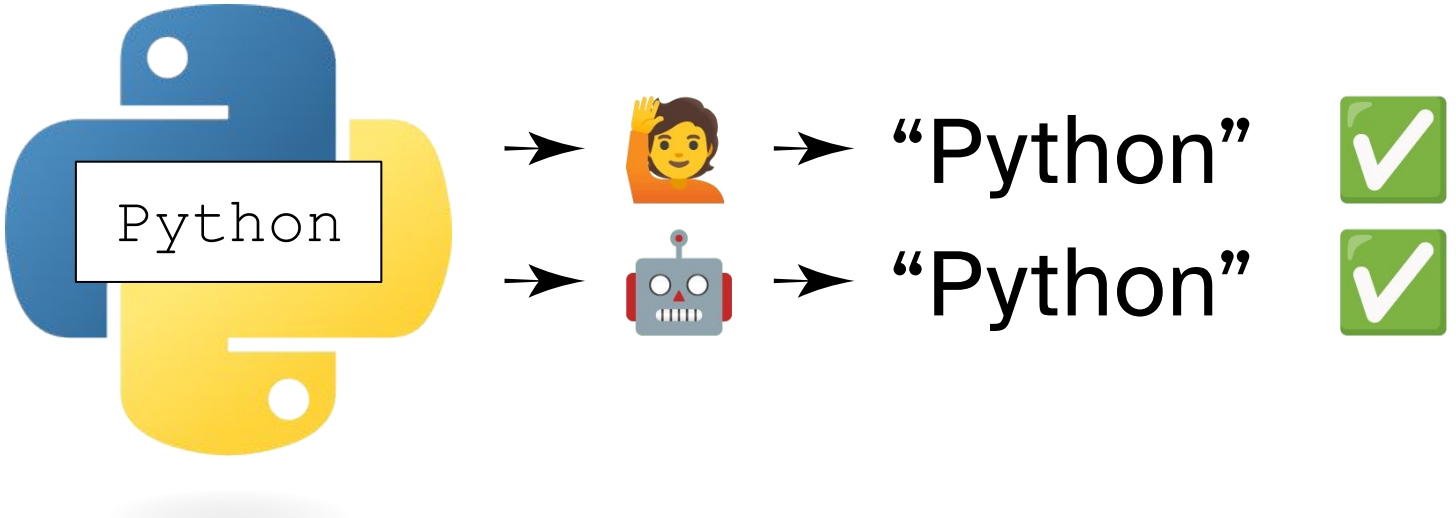
Software only sees **metadata**



Why are Phantom Dependencies a problem?

Software can’t see ~~ghosts~~ software

Software only sees **metadata**



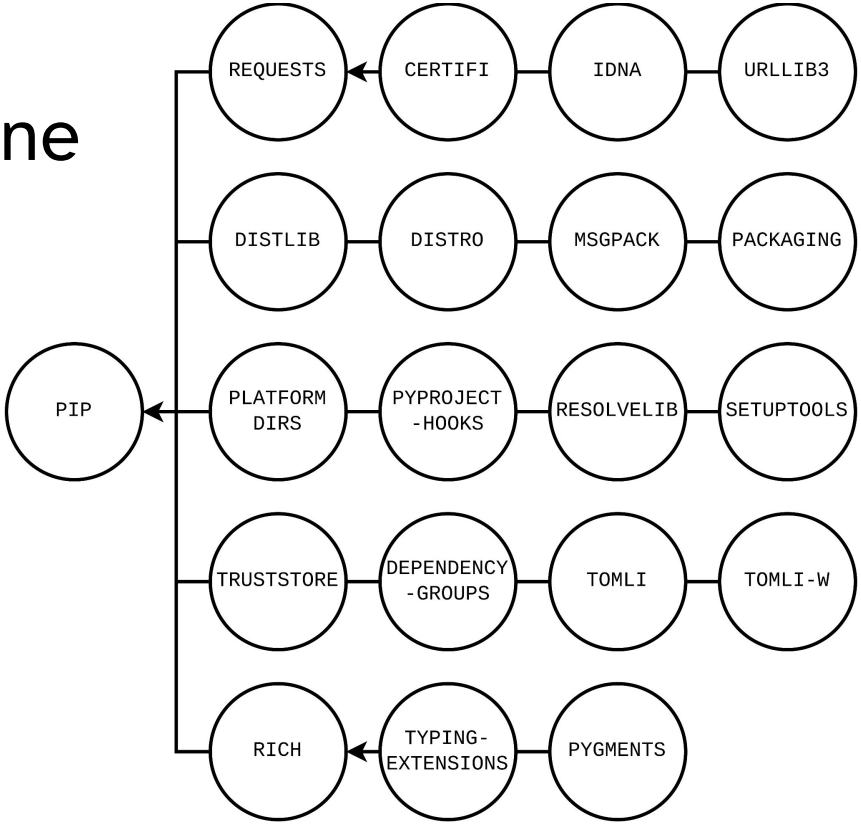
Why do projects have Phantom Dependencies?

Why Phantom Dependencies? ➛ Bundling

One “package” isn’t always one “dependency”…

Scanners only “see” pip, not as 20 distinct projects…

Static linking is “bundling”



Why Phantom Dependencies? ➛ Bundling

Why do projects bundle dependencies?

- Bootstrapping
- 1 dependency version per environment
- Backporting standard library features
- Portability / Deployment

Why Phantom Dependencies? ➛ Manylinux

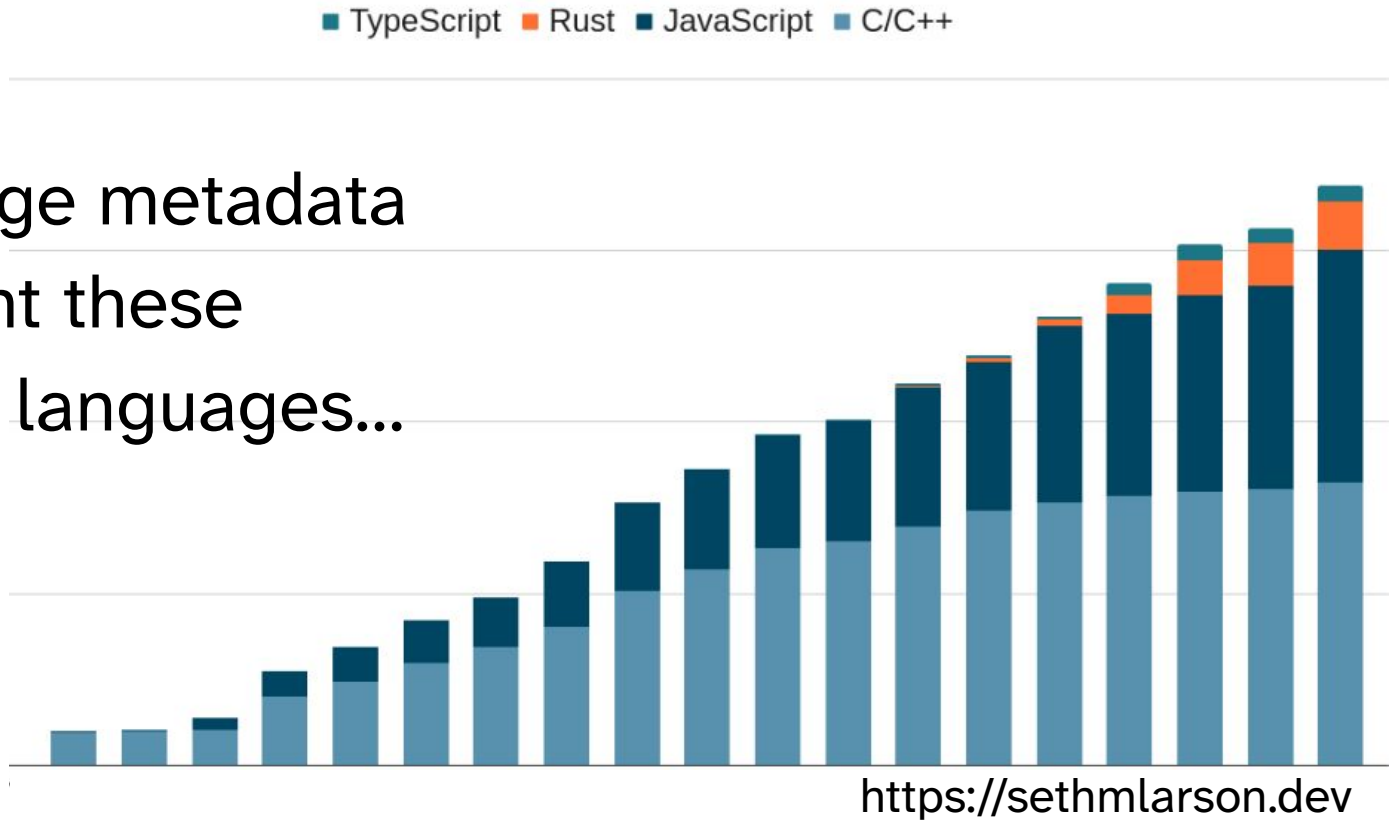
“Wheels that work on (almost) any Linux”

Secret sauce: Assume runtime is glibc/musl X.Y
and a few libraries exist. **Bundle all other libraries.**

Auditwheel is the key tool for manylinux!

Why Phantom Dependencies? ➛ JS, C, Rust

Python package metadata can’t represent these programming languages...



Solving the Phantom Dependency problem?

We need a record (**manifest**) for
ecosystem-agnostic software
inside a Python package...

Solving the Phantom Dependency problem?

We need a record (**manifest**) for
ecosystem-agnostic software
inside a Python package...

...did somebody say SBOM? 🦸

PEP 770 – Improving measurability of Python packages with Software Bill-of-Materials

Author:	Seth Larson <seth at python.org>
Sponsor:	Brett Cannon <brett at python.org>
PEP-Delegate:	Brett Cannon <brett at python.org>
Discussions-To:	Discourse thread
Status:	Accepted
Type:	Standards Track
Topic:	Packaging
Created:	02-Jan-2025

“.dist-info/sboms/”

PEP accepted April 11th

Thank you, reviewers!

https://sethmlarson.dev

Generate SBOMs for repaired libraries #577

Open **sethmlarson** wants to merge 5 commits into `pypa:main` from `sethmlarson:sboms`

sethmlarson commented 2 weeks ago Member

(Draft, do not merge) This is my initial pull request for adding automatic SBOM generation based on package manager info for libraries that are repaired into wheels. I've tested locally by building wheels from various projects and operating systems, will work on getting those pulled into the test suite.

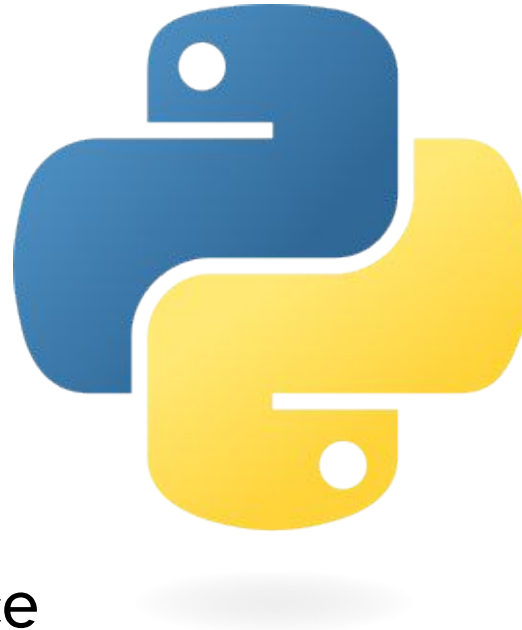
The majority of the "logic" comes from the project "whichprovides" which gets bundled into auditwheel as a single file. You can review that project in its entirety within this pull request. If you'd like to submit comments that you have about "whichprovides" you can do so here and I'll get them addressed in the upstream project.

LET'S MANIFEST
(LIVE DEMO)

https://sethmlarson.dev

Support the Python Software Foundation!

Developers-in-Residence Program
Security sponsored by **Alpha-Omega**



- Supply-Chain Security
- Python Package Index
- Python Language

https://python.org/psf/developersinresidence

“Is my software vulnerable?”

Start with an SBOM, not a scan

I recommend you use “Syft”, *but*...

Human in the loop: don’t trust, verify!

- Are these the packages I expect? What’s missing?
- Are there software identifiers? (PURL, CPE)
- Check out your virtual environments

https://sethmlarson.dev

“Is my software vulnerable?”

Scanner should support non-Python software.

My recommendation: “Grype”

grype

Only use pip-audit if you’re using pure Python. *Check your manifest!*

https://sethmlarson.dev

“Is vulnerability data available?”

For each package ecosystem (PyPI, deb, cargo)...

Find a known vulnerability **and make it show up!**

Downgrade a version in SBOM (version, PURL, and CPE) and rescan, does the vulnerability appear?

Key Takeaways

- They were friendly ghosts all along 👻
- Software dependencies can be complicated
- Know your dependencies: Manifest!
- Python packages measurability improves as more projects adopt PEP 770

https://sethmlarson.dev

References

- https://www.endorlabs.com/learn/dependency-resolution-in-python-beware-the-phantom-dependency
- https://py-code.org
- https://sethmlarson.dev/security-developer-in-residence-weekly-report-16
- https://github.com/pypa/auditwheel
- https://github.com/anchore/syft
- https://github.com/anchore/grype
- https://peps.python.org/pep-0770/