TRAIL OF BITS

The Next 5 Years of Supply Chain Security on PyPI

William Woodruff, Trail of Bits

Hello!

- **William Woodruff (william@trailofbits.com)**
  - open source group engineering director @ trail of bits
  - long-term OSS contributor (Homebrew, LLVM, Python) and maintainer (pip-audit, sigstore-python)
  - @yossarian@infosec.exchange
- **Trail of Bits**
  - ~130 person cybersecurity engineering and auditing consultancy
  - specialities: cryptography, compilers, program analysis research, “supply chain”, OSS package management, general high assurance software development

thank-yous

- our work on PyPI and Python packaging security over the years has been funded by Google's Open Source Security Team, the Sovereign Tech ~~Fund~~ Agency, and the PSF themselves
- special thanks to Dustin Ingram & Hayden Blauzvern

Sovereign Tech Agency

agenda

- **background on PyPI/Python packaging/Python community**
- **systematically securing Python packaging**
  - or: “treating security first and foremost as a usability problem”
- **previous efforts**
- **current and ongoing efforts**
- **the next 5 years**
  - disclaimer: my opinions

PyPI

- **the Python Package Index**
  - “Pie-pea-eye”
- **the index behind `pip install`**
- **~860K users, ~575K projects, ~6.1M releases, ~12.2M files**
- **~51B downloads/month, >2TB traffic/day**
  - somewhat inflated by tools and large CI/CD providers not caching as much as they could!

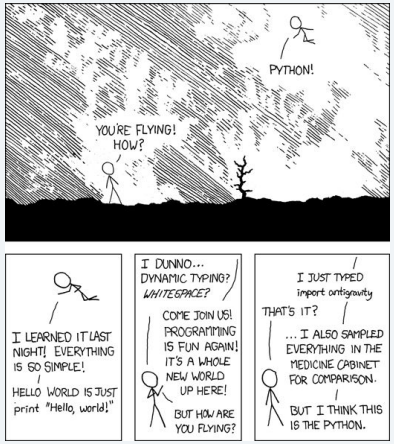
Downloads last day: 2,016,247,697
Downloads last week: 11,901,433,314
Downloads last month: 50,946,221,288

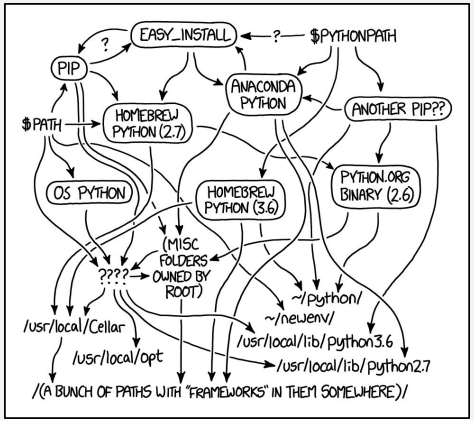
Python packaging/community

- **PyPI is just one piece of the Python packaging constellation**
  - installers: `pip`, `uv`, `poetry`, etc.
  - build tools and backends: `setuptools`, `hatch`, `flit`, `poetry`, etc.
  - upload tools: `twine`, `uv`, `poetry`, etc.
  - very few single tools/toolchains do everything, although this is slowly changing
- **Python packaging is standards driven**
  - Python Enhancement Proposals (PEPs) become living PyPA standards once accepted
  - **not** top-down or tool-driven, unlike Rust (`cargo`) and JS (`npm`)
- **Python's community is massive and diverse**
  - every possible combination of experience, background, interest, skill, time commitment, etc.
  - this makes it hard to establish a common denominator for *anything*, much less security!
  - more closely resembles a country/countries than most other ecosystems!

systematically securing Python packaging

- **diversity is both Python's strength and also a challenge to *systematic* security changes**
  - a lack of systematic planning is ***self-fulfilling***, since new systematic plans need to accommodate conventions/expectations
  - systematic planning ***can be dangerous*** to the things that keep Python's community healthy: onerous security requirements scare newcomers away!





MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

systematically securing Python packaging

to make systematic progress on packaging security, we need to treat security as a *usability* problem!

two core demographics:

- **enthusiasts: 1-5% who will take the initiative to opt into things**
  - good news: most of the top packages are maintained by this demographic!
  - bad news: they're still a tiny overall percentage of package maintainers
- **everyone else: 95-99% who only care about the reliability of their workflow; security changes must not require significant behavioral changes unless those changes make things easier for them**
  - good news: ***if*** we make things easier for them, then they'll happily adopt changes

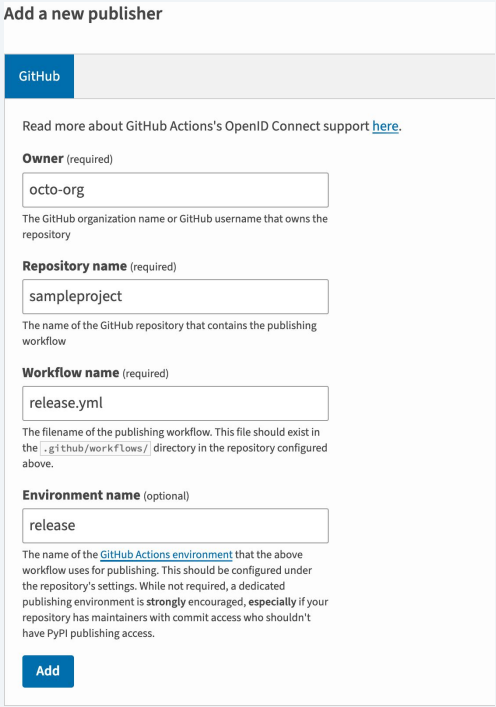
previous efforts

early steps

- **2003-ish: PyPI comes into existence**
  - predated by less-formal packaging, back to 1998
- **2005: PyPI begins to host files**
  - previously was **just** an index, pointing to other hosts
  - HTTP + HTTP redirects to random servers!
- **2013-ish: PyPI adds HTTPS**
  - made mandatory for Web users + default for `pip` and `easy_install`
- **2015: PEP 503 standardizes the index API**
  - including digests!
  - PEP 470 formally deprecates external hosting and removes remaining external links
- **2017: PyPI is rewritten (from “legacy PyPI” to Warehouse)**
- **2018: current PyPI backend (Warehouse) goes live**

the *last* 5 years

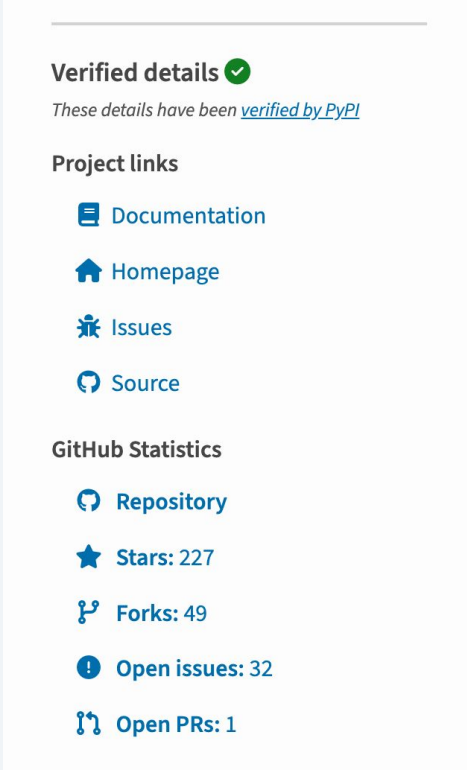
- **2019: PyPI implements and enables MFA**
  - TOTP + WebAuthn (physical tokens)
- **2019: PyPI implements API tokens**
  - includes project scoping
  - complements user/password authentication for uploads
- **2020: initial attempt to implement TUF (PEP 458)**
  - initial attempts at malware detection/scanning on PyPI
- **2022: MFA enforced for “critical” projects**
  - WebAuthn tokens given to designated projects for free
- **2023: Trusted Publishing implemented**
  - complements manual API tokens
- **Jan 1, 2024: MFA mandatory for all projects**
  - also makes API tokens/Trusted Publishing mandatory for uploads!



current and ongoing efforts

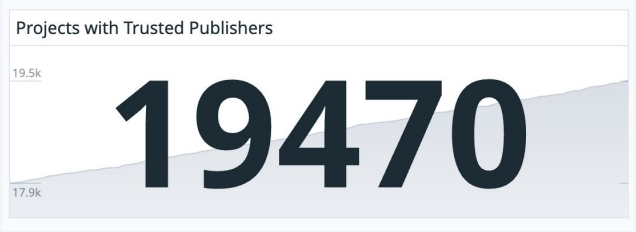
verified project URLs

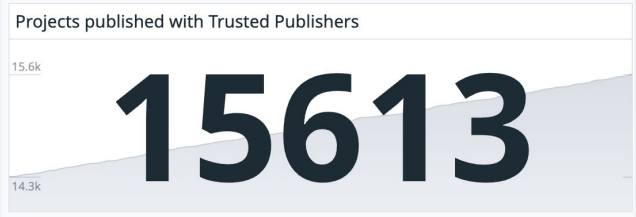
- **where “verified” means “PyPI can corroborate this URL”**
  - uses Trusted Publishing as the source of ground truth!
- **makes it harder for typosquatters, etc. to impersonate a project by submitting a project with malicious contents but valid-looking links**
  - a/k/a “starjacking”

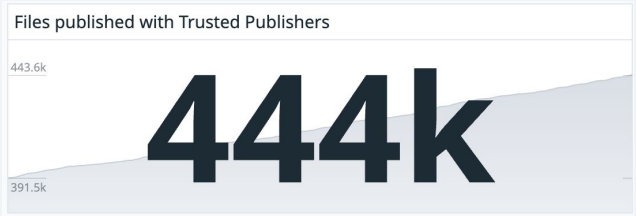


attestations!

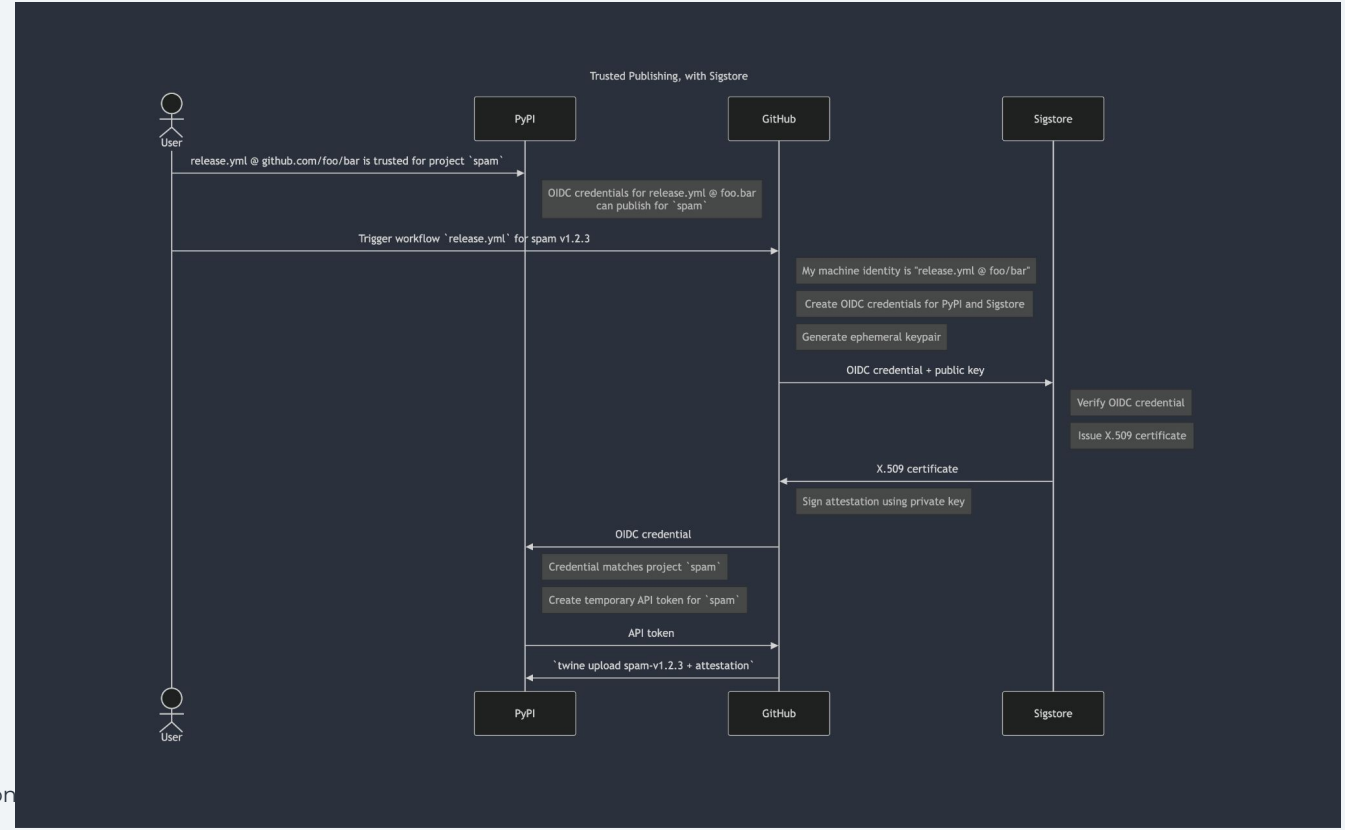
- **Trusted Publishing was a huge success**
  - ***because*** it was a usability ***and*** security win
- **PEP 740 bootstraps package signing on top of Trusted Publishing**
  - means it can be ***enabled by default*** for ~20,000 publishing workflows, with no additional configuration!
  - built on top of Sigstore → no long-term signing keys, only machine identities
  - ***live on PyPI as of last month!***





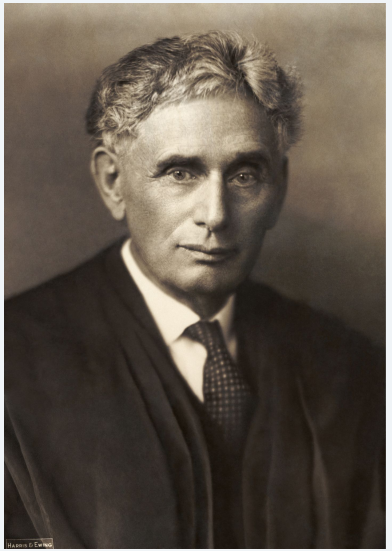


Trusted Publishing + attestations



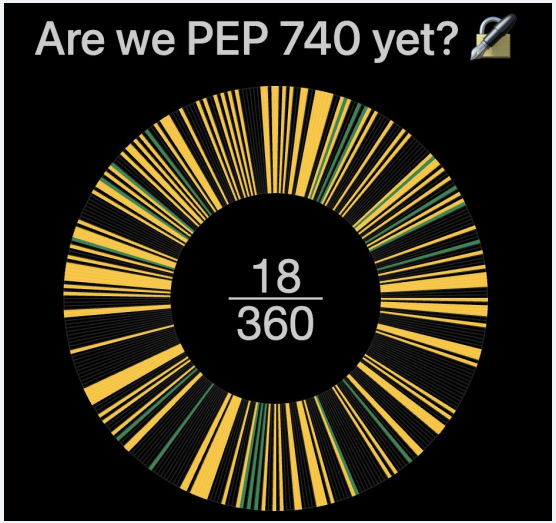
what do attestations actually get us?

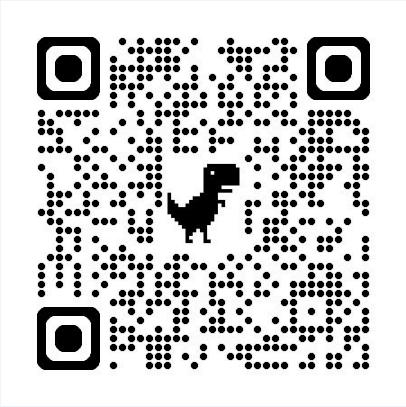
- ***publicly verifiable*** **proof of a Python package's provenance**
  - equivalent to the sentence "package `foo` version `1.2.3` was produced by GitHub repository `foo/foo` via workflow `pypi.yml` at commit `cafe…`"
- **external transparency and auditability for the index and individual publishers**
  - each attestation has a corresponding transparency log entry that can be audited/reviewed
    - …and a corresponding machine identity that can be ***proactively monitored***
  - makes it harder for an attacker to ***get away*** with compromising a package or its repository
  - not perfect, since attestations aren't (and can't be made) mandatory!



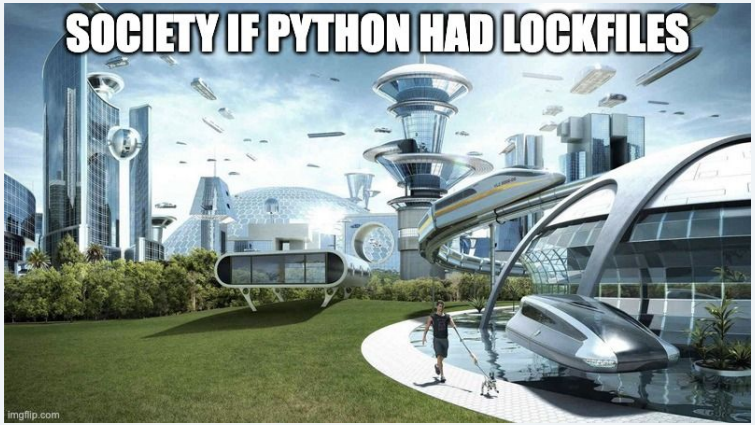
attestation adoption so far?

- **>10,000 attestations uploaded to PyPI since enablement**
- **18/360 (5%) of top projects have attestations uploaded**
  - 2/3rds haven't been uploaded since enablement, so could also be seen as 15%
- **not bad considering it's *only been enabled for 2 weeks*!**





the *next* 5 years



extending attestations

- **as defined in PEP 740, attestations are (intentionally) very limited:**
  - only Trusted Publishers can be used as signing identities
  - attestation predicates intentionally limited for a MVP to just SLSA attestations and PyPI "publish" attestations
- **more attestation types!**
  - VSAs?
  - third-party attestations, e.g. attesting that a package has been reviewed?
- **more signing identities!**
  - email identities, since PyPI knows which email addresses have been verified
  - domain identities, possibly?

goal #1: expand the breadth/depth of meaningful attestations

it's one thing to sign; another to verify

- **generating attestations is the "easy" part!**
- **the index (= PyPI) verifies attestations on upload, but getting *user agents* (e.g. `pip`) to verify is *much harder***
  - architectural constraints: `pip` needs to be pure Python for vendoring reasons, Sigstore's Python client isn't (uses Rust via PyCA cryptography)
  - standards/tooling limitations: lack of a standard lockfile format makes a "TOFU" verification story for users nontrivial; need to first establish a way to lock expected signing identities for packages

goal #2: bring attestation verification to downstream users!

attestations don't go far enough

- **attestations are cool but *fundamentally limited in scope***
  - not everybody uses Trusted Publishing (or can use it), and this is fine!
  - ***index-wide*** monitoring is ~impossible to accomplish with individual user-side attestations
    - unbounded number of signing identities that need to be monitored for
  - attestations don't reduce trust in the index itself, meaning PyPI is still a juicy target
- **want *blanket* transparency and authenticity properties**
- **binary transparency for PyPI!**
  - prior art: Go's sumdb, npm's "publish attestations" (opposite of PyPI's)
  - desirable client-side properties: verification is 1+ ECC signatures + inclusion proof
  - downsides too: needs monitoring & witnessing ecosystem, evidence for these is scant

goal #3: index-level transparency for PyPI!

summary

the dream

today, **have misuse-resistant credentials and opportunistic, on-by-default publish attestations:**

- **thanks to Trusted Publishing + Sigstore based index-attestations (PEP 740)**

tomorrow, **we want:**

- **downstream verification of attestations**
  - possible today, but not within official package tooling (yet)
- **upstream (maintainer) *monitoring* of publishing identities**
  - possible today, but UX is not good: Sigstore needs the equivalent of Cert Spotter for CI
- **index-level transparency, with a healthy witnessing network**
  - needs lots of design thought! you can help!

the dream

the ultimate goal is to be able to express sentences like this:

✅ ✅

“when i download sampleproject v4.0.0,
i know that it came from the repository 🚧
it’s supposed to come from and that
❌ **PyPI served the *same file* to me as**
everyone else”

thank you!

- **these slides will soon be available here:**

https://yossarian.net/publications#sigstorecon-2024

resources:

- **PEP 740: https://peps.python.org/pep-0740/**
- **Trusted Publishing docs: https://docs.pypi.org/trusted-publishers/**
- **PyPI Attestation docs: https://docs.pypi.org/attestations/**

contact:

- **william@trailofbits.com**
- **@yossarian@infosec.exchange**