OpenSSF

OPEN SOURCE SECURITY FOUNDATION

We Make Python Safer Than Ever

Cheuk Ting Ho

and

Seth Michael Larson

Alpha-Omega

Alpha-Omega is an associated project of the OpenSSF, established in February 2022, with a mission to **protect society by improving the security of open source software** through direct maintainer engagement and expert analysis, trying to build a world where **critical open source projects are secure** and that security vulnerabilities are found and fixed quickly.

Alpha-Omega

Alpha will work with the maintainers of the most critical open source projects to help them identify and fix security vulnerabilities, and improve their security posture.

Omega will identify at least 10,000 widely deployed OSS projects where it can apply automated security analysis, scoring, and remediation guidance to their open source maintainer communities.

Engagements with projects:

- Node.js
- Eclipse Foundation
- Rust Foundation
- jQuery
- Python Software Foundation

Met our first 🥁🥁🥁🥁

Seth Michael Larson

Security Developer-in-Residence

Improve the security of
Python, **Python Packaging**
and more generally the **Python ecosystem** as a whole

- ➔ Challenges
- ➔ Accomplishments
- ➔ What’s next?
- ➔ What can you do?

Challenges securing Open Source

Challenges securing Open Source

👋 Many folks are volunteers

Time is limited, people come and go.

Challenges securing Open Source

👋 **Many folks are volunteers**

Time is limited, people come and go.

🔗 **Open Source is many things**

Decentralized, different sizes and types.
Changing behavior and mandates are difficult.

Challenges securing Open Source

👋 Many folks are volunteers

Time is limited, people come and go.

🔗 Open Source is many things

Decentralized, different sizes and types.
Changing behavior and mandates are difficult.

⛰️ Open Source is huge, “Long tail”

>400K Projects on PyPI

Challenges securing Python

Challenges securing Python

🩹 **Python is *the* glue language**

C, C++, ASM, Fortran, Rust, Go, WASM, JS...

Challenges securing Python

- 🩹 **Python is *the* glue language**

  C, C++, ASM, Fortran, Rust, Go, WASM, JS...

- 🍱 **Python packaging is a diverse ecosystem**

  PyPI, conda, distros, tools, standards

Challenges securing Python

- 🩹 **Python is *the* glue language**

  C, C++, ASM, Fortran, Rust, Go, WASM, JS...

- 🍱 **Python packaging is a diverse ecosystem**

  PyPI, conda, distros, tools, standards

- 🛰️ **Python user-base is also diverse**

  Scientists, Analysts, AI, Web, Space Helicopters...

That’s a lot of challenges…

😰

Sustainability, Clarity, & Visibility

What have we accomplished so far?

✍️ Signed Releases with Sigstore

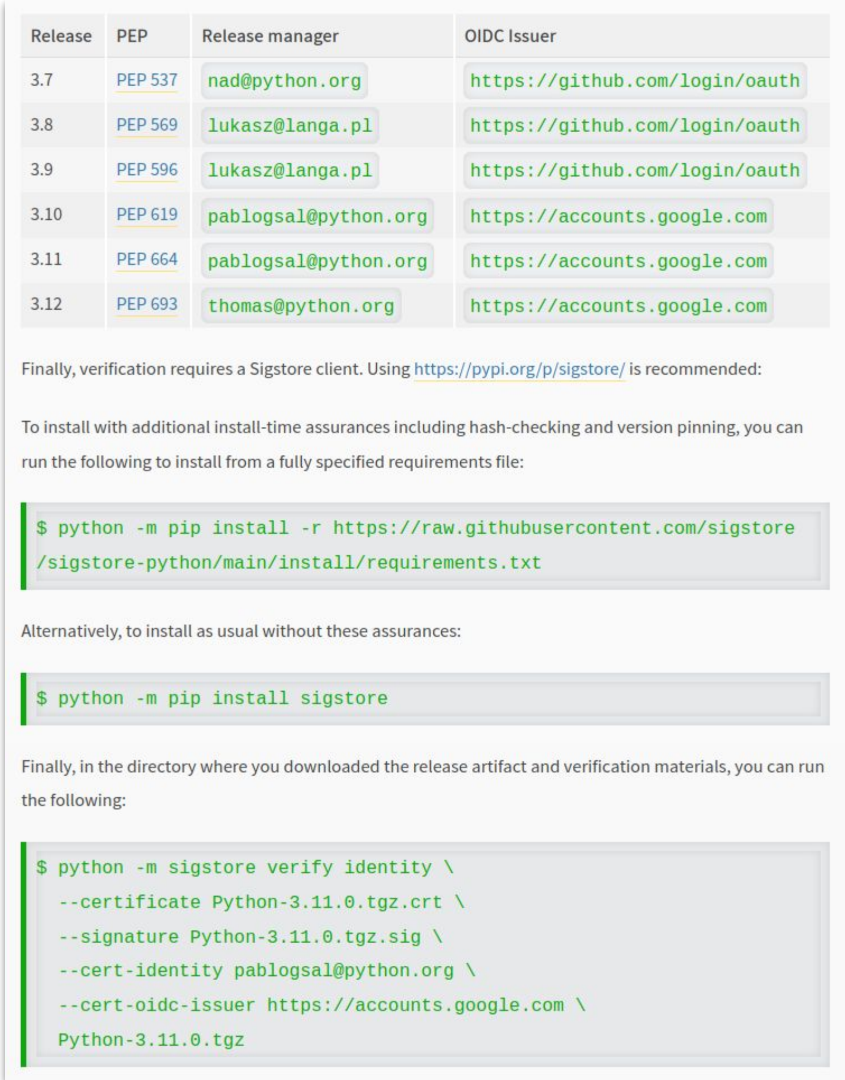
Q: How do you know if a Python release artifact is legitimate?

✍️ Signed Releases with Sigstore

Q: How do you know if a Python release artifact is legitimate?

A: Verify the signatures!

Information on Sigstore signatures:
https://python.org/download/sigstore



Release	PEP	Release manager	OIDC Issuer
3.7	PEP 537	`nad@python.org`	`https://github.com/login/oauth`
3.8	PEP 569	`lukasz@langa.pl`	`https://github.com/login/oauth`
3.9	PEP 596	`lukasz@langa.pl`	`https://github.com/login/oauth`
3.10	PEP 619	`pablogsal@python.org`	`https://accounts.google.com`
3.11	PEP 664	`pablogsal@python.org`	`https://accounts.google.com`
3.12	PEP 693	`thomas@python.org`	`https://accounts.google.com`

Finally, verification requires a Sigstore client. Using https://pypi.org/p/sigstore/ is recommended:

To install with additional install-time assurances including hash-checking and version pinning, you can run the following to install from a fully specified requirements file:

```
$ python -m pip install -r https://raw.githubusercontent.com/sigstore
/sigstore-python/main/install/requirements.txt
```

Alternatively, to install as usual without these assurances:

```
$ python -m pip install sigstore
```

Finally, in the directory where you downloaded the release artifact and verification materials, you can run the following:

```
$ python -m sigstore verify identity \
  --certificate Python-3.11.0.tgz.crt \
  --signature Python-3.11.0.tgz.sig \
  --cert-identity pablogsal@python.org \
  --cert-oidc-issuer https://accounts.google.com \
  Python-3.11.0.tgz
```

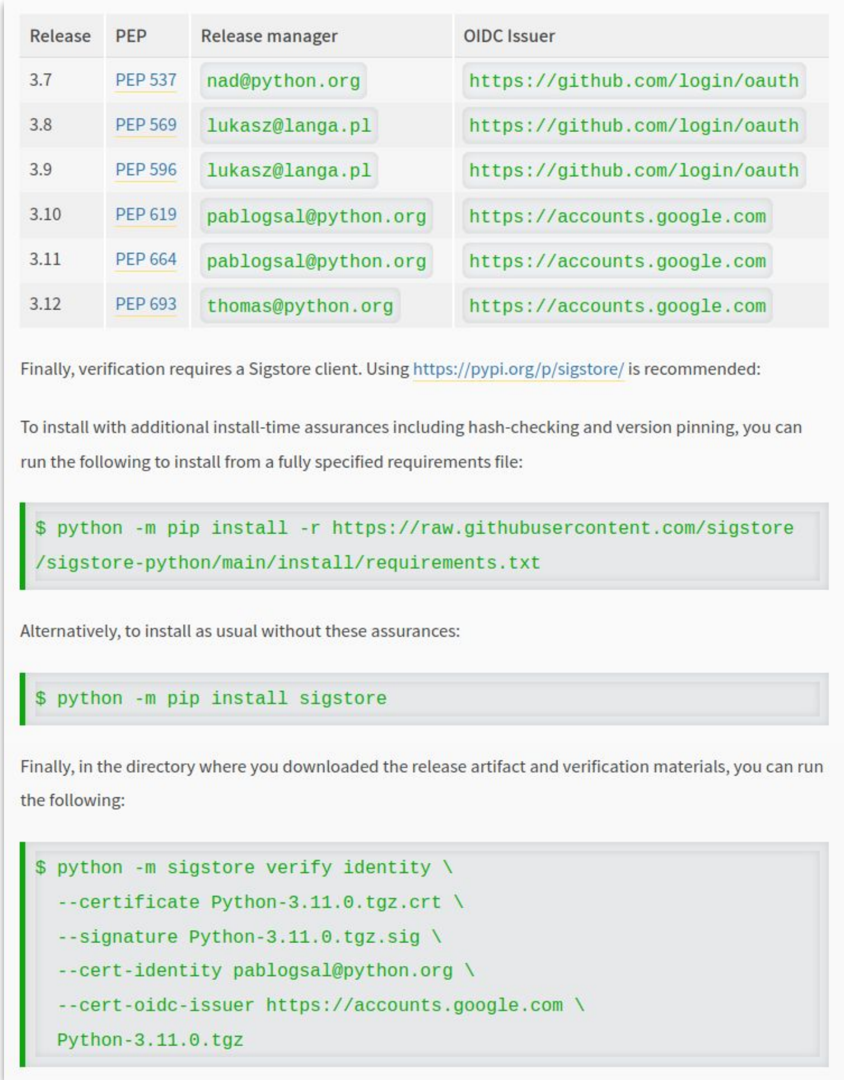
✍️ Signed Releases with Sigstore

Q: How do you know if a Python release artifact is legitimate?

A: Verify the signatures!

"You know the chef, not the ingredients"

Information on Sigstore signatures:
https://python.org/download/sigstore



Release	PEP	Release manager	OIDC Issuer
3.7	PEP 537	nad@python.org	https://github.com/login/oauth
3.8	PEP 569	lukasz@langa.pl	https://github.com/login/oauth
3.9	PEP 596	lukasz@langa.pl	https://github.com/login/oauth
3.10	PEP 619	pablogsal@python.org	https://accounts.google.com
3.11	PEP 664	pablogsal@python.org	https://accounts.google.com
3.12	PEP 693	thomas@python.org	https://accounts.google.com

Finally, verification requires a Sigstore client. Using https://pypi.org/p/sigstore/ is recommended:

To install with additional install-time assurances including hash-checking and version pinning, you can run the following to install from a fully specified requirements file:

```
$ python -m pip install -r https://raw.githubusercontent.com/sigstore
/sigstore-python/main/install/requirements.txt
```

Alternatively, to install as usual without these assurances:

```
$ python -m pip install sigstore
```

Finally, in the directory where you downloaded the release artifact and verification materials, you can run the following:

```
$ python -m sigstore verify identity \
  --certificate Python-3.11.0.tgz.crt \
  --signature Python-3.11.0.tgz.sig \
  --cert-identity pablogsal@python.org \
  --cert-oidc-issuer https://accounts.google.com \
  Python-3.11.0.tgz
```

🛡️ Python Security Response Team (PSRT)

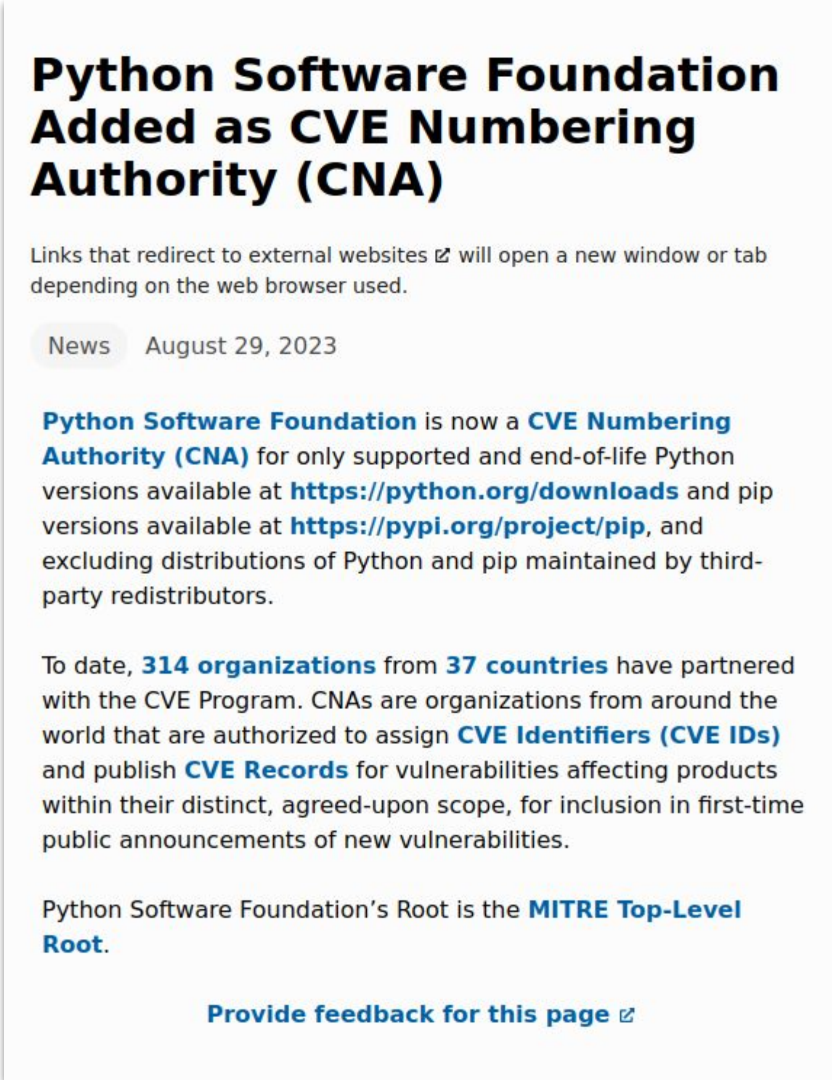
The fine folks behind security@python.org

Got vulns? 👉 https://python.org/dev/security

- Joined PSRT, coordinating, authoring advisories: security-announce@python.org
- Documented end-to-end handling of **CVE-2023-40217** from disclosure to releases. Now we improve the process! 💪
- Investigating adoption of a ticketing system for reports (GHSA?)

🎉 CVE Numbering Authority (CNA)

- **CVE IDs issued for Python and pip according to security policies.**
- Staffing investment supplied by Python Software Foundation! 🙇
- Guidance for other Open Source orgs and projects wanting to become and operate a CNA.



Python Software Foundation Added as CVE Numbering Authority (CNA)

Links that redirect to external websites will open a new window or tab depending on the web browser used.

News August 29, 2023

Python Software Foundation is now a **CVE Numbering Authority (CNA)** for only supported and end-of-life Python versions available at **https://python.org/downloads** and pip versions available at **https://pypi.org/project/pip**, and excluding distributions of Python and pip maintained by third-party redistributors.

To date, **314 organizations** from **37 countries** have partnered with the CVE Program. CNAs are organizations from around the world that are authorized to assign **CVE Identifiers (CVE IDs)** and publish **CVE Records** for vulnerabilities affecting products within their distinct, agreed-upon scope, for inclusion in first-time public announcements of new vulnerabilities.

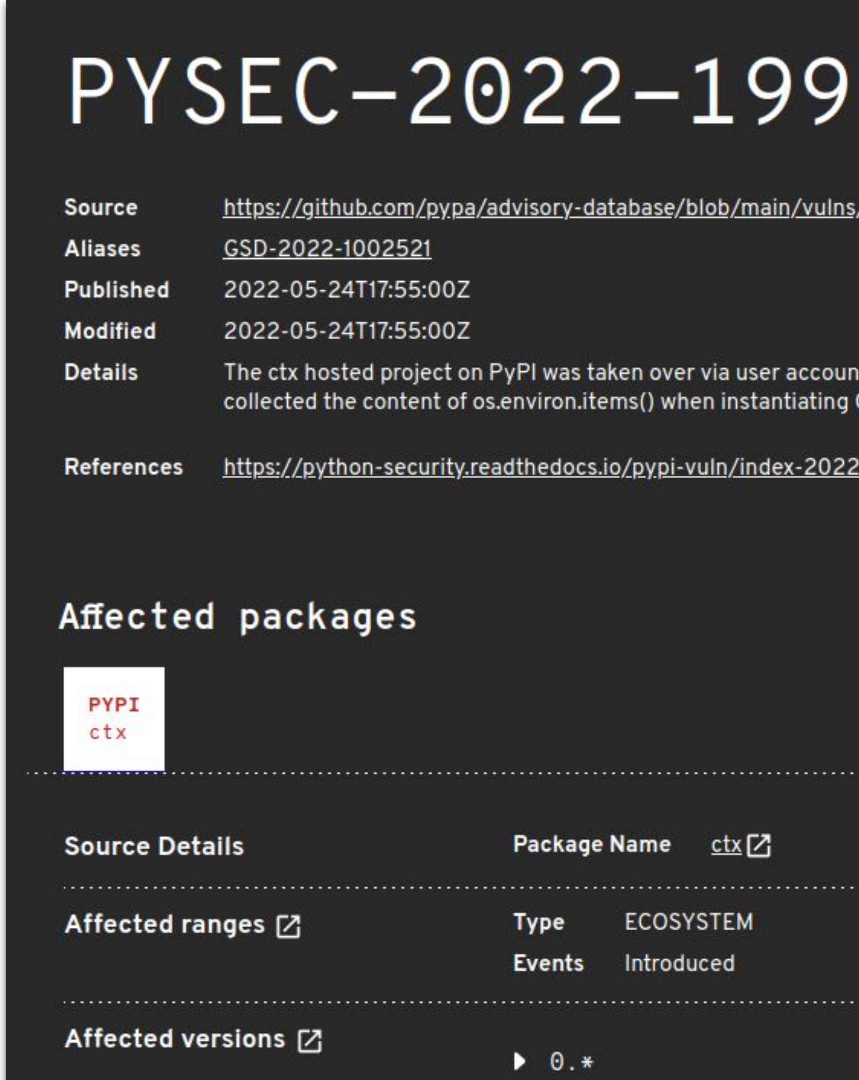
Python Software Foundation's Root is the **MITRE Top-Level Root**.

Provide feedback for this page

🪳 Open Source Vulnerability DBs

Advisories with ecosystem-specific names and version ranges.

- Back-filled historical advisories (thanks to **Victor Stinner**!)
- PSF Advisory Database for CPython from CVEs.
- PyPA Advisory Database and **pip-audit** for Python packages



PYSEC-2022-199

Source	https://github.com/pypa/advisory-database/blob/main/vulns
Aliases	GSD-2022-1002521
Published	2022-05-24T17:55:00Z
Modified	2022-05-24T17:55:00Z
Details	The ctx hosted project on PyPI was taken over via user accoun collected the content of os.environ.items() when instantiating
References	https://python-security.readthedocs.io/pypi-vuln/index-2022

Affected packages

PYPI ctx

Source Details — Package Name: ctx

Affected ranges — Type: ECOSYSTEM; Events: Introduced

Affected versions — 0.*

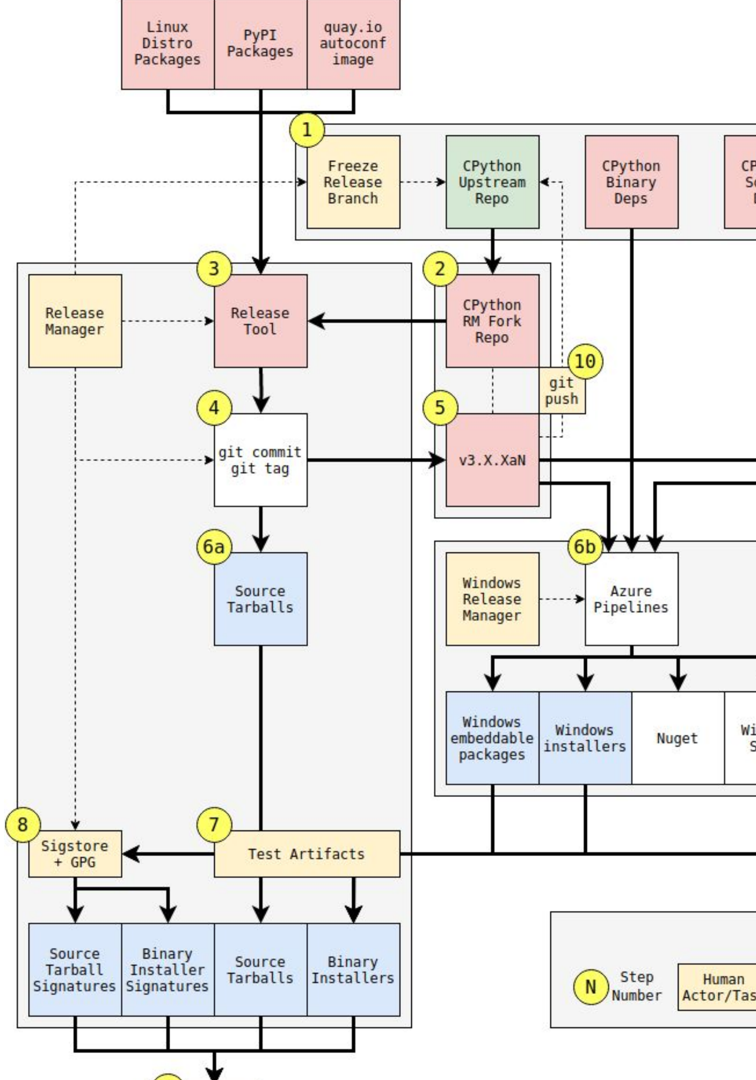
What’s on the horizon for Python? 🌅

⚙️ CPython and pip Release Processes

- Non-trivial release processes involving multiple people and projects.

- **Make recommendations to avoid known supply chain threats.**

- Improve reproducibility through automation (win-win!)



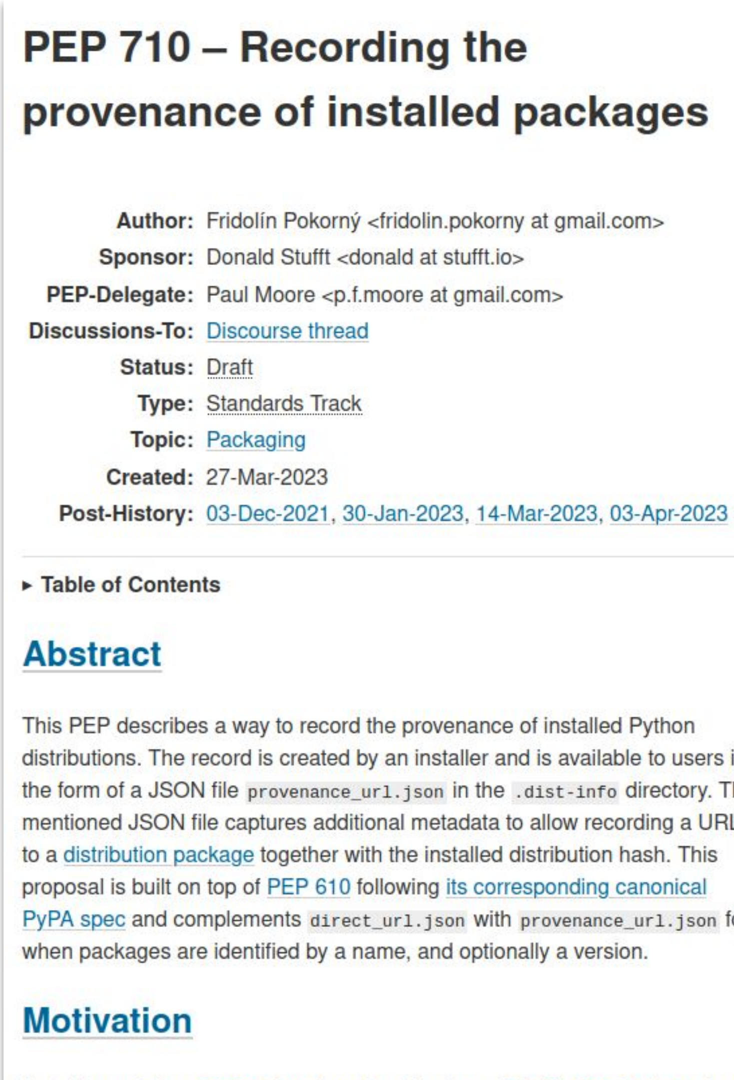
📰 Standards, Guidance, and Metrics

Standards (PEPs)

- PEP 710 (Package Provenance)
- PEP 639 (SPDX License Identifiers)
- **Metadata for Bundled Projects**

Guidance (OpenSSF)

- **Best Practices for Using and Developing Python Projects**
- Becoming a CNA as an Open Source Organization or Project



PEP 710 – Recording the provenance of installed packages

Author:	Fridolín Pokorný <fridolin.pokorny at gmail.com>
Sponsor:	Donald Stufft <donald at stufft.io>
PEP-Delegate:	Paul Moore <p.f.moore at gmail.com>
Discussions-To:	Discourse thread
Status:	Draft
Type:	Standards Track
Topic:	Packaging
Created:	27-Mar-2023
Post-History:	03-Dec-2021, 30-Jan-2023, 14-Mar-2023, 03-Apr-2023

▸ **Table of Contents**

Abstract

This PEP describes a way to record the provenance of installed Python distributions. The record is created by an installer and is available to users i the form of a JSON file `provenance_url.json` in the `.dist-info` directory. T mentioned JSON file captures additional metadata to allow recording a URL to a distribution package together with the installed distribution hash. This proposal is built on top of PEP 610 following its corresponding canonical PyPA spec and complements `direct_url.json` with `provenance_url.json` f when packages are identified by a name, and optionally a version.

Motivation

📝 Software Bill of Materials (SBOM)

- SBOMs are important to consumers for compliance and vuln management.

- The “soul” of SBOMs: **Visibility into the software you’re building and running.** This also happens to be the tough part.

- Plan to work on SBOMs for CPython, pip, and making them easier to create for Python packages.

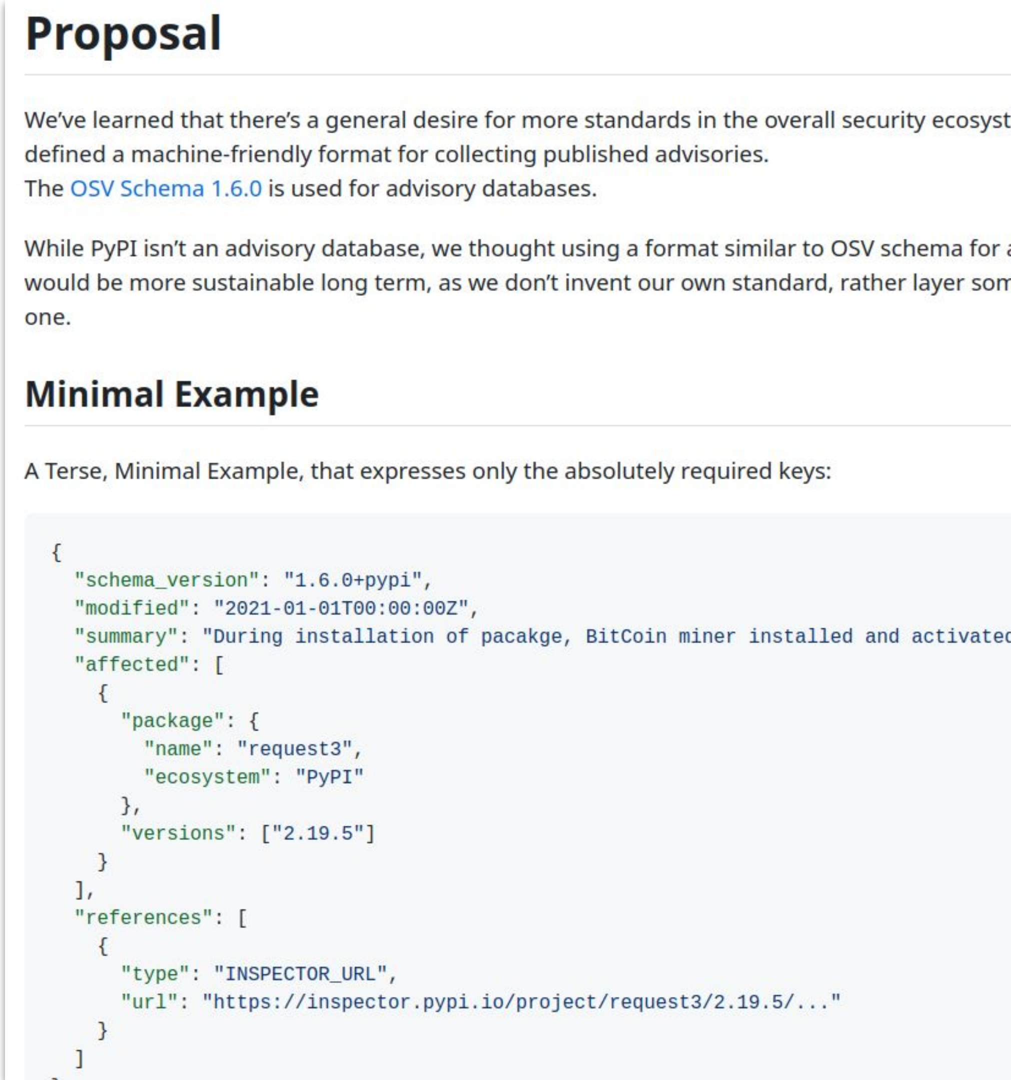
SPDX

CycloneDX

⚠ PyPI Malware Reporting API

Can we reduce the amount of malware on PyPI to effectively zero?

- Third-parties already report malware to PyPI via email.
- What if they trusted third parties could report via an API?
- What if PyPI could take action autonomously? 🤖



Proposal

We've learned that there's a general desire for more standards in the overall security ecosyst
defined a machine-friendly format for collecting published advisories.
The OSV Schema 1.6.0 is used for advisory databases.

While PyPI isn't an advisory database, we thought using a format similar to OSV schema for a
would be more sustainable long term, as we don't invent our own standard, rather layer som
one.

Minimal Example

A Terse, Minimal Example, that expresses only the absolutely required keys:

```
{
  "schema_version": "1.6.0+pypi",
  "modified": "2021-01-01T00:00:00Z",
  "summary": "During installation of pacakge, BitCoin miner installed and activatec
  "affected": [
    {
      "package": {
        "name": "request3",
        "ecosystem": "PyPI"
      },
      "versions": ["2.19.5"]
    }
  ],
  "references": [
    {
      "type": "INSPECTOR_URL",
      "url": "https://inspector.pypi.io/project/request3/2.19.5/..."
    }
  ]
```

📢 Why is it important?

Python

- Over 400,000 Python packages on Python Package Index (PyPI)
- Used by researchers: NASA, CERN and many universities and institutes
- Used by financial operations: Bloomberg, Capital One and many banks
- Used to handle data in many industries and journalists
- Many user’s first programming language
- Many users does not have an software engineering background

It is great to have a **broad adaptation** of Python in different industries. 💼

This make security in the Python ecosystem more important.

Thanks to Alpha-Omega and OpenSSF we have Seth to help us.

But the work **doesn’t stop there**. How can we amplify his work? 🤔

💡 What can you do today?

Maintainers of Python projects:

- **Enable 2FA everywhere** (email, PyPI, GitHub, GitLab, etc)
- Learn about secure development best practices (OpenSSF Guides!)
- Subscribe to the PyPI Blog for new security features

Users of Python projects:

- **Keep your dependencies locked and up-to-date.**
- Subscribe for advisories: security-announce@python.org
- Use **pip-audit** to audit your dependencies for known vulnerabilities.

🎓 Education to the community

- More security related tracks at conferences
- Security summit
- Promote adaptation of security practices
- Amplify security alert on social media

We have filled a new role last month
👏👏👏👏

Mike Fiedler

PyPI Safety & Security Engineer

PyPI Safety & Security Engineer

- Funded by Amazon Web Services (AWS)
- Focus on Python Package Index (PyPI)
- Increased support for package maintainers
- Reduced response time for malware reports
- Work closely with Seth

We have made Python safer than ever,

but we will keep making Python even safer