software stewardship lab

06 JUL 2026
VLAD-STEFAN HARBUZ

PROPOSAL

Securing Python's Binary Dependencies

Python is valued for its ability to easily interface with compiled libraries, but Python programs' dependencies on compiled libraries are not properly recorded, and are managed by Python tools instead of by the system package manager. This makes these dependencies difficult to identify, verify for integrity, and update, potentially causing catastrophic security issues.

I aim to enable Python users to easily identify their (transitive) binary dependencies to verify that these files came from a trusted source; and to keep these dependencies up to date with minimal effort. I plan to do this by creating new tools and making targeted improvements to key pieces of the Python ecosystem such as uv/pip.

My roadmap is as follows. I aim to complete points 1–3 as part of this grant if awarded, leaving 4–6 for future grant applications.

1. **Describe the issue thoroughly.** Write a paper describing the aforementioned security issues in detail, and precisely identifying which combinations of improvements to the Python ecosystem would achieve the above goals. For my current work on this problem, see my FOSDEM 2026 talk.
2. **Identify.** Create a tool that allows users to list a project's binary dependencies. auditwheel is already able to identify a project's required dynamic libraries, but is not able to output this information in human-readable format, and also does not have a programmatic API. I aim to implement such an API, which would allow downstream users and researchers to understand existing binary dependency data. Hopefully this would get integrated into auditwheel long-term — see my issue 676.
3. **Prototype.** Create a rough PoC tool that rewrites the vendored binary dependencies of installed packages, making them point to versions provided by the system package manager. I hope that this step will prove the viability of a future where Python binary dependencies are managed by the system package manager.
4. **Record.** Create tooling that automatically records binary dependencies in pyproject.toml according to PEP 725. This may happen via improvements to uv/pip, or as a standalone tool. These records would allow registries and researchers to analyse

binary dependency relationships and their security status, and to enrich SBOMs and store them as specified in PEP 770.

5. **Connect the dots on vulnerabilities.** Improve uv/pip to enable them to warn users when their binary dependencies are out of date or present known vulnerabilities.
6. **Defer to system package managers.** Build on the previous steps to ensure binary dependencies are managed not by language-specific package managers, but by system package managers. I aim to develop interoperability between Python tooling and system package managers to make this possible.

I will consult Seth Larson and Andrew Nesbitt to take advantage of their expertise.

Additional Reading

- *How Binary Dependencies Work Across Different Languages* by me
- *The C-Shaped Hole in Package Management* by Andrew
- A description of phantom dependencies by Anand Sawant
- PEP 770
- PEP 725
- PEP 804
- elfdeps
- More resources available here

Estimate for points 1–3

Duration	Cost
6 weeks	50,000 USD