TRAIL OF BITS

Python packaging mystery meat

William Woodruff
Trail of Bits

hello

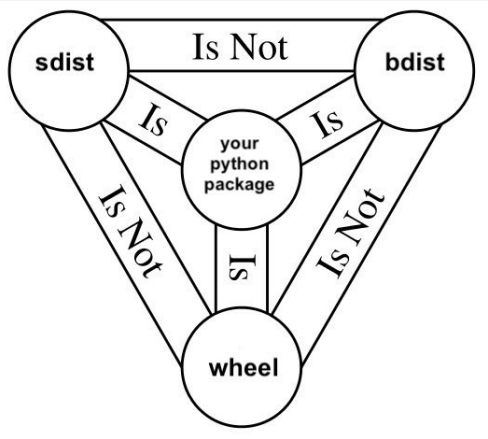
- **william woodruff**
  - senior security engineer @ Trail of Bits
  - Homebrew, pip-audit, sigstore-python maintainer, contributor to many things
  - @yossarian@yossarian.net (Mastodon)
  - @8x5clPW2 (blue bird site)
- **Trail of Bits**
  - ~~small~~ mid-sized security consultancy (~130 people)
  - NYC based, but >80% remote
  - areas:
    - foundational program analysis and cryptography research
    - applied cryptography, supply-chain, general security engineering
    - client security audits
  - we're hiring!

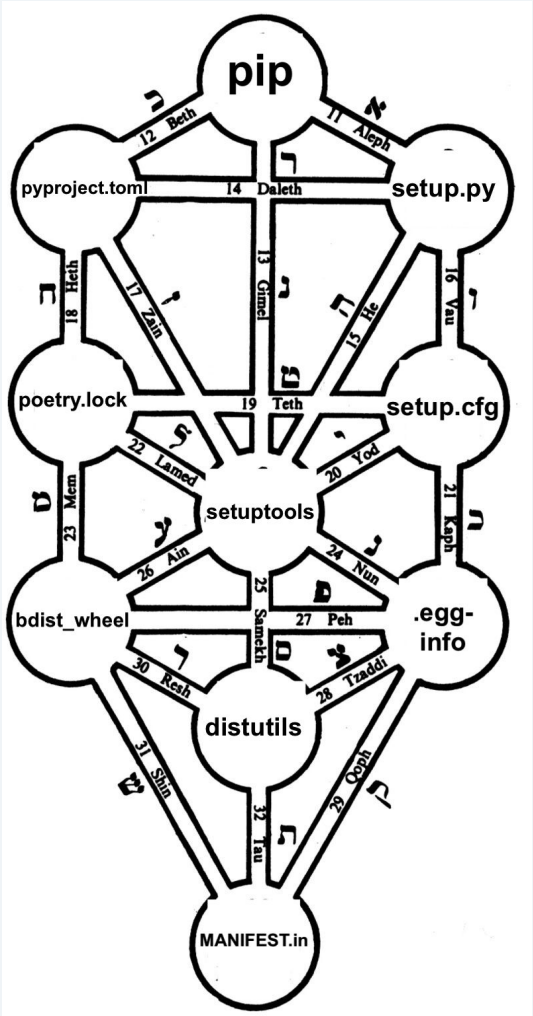
what engineers think a python package is

```
my-package/
    setup.py
    my_package/
        __init__.py
        some_mod.py
        another/
            __init__.py
            junk.py
```

```python
>>> from my_package.another import junk
```

what a python package actually is





a python package is “just” a name

this name has nothing to do with the module name, and has different rules

```
$ python -m pip install Pillow
Collecting Pillow
  Downloading
Pillow-9.3.0-cp310-cp310-macosx_10_
10_x86_64.whl (3.3 MB)
Installing collected packages:
Pillow
Successfully installed Pillow-9.3.0

$ python
>>> import PIL
>>> PIL.__version__
‘9.3.0’
```

a python package is composed of **versions**, and versions are composed of **distributions**

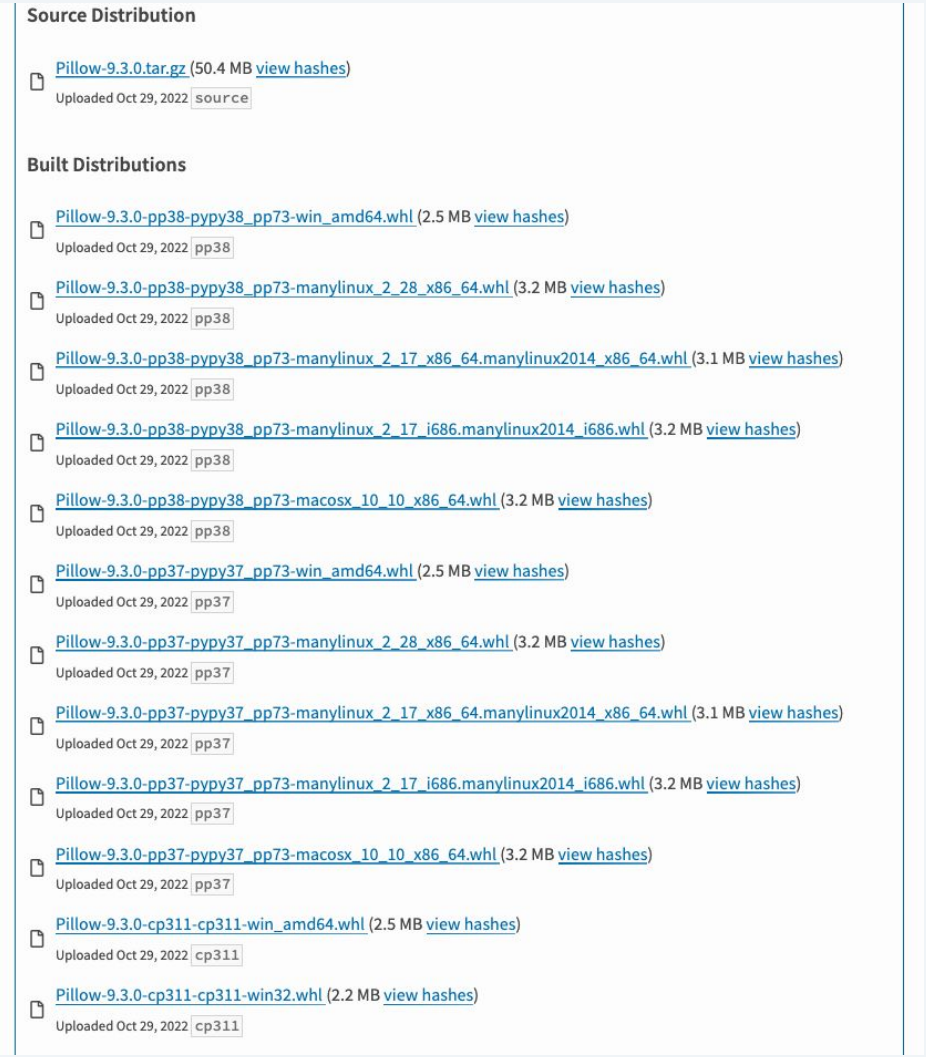
```
$ python -m pip install Pillow
Collecting Pillow
  Downloading
Pillow-9.3.0-cp310-cp310-macosx_10_
10_x86_64.whl (3.3 MB)
Installing collected packages:
Pillow
Successfully installed Pillow-9.3.0

$ python
>>> import PIL
>>> PIL.__version__
‘9.3.0’
```

PyPI supports two distribution formats: sdists and wheels

sdist: source distributions (one per version)

wheels: “built” distributions (many per version)

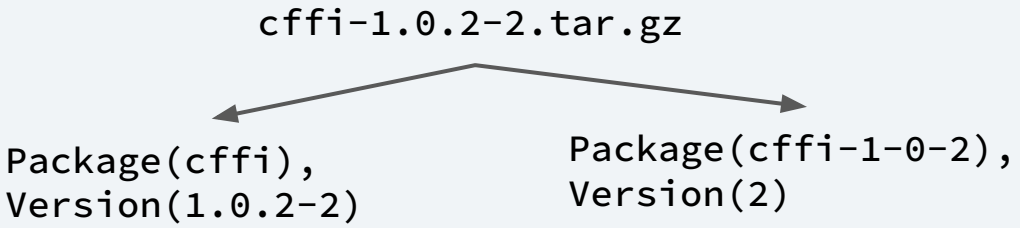


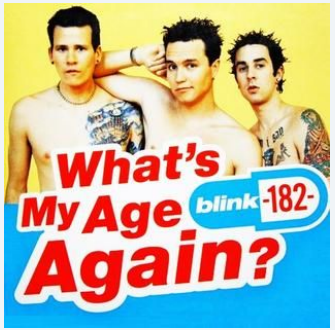
source distributions

the “fuck around and find out” approach to packaging

Python doesn't know what your distribution's name is

- **Until (very) recently, sdist filenames were only loosely standardized as `{project}-{version}.tar.gz`**
- **`version` is a PEP 440 version, meaning it can contain dashes**
- **…but `project` can also include dashes, per PEP 345 and forwards!**
- **Result: there are packages on PyPI whose name/version can't be unambiguously parsed from just the sdist index entry!**
  - **pip et al. hack around this by keeping additional parse context**
- **Fixed in PEP 625, which was accepted ~2 months ago**
  - **…but hacks will stay in place because of old distributions**





sdists are “just” a tarball of the Python source plus a build/install script

that script is usually `setup.py`

pictured: a reasonable looking `setup.py`

```python
from setuptools import setup, find_packages

with open("README.md") as f:
    long_description = f.read()

setup(
    author="William Woodruff",
    author_email="william@trailofbits.com",
    classifiers=[
        "License :: OSI Approved :: Apache Software License",
        "Programming Language :: Python :: 3 :: Only",
    ],
    description="A short description",
    install_requires=["some-dep>=1.2.3"],
    long_description=long_description,
    long_description_content_type="text/markdown",
    name="example",
    packages=find_packages(),
    package_data={
        "example": ["py.typed"],
    },
    python_requires=">=3.7",
    url="http://github.com/example/example",
    version="0.0.30",
)
```


setup.py can do anything it wants!

including hooking subcommands!

```python
from setuptools import setup
from setuptools.command.install import install

class CustomInstall(install):
  def run(self):
    install.run(self)
    subprocess.run(["pip", "uninstall", "example"])

setup(
    author="William Woodruff",
    ...,
    cmdclass={
        "install": CustomInstall,
    }
    name="example",
)
```

Experts found 11 malicious Python packages in the PyPI repository

November 21, 2021 By Pierluigi Paganini

Multiple malicious packages in PyPI repository found stealing AWS secrets

June 25, 2022 By Pierluigi Paganini

Experts found 10 malicious packages on PyPI used to steal developers' data

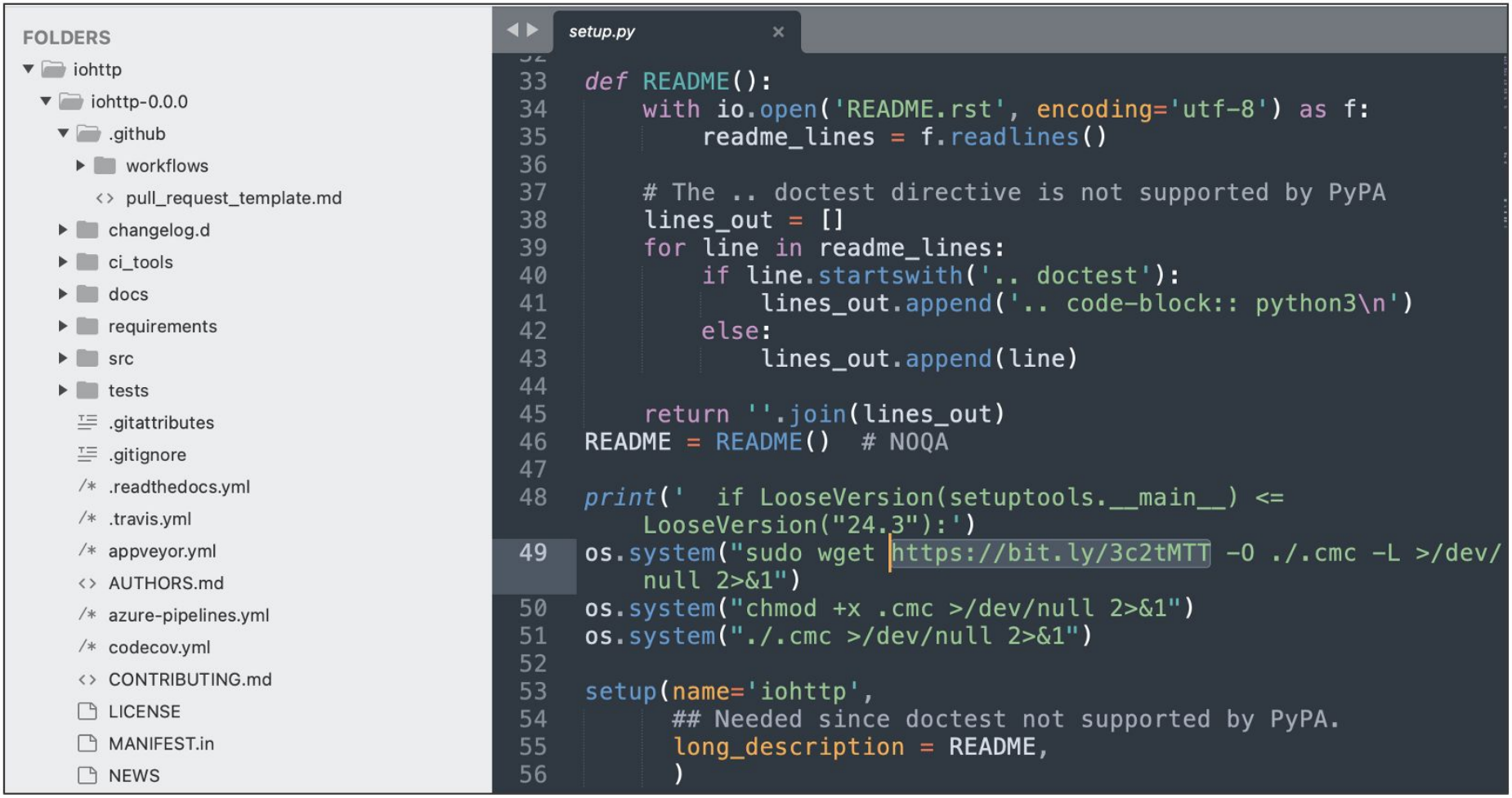
August 10, 2022 By Pierluigi Paganini

More Than 200 Cryptomining Packages Flood npm and PyPI Registry

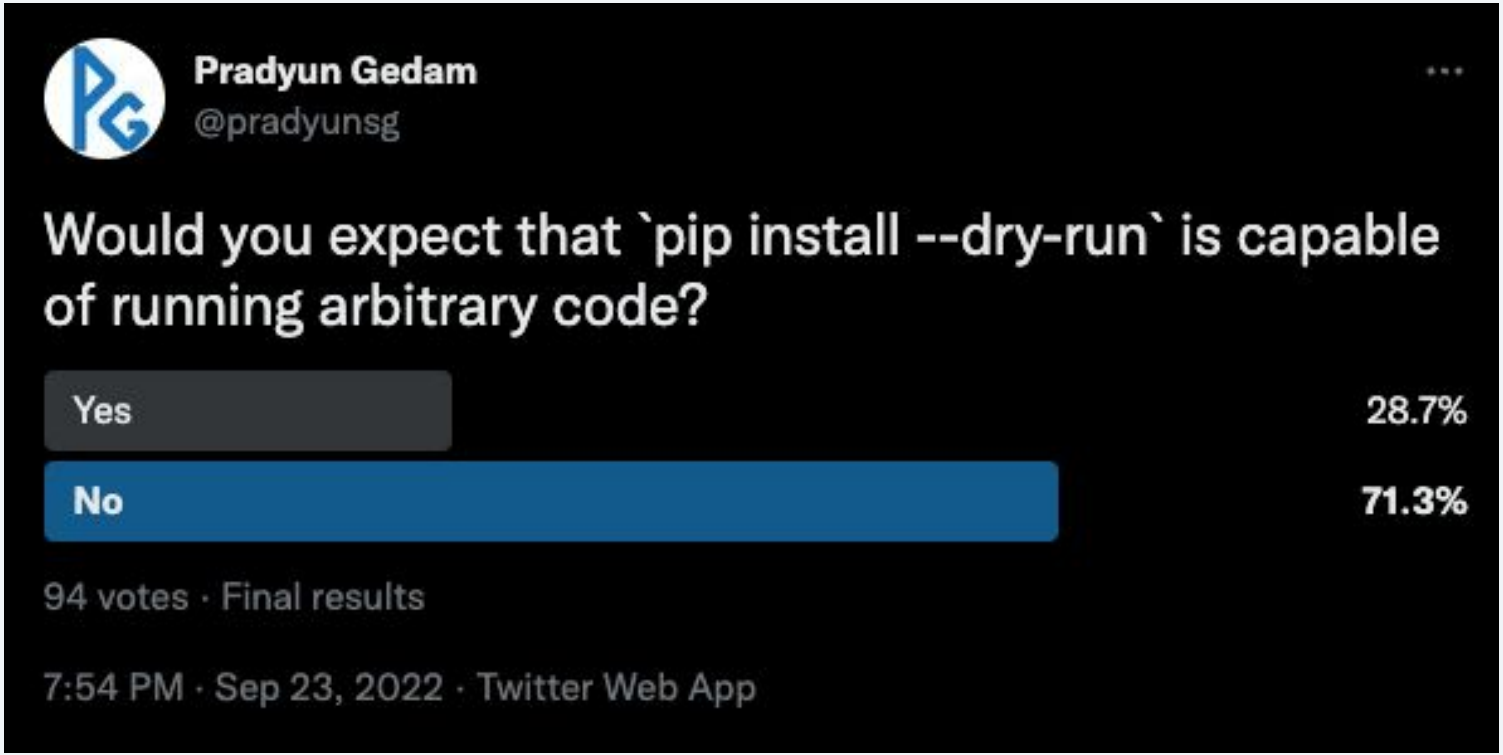
August 19, 2022 By **Ax Sharma**

6 minute read time

From Sonatype:



`pip install --dry-run` — ACE or not?



`pip download` — ACE or not?

Uses `setup.py egg_info` to generate package metadata!

```
    args = make_setuptools_egg_info_args(
        setup_py_path,
        egg_info_dir=egg_info_dir,
        no_user_config=isolated,
    )

    with build_env:
        with open_spinner("Preparing metadata (setup.py)") as spinner:
            try:
                call_subprocess(
                    args,
                    cwd=source_dir,
                    command_desc="python setup.py egg_info",
                    spinner=spinner,
                )
            except InstallationSubprocessError as error:
                raise MetadataGenerationFailed(package_details=details) fro

    # Return the .egg-info directory.
    return _find_egg_info(egg_info_dir)
```

```
from setuptools import setup
from setuptools.command.egg_info import egg_info

class EvilEggInfo(install):
  def run(self):
    egg_info.run(self)
    print("im in ur metadataz")

setup(
    author="William Woodruff",
    ...,
    cmdclass={
        "egg_info": EvilEggInfo,
    }
    name="example",
)
```

$ pip download --no-deps issue7325

```
Collecting issue7325
  Downloading issue7325-0.1.tar.gz (773 bytes)
    ERROR: Command errored out with exit status 1:
     command: /private/tmp/env/bin/python -c 'import io, os, sys, setuptools, token
= '"'"'/private/var/folders/zj/hy934vnj5xs68zv6w4b_f6s40000gn/T/pip-download-xqh9hg
fb8aba0b4a70b45d490b60a6607c/setup.py'"'"'; __file__='"'"'/private/var/folders/zj/h
b_f6s40000gn/T/pip-download-xqh9hgvc/issue7325_8896fb8aba0b4a70b45d490b60a6607c/set
tattr(tokenize, '"'"'open'"'"', open)(__file__) if os.path.exists(__file__) else io
om setuptools import setup; setup()'"'"');code = f.read().replace('"'"'\r\n'"'"', '
ose();exec(compile(code, __file__, '"'"'exec'"'"'))' egg_info --egg-base /private/v
34vnj5xs68zv6w4b_f6s40000gn/T/pip-pip-egg-info-4wv6vong
         cwd: /private/var/folders/zj/hy934vnj5xs68zv6w4b_f6s40000gn/T/pip-download-
25_8896fb8aba0b4a70b45d490b60a6607c/
    Complete output (8 lines):

    WARNING: use of "pip download --no-deps" allowed arbitrary code execution
             see https://github.com/pypa/pip/issues/7325

    WARNING: use of "pip download --no-deps" allowed arbitrary code execution
             see https://github.com/pypa/pip/issues/7325
    setup.py was executed :(
    setup.py is executing
    ----------------------------------------
WARNING: Discarding https://files.pythonhosted.org/packages/c0/51/bd28cda650e3f0123
a90ec8b2af89a9029e25768647/issue7325-0.1.tar.gz#sha256=ff4bca2ab7301cf36b6e6880c51f
46e83cd8228db955e18 (from https://pypi.org/simple/issue7325/). Command errored out
1: python setup.py egg_info Check the logs for full command output.
ERROR: Could not find a version that satisfies the requirement issue7325 (from vers
ERROR: No matching distribution found for issue7325
WARNING: You are using pip version 21.2.3; however, version 22.3.1 is available.
You should consider upgrading via the '/private/tmp/env/bin/python -m pip install --upgrade pip' com
mand.
```

built distributions

or: i think i can build it better than you can

recap: wheels are “built” distributions

“built” doesn’t actually mean “binary code”

it means structured metadata and layout, rather than a YOLO’d tarball

Inspector

pip-audit

pip-audit==2.4.6

pip_audit-2.4.6-py3-none-any.whl

- ./pip_audit/__init__.py
- ./pip_audit/__main__.py
- ./pip_audit/_audit.py
- ./pip_audit/_cache.py
- ./pip_audit/_cli.py
- ./pip_audit/_fix.py
- ./pip_audit/_state.py
- ./pip_audit/_subprocess.py
- ./pip_audit/_util.py
- ./pip_audit/_virtual_env.py
- ./pip_audit/_dependency_source/__init__.py
- ./pip_audit/_dependency_source/interface.py
- ./pip_audit/_dependency_source/pip.py
- ./pip_audit/_dependency_source/pyproject.py
- ./pip_audit/_dependency_source/requirement.py
- ./pip_audit/_dependency_source/resolvelib/__init__.py
- ./pip_audit/_dependency_source/resolvelib/pypi_provider.py
- ./pip_audit/_dependency_source/resolvelib/resolvelib.py
- ./pip_audit/_format/__init__.py
- ./pip_audit/_format/columns.py
- ./pip_audit/_format/cyclonedx.py
- ./pip_audit/_format/interface.py
- ./pip_audit/_format/json.py
- ./pip_audit/_format/markdown.py
- ./pip_audit/_service/__init__.py
- ./pip_audit/_service/interface.py
- ./pip_audit/_service/osv.py
- ./pip_audit/_service/pypi.py
- ./pip_audit-2.4.6.dist-info/entry_points.txt
- ./pip_audit-2.4.6.dist-info/LICENSE
- ./pip_audit-2.4.6.dist-info/WHEEL
- ./pip_audit-2.4.6.dist-info/METADATA
- ./pip_audit-2.4.6.dist-info/RECORD

Source: https://github.com/pypi/inspector

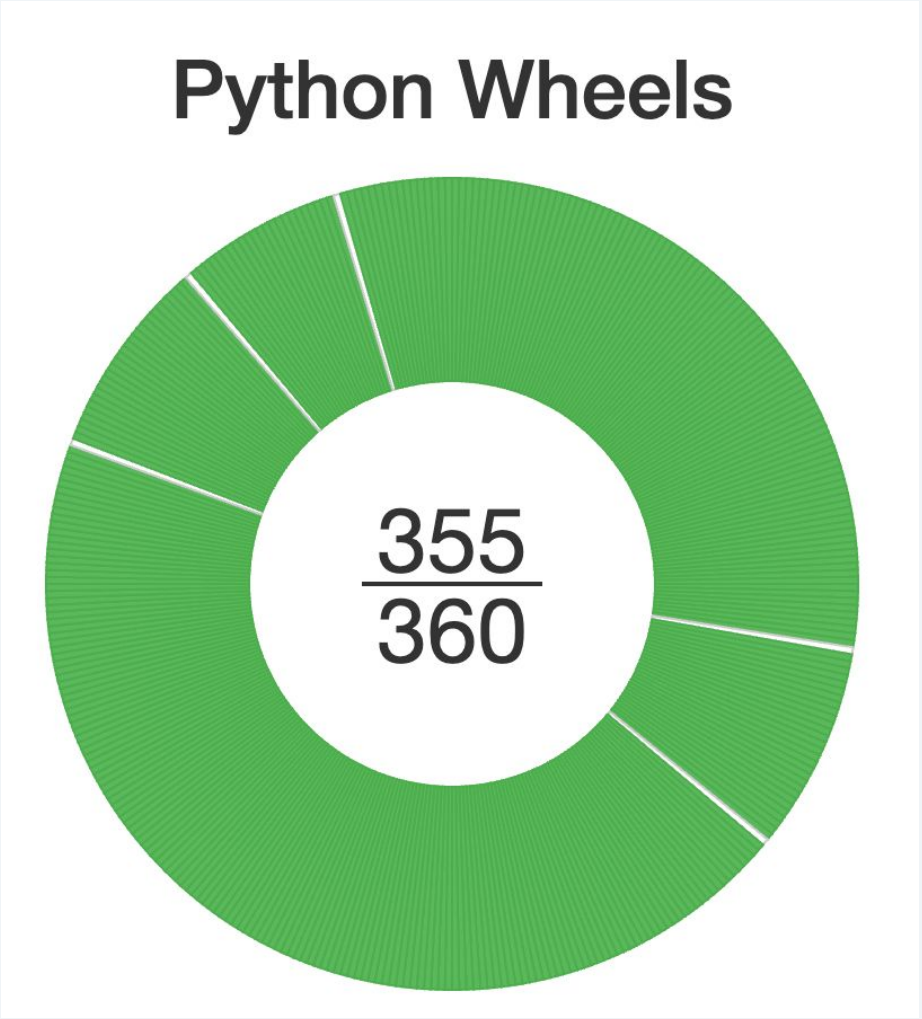
pip_audit-2.4.6.dist-info/WHEEL

```
1 Wheel-Version: 1.0
2 Generator: flit 3.8.0
3 Root-Is-Purelib: true
4 Tag: py3-none-any
```

wheels can be pure Python!

wheels are the **correct** way to distribute pure Python packages in 2022 — no arbitrary code exec via `setup.py`!

tooling for wheels is mature; python -m build should just work



…but they can also contain binaries

loadable shared objects for python extensions + loadable shared objects for vendored native dependencies

tools like auditwheel assist with this by automatically relocating deps into the bdist

`import foo` searches a bunch of paths, including `foo.so` and `foo.abi3.so`

- ./Pillow.libs/libwebpmux-d318e4b4.so.3.0.10
- ./Pillow.libs/libopenjp2-655fe86a.so.2.5.0
- ./Pillow.libs/libjpeg-83a26697.so.62.3.0
- ./Pillow.libs/libharfbuzz-2034b74b.so.0.50301.0
- ./Pillow.libs/liblcms2-ff0f7acb.so.2.0.13
- ./Pillow.libs/liblzma-26c6a2c3.so.5.2.7
- ./Pillow.libs/libpng16-4e1696c6.so.16.38.0
- ./Pillow.libs/libwebpdemux-9079a8bd.so.2.0.11
- ./Pillow.libs/libwebp-944f9a6c.so.7.1.5
- ./Pillow.libs/libxcb-62f08786.so.1.1.0
- ./Pillow.libs/libfreetype-469dac36.so.6.18.3
- ./Pillow.libs/libXau-21870672.so.6.0.0
- ./Pillow.libs/libtiff-15db6496.so.5.8.0

Alex (thankful for people addressing systemic issues) Gaynor

 11:31 AM

If I had to profile our "uses sdist users" I'd guess a) ding dongs on weird ass platforms, b) people with actual security programs that realize auditing binary wheels is basically impossible, so you probably shouldn't trust them.

(It's *wild* me that anyone would ever use a binary wheel in production.)

binary wheels: good and bad

good: saves end users from running complex source builds/building things wrong/toolchain support

bad: massive build/support matrices: 5 python versions (incl PyPy) × 3 host OSes × 2-3 libc builds per OS; hard to test!

tools like cibuildwheel assist with this by automating wheel builds across large matrices

CPython also tries to help by providing a stable ABI ("abi3"): wheels built for `abi3-py36` are CPython 3.6+ compatible

the not-so-stable ABI

to use the stable ABI, you define `Py_LIMITED_API` to the version you'd like to use…

…which is completely disconnected from the CLI option used to tag a wheel as stable-ABI-compatible!

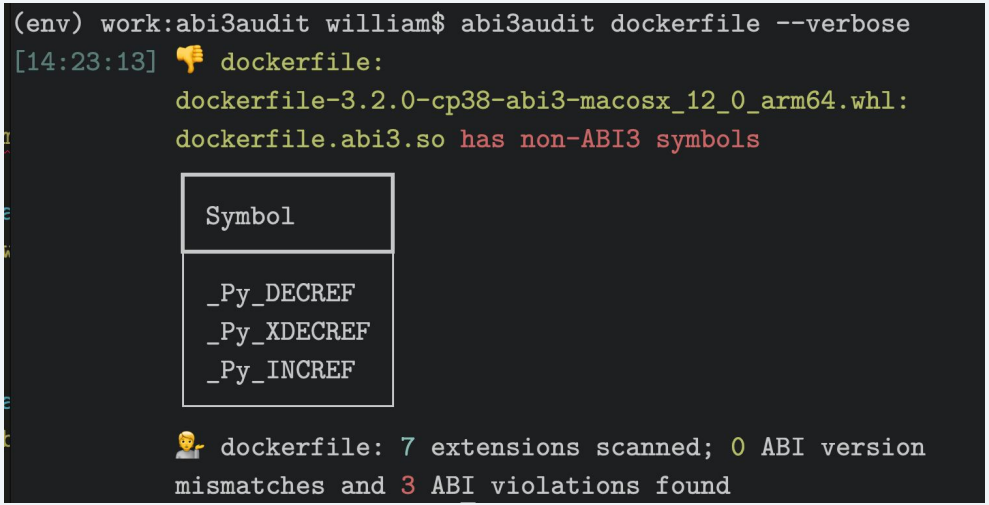
```
15  //#define Py_LIMITED_API 0x03020000
16  #include <Python.h>
```

```
return Extension('refl1d.reflmodule',
                 sources=sources,
                 depends=depends,
                 language="c++",
                 py_limited_api="cp32",
                 )
```

the not-so-stable ABI

we made a tool (abi3audit) to look for these incorrectly tagged wheels, and found that 15% of all PyPI projects have at least one mis-tagged wheel

ABI mismatches = potential crashes and memory corruption!



looking forwards

a few big trends happening on the horizon:

- ✅ **it's becoming harder to misconfigure packages, and easier to make and distribute wheels**
  - PEP 517 (`pyproject.toml`!), pypa/build, pypa/flit
- ✅ **it's easier to audit your dependencies for vulnerabilities**
  - pypa/pip-audit
- 🚧 **it's becoming easier to codesign for Python packages without PGP**
  - Sigstore
- 🚧 **it's becoming easier to publish to PyPI without managing credentials**
  - **OIDC federation with GitHub Actions and other CI providers**

talk to me about these after this talk!

addenda: cursed things to look up on your own

- **the URL requirement format, including `#egg` fragments**
  - `foo[bar] @ git+https://example.com/baz#egg=quux[lol]`
- **no specified limit on version length (see below)**
- **some wheels are built with `-ffast-math`, breaking FP in remote code**
  - **Brendan Dolan-Gavitt: someone's been messing with my subnormals!**
- **there is no guaranteed fixed point for dependency resolution**
  - **thanks to setup.py of course**
  - **Dustin Ingram: why PyPI doesn't know your project's dependencies**

thank you!