Path traversal in tar extraction on Python 3.10.0 - 3.10.12 and 3.11.0 - 3.11.4

Low **radoering** published **GHSA-73h3-mf4w-8647** last week

Package

poetry (pip)

Affected versions	Patched versions
<= 2.3.3	2.3.4

Description

Summary

The `extractall()` function in `src/poetry/utils/helpers.py:410-426` extracts sdist tarballs without path traversal protection on Python versions where `tarfile.data_filter` is unavailable. Considering only Python versions which are still supported by Poetry, these are 3.10.0 - 3.10.12 and 3.11.0 - 3.11.4.

Impact

Arbitrary file write (path traversal) from untrusted sdist content.

In practice, the impact is low because an attacker who exploits this vulnerability can as well include arbitrary code in a `setup.py`, which will be executed when the sdist is built after tar extraction. In other words, a malicious sdist can write arbitrary files by design. However, since it is unexpected and not by design that the file write already happens during tar extraction, this is still considered a vulnerability.

On Python 3.11.2 (Debian Bookworm default, directly tested), a crafted sdist with `../../` tar member paths writes files outside the intended extraction directory. The traversal occurs during metadata resolution (`poetry add --lock`), before the build backend is run.

Affected Environments:

- **Python 3.10.0 through 3.10.12** (inclusive): `tarfile.data_filter` absent or broken

- **Python 3.11.0 through 3.11.4** (inclusive): `tarfile.data_filter` absent or broken
- **Debian Bookworm**: Python 3.11.2 (default)
- **Ubuntu 22.04 LTS**: Python 3.10.6 (default)

Patches

Versions 2.3.4 and newer of Poetry ensure that paths are inside the target directory.

Root Cause

File: `src/poetry/utils/helpers.py`, lines 410-426:

```
def extractall(source: Path, dest: Path, zip: bool) -> None:
    """Extract all members from either a zip or tar archive."""
    if zip:
        with zipfile.ZipFile(source) as archive:
            archive.extractall(dest)
    else:
        broken_tarfile_filter = {(3, 9, 17), (3, 10, 12), (3, 11, 4)}
        with tarfile.open(source) as archive:
            if (
                hasattr(tarfile, "data_filter")
                and sys.version_info[:3] not in broken_tarfile_filter
            ):
                archive.extractall(dest, filter="data")
            else:
                archive.extractall(dest)  # <-- NO FILTER: path traversal
```

On Python versions without a working `tarfile.data_filter`, the `else` branch at line 426 calls `tarfile.extractall()` without any filter or path validation. This enables three attack vectors:

1. **Direct path traversal**: Tar members with `../../` path components write files outside the extraction directory.
2. **Symlink traversal**: A symlink member pointing outside dest, followed by a file written through that symlink, escapes the boundary.
3. **Hardlink attacks**: Hardlink members can read arbitrary files (same inode) or overwrite targets outside dest.

Call Sites

This function is called from two locations:

1. **`src/poetry/installation/chef.py:104`** (`_prepare_sdist`): During `poetry install` / `poetry add` when building a package from sdist. Only triggered when the executor is enabled (actual installation).

2. **`src/poetry/inspection/info.py:322`** (`_from_sdist_file`): During dependency resolution (`poetry lock` / `poetry add`). This path is reached when the sdist's `PKG-INFO` lacks `Requires-Dist` metadata, forcing Poetry to extract the archive (and afterwards build the package).

Suggested Fix

Apply path validation in the `else` branch, covering direct traversal, symlinks, and hardlinks:

```
def extractall(source: Path, dest: Path, zip: bool) -> None:
    """Extract all members from either a zip or tar archive."""
    if zip:
        with zipfile.ZipFile(source) as archive:
            archive.extractall(dest)
    else:
        broken_tarfile_filter = {(3, 9, 17), (3, 10, 12), (3, 11, 4)}
        with tarfile.open(source) as archive:
            if (
                hasattr(tarfile, "data_filter")
                and sys.version_info[:3] not in broken_tarfile_filter
            ):
                archive.extractall(dest, filter="data")
            else:
                # Validate all member paths before extraction
                dest_resolved = dest.resolve()
                safe_members = []
                for member in archive.getmembers():
                    member_path = (dest_resolved / member.name).resolve()
                    if not member_path.is_relative_to(dest_resolved):
                        raise ValueError(
                            f"Refusing to extract {member.name}: "
                            f"would write outside {dest}"
                        )
                    if member.issym():
                        link_target = (member_path.parent / member.linkname).resolve()
                        if not link_target.is_relative_to(dest_resolved):
                            raise ValueError(
                                f"Refusing symlink {member.name}: "
                                f"target {member.linkname} outside {dest}"
                            )
                    elif member.islnk():
                        link_target = (dest_resolved / member.linkname).resolve()
                        if not link_target.is_relative_to(dest_resolved):
                            raise ValueError(
                                f"Refusing hardlink {member.name}: "
                                f"target {member.linkname} outside {dest}"
                            )
                    safe_members.append(member)
                archive.extractall(dest, members=safe_members)
```

Severity

Low

CVE ID

CVE-2026-41140

Weaknesses

▶ CWE-22

Credits

kodareef5	Reporter
radoering	Remediation reviewer