

# spip: A State-of-the-Art Version-Controlled Pip Replacement

Antigravity

January 2026

## 1 Introduction

Current Python package management using `pip` and `venv` suffers from environment drift, lack of versioning, and hidden security risks. `spip` introduces a pure implementation that provides deep dependency analysis, automatic repair/verification, deterministic housekeeping, and real-time security auditing.

## 2 The spip Architecture

`spip` addresses these via a novel architecture combining:

- **Centralized Vault:** All environments are stored in `~/.spip/envs/`.
- **Git-as-Database:** A central Git repository tracks all environment states.
- **Real-Time Security Auditing:** The `spip audit` command performs a batch query of all installed libraries against the \*\*OSV (Open Source Vulnerability) database\*\*. It identifies active CVEs, providing summaries and CVSS severity scores to ensure the environment remains secure against known exploits.
- **Orphan Pruning:** Recursive graph traversal to identify and remove unused sub-dependencies.
- **Environment Portability (Freeze):** Bundles the entire environment into a compressed `.tgz` archive.
- **Bulk Environment Testing:** Automated discovery and execution of package internal tests.
- **Self-Healing Verification:** Automated patching of legacy syntax ambiguities.
- **Atomic Commits and Rollbacks:** Every change is an atomic Git commit.

### **3 Formal Verification**

The isolation, auditing, and maintenance strategies have been formally verified in Coq (`safety.v`), proving that the environment lifecycle maintains strict projection on the user’s branch and preserves global system state.

### **4 Implementation**

`spip` is implemented in C++23. It integrates with native system tools and advanced static analysis to provide a robust, version-controlled, and secure Python development experience.