Veil – Linux Filesystem Sandboxing Tool

Project Overview

Veil is a Linux command-line tool that allows users to run applications inside a temporary filesystem sandbox. Only explicitly allowed directories are visible to the application. Once Veil exits, the system is left completely unchanged.

Problem Statement

On Linux, desktop applications usually have unrestricted access to the user home directory. Running untrusted binaries, installers, or scripts requires blind trust and exposes personal data.

Solution Summary

Veil enforces least-privilege filesystem access by restricting what an application can see. If a directory is not visible, it is not accessible. Security is achieved without system-wide changes.

Core Design Principles

1. Deny by default

2. Filesystem visibility equals permission

3. No system-wide impact

4. Stable behavior for GUI applications

5. Explicit user control

High-Level Working Flow

1. User launches an application using Veil.

2. Veil creates a temporary sandbox directory.

3. A FUSE filesystem is mounted.

4. System directories are exposed as read-only.

5. One user-selected directory is exposed as read-write.

6. The application runs inside the sandbox.

7. Denied accesses are logged.

8. Sandbox is removed cleanly after exit.

Filesystem View Inside Sandbox

Read-only: /usr, /lib, /bin, /etc

Read-write: /tmp, user selected directory

All other paths are invisible.

Denied Access Handling

If the application tries to access a restricted path, the operation fails safely. Veil logs the denied access and explains the failure after the application exits.

Post-Run Suggestions

Veil provides clear suggestions on which directories to allow if the user wants to rerun the application successfully.

Optional Interactive Mode (CLI Only)

Veil includes an experimental interactive permission mode for command-line tools. This mode allows blocking file access and asking the user for permission at runtime.

This mode is disabled by default and is not supported for GUI applications.

Smart Engineering Approach

Veil is built using battle-tested tools to avoid reinventing complex and error-prone components.

Key Libraries and Tools Used

Filesystem: fuser crate for FUSE handling

Sandboxing: bubblewrap for secure process isolation

Concurrency: DashMap for thread-safe permission storage

Communication: JSON over Unix sockets using serde

CLI Interface: clap for argument parsing

Logging: log and env_logger

Implementation Strategy

1. Start with a FUSE passthrough filesystem using fuser.

2. Add allow and deny checks for paths.

3. Store permissions using DashMap.

4. Run applications inside bubblewrap.

5. Log denied accesses and generate summaries.

6. Keep interactive permissions optional and CLI-only.

What Veil Does Not Do

Veil does not perform malware detection, antivirus scanning, or system-wide enforcement. It does not modify file permissions or install persistent policies.

Use Cases

Running untrusted binaries safely

Testing installers and scripts

Developer workspace isolation

Preventing accidental file corruption

Why This Fits FOSS Hack

Veil is fully open source, avoids proprietary APIs, uses standard Linux primitives correctly, and demonstrates disciplined systems engineering.

Project Timeline

Preparation Phase: Learning and prototyping before March

Development Phase: Core implementation during March

Final Phase: Documentation, demo, and cleanup

Future Work

Graphical interface

Policy presets

Improved visualization of logs

Extended interactive permissions for CLI tools

End of Report