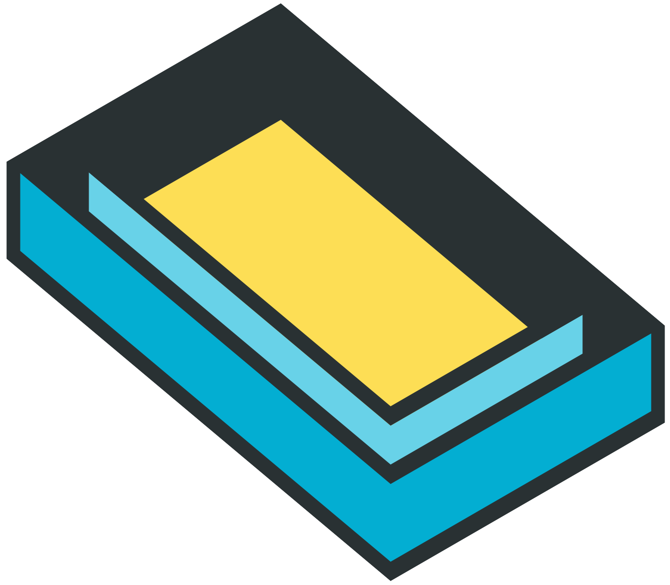
Island: Sandboxing tool powered by Landlock



FOSDEM

Mickaël Salaün – kernel maintainer

2026-01-31

Island’s goal

Protect users’ data access **from buggy, malicious, or exploited software** by restricting most commands.

Island leverages **Landlock**, the unprivileged Linux sandboxing mechanism.

Sandbox

“A **restricted**, controlled **execution environment** that prevents potentially malicious software [...] from accessing any system resources except those for which the software is authorized.”

Landlock

Landlock helpers

Examples of sandbox tools:

- setpriv
- Minijail
- Firejail

Examples of sandbox libraries:

- Landlock Rust crate
- Landlock Go library
- Minijail
- Pledge for Linux

Landlocked apps

Examples of various sandboxed apps:

- Zathura (document viewer)
- Pacman (package manager)
- Cloud Hypervisor (VM monitor)
- Suricata (network IDS)
- Polkadot (blockchain SDK)
- wireproxy (Wireguard client)
- GNOME LocalSearch (search engine)
- XZ Utils (archive manager)

Key Landlock features

Unprivileged

- Dynamic and **ephemeral** restrictions: no persistent state, no file labels
- **Independent** restrictions: the kernel manages a set of standalone policies per user, service, program…
- **Nested** sandboxes
- **One-way** restrictions: cannot be disabled once enabled for a process hierarchy.

Access control

- Configuration not explicitly tied to system calls, but to the **kernel semantic**: no need to synchronize with library/code updates using new syscalls
- Orthogonal to namespaces: only **restrict access**, do not build “views” of kernel resources (e.g. filesystem, network)

How does Landlock work?

Restrict ambient rights according to the kernel semantic (e.g., global filesystem access) for a set of processes, thanks to **3 dedicated syscalls**.

Security policies are **inherited** by all new children processes without being able to escape their sandbox.

Current access control

Implicit restrictions

- Process impersonation (e.g., ptrace)
- Filesystem topology changes (e.g., mounts), when it makes sense

Explicit access rights

- Filesystem
- Networking
- Signaling
- UNIX socket

Use case #1

Exploitable bugs in trusted programs: protect from vulnerable code maintained by developers.

Candidates:

- Parsers: archive tools, file format conversion, renderers…
- Web browsers
- Network and system services

Use case #2

Untrusted programs: protect from potentially malicious third-party code.

Candidates:

- **Sandboxer tools**
- Container runtimes
- Init systems

Landlock ABI versions

1. Linux 5.13: Initial set of FS access rights
2. Linux 5.19: Rename and link
3. Linux 6.2: Truncation
4. Linux 6.7: TCP connect and bind
5. Linux 6.10: IOCTL for devices
6. Linux 6.12: Signal and abstract UNIX socket
7. Linux 6.15: Log configuration

Island

Why Island?

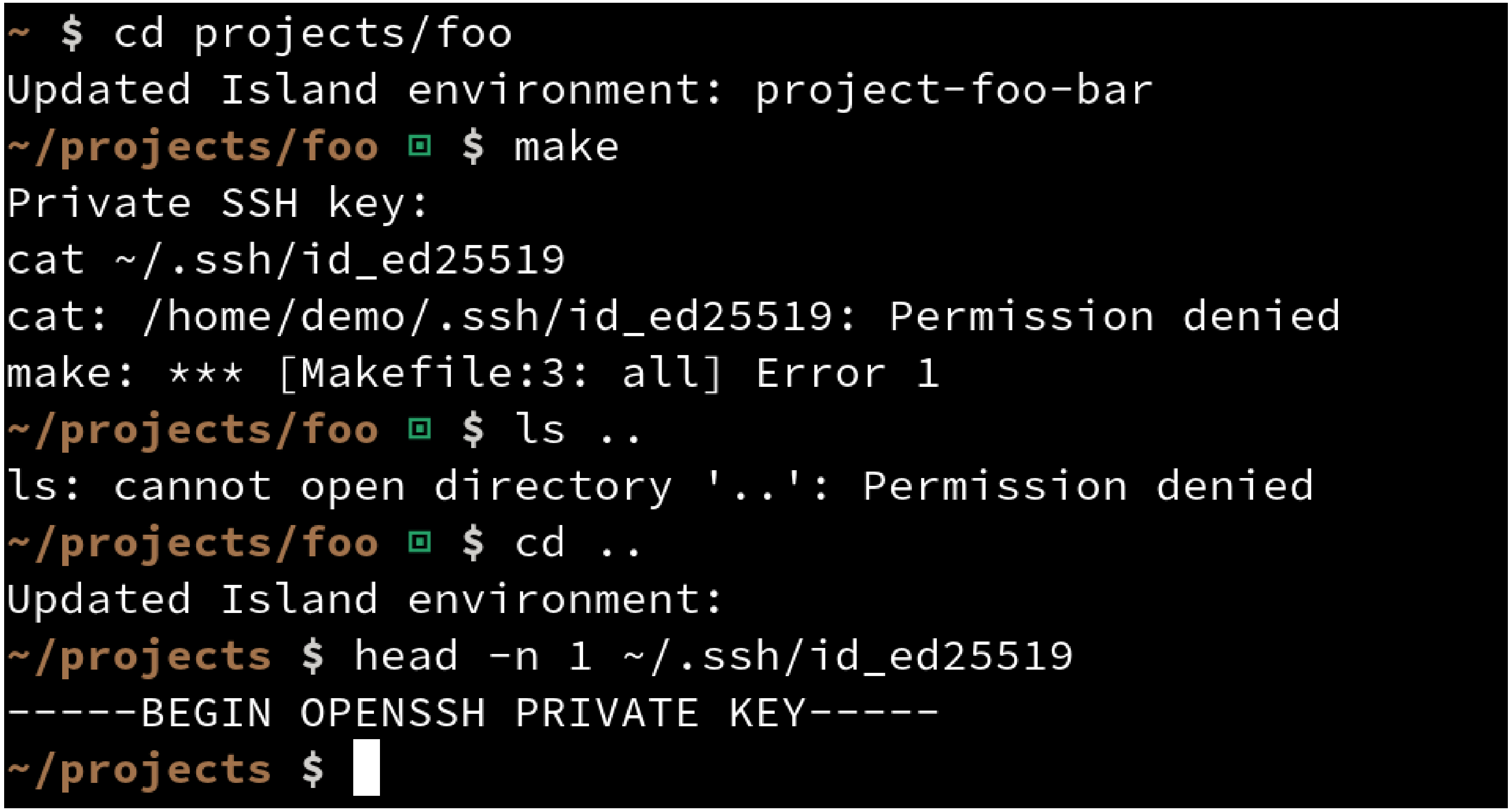
Make Landlock practical for everyday workflows by acting as a high-level wrapper and policy manager.

Island is designed to be available to everyone (using a terminal), and to avoid cognitive load (once configured).

Island’s main properties

- **Zero-code integration**: Runs existing binaries without modification.
- **Declarative, flexible, and sharable policies**: Uses TOML profiles instead of code-based rules.
- **Context-aware activation**: Automatically applies security profiles based on your current working directory.
- **Dedicated environments per sandbox**: Manages isolated workspaces (XDG directories, TMPDIR) in addition to access control.

Demo



```
~ $ cd projects/foo
Updated Island environment: project-foo-bar
~/projects/foo ▣ $ make
Private SSH key:
cat ~/.ssh/id_ed25519
cat: /home/demo/.ssh/id_ed25519: Permission denied
make: *** [Makefile:3: all] Error 1
~/projects/foo ▣ $ ls ..
ls: cannot open directory '..': Permission denied
~/projects/foo ▣ $ cd ..
Updated Island environment:
~/projects $ head -n 1 ~/.ssh/id_ed25519
-----BEGIN OPENSSH PRIVATE KEY-----
~/projects $ █
```

Current limitations

Missing features

- Not full isolation on most **desktop** or (unsandboxed) Tmux environments because of unrestricted access to local services
- No restriction to access file’s metadata, but data is well handled

The upcoming kernels will address these issues, but in the meantime, get ready!

Island is pretty young and looking for feedbacks!

Landlock Config

Simple example (FS-only)

```
abi = 6

[[variable]]
name = "writable"
literal = ["/tmp", "/var/tmp", "/home/user/tmp"]

# Main system file hierarchies can be read and executed.
[[path_beneath]]
allowed_access = ["abi.read_execute"]
parent = ["/bin", "/lib", "/usr", "/dev", "/proc", "/etc", "/home/user/bin"]

# Only allow writing to temporary and home directories.
[[path_beneath]]
allowed_access = ["abi.read_write"]
parent = ["${writable}"]
```

Properties

- Ease sharing and maintaining security policies
- Declarative, deterministic, idempotent
- Customizable
- Handle variables and compose them commutatively:
  - Variables are a set of values
  - Must be defined when using it, but can be empty
- Individual access rights or groups scoped to a specific Landlock ABI: read_execute, read_write, all

Composed and shared policies

Requirements

- Standalone files/snippets tailored to specific programs
- Handle different set of access rights

Several sources

- Provided by upstream developers (independent from distros)
- Provided by distro packages
- Provided by end users, communities

Wrap-up

Try Island (with Zsh)

```
git clone https://github.com/landlock-lsm/island
cd island
cargo install --path .
export PATH="$PATH:$HOME/.cargo/bin"
rehash
source <(island completion zsh)
source <(island hook zsh)
cd ~/my-project
island create my-project
ls /
```

Island

- Protects from malicious or unattended actions per activity
- Dedicated to Linux users with a terminal
- Easy to use

Questions?

landlock@lists.linux.dev

Thank you!