

Syd

Writing an Application Kernel in Rust

Ali Polatel



FOSDEM 2026

whoami

I've got a bike, you can ride it if you like.

- Exherbo Linux dev, ex-Gentoo dev
- Main author of Sydbox
- Chess trainer, Co-founder of `chesswob.org`
- Interests: Linux, BSD, Sandboxing, Security, Board games, Translation
- E-mail: `alip@chesswob.org`

Outline

Long you live and high you fly, smiles you'll give and tears you'll cry.

- What is an Application Kernel?
- Syscall Interception
- Why Rust?
- Memory Safety Patterns
- Safety and Performance
- Testing Infrastructure
- Q&A

What is an Application Kernel?

We call it riding the gravy train.

Definition: A Library OS variant that intercepts, emulates, and transforms syscalls in user-space for sandboxed processes. (cf. Exokernel, SOSP'95)

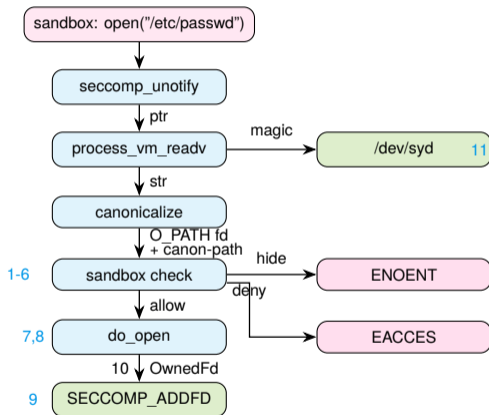
- Intercepts syscalls via `seccomp_unotify(2)`, `ptrace(2)`, `landlock(7)`
- Emulates file system, network, process operations
- Transforms paths, flags, credentials at runtime
- Configuration via `/dev/syd`

Similar projects: gVisor (Google, Go), rump kernels (NetBSD), Nabla containers, OSv, MirageOS

application + kernel \neq application kernel — “You must take your opponent into a deep dark forest where $2 + 2 = 5$, and the path leading out is only wide enough for one.” — Mikhail Tal

The Path of open(2)

Hello, is there anybody in there?



Transformations

- 1 Path hiding: deny → ENOENT
- 2 Mask: path → /dev/null
- 3 Crypt: transparent encryption
- 4 Append: force O_APPEND
- 5 Filter: rewrite proc/status
- 6 FS sandbox: block by fs type
- 7 rand_fd: randomize fd number

TOCTOU Prevention

- 8 Open via /proc/thread-self/fd/
- 9 Inject fd with SECCOMP_ADDFD
- 10 Never CONTINUE syscall
- 11 /dev/syd = sealed memfd

Why Rust?

Mother did it need to be so high?

- `#![forbid(unsafe_code)]` modules (ELF parser, glob matcher)
- `#![forbid(clippy::arithmetic_side_effects)]` for DoS prevention
- Type-state patterns: `SealBox<T> → Sealed<T>`
- Ownership = resource tracking (file descriptors, memory mappings)
- Zero-cost abstractions in hot paths
- Fearless concurrency for thread pool

Memory Safety Patterns

No one told you when to run, you missed the starting gun.

Type-State Pattern (sealbox.rs)

```
1 // Linear consumption: SealBox<T> -> Sealed<T>
2 pub fn seal(self, vma_name: Option<&CStr>) -> Result<Sealed<T>, Errno> {
3     mprotect_readonly(self.map_ptr, self.map_len)?;
4     mseal(self.map_ptr, self.map_len)?; // Linux 6.10+
5     mem::forget(self); Ok(sealed)
6 }
7
8 // Compile-time + runtime state tracking
9 enum Sealable<T: Copy> {
10     Unsealed(T), // DerefMut allowed
11     Sealed(Sealed<T>), // DerefMut panics!
12 }
```

ELF Parser (elf.rs) — 887 lines, zero unsafe

```
1 #![forbid(unsafe_code)]
2 #![forbid(clippy::arithmetic_side_effects)]
3 #![forbid(clippy::cast_possible_truncation)]
```

SyscookiePool (cookie.rs) — Guard pages + fillrandom(2) + mseal(2)

Safety and Performance

Together we stand, divided we fall.

Glob Matcher (wildmatch.rs) — rsync (1986), Kirk Krauss's FastWildCompare

```
1 #![forbid(unsafe_code)]
2 // SIMD via memchr crate, dual backup for * vs **
```

Method	Mean	Samples	2.3x faster
wildmatch	37.12 ms	807/808	
fnmatch (libc)	86.64 ms	347/347	

Custom Path Types (path.rs)

```
1 pub struct XPathBuf(TinyVec<u8; 400>); // stack alloc <400 bytes
2 pub struct XPath([u8]);               // DST, SIMD compare
```

- Why not `std::path::Path`? Bytes not `OsStr`, stack alloc, SIMD compare

Testing Infrastructure

Can you tell a green field from a cold steel rail?

Multi-Architecture CI Pipeline (.gitlab-ci.yml)

- Native runners: x86_64, aarch64, armv7, s390x
- Stages: build → test → ltp → compat → release
- 32-bit cross-compile tests (i686 under x86_64 sandbox)

External Test Suites


- **LTP**: 4000+ Linux syscall tests under -pltp profile
- **gnulib**: 250+ POSIX compatibility tests under -ppaludis

Sandbox Escape Tests (t/do.rs — 20+ attack vectors)

```
1 // TOCTOU attacks that no longer work:
2 ptrmod_toctou_exec_* // pointer modification during execve
3 symlink_exchange_toctou_* // RENAME_EXCHANGE race
4 vfsmod_toctou_* // VFS fd swap attacks
5 magiclink_toctou // /proc/self/fd race
```

Thanks for watching! Questions?

So you think you can tell heaven from hell?

- Gitlab: <https://gitlab.exherbo.org/sydbox/sydbox.git>
- Manual: <https://man.exherbo.org>
- IRC: #sydbox at Libera
- Matrix: #sydbox:mailstation.de
- Thanks to friends at  ^{oneM2M} MORE DATA for sponsoring my attendance!