

Syd

An Introduction to Secure Application Sandboxing for Linux

Ali Polatel



FOSDEM, 2025

Before we start...

The game is on! Viva la revolución!

- CTF: {https,ssh}://syd.chesswob.org
 - user/pass: syd
 - rules: /etc/user.syd-3
 - goal: read /etc/CTF & get 200€!
- GIT: <https://gitlab.exherbo.org/sydbox/sydbox.git>
- DOC: <https://man.exherbolinux.org>
- ML: <https://lists.sr.ht/~alip/exherbo-dev>
- IRC: #sydbox at Libera
- Matrix: #sydbox:mailstation.de

History: Exherbo

All you touch and all you see is all your life will ever be.

- Gentoo Linux: because no penguin can swim faster!
 - Source-based, rolling-release distribution
 - Sandboxing required to detect package build mishaps
 - Gentoo Sandbox: LD_PRELOAD, no network restrictions
- Exherbo Linux: when you hear hoofbeats, think of a zebrapig!
 - s/Gentoo fork/Gentoo done right/
 - Recommended watch by Bryan Østergaard, aka kloeri:
 - “10 cool things about Exherbo”
 - “You’re doing it wrong!”
 - Package testing by default

History: Syd

Did you exchange a walk-on part in the war for a lead role in a cage?

- SydB`ox`: The `Ⓜ`other `SⓂ`ndb`ox`
 - SydB`ox`-1: `c`, `ptrace`
 - Network sandboxing: Builds restricted to loopback
 - Exec sandboxing: Metadata phase restricted to shell builtins
 - SydB`ox`-2: `c`, `seccomp`
 - Initial experiments to replace `ptrace` with `seccomp`
 - Initial experiments to make SydB`ox` a security boundary
 - Read sandboxing & Path hiding
 - SydB`ox`-3, aka Syd: `rust`, `seccomp`, `landlock`, `namespaces`

Overview: What?

Welcome my son, welcome to the machine.

- An application kernel to sandbox applications on Linux
- Written in Rust, `cargo install --locked syd`, requires `libseccomp`
- Licensed GPL-3.0, forever free
- Requires Linux ≥ 5.19 with `CONFIG_SECCOMP_FILTER`
- Good portability across architectures
 - Tested on `arm64`, `armv7`, `ppc64le`, `riscv64`, `s390x`, `x86`, and `x86-64`, with `mips`, `m68k`, `superh` and `loongarch` support on the way!
 - Should work on any `libseccomp` supported architecture with minimal work

Overview: How?

You dreamed of a big star, he played a mean guitar.

- Make sandboxing as easy as text searching is with `grep(1)`!
- UNIX philosophy: Do one thing and do it well
- Simple interface for complex sandboxing mechanisms
- Secure by default with minimal overhead
- No extra privileges required: No SETUID, EBPF, or LKM
- Can be used as login shell

Features: Basics

Breathe, breathe in the air. Don't be afraid to care.

- Path sandboxing
 - Read sandboxing and Path Masking
 - Write sandboxing and Append-only Paths
 - Stat sandboxing and Path Hiding
 - Ioctl sandboxing
 - Contain AI/ML workloads
 - Safe access to PTY, DRM, and KVM
- Network sandboxing
 - feat. UNIX, IPv4, IPv6, Netlink, and KCAPi sockets
 - Application level firewalls with IP blocklists
- `pledge(2)` like refined sandboxing categories
 - `stat`, `read`, `write`, `exec`, `chdir`, `readdir`, `create`, `delete`, `rename`, `link`, `truncate`, `tmpfile`, `ioctl`, `node`, `attr`, `chown`, `chgrp`, `chroot`, `net/bind`, `net/connect`, `net/send`

Features: Execution Control

One slip, and down the hole we fall, it seems to take no time at all.

- Exec sandboxing
 - Requires `PTRACE_EVENT_EXEC` to be safe
- SegvGuard
 - Block execution if binary is crashing repeatedly
 - Wider range of trigger signals than Grsecurity
 - Can also be triggered using sandbox rules
 - `kill/read+/etc/shadow`
- Force sandboxing (aka Verified Execution)
 - Verify binary/library integrity at `exec(3)/mmap(2)` time
 - like Veriexec (NetBSD), and Integriforce (HardenedBSD)
 - `sha3-512`, `sha3-384`, `sha3-256`, `sha1`, `md5`, `crc64` and `crc32`
- Trusted Path Execution
 - like Grsecurity, HardenedBSD
 - Execution only allowed from “Trusted directories”
 - Not writable by group or others
 - Optionally owned by root or current user

Features: Nice-to-haves

Each small candle lights a corner in the dark.

- Sandbox lock and dynamic configuration
- `AT_SECURE` set by default
- Executable restrictions:
 - PIE and non-executable stack enforced by default
 - Deny based on bitness: `trace/deny_elf32`
 - Deny based on linkage:
 - `trace/deny_elf_static`
 - `trace/deny_elf_dynamic`
 - Deny scripts: `trace/deny_script`
- Fake root with `root/fake`
- Forcing umask with `trace/force_umask`
- Deny directory-traversal with `trace/deny_dotdot`
- Deny access to TSC with `trace/deny_tsc` and `libsydttime`

Features: Bonus

Click clack, ride on the rail track.

- Lock sandboxing, uses Landlock LSM
- Proxy sandboxing
 - SOCKS proxy forwarding with network namespace isolation
 - Defaults to TOR
- Memory & PID sandboxing
 - Simple alternatives to Control Groups
- SafeSetID
 - Safe user/group switching
 - Predefined UID/GID transitions
- Ghost mode
 - Similar to `Seccomp` Level 1, aka Strict Mode
- Namespaces, Containerization, and `syd-oci`
- Learning mode with `pandora`

The End

You'll lose your mind and play free games for May!

- CTF: {https,ssh}://syd.chesswob.org
 - user/pass: syd
 - rules: /etc/user.syd-3
 - goal: read /etc/CTF & get 200€!
- GIT: <https://gitlab.exherbo.org/sydbox/sydbox.git>
- DOC: <https://man.exherbolinux.org>
- ML: <https://lists.sr.ht/~alip/exherbo-dev>
- IRC: #sydbox at Libera
- Matrix: #sydbox:mailstation.de