

# Sydbox: A modern Sandbox

Patrick Lauer, Ali Polatel



FOSDEM 2026

# whoami

I am he as you are he as we are all together

- Patrick Lauer
- Gentoo Linux developer
- Evangelist, tester of Sydbbox
- E-mail: `me@patricklauer.dev`

- Ali Polatel
- Exherbo Linux dev, ex-Gentoo dev
- Main author of Sydbox
- Chess trainer, Co-founder of [chesswob.org](http://chesswob.org)
- Interests: Linux, BSD, Sandboxing, Security, Board games, Translation
- E-mail: [alip@chesswob.org](mailto:alip@chesswob.org)

- Founded 1999 in Mönchengladbach
- Close ties to Open-Source Community
- 40 Open-Source experts
- Consulting, development, training, support (3rd-level / 24x7)
- Open-Source infrastructure with Debian, Kubernetes and Proxmox
- Open-Source databases with PostgreSQL
- DevOps with Ansible, Puppet, Terraform and others
- Since 2025 independent owner-managed company again

- Bootstrapped startup for Big Data ideas
- "Infinitely scalable" log storage built on PostgreSQL
- on your server, on our server (managed), or on someone else's server (cloud)

# Outline

Good day, sunshine

- What is sandboxing
- Syscall Interception
- Sydbbox: The Syscall rewriter
- TOCTOU and Security
- Performance
- Q&A

# What is sandboxing

You never give me your money

- Linux distro development: Want to build software
- Build systems are confused and do strange things
- Sandbox: Contain software to only read/write to specific paths
- Example: Don't allow writing to `/etc/passwd`
- Example: Don't allow reading `/.ssh/`

# Secure?

I'm only sleeping

- Safety: Avoid "bad" outcomes
- Confidentiality: Avoid access to sensitive data
- Security: Avoid privilege escalation
- Syd tries to cover all aspects

# History: Syd

Ob-la-di, Ob-la-da ...

- SydBox: The ☺ther S@ndbøx
  - SydBox-1: c, ptrace
    - Network sandboxing: Builds restricted to loopback
    - Exec sandboxing: Metadata phase restricted to shell builtins
  - SydBox-2: c, seccomp
    - Initial experiments to replace ptrace with seccomp
    - Initial experiments to make SydBox a security boundary
    - Read sandboxing & Path hiding
  - SydBox-3, aka Syd: rust, seccomp, landlock, namespaces

# Requirements

It doesn't really matter what chords I play / What words I say or time of day it is

- Linux kernel, v.5.19 or later
- we want/need `SECCOMP_FILTER_FLAG_WAIT_KILLABLE_RECV`
- ptrace, seccomp, seccomp-bpf support
- memfd
- Optionally landlock LSM
- Newer kernels add some useful extra features
- `syd` runs unprivileged, no need for `suid` or other dangerous magic

# Related projects / competition

All together now

- User Mode Linux, full VMs (qemu-kvm etc.), firecracker
- sandboxes (bubblewrap, firejail, gentoo sandbox)
- emulators (qemu, gvisor)
- unikernels (nabla, quark) ?

# Excursion: Syscalls

Picture yourself in a boat on a river

- Interface between userspace applications, and the kernel
- Linux has ~450 syscalls
- Interface can only be extended, but not changed (would break existing binaries\*)
- So new syscalls are added: clone -> clone2 -> clone3, open -> openat -> openat2
- Usually libc hides implementation details, libc open() may use any of the open\* syscalls

\* linux-specific, Windows/MacOS/OpenBSD avoid that with different architecture

# Excursion: ptrace

Take a sad song, and make it better

- Process Trace
- Mechanism to attach to a process and observe syscalls (debugging)
- ... soon extended to allow manipulating syscalls
- all in all good tooling, but can be slow/fragile
- multithreaded apps and ptrace are difficult

# Excursion: seccomp

Yesterday, all my troubles seemed so far away

- "Secure Computation", originally designed to limit subprocesses to a whitelist of syscalls
- Example: Converting an image, data comes on stdin, goes out on stdout, why would you ever open() any file?
- seccomp() syscall transitions a process into "secured" state

# Excursion: seccomp-bpf

Happiness is a warm gun

- Allows attaching a BPF program to each syscall
- (c|e)BPF: Kernel magic that allows to efficiently run code in the kernel
- We can run arbitrary code, on the kernel side, when a syscall is triggered

# Sydbox: The syscall rewriter

Twist and Shout

- Any process (and thread!) started under syd is seccomp-bpf sandboxed
- Threadpool of syd-emulator threads to handle notifications, no bottleneck
- A call to `open()` may get filtered:
  - Syd canonicalizes path names ( `../foo/../foo/../foo/bar` -> `../foo/bar` )
  - Bind mounts in Mount Namespaces enforce Virtual File System (VFS) restrictions, such as read-only, nodev, noexec, nosuid, and nosymfollow.
  - Syd allow/denylists based on glob/regex patterns: `allow/readdir+/home/`
  - Process sees the result that syd wants it to see

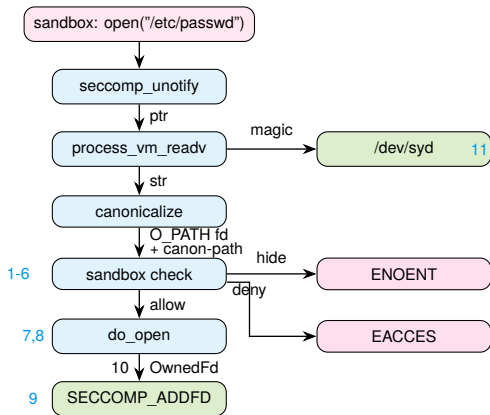
# Sydbox: The syscall rewriter

Help! I need somebody's help!

- Backchannel: `seccomp_unotify`
- Any `seccomp-bpf` action may trap to userspace
- `syd` listens on a FD, `open()` in a process triggers `seccomp-bpf` rule, which triggers `syd`
- Complex logic like path canonicalization lives in `syd`, `seccomp-bpf` rules can remain simple

# The Path of open(2)

Take a sad song and make it better



## 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

# Our Security model

Fixing a hole

- We have to trust the kernel
- Assume no malicious modification of the kernel and kernel interfaces
- Anything that happens within an application we “don’t care” about
- Boundary: Syscalls

# Sydbox: The TOCTOU avoider

Hey Bulldog

- TOCTOU: Time of check to time of use
- Race condition between an object being checked, and then later used
- source of so many security issues, especially in sandboxes (docker, podman, ...)

# Sydbox: The TOCTOU avoider

For the benefit of Mr.Kite

- A path is not a single pointer, but a sequence of names
- `/home/me/docs/pdf/0815.pdf` is at least five different items
- Walking such a path is always a snapshot in time
- Imagine docs being a symlink - I can delete and recreate it as I want
- A path walker might `stat()` to see if we have access to the file
- ... and then later `open()` ... but in that time the symlink was rewritten
- Object at Time Of Check is not the object we have at Time Of Use!

# Paranoia

Here, there, and everywhere

- Any new/unknown syscalls are blocked by default
- Trying to always return the right errorcodes as the kernel would
- A lot of effort to prevent race conditions / TOCTOU
- Onion design: many layers, even if one layer breaks we hope to survive intact
- Gluetraps and self-destruct: syscall cookies, signal handlers, landlock
- Drop privileges hard!
- Lots of regression tests, CI
- Public CTF to motivate external stresstesting:
- Try to read the file `/etc/CTF` on `syd.chesswob.org` with `ssh user/pass: syd`

# Performance

One for you, nineteen for me

- Extra cost: one in-kernel seccomp-bpf rule, sometimes one context switch and possibly extra syscalls
- Most of the overhead is in the kernel and can't be improved in sydbox
- Example: Building git, no sandbox:  $103.719 \pm 3.850$
- `syd -ppaludis -plandlock`,  $104.507 \pm 1.829$
- `syd -puser -phide -plandlock`,  $149.308 \pm 1.014$
- `gvisor --network=host -platform ptrace`,  $405.410 \pm 14.433$
- `gvisor --network=host -platform kvm`,  $959.115 \pm 63.610$

# Sydbox profiles

Martha, my dear

- Config file for specific tasks
- Contains rules for path access: allow/deny
- Contains checksum of binaries
- Enables or disables specific features like LandLock LSM

# Pandora's box

It's getting better all the time

- Pandora: Profile generator for sydbox
- Runs sydbox in permissive mode and logs events, then synthesizes a config out of it
- Last action wins
- Run: `PANDORA_OUT=wc-profile pandora profile wc -l .`
- Config can be used to confine a process: `syd -P wc-profile /usr/bin/wc -l`

# Sydbox profiles

A day in the life

```
allow/read,exec+/usr/lib64/libc.so.6
allow/readdir+/home/me/syd-test
allow/exec+/usr/bin/wc allow/exec+/usr/lib64/ld-linux-x86-64.so.2
allow/read+/etc/ld.so.cache
sandbox/force:on
force+/usr/bin/wc:sha3-512:05d7[...]9ec6
force+/usr/lib64/libc.so.6:sha3-512:d460[...]02a3
force+/usr/lib64/ld-linux-x86-64.so.2:sha3-512:2c93[...]a656
```

# The Syd device node

When I'm sixty four

- Within the sandbox: `/dev/syd` virtual device node
- Can be used to rewrite config (if allowed) and lock config
- Allows dynamically adding/removing paths
- Allows dropping permissions after startup/init

# Bonus features

Baby you can drive my car

- OCI runtime - Syd is also a container engine
- SegV-Guard: rate-limit process restart on crash
- ioctl sandboxing: Allows using CUDA/ROCm
- Path hiding, path masking, append-only paths
- Network sandboxing, network port rules
- SafeSetID, Ghost Mode
- And many other features ...

# Thanks for watching! Questions?

And in the end / the love you take / is equal to the love you make

- Gitlab: <https://gitlab.exherbo.org/sydbbox/sydbbox.git>
- Manual: <https://man.exherbo.org>
- IRC: #sydbbox at Libera
- Matrix: #sydbbox:mailstation.de
- Thanks to credativ for supporting Patrick's attendance!
- Thanks to friends at  <sup>MORE DATA</sup> for sponsoring Ali's attendance!