



how can we make procs safe?

current state of the art and a few libpathrs updates



**Linux
Plumbers Conference** | Dublin, Ireland **Sept. 12-14, 2022**

Aleksa Sarai (SUSE)
cyphar@cyphar.com

A decorative graphic of a green pipe network with various fittings, valves, and elbows, framing the central text.

threat model

We are executing in an environment where a user has managed to mess with the filesystem and possibly mounts.

We want to be able to **detect** if we are being tricked into operating on a different path than the one we expected.

Main usecase is container runtimes, but basically any program operating in or on chroots would benefit.



Linux

Plumbers Conference | Dublin, Ireland Sept. 12-14, 2022

A decorative graphic of a green pipe network with various fittings, elbows, and valves, framing the text on the slide.

what's special about procfs?

With `openat2(RESOLVE_*)`, any non-`procfs` filesystem path can be accessed safely with various resolution restrictions.

With `procfs`, we require more than just the path be resolved “reasonably”, we want a **specific** `procfs` endpoint to be reached. The core issue is that `/proc/self/environ` and `/proc/self/sched` exist.



Linux

Plumbers Conference | Dublin, Ireland Sept. 12-14, 2022

A decorative graphic of a green pipe network with various fittings, elbows, and valves, framing the central text.

current status

(non-magiclinks)

In Linux 5.6, `openat2(2)` made this safe:

- `openat("/proc", O_PATH)`
- Check the `f_type` (`fstatfs`) and `stx_ino` (`statx`).
- `openat2(<procfsfd>, "<pid>/attr/current",
{RESOLVE_NO_XDEV|RESOLVE_NO_SYMLINKS})`



Linux

Plumbers Conference | Dublin, Ireland Sept. 12-14, 2022

A decorative graphic of a green pipe network with various fittings, valves, and elbows, set against a light gray background. The pipes are arranged in a complex, interconnected pattern, with some sections being thicker than others. The overall style is clean and technical.

what's the issue with magiclinks?

We can't use `RESOLVE_NO_XDEV` because they are almost always crossing a mountpoint. **But you can also mount on top of them.**

We can't use `RESOLVE_NO_SYMLINKS` for obvious reasons.



Linux

Plumbers Conference | Dublin, Ireland Sept. 12-14, 2022

A decorative green pipe graphic runs vertically on the left side of the slide, with various fittings and valves. It starts at the top left, goes down, then right, then down again, and finally right at the bottom. There are several elbows, tees, and valves along its path.

current status

(magicklinks)

It turns out this is safe since Linux 5.8 (w/CAP_SYS_ADMIN):

- `open_tree("/proc", OPEN_TREE_CLONE | AT_RECURSIVE)`
- Check the `f_type` (`fstatfs`) and `stx_ino` (`statx`).
- `openat2(<procfsfd>, "<pid>", {O_PATH, RESOLVE_NO_XDEV|RESOLVE_NO_SYMLINKS})`
- `statx(<proccselfd>, "exe", AT_SYMLINK_NOFOLLOW)`
 - Check whether `STATX_ATTR_MOUNT_ROOT` is set.
 - If not, safe to use (no races because of `OPEN_TREE_CLONE`).

(See [this example program](#).)



Linux

Plumbers Conference | Dublin, Ireland Sept. 12-14, 2022

A decorative graphic of a green pipe network with various fittings, valves, and elbows, framing the central text.

why does this work?

(and why might it break in the future?)

The safety of this setup relies on several undocumented behaviours:

- `OPEN_TREE_CLONE` mounts are an `O_PATH` to a bind-mount and there is no way for an external process to change any overmounts.
 - Returned tree is an anonymous mount namespace.
 - Even a `CAP_SYS_ADMIN` user in a different mountns cannot mount into it.
 - Mount propagation to the clone is [explicitly disabled](#).



Linux

Plumbers Conference | Dublin, Ireland Sept. 12-14, 2022

A decorative graphic of a green pipe network with various fittings, valves, and elbows, set against a white background. The pipes are arranged in a complex, interconnected pattern, with some sections being vertical and others horizontal. The pipes have a slight 3D effect with shadows.

libpathrs

- Using `openat2` (`RESOLVE_IN_ROOT`) correctly is non-trivial.
 - Lots of messing around with `O_PATH`.
 - No other syscalls support `RESOLVE_IN_ROOT`.
 - How do we deal with old kernels?
- **Solution:** Rust library that provides “nice” helpers that Do The Right Thing™.
 - ... and it emulates `RESOLVE_IN_ROOT` on old kernels!
 - ... but this requires we port programs to use it.



Linux

Plumbers Conference | Dublin, Ireland Sept. 12-14, 2022

A decorative graphic of green pipes with valves and elbows, running vertically on the left side of the slide and curving at the top and bottom.

an update on libpathrs

Safe wrappers for procs.

- Cache a safe open_tree'd procs handle.
- `procsel_get("exe")` - or something...

Sane C API (which then can be used with Go nicely).

- Go programs are used to being able to call `.Close()` many times.
- Should we remove the footgun-guards and just pass fds?



Linux

Plumbers Conference | Dublin, Ireland Sept. 12-14, 2022

A decorative graphic of a green pipe network with various fittings, elbows, and valves, framing the central text.

an update on libpathrs

How to model the (new and old) mount API?

- We want to use the new one to avoid using /proc.
- Should we abstract the whole thing or just expose the old one?

How much of the VFS API should be replicated by libpathrs?

- In theory, any operation which has `AT_EMPTY_PATH` is okay.
- How much should we trust library users to not footgun themselves?



Linux

Plumbers Conference | Dublin, Ireland Sept. 12-14, 2022



an update on libpathrs

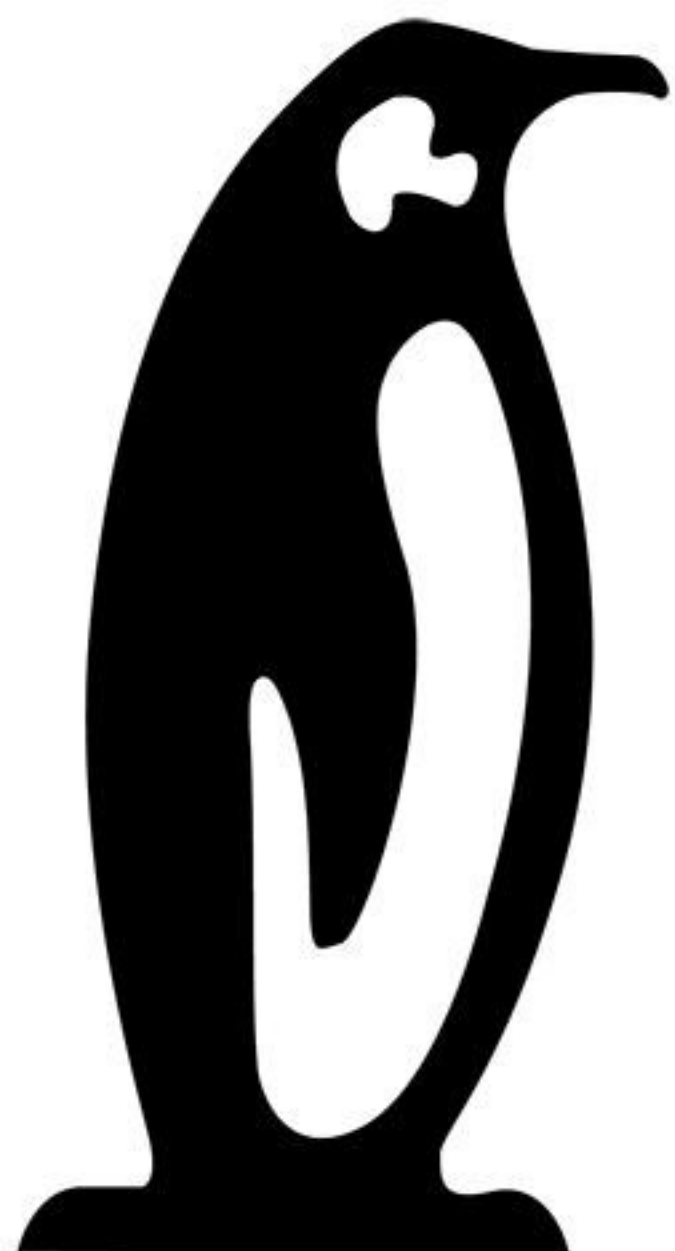
<https://github.com/openSUSE/libpathrs>

Still being worked on. Plan to port umoci to this first.



Linux

Plumbers Conference | Dublin, Ireland Sept. 12-14, 2022



Linux Plumbers Conference

Dublin, Ireland **September 12-14, 2022**