Microsoft

Update on Landlock: Audit, Debugging and Metrics

Kernel Recipes

Mickaël Salaün

2023-09-25

Update on Landlock: Audit, Debugging and Metrics

Landlock is available in mainline since 2021 (Linux 5.13), but with some limitations due to the iterative approach.

Landlock is now enabled by default on multiple distros: Ubuntu 22.04 LTS, Fedora 35, Arch Linux, Alpine Linux, Gentoo, Debian Sid, chromeOS, CBL-Mariner, WSL2

This talk is about audit support for Landlock

Sandboxing

A security approach to **isolate** a software component **from the rest of the system**.

Sandboxing

A security approach to **isolate** a software component **from the rest of the system**.

An innocuous and trusted process can become malicious during its **lifetime** because of bugs exploited by attackers.

Sandboxing

A security approach to **isolate** a software component **from the rest of the system**.

An innocuous and trusted process can become malicious during its **lifetime** because of bugs exploited by attackers.

Sandbox properties:

- Follow the least privilege principle
- Innocuous and composable security policies

What is Landlock?

Landlock is the first Mandatory Access Control available to **unprivileged** processes on Linux.

It enables developers to add **built-in** application **sandboxing** to protect against:

- Untrusted applications (sandbox managers or container runtimes)
- Exploitable bugs in trusted applications (embedded policy)

Filesystem access-control

Filesystem restrictions

Access-control rights:

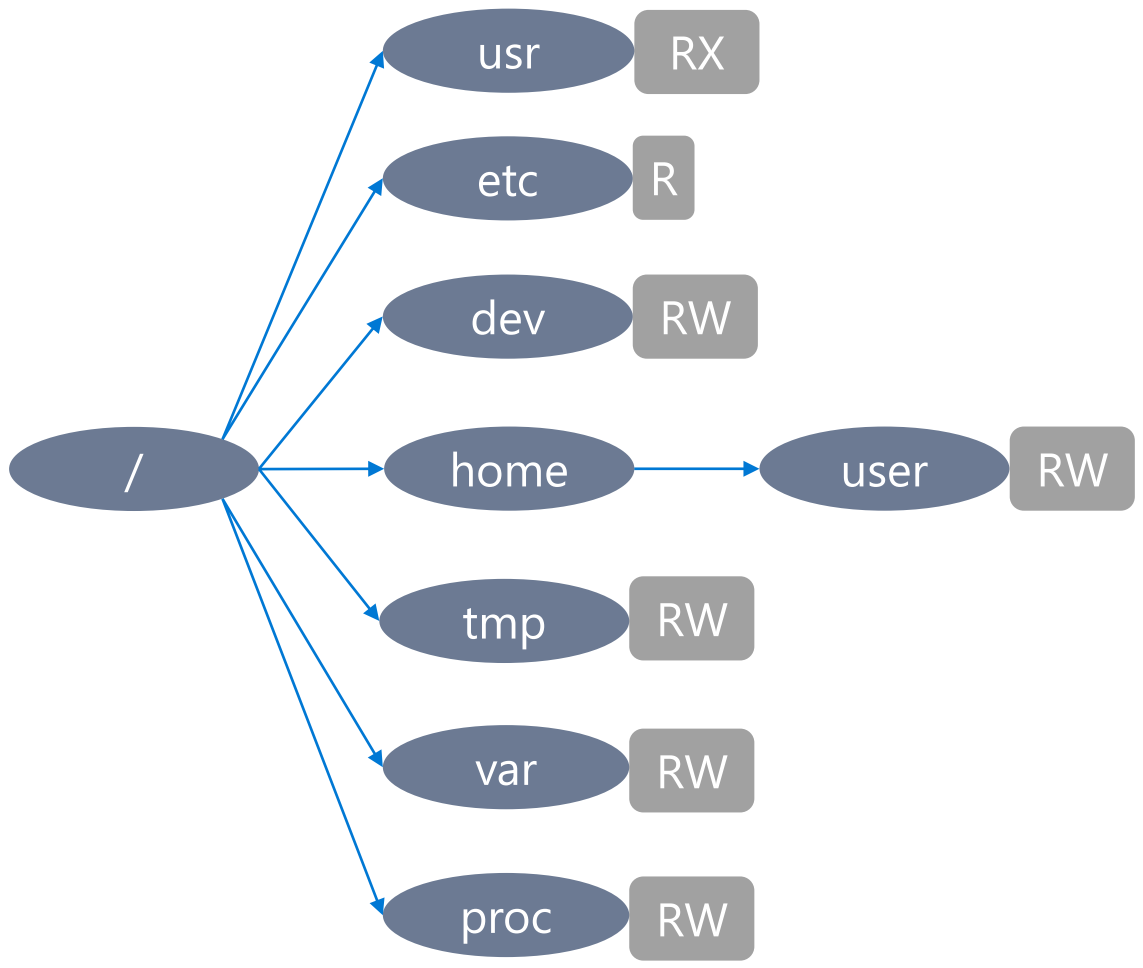
- Execute, read or write to a file
- List a directory or remove files
- Create files according to their type
- Rename or link files

File hierarchy identification: ephemeral inode tagging

Example of filesystem policy composition

Example of filesystem policy composition

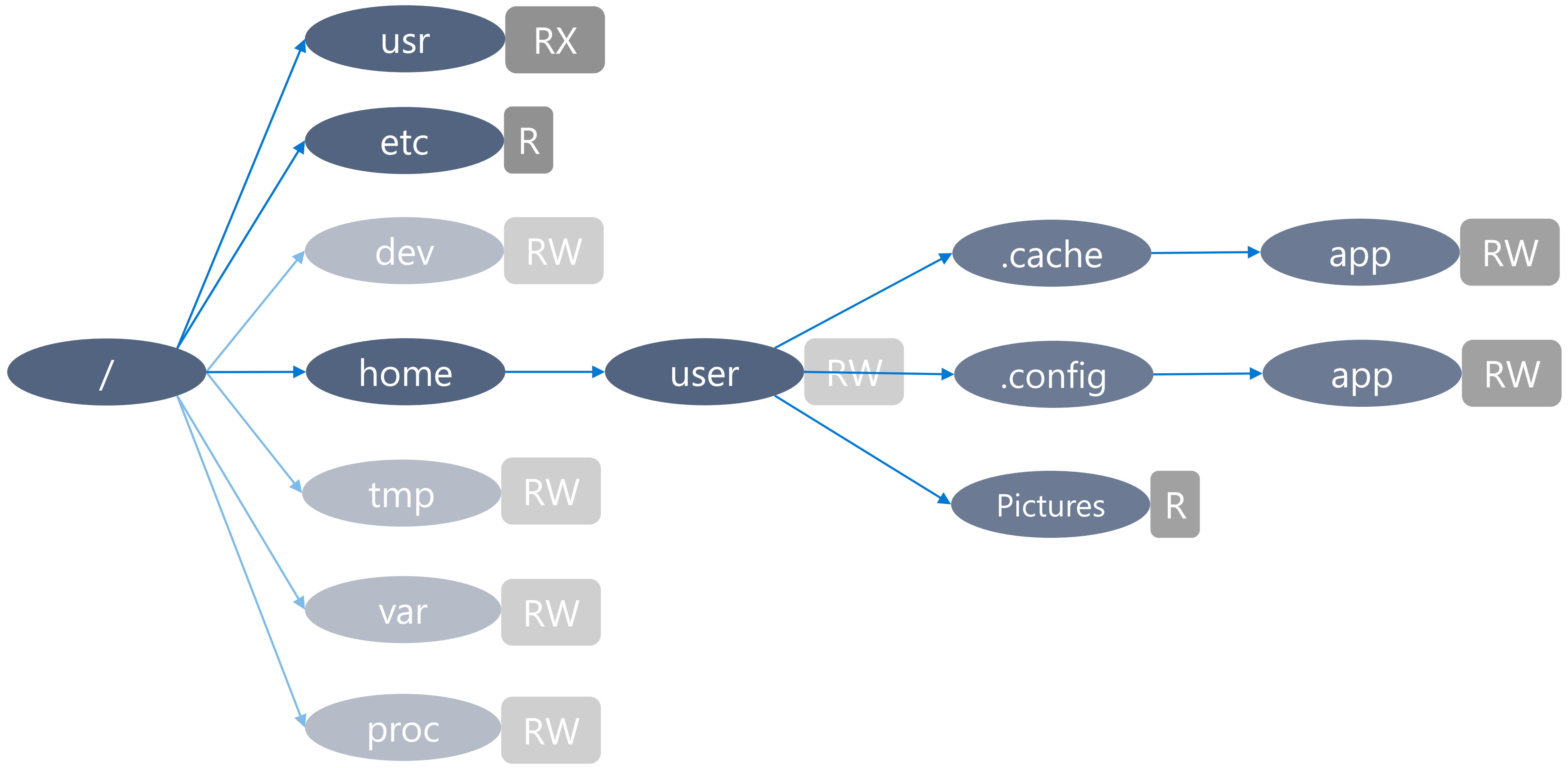
1st layer



R	Read
W	Write
X	eXecute

Example of filesystem policy composition

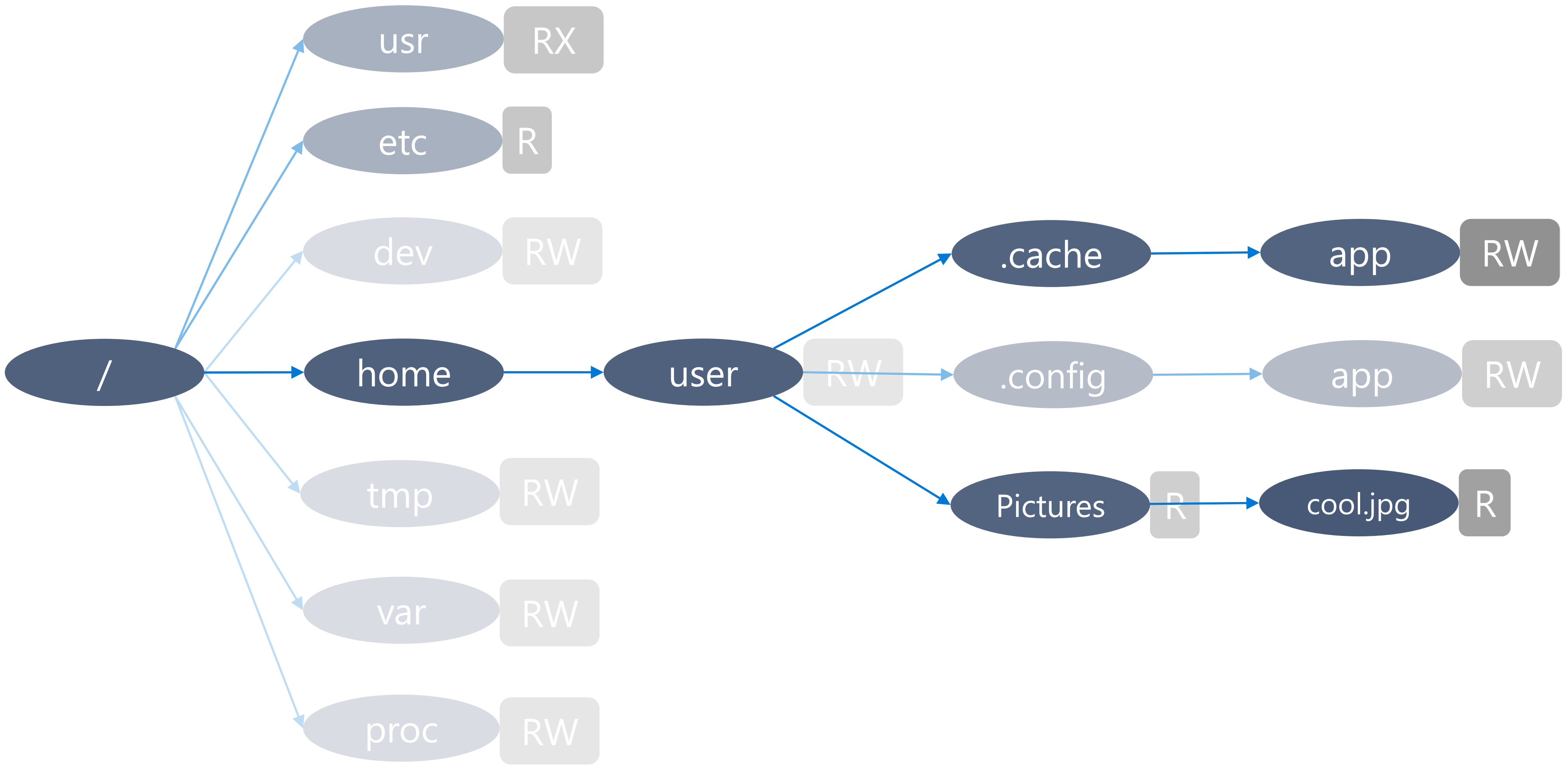
2nd layer



R	Read
W	Write
X	eXecute

Example of filesystem policy composition

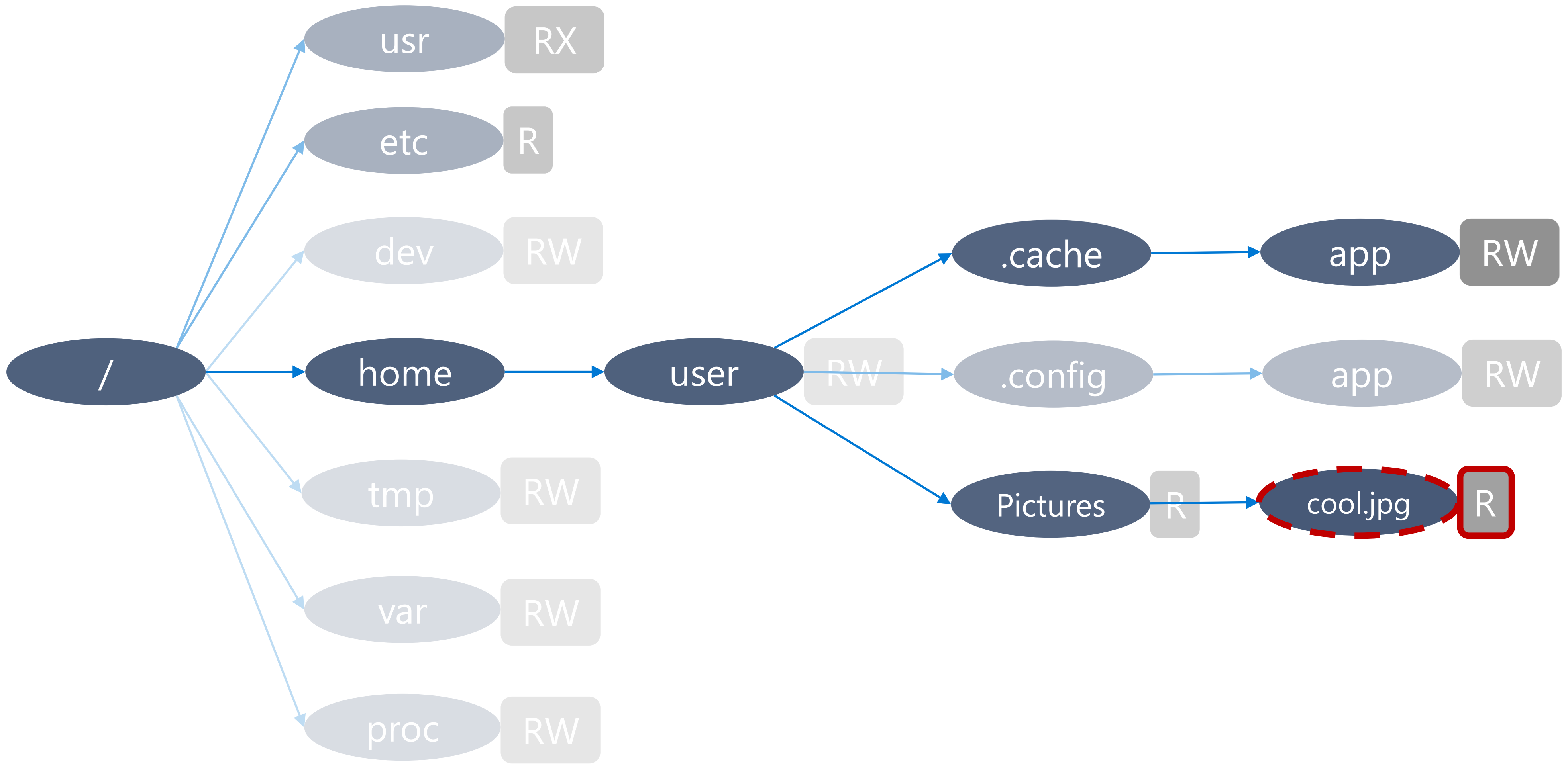
3rd layer



R	Read
W	Write
X	eXecute

Example of filesystem policy composition

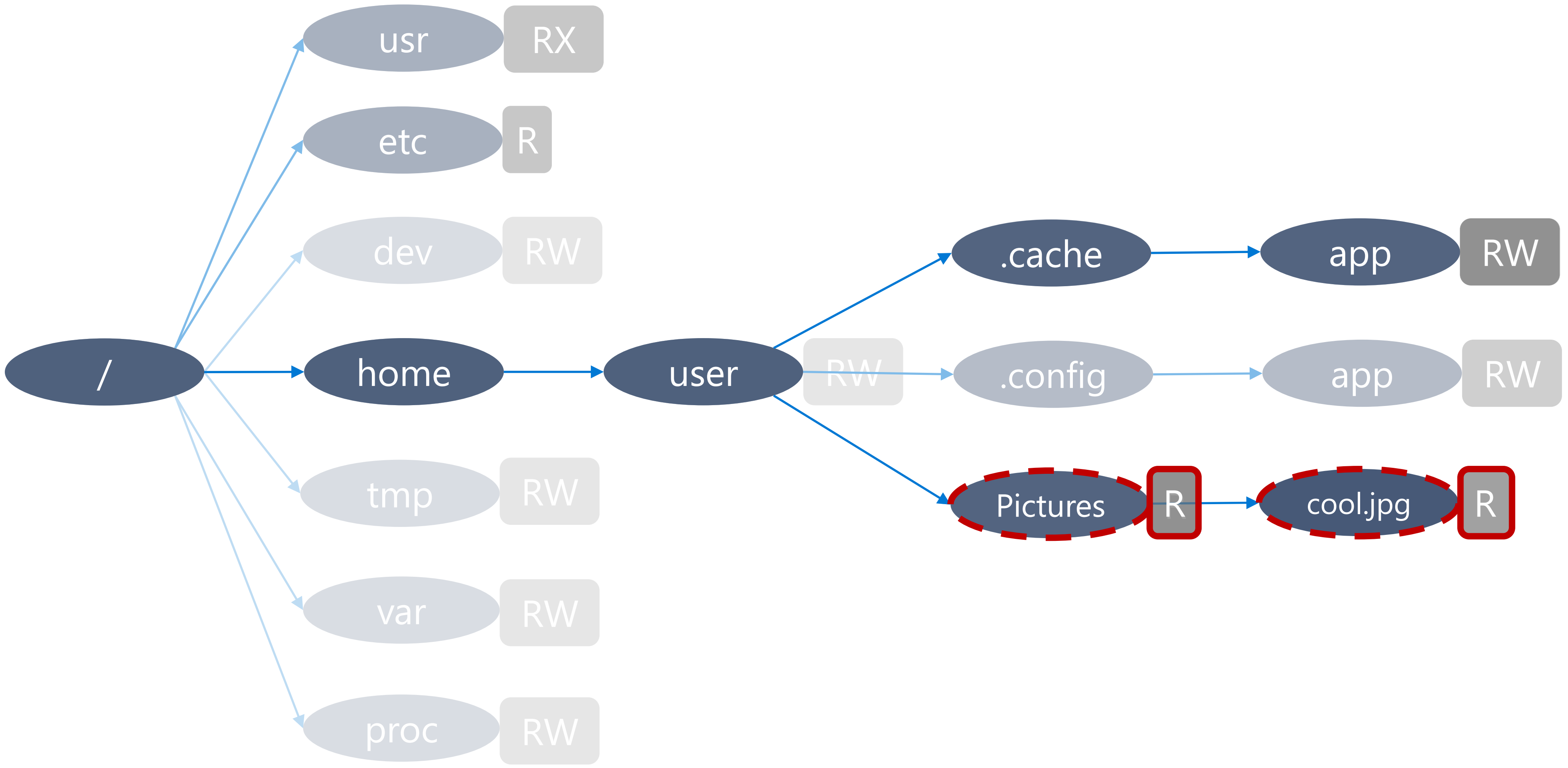
3rd layer ✓



R	Read
W	Write
X	eXecute

Example of filesystem policy composition

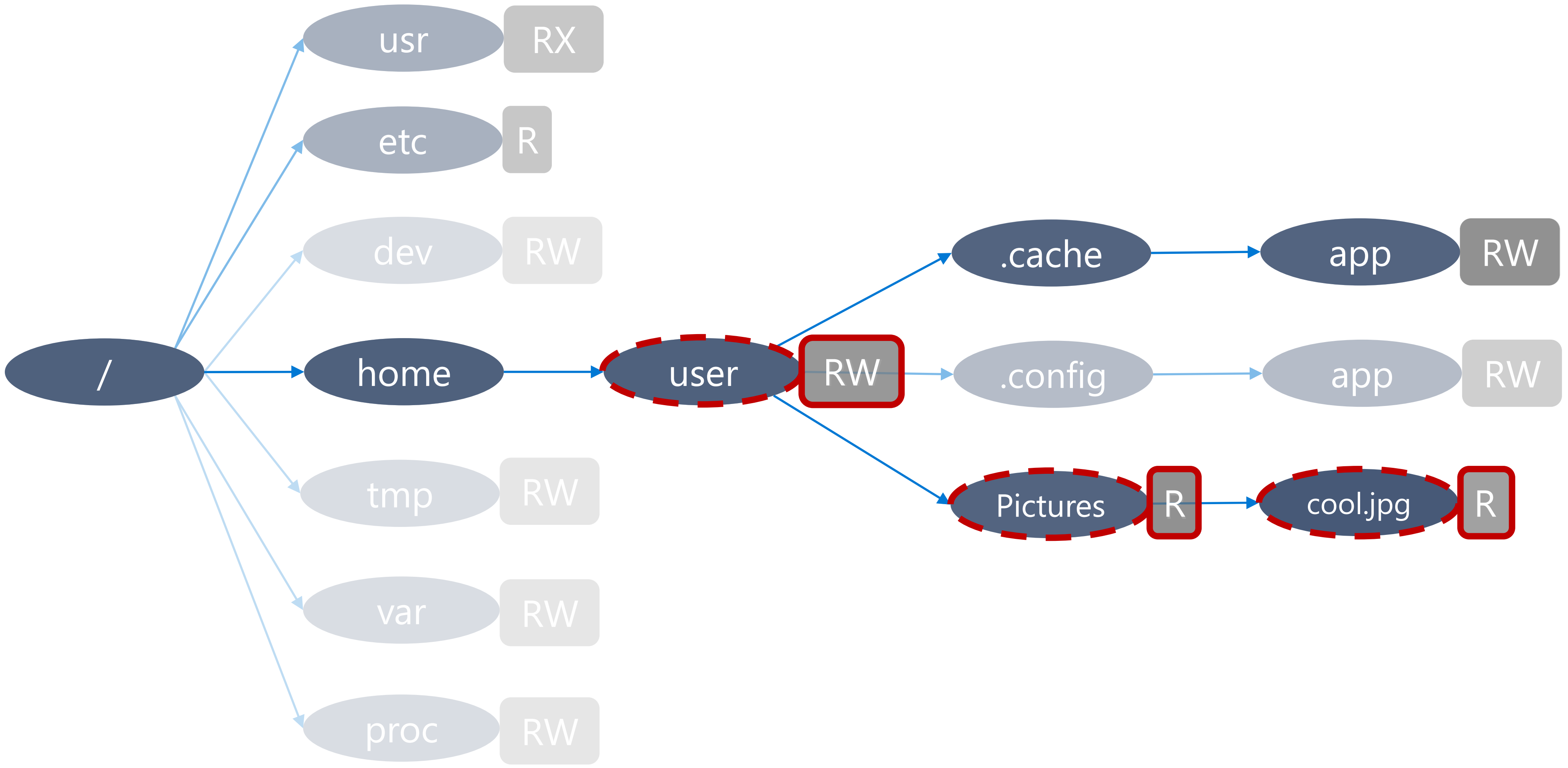
3rd layer ✓
2nd layer ✓



R Read
W Write
X eXecute

Example of filesystem policy composition

3rd layer ✓
2nd layer ✓
1st layer ✓



R	Read
W	Write
X	eXecute

Bringing access logs to sandboxes

Non-goal: Track access requests

- Not the goal of Landlock
- The LSM framework is not design to see everything, but mainly to deny actions

Other kernel features and related tools are available: e.g. trace-cmd, bpftrace

Goal: Log Landlock denials and their reasons

Help users with different use cases:

- App developers: to ease and speed up sandboxing support
- Power users: to understand denials
- Sysadmins: to look for users' issues
- Tailored distro maintainers: to get usage metrics from their fleet
- Security experts: to detect attack attempts

Challenges of dynamic policy compositions

Security policies:

- Unprivileged
- Multiple and standalone
- Nested
- Dynamic

What logs should enable

- Identify denied access requests and their reasons
  - Most relevant Landlock domain: youngest
  - Relevant access rights: those denied by this domain
- Identify domain hierarchy
- Follow the lifetime of rulesets and domains

Not available to unprivileged users

Relying on the Linux audit mechanism

Demo

What’s next?

Next patch series

Similar to SECCOMP_FILTER_FLAG_LOG, SECCOMP_RET_LOG, and /proc/sys/kernel/seccomp/actions_logged

What to expect from the next patch series? New syscall flags:

- For landlock_create_ruleset() to opt-in for logging ruleset-related and domain-related use
- For landlock_add_rule() to opt-in for logging this rule if it granted the requested access
- For landlock_restrict_self() to opt-in for
  - not log anything
  - handle a **permissive mode** to log actions that would have been denied: very useful to build a sandbox

Future work

Enable processes to get useful Landlock domain information thanks to a **new filesystem**:

- Custom view per domain to introspect nested domains (like /proc/self)
- Need to be careful about IDs:
  - Unique (and then global) IDs would be useful to tie to other views and logs
  - Should not leak information from parent or sibling sandboxes: not sequential IDs
  - No race condition

Missing CRIU support

Being able to efficiently restore Landlock states, especially Landlock rulesets and domains:

- Filesystem rules (file descriptors)
- **IDs**

Proposal:

- File system exposing internal data and being able to (safely) update IDs
- Who should have access to it?
- Could be useful for unprivileged users to debug too

Any though?

- What would you like to see (or not) in your logs?
- Which kind of tool integration could be useful to debug or audit?

See the first RFC patch series

Landlock roadmap

Ongoing and next steps:

- Add new access-control types: IOCTL, networking…
- Update and merge audit features to ease debugging
- Improve kernel performance

Contribute

- Develop new (kernel) features (e.g., new access types)
- Write new tests (e.g., kunit)
- Challenge the implementation
- Improve documentation
- **Sandbox your applications** and others'
  - Secure Open Source Rewards
  - Google Patch Rewards

Questions?

https://docs.kernel.org/userspace-api/landlock.html

Past talks: https://landlock.io

landlock@lists.linux.dev

Thank you!