Microsoft

Update on Landlock: Lifting the File Reparenting Limits and Supporting Network Rules

Linux Security Summit North America

Mickaël Salaün

2022-06-24

Update on Landlock

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 36, Arch Linux, Alpine Linux, Gentoo

This talk is about the challenge of renaming and linking files, and network access-control for the sandboxing use case.

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

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

Lifting the File Reparenting Limits

The rename(2) and link(2) use cases

Initial filesystem access-control types

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

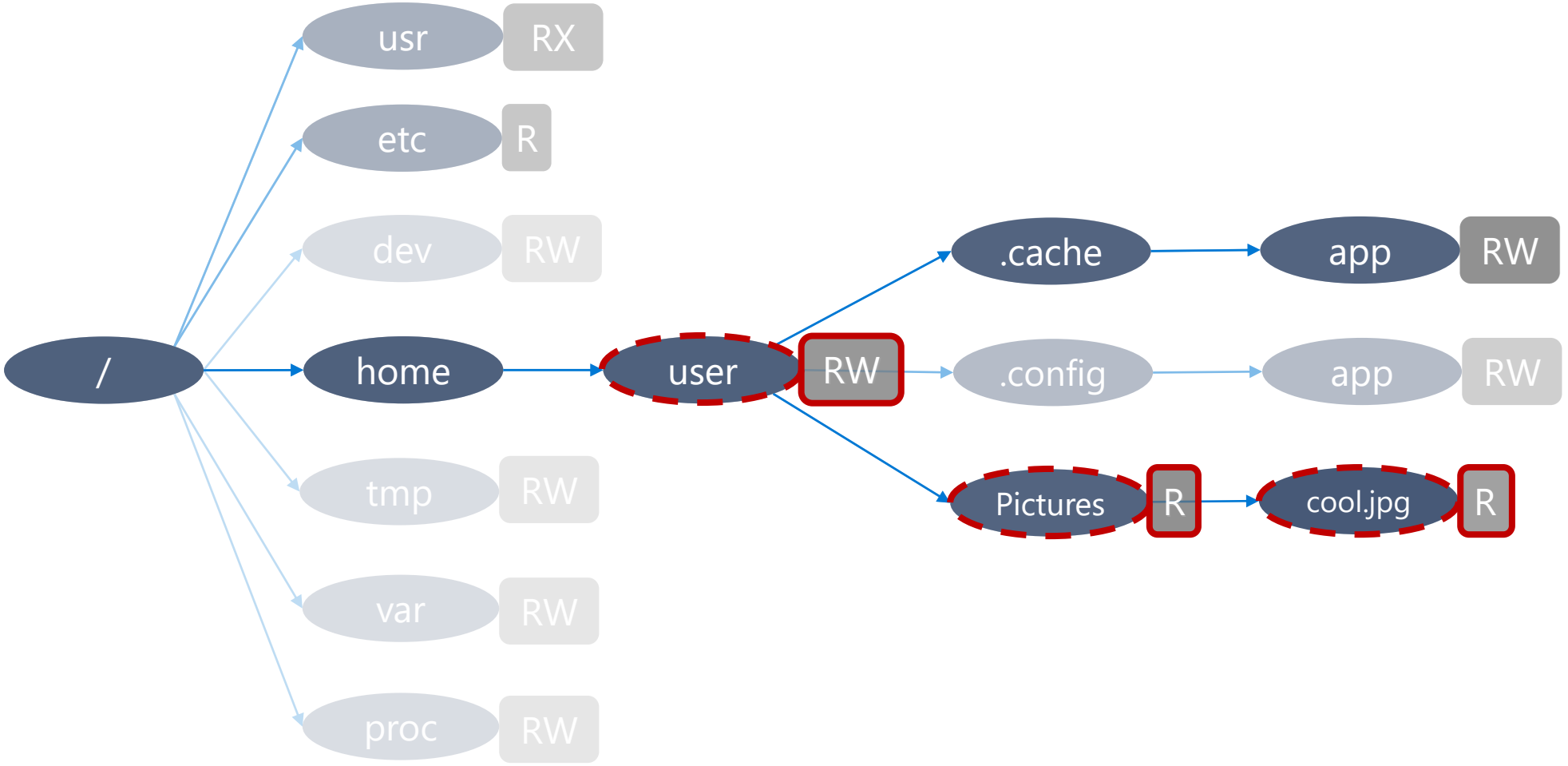
File hierarchy identification

Because of the unprivileged nature of Landlock, file hierarchy are identified thanks to ephemeral inode tagging.

Constraints for a sandboxed processes are a composition of policy layers.

Example of filesystem policy composition

3rd layer ✔
2nd layer ✔
1st layer ✔



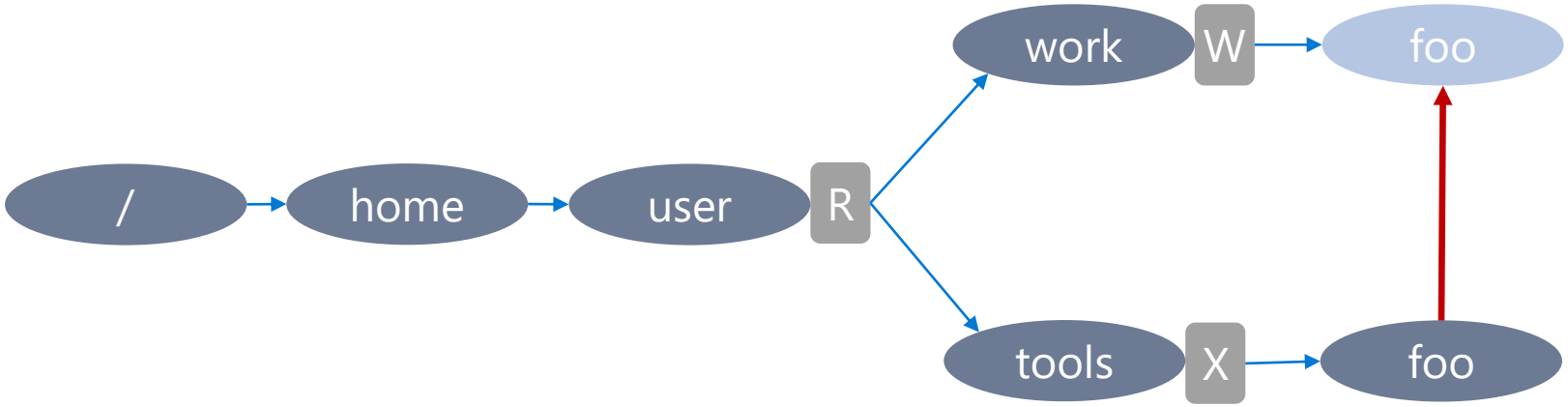
R Read
W Write
X eXecute

File hierarchy modification

Sandboxed processes could initially only link and rename files to the same directory (i.e., file reparenting is always denied).

Arbitrary renaming and linking may indeed change the security policy in unexpected ways.

What could go wrong with file reparenting?



R Read

W Write

X eXecute

```
$ ln ~/tools/foo ~/work/foo
```

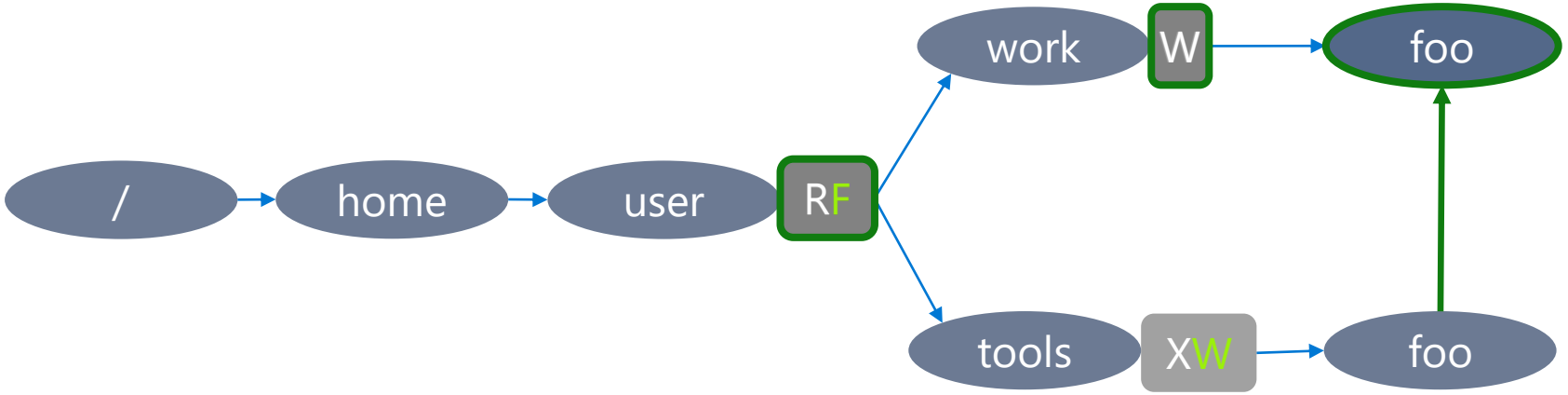
foo (content) would get RWX permissions!

A new filesystem access-control type: FS_REFER

Requirements:

- FS_REFER to link or rename a file from or to a different directory (i.e., reparent a file hierarchy).
- Linking also requires a FS_MAKE_* access right on the destination directory, and renaming also requires a FS_REMOVE_* access right on the source's (file or directory) parent.
- The destination directory hierarchy must also always have the same or a superset of restrictions of the source hierarchy.

What would be legitimate?



R Read

F reFer

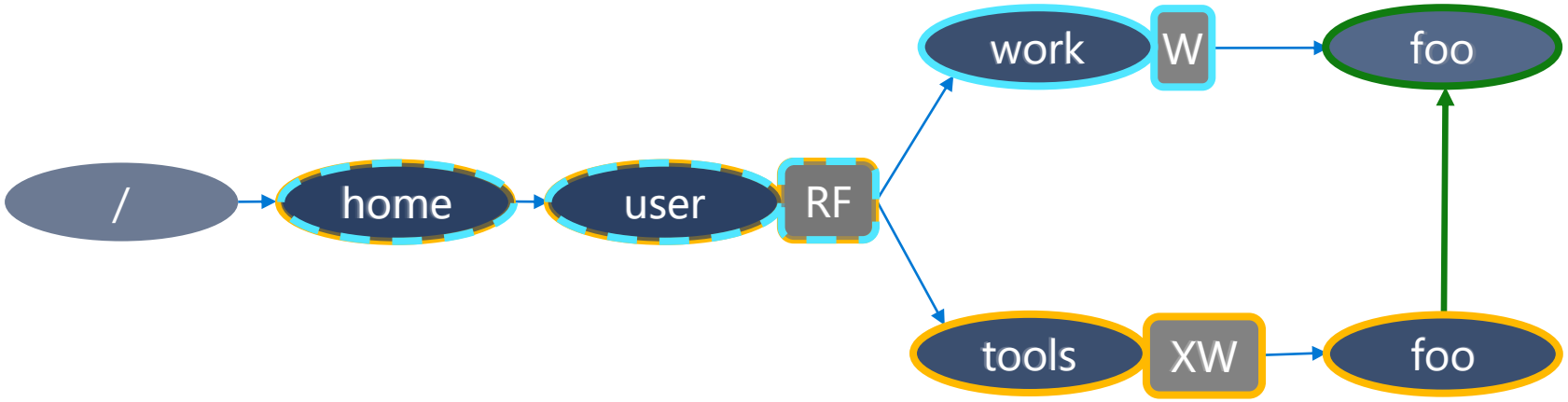
W Write

X eXecute

```
$ ln ~/tools/foo ~/work/foo
```

~/work/foo only gets RW, less than RWX

How to compare path permissions?



R Read

F reFer

W Write

X eXecute

```
$ ln ~/tools/foo ~/work/foo
```

rights(~/tools/foo, layer1) ≥ rights(~/work/foo, layer1)

Multilayer partial ordering problem

- Collect all access-rights at once and finally compare the combination of all of them.
- Impact: 16 layers max

Gracefully handling rename or link denials

If a reparenting action is denied because of FS_MAKE_* or FS_REMOVE_* deny with EACCES, otherwise with EXDEV.

A new ABI version: 2

Tied to the new FS_REFER access-control type, a new Landlock ABI version is returned by the kernel: 2

```
int abi = landlock_create_ruleset(NULL,
0, LANDLOCK_CREATE_RULESET_VERSION);
```

Useful to leverage available features in a **best-effort security** approach.

Supporting Network Rules

Main author: Konstantin Meskhidze (Huawei)

A minimal firewall

Goal: **restrict** the sandboxed processes and **protect** outside ones

A sandbox is not a system-wide firewall:

- Applications (developers) know protocols and (configured) ports ⇒ **what**
- but probably not IP addresses (e.g., local network, NAT, IPv4/IPv6) resolved with DNS ⇒ **who**

The in-review version is focused on TCP:

- New rule types:
  - TCP_CONNECT + port
  - TCP_BIND + port
- Minimal overhead

Any though?

- Is handling port range worth it?
- How to meaningfully and efficiently restrict UDP?
- What if we only handle *bind*, *recvfrom*, *sendto* and deny “unconnected” UDP?
- What about other protocols?
- What about other socket types (e.g., *vsock*)?

Demo

What’s next?

Roadmap

Short term:

- Add audit features to ease debugging
- New minimal access-control types:
  - Network
  - Process signaling
- Improve kernel performance

Roadmap

Medium term:

- Extend access-control types to address the current limitations:
  - Filesystem
  - Network
- Add the ability to follow a deny listing approach

Questions?

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

Past talks: https://landlock.io

landlock@lists.linux.dev

Thank you