



Constraints of process migrations

Michal Koutný <mkoutny@suse.com>
LPC 2026



Why migrations?

- Starting a new job
 - New container
 - New service/application/session
- Change of rules
 - Dynamic decision (cgrulesengd)
 - Effect of singular controllers (v1 hierarchies)
- Implicit migrations
 - `cgroup_update_dfl_csses()`
- Misc
 - Process (thread) granularity, not (process) groups
 - `[setns(2)]`
 - Resource transfers, proxy execution?

```
`-kubepods.slice
  |-kubepods-burstable.slice
  | |-kubepods-burstable-podff36... .slice
  | | |-cri-containerd-342f6f1eb... .scope ...
  | | `--cri-containerd-94981a771... .scope ...
  | |-kubepods-burstable-pod793a... .slice
  | | |-cri-containerd-5c1276c27... .scope ...
  | | `--cri-containerd-2c8502882... .scope ...
  `--kubepods-besteffort.slice
    |-kubepods-besteffort-pod12... .slice
    | |-cri-containerd-29012107... .scope ...
    | `--cri-containerd-03a7d1e7... .scope ...
```



Migration rules

- v2 vs v1 differences

- dst owner
- object owner
- (src,dst) owner
 - nsdelegate
- .can_attach()

```
-.slice
├user.slice
│   ├──user-1000.slice
│   │   ├──user@1000.service ...
│   │   └session.slice
│   └user-1001.slice
│       └user@1001.service ...
│           └session.slice
```

```
-.slice
├user.slice
│   ├──user-1000.slice
│   │   ├──session-1.scope
│   │   └session-2.scope
```

```
$ ls -l user@1000.service/ | grep -v "^d" | cut -c1-10,15-22|sort|uniq -c| sort -rn
```

```
19 -rw-r--r--root
18 -r--r--r--root
 5 -rw-r--r--mkoutny
 1 --w-----root
```

```
cgroup.procs
cgroup.subtree_control
cgroup.threads
memory.oom.group
memory.reclaim
```

```
cgroup.kill
```

.can_attach()

- pids_can_attach()
- cpuset_can_attach()
 - cpumask_empty()
 - 🤖 security_task_setscheduler() ([\[PATCH v4\] security: Expand task_setscheduler LSM hook](#))
 - ...
- cpu_cgroup_can_attach()
 - sched_rt_can_attach()
 - scx_cgroup_can_attach() → eBPF

Challenges of migration

- PAM
- Asynchronicity
- Self vs 3rd-party migration
- Delegation (only) via tree
- Internal group constraint
 - Entering a container
 - Threaded subtrees

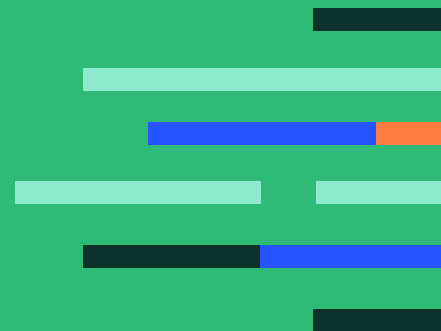
Proposed approach

- [PATCH 2/4] cgroup: add bpf hook for attach

```
__weak noline int bpf_lsm_cgroup_attach(struct task_struct *task,  
                                         struct cgroup *src_cgrp,  
                                         struct cgroup *dst_cgrp,  
                                         struct super_block *sb,           // of cgroup.procs  
                                         bool threadgroup,  
                                         struct cgroup_namespace *ns);    // of caller/"writer"
```



Discussion



Topics

- confused deputy (root), setup & seal
- uids vs pids assignments
- xattrs
- mandatory access control
- single writer rule / package managers

Expected outputs

- The new specific BPF hook
 - Clarified arguments and output (AND/OR/overwrite)
- Stick with potential of cgroup FS-based permissions
- Unification/connection with other existing LSM hooks
- Mechanism by an analogy
- Yet another proposal(s)



Thank you!

