Isolated user namespaces & cgroupfs

Stéphane Graber
Owner, Zabbly
stgraber@stgraber.org

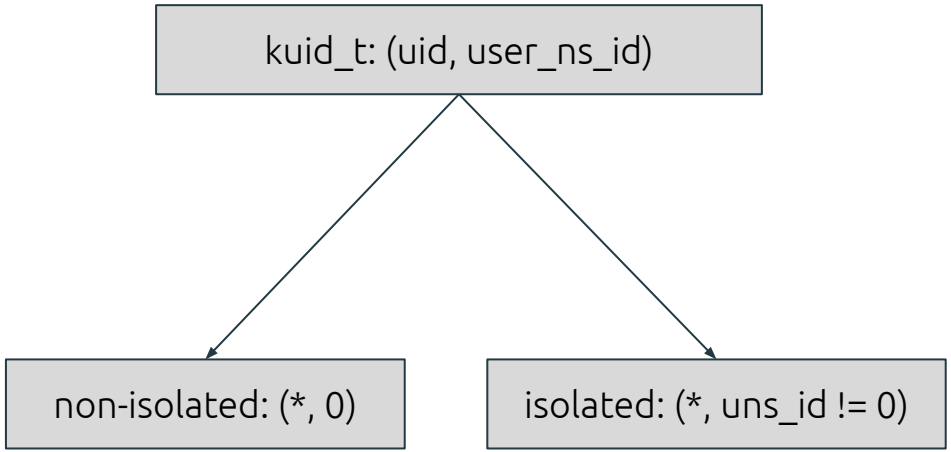
Aleksandr Mikhalitsyn
Software engineer, Canonical
aleksandr.mikhalitsyn@canonical.com

Intro

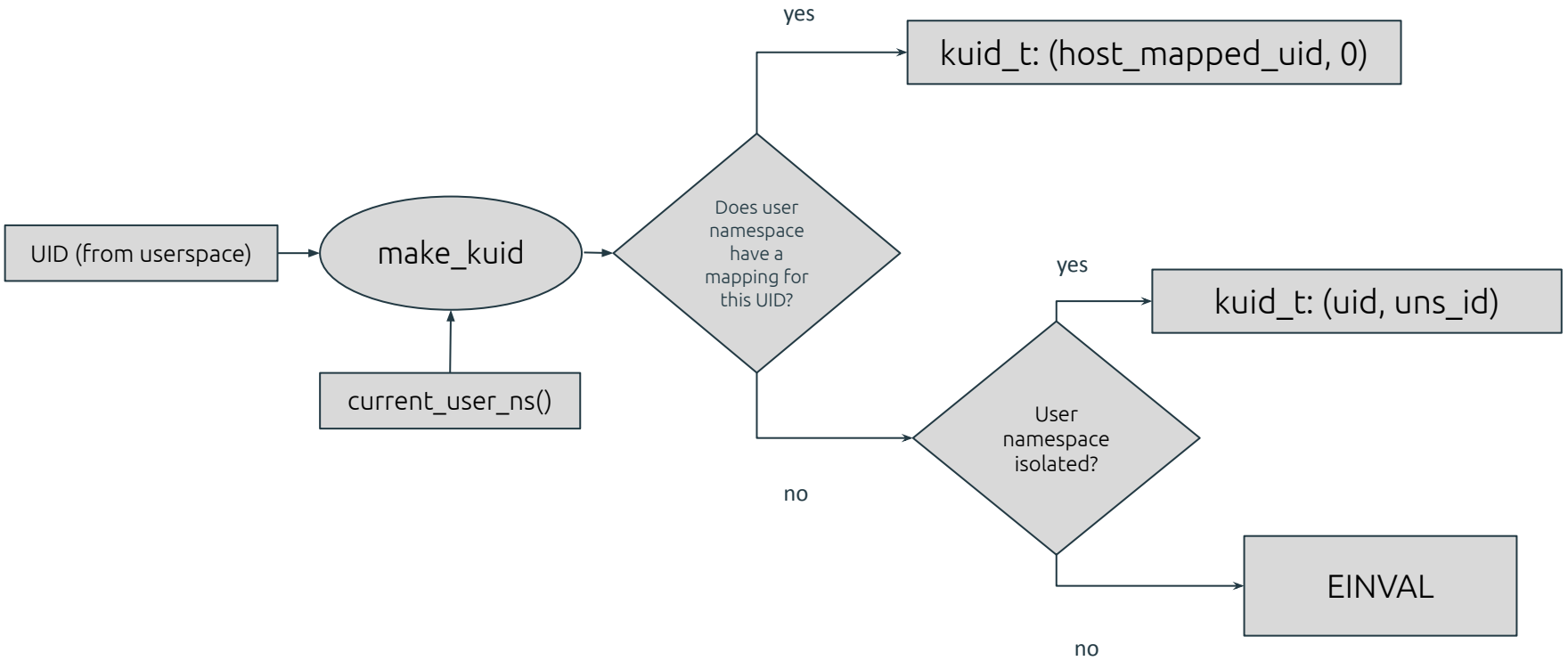
Quick recap

- Currently, each usable UID (or GID) **must** have a corresponding UID on the host
- We use 32-bit-wide type to represent UID
- We **may** want to have different containers on the machine to have non-intersecting UID ranges
- ⇒ we can not provide all the containers with full 32-bit UID space

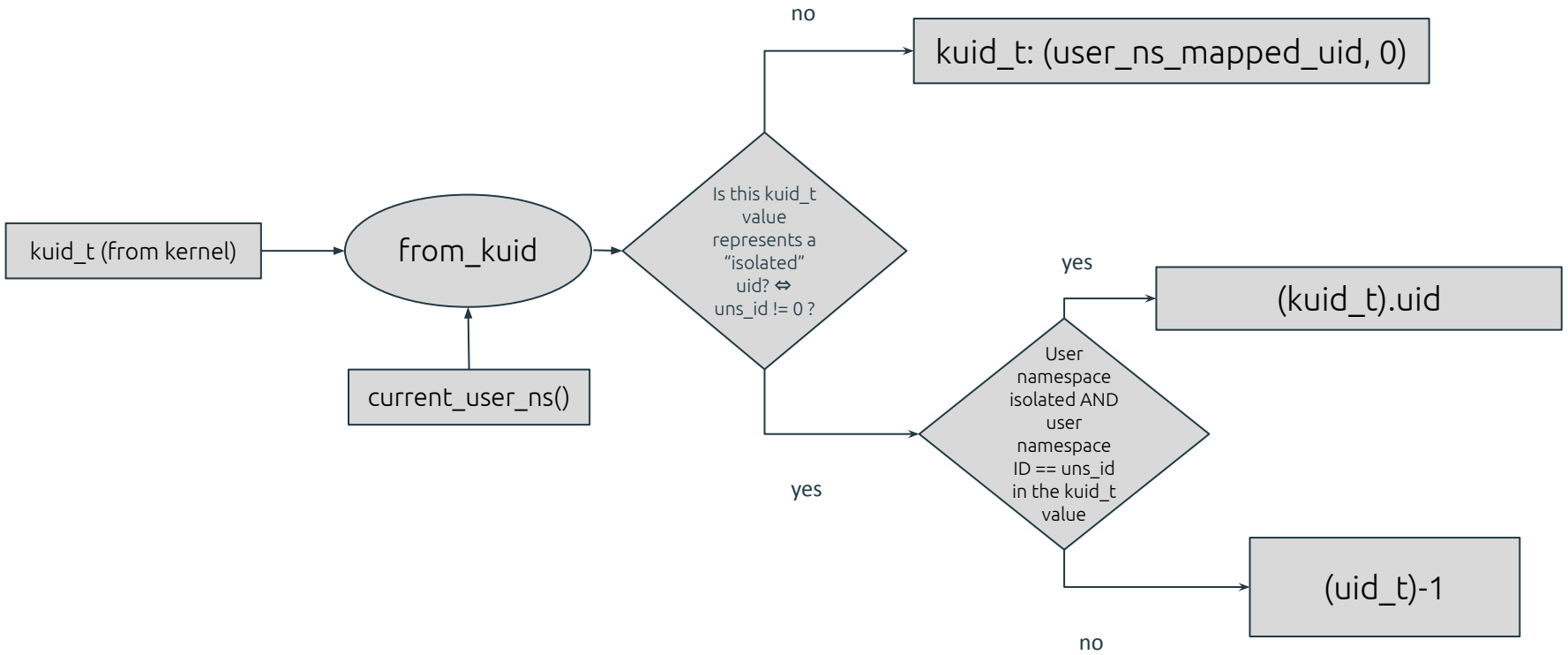
Let’s make k{u,g}id_t to be 64bit



Mapping procedure (setuid syscall)



Inverse mapping procedure (getuid syscall)



overflowuid case

- When from_kuid() fails to map (kuid_t) back to the userspace (uid_t) type it returns (-1)
- In many places we use from_kuid_munged() function which replaces this (-1) with an overflowuid value (usually, 65534, but configurable through /proc/sys/kernel/overflowuid)
- For isolated user namespace, instead of going with the overflowuid we returning a UID of a user namespace owner

Problems

- Need to integrate with filesystems
  - Use VFS idmaps or container-ized filesystems
  - cgroupfs!
- Nesting
- Cross-container interaction
  - SCM_CREDENTIALS between two isolated containers

Why cgroupfs is so special?

- A cgroup object is a host-level thing
  - each cgroup has an owner (stored in a *kernfs_iattrs* structure)
  - it ***must*** be controllable from the host (and parent userns ?)
- A superblock remains the same even when cgroup namespace is used
  - ⇒ one *struct inode* and *struct kernfs_node* per cgroup

File system’s idmapping

- uid_t i_uid_read(const struct inode *inode)
- void i_uid_write(struct inode *inode, uid_t uid)
  - inode->i_uid = make_kuid(inode->i_sb->s_user_ns, uid)
  - There is a check that prevents writing an “unmappable” uid to the inode->i_uid (see vfsuid_has_fsmapping() function)

Okay, what's about task_struct then?

- task_struct has struct cred
  - ⇒ user namespace ⇒ we can do permission checks based on capabilities and not UID/GIDs!
  - see, for example, kill_ok_by_cred() function
- cgroup has no creds attached to it!
  - Instead, it has struct kernfs_node which keeps k{u,g}id_t values (through struct kernfs_iattrs)
- To conclude, for task_struct, owner UID/GIDs are not playing a fundamental role

Ideas

- Attach struct cred (or struct user_namespace) to a cgroup
- Introduce a concept of multiple owners for a cgroupfs files
  - A set of kuid_t values depending on the superblock
    - We need multiple superblocks!
    - Instead of one struct kernfs_iattrs per kernfs_node we need an array/hashtable with them
    - In the initial user namespace we take cgroup->kn->iattr[0].ia_uid

What can we do?

1. During a cgroup namespace creation from an isolated user namespace we can change an ownership of a cgroup to an uid=0 in the isolated user ns
2. We need to introduce a separate (per-cgroupns) superblocks to make VFS layer happy when user does chown() inside the user namespace to an isolated UID/GIDs

Linux kernel patches

Thank you!
It’s time for a discussion

Stéphane Graber
Owner, Zabbly
stgraber@stgraber.org

Aleksandr Mikhalitsyn
Software engineer, Canonical
aleksandr.mikhalitsyn@canonical.com

Links

1. More flexible user namespaces
   https://fosdem.org/2024/schedule/event/fosdem-2024-2987-more-flexible-user-namespaces/
2. User namespaces with host-isolated UIDs/GIDs
   https://lpc.events/event/17/contributions/1569/
3. Isolated dynamic user namespaces
   https://lpc.events/event/7/contributions/836/
4. Simplified user namespace allocation
   https://lpc.events/event/11/contributions/982/