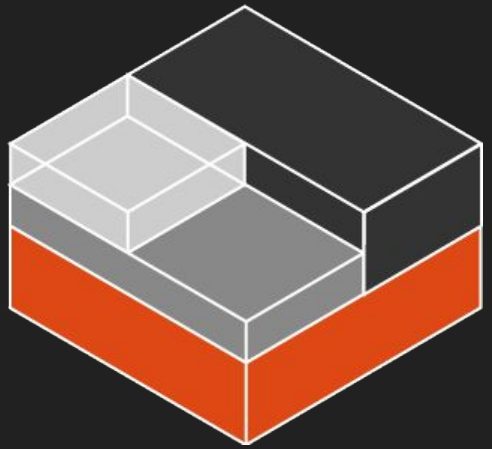
Linux Device Management

Tuebix
Tübingen, Germany



Christian Brauner
christian@brauner.io
@brau_ner
https://brauner.github.io

whoami

Devices

- "everything is a file": mimic I/O model of regular files
- can have optimized I/O methods (`splice()`, `sendmsg()` etc.)
- there are different types of devices
- created with the `mknod()` syscall
  - `int mknod(const char *pathname, mode_t mode, dev_t dev);`
  - rough call chain is `vfs_mknod()` and then down to the fs specific method (e.g. `ext4_mknod()`)
- Linux standard devices: `/dev/{full,null,random,tty,urandom,zero}`

devtmpfs

- pseudo filesystem
- mounted at `/dev`
- kernel maintained
- ```
  modprobe kvm_intel
  ls -al /dev/kvm
  rmmod kvm_intel
  ls -al /dev/kvm
  modprobe kvm_intel
  ls -al /dev/kvm
  ```


  ```

udev

- userspace part of device management
- implementations: `systemd-udevd`, `eudevd`, `ueventd`
- manages permissions, symlinks, persistent device naming

uevents

- interesting bits are located in lib/kobject_uevent.c
- `int kobject_uevent_env(struct kobject *kobj, enum kobject_action action, char *envp_ext[])`
- `static int kobject_uevent_net_broadcast(struct kobject *kobj, struct kobj_uevent_env *env, const char *action_string, const char *devpath)`

  This function won't be present from 4.18 onwards.
- KEY=<value> messages separated by \0-bytes using the following schema:

  `<action>@<devpath>\0ACTION=<action>\0DEVPATH=<devpath>\0SUBSYSTEM=<subsystem>\0...\0SEQNUM0=<seqnum>`

Netlink

- socket protocol
- `NETLINK_KOBJECT_UEVENT`
  - unprivileged socket protocol, i.e. everyone can listen to uevent messages

Containers: A userspace fiction

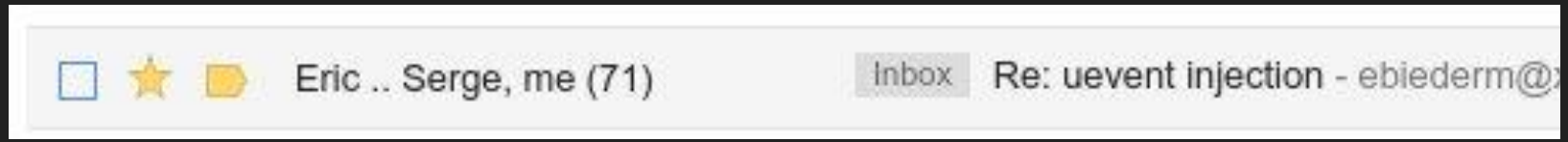
https://www.youtube.com/watch?v=wiFWBhmFyOM

Containers, Devices, (time-permitting also SR-IOV)

- container do allow for easy device passthrough but few problems:
  - `devtmpfs` is not namespaced
  - `devtmpfs` not mountable in non-init user namespaces
  - missing `CAP_MKNOD`
  - user namespaces
- privileged actions for unprivileged containers
  https://lwn.net/Articles/756233/

Namespacing Devices

- kernel solution: *namespacing devtmpfs and kobjects*
- userspace solution: *namespace uevents/uevent injection*
- mails required just to agree on an initial design:



Uevent Injection

- Things we can already do:
  - device injection: devtmpfs from userspace
- Status Quo
  - missing isolation: uevents broadcast into all network namespaces
  - wrong credentials: uevents ignored by udev
- Status new
  - isolation: by owning user namespace of the network namespace a uevent socket resides in
  - credentials: per user-namespace credentials
  - injection: sending uevents from userspace

Future Work

- namespacing devtmpfs: #controversial
- seccomp from userspace
- remove global locking
  - global lock on list of list
  - partially done in

Demo Time