Embedded data structures and lifetime management

LinuxCon NA, August 24th 2016

Shuah Khan
Samsung Open Source Group
shuahkh@osg.samsung.com

Abstract ...

Embedded data structures are a common occurrence in Linux Kernel code. When they include multiple ref-counted objects with different lifetimes, use-after-free errors can creep in if the data structure is released prematurely. In this talk, Shuah will share her experience solving use-after-free errors in Media Controller ioctls. In addition, she will talk about the benefits of pro-actively testing for lifetime problems. Knowing the best practices about what to do and what not to do can help avoid days of debugging problems that result in inadequate fixes, and will result in better code quality that stands the test of time.

Agenda ...

- Embedded Data Structures
- Lifespan Requirements
- Resource Usage
- Pro-actively testing
- Reproducing the problem
- Fix
- Summary

Embedded Data Structures ...

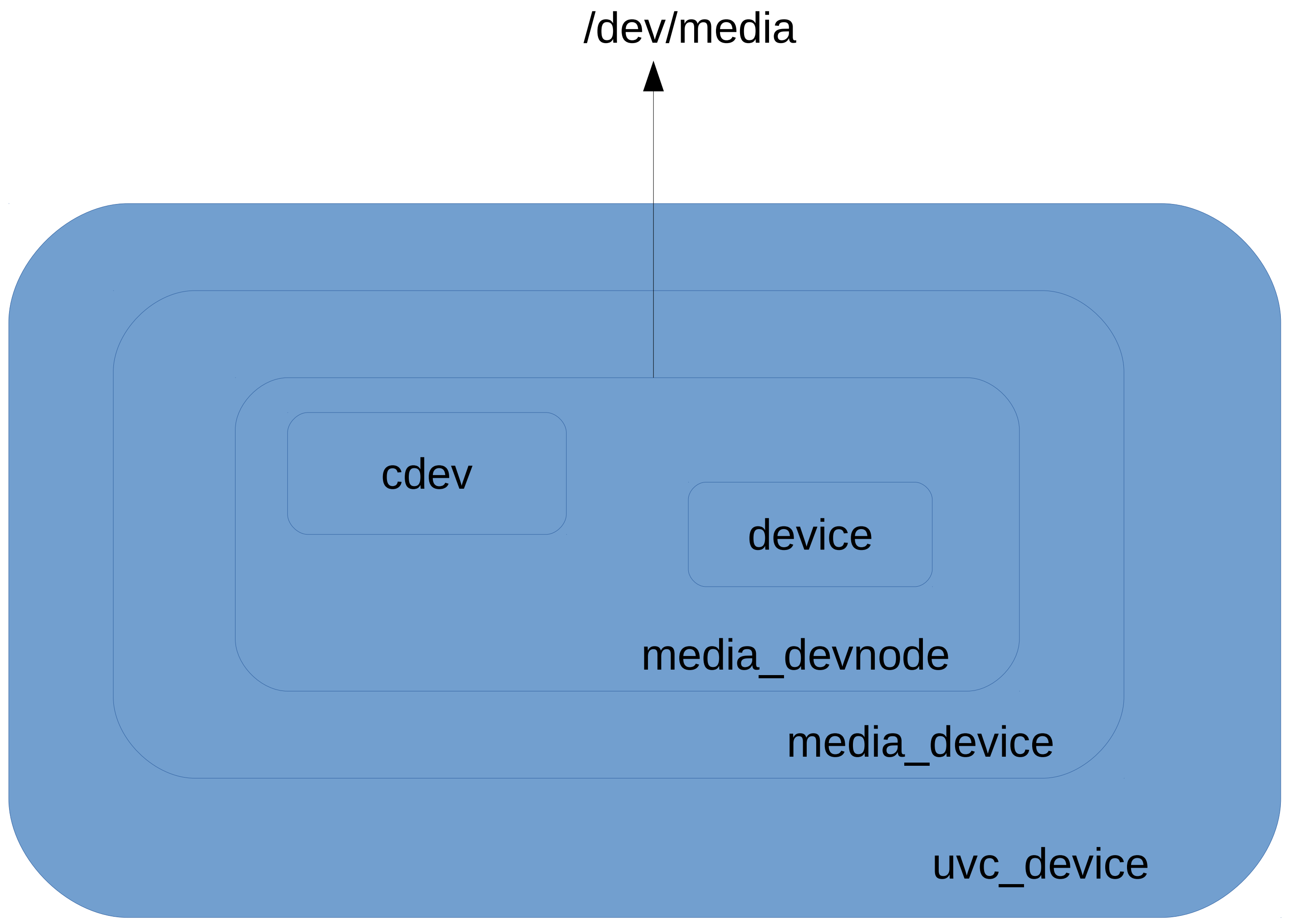
- Structure A embeds structure B
- Structure B embeds structure C
- Structure C embeds structures D and E

Okay! Why does that matter ?

Resources (struct foo memory) ...

- Do the resources:
  - include ref-counted objects?
- Do the resources have:
  - identical lifespans?
  - different lifespans overall or only in some scenarios?

Embedded Data Structure Examples ...


/dev/media
cdev
device
media_devnode
media_device
uvc_device

```
struct uvc_device {
    struct media_device mdev;
    ………
}
```

```
struct media_device {
    struct media_devnode devnode;
    ........
}
```

```
struct media_devnode {
    struct device dev;
    struct cdev cdev;
    ……..
}
```

```
struct device {
    struct kobject kobj;
    .........
}
```

```
struct cdev {
    struct kobject kobj;
    ………
}
```

Lifespan Requirements ...

- Application doesn’t have device files open and
  - driver is unbound (unbind) or
  - driver is removed (rmmod) or
  - device is removed

Lifespan Requirements ...

- Driver does unregister and cleanup
  - All embedded resources (media_device, media_devnode, device, and cdev) get released when uvc_device is free'd
  - Safe scenario – no issues.

Lifespans requirements – What happens if ...

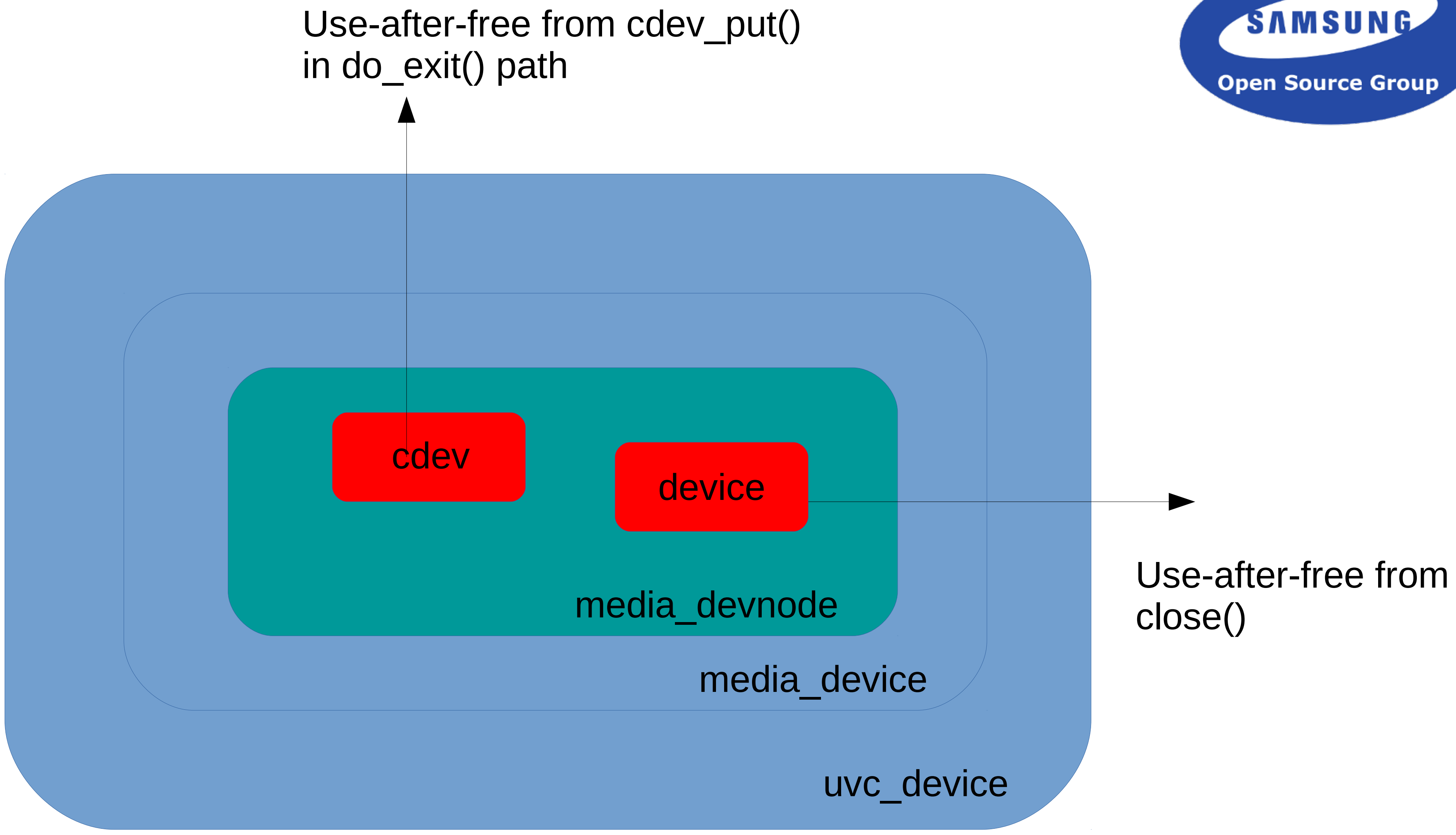
- Application has device file open with ioctl or syscall in progress and
  - driver is unbound (unbind) or
  - device is removed
- driver is removed (rmmod)
  - this is is not a problem. rmmod fails with device in use.

Resource Usage ...

- media_devnode→device is use until application closes the device file
- media_devnode→cdev is in use until application exits

What happens ...

- Driver does unregister and cleanup
  - All embedded resources (media_device, media_devnode, device, and cdev) get released when uvc_device is free'd
- Application closes the file and exits after driver completes cleanup
- Use-after-free errors on free'ed resources that are in use


Use-after-free from cdev_put()
in do_exit() path
cdev
device
media_devnode
media_device
uvc_device
Use-after-free from
close()

Root Cause ...

- Resources with differing lifespan requirements embedded in the parent resource
- Resources that are use get released prematurely

Pro-actively Testing ...

Reproducing the problem ...

- Writing a test to reproduce the problem reliably is crucial
- Run a test that opens the device file and does ioctls calls in a loop. While this test is running, unbind the driver
- Use-after-free errors will occur either during an ioctl or when application exits
- Refer to tests and test procedure to find and fix this problem in tools/testing/selftests/media_tests

What’s the fix ...

Fix ...

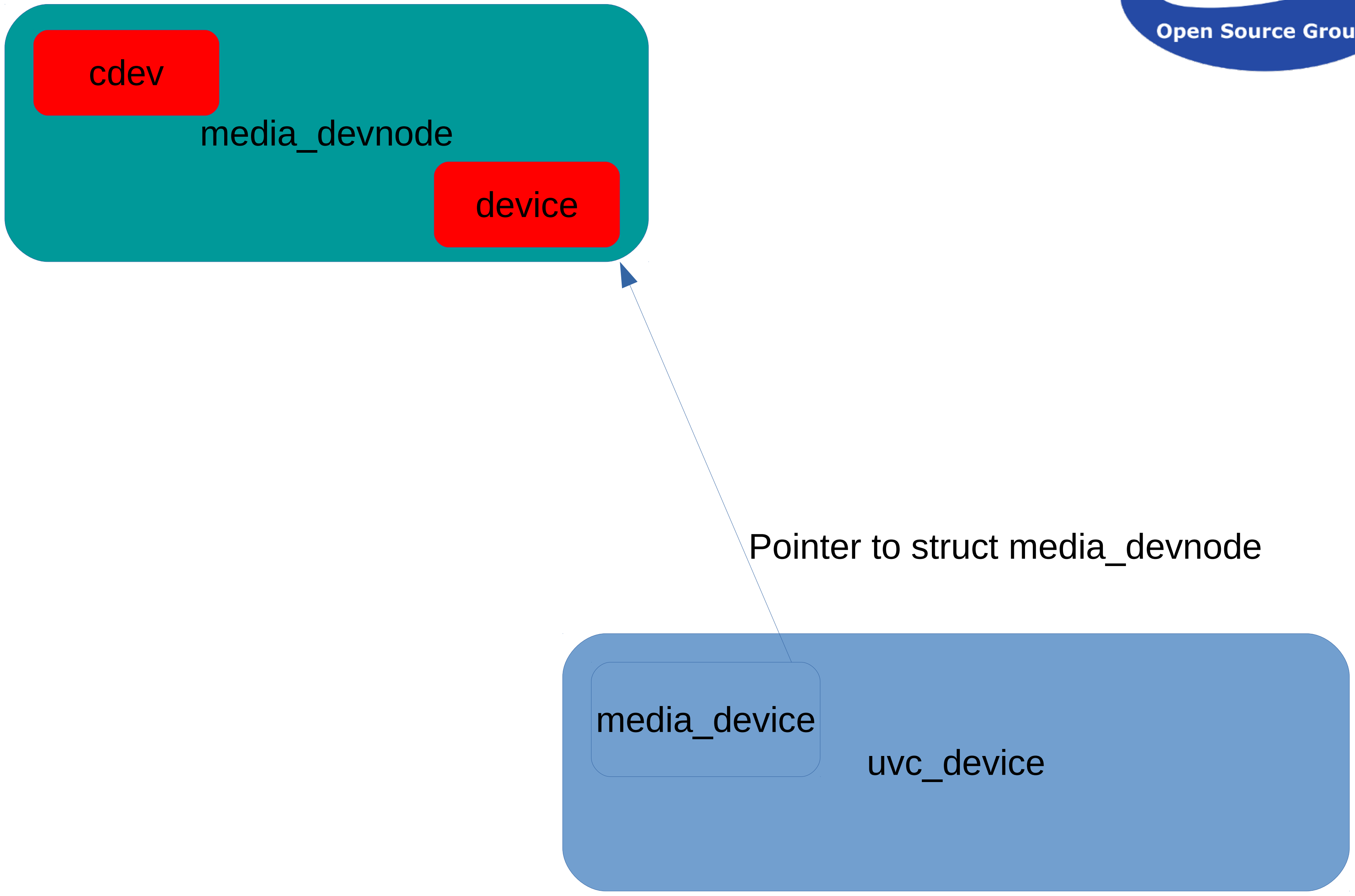
- Dynamically allocate the resource(s) with different lifespans
- Ensure resource that embeds cdev doesn’t get released before cdev_del()

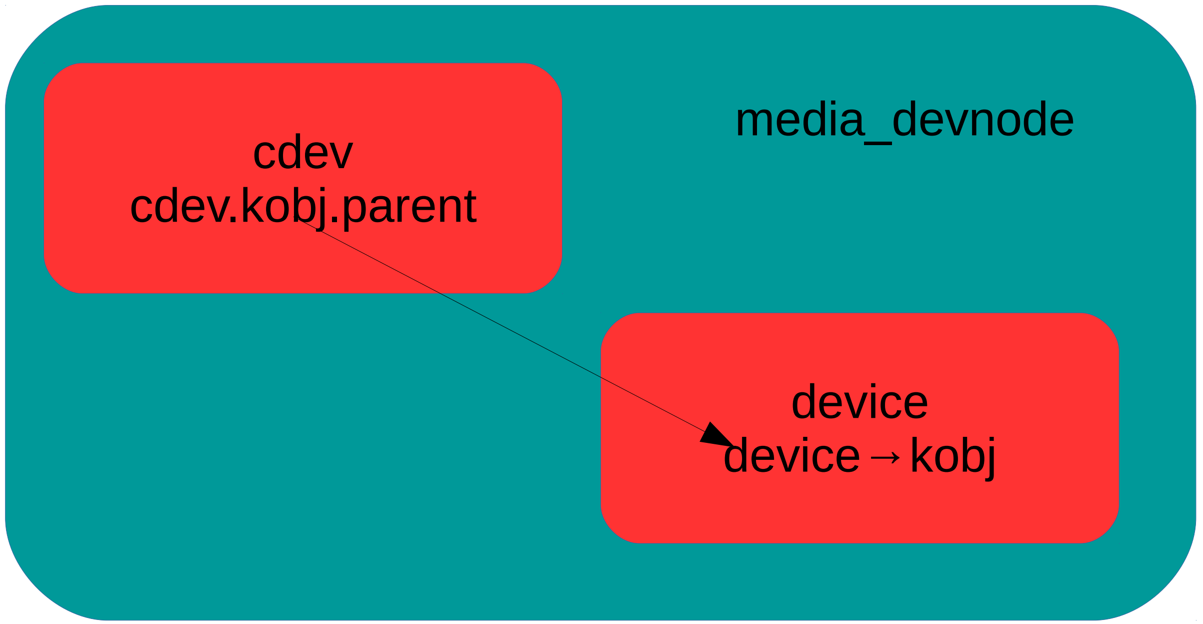
Fix ...

- Dynamically allocate media_devnode
  - detaches media_devnode lifetime from media_device
  - media_devnode will not be free'd when driver does its cleanup and frees uvc_device along with media_device

Fix ...

- Set devnode struct device kobj as the cdev parent kobject
  - media_devnode->cdev.kobj.parent = &media_devnode->dev.kobj;
- cdev_add() gets a reference to dev.kobj
- cdev_del() releases the reference to dev.kobj
- ensuring that the devnode is not deallocated as long as the application has the device file open


cdev
media_devnode
device
Pointer to struct media_devnode
media_device
uvc_device


media_devnode
cdev
cdev.kobj.parent
device
device→kobj

Current status ...

- Problem fixed in media-core in 4.8-rc1
- Drivers affected - uvcvideo, em28xx, and au0828

Summary ...

- Pay attention to resources lifespan requirements
- Do not embed resources with differing lifespan requirements
- Pro-actively test resource lifetimes

Questions ?

Thank You!