Device Whitelist Controller

1. Description

Implement a cgroup to track and enforce open and mknod restrictions on device files. A device cgroup associates a device access whitelist with each cgroup. A whitelist entry has 4 fields. 'type' is a (all), c (char), or b (block). 'all' means it applies to all types and all major and minor numbers. Major and minor are either an integer or * for all. Access is a composition of r (read), w (write), and m (mknod).

The root device cgroup starts with rwm to 'all'. A child device cgroup gets a copy of the parent. Administrators can then remove devices from the whitelist or add new entries. A child cgroup can never receive a device access which is denied by its parent.

2. User Interface

An entry is added using devices.allow, and removed using devices.deny. For instance:

```
echo 'c 1:3 mr' > /sys/fs/cgroup/1/devices.allow
```

allows cgroup 1 to read and mknod the device usually known as /dev/null. Doing:

```
echo a > /sys/fs/cgroup/1/devices.deny
```

will remove the default 'a *:* rwm' entry. Doing:

```
echo a > /sys/fs/cgroup/1/devices.allow
```

will add the 'a *:* rwm' entry to the whitelist.

3. Security

Any task can move itself between cgroups. This clearly won't suffice, but we can decide the best way to adequately restrict movement as people get some experience with this. We may just want to require CAP_SYS_ADMIN, which at least is a separate bit from CAP_MKNOD. We may want to just refuse moving to a cgroup which isn't a descendant of the current one. Or we may want to use CAP_MAC_ADMIN, since we really are trying to lock down root.

CAP_SYS_ADMIN is needed to modify the whitelist or move another task to a new cgroup. (Again we'll probably want to change that).

A cgroup may not be granted more permissions than the cgroup's parent has.

4. Hierarchy

device cgroups maintain hierarchy by making sure a cgroup never has more access permissions than its parent. Every time an entry is written to a cgroup's devices.deny file, all its children will have that entry removed from their whitelist and all the locally set whitelist entries will be re-evaluated. In case one of the locally set whitelist entries would provide more access than the cgroup's parent, it'll be removed from the whitelist.

Example:

```
  A
 / \
    B

group        behavior       exceptions
A            allow          "b 8:* rwm", "c 116:1 rw"
B            deny           "c 1:3 rwm", "c 116:2 rwm", "b 3:* rwm"
```

If a device is denied in group A:

```
# echo "c 116:* r" > A/devices.deny
```

it'll propagate down and after revalidating B's entries, the whitelist entry "c 116:2 rwm" will be removed:

```
group        whitelist entries                        denied devices
A            all                                      "b 8:* rwm", "c 116:* rw"
B            "c 1:3 rwm", "b 3:* rwm"                 all the rest
```

In case parent's exceptions change and local exceptions are not allowed anymore, they'll be deleted.

Notice that new whitelist entries will not be propagated:

```
  A
 / \
        B

   group        whitelist entries                        denied devices
   A            "c 1:3 rwm", "c 1:5 r"                   all the rest
   B            "c 1:3 rwm", "c 1:5 r"                   all the rest
```

when adding `c *:3 rwm`:

```
   # echo "c *:3 rwm" >A/devices.allow
```

the result:

```
   group        whitelist entries                        denied devices
   A            "c *:3 rwm", "c 1:5 r"                   all the rest
   B            "c 1:3 rwm", "c 1:5 r"                   all the rest
```

but now it'll be possible to add new entries to B:

```
   # echo "c 2:3 rwm" >B/devices.allow
   # echo "c 50:3 r" >B/devices.allow
```

or even:

```
   # echo "c *:3 rwm" >B/devices.allow
```

Allowing or denying all by writing 'a' to devices.allow or devices.deny will not be possible once the device cgroups has children.

4.1 Hierarchy (internal implementation)

device cgroups is implemented internally using a behavior (ALLOW, DENY) and a list of exceptions. The internal state is controlled using the same user interface to preserve compatibility with the previous whitelist-only implementation. Removal or addition of exceptions that will reduce the access to devices will be propagated down the hierarchy. For every propagated exception, the effective rules will be re-evaluated based on current parent's access rules.