
Subject: [PATCH 7/8] CGroup Files: Convert devcgroup_access_write() into a cgroup write_string() handler

Posted by [Paul Menage](#) on Fri, 20 Jun 2008 23:44:05 GMT

[View Forum Message](#) <> [Reply to Message](#)

This patch converts devcgroup_access_write() from a raw file handler into a handler for the cgroup write_string() method. This allows some boilerplate copying/locking/checking to be removed and simplifies the cleanup path, since these functions are performed by the cgroups framework before calling the handler.

Signed-off-by: Paul Menage <menage@google.com>

security/device_cgroup.c | 103 ++++++-----
1 file changed, 39 insertions(+), 64 deletions(-)

Index: cws-2.6.26-rc5-mm3/security/device_cgroup.c

=====

--- cws-2.6.26-rc5-mm3.orig/security/device_cgroup.c

+++ cws-2.6.26-rc5-mm3/security/device_cgroup.c

@@ -59,6 +59,11 @@ static inline struct dev_cgroup *cgroup_
return css_to_devcgroup(cgroup_subsys_state(cgroup, devices_subsys_id));
}

+static inline struct dev_cgroup *task_devcgroup(struct task_struct *task)

+{
+ return css_to_devcgroup(task_subsys_state(task, devices_subsys_id));
+}

+
struct cgroup_subsys devices_subsys;

static int devcgroup_can_attach(struct cgroup_subsys *ss,
@@ -312,10 +317,10 @@ static int may_access_whitelist(struct d
* when adding a new allow rule to a device whitelist, the rule
* must be allowed in the parent device
*/

-static int parent_has_perm(struct cgroup *childcg,
+static int parent_has_perm(struct dev_cgroup *childcg,
struct dev_whitelist_item *wh)

{
- struct cgroup *pcg = childcg->parent;
+ struct cgroup *pcg = childcg->css.cgroup->parent;
struct dev_cgroup *parent;
int ret;

@@ -341,39 +346,18 @@ static int parent_has_perm(struct cgroup
* new access is only allowed if you're in the top-level cgroup, or your

```

* parent cgroup has the access you're asking for.
*/
-static ssize_t devcgroup_access_write(struct cgroup *cgroup, struct cftype *cft,
- struct file *file, const char __user *userbuf,
- size_t nbytes, loff_t *ppos)
-{
- struct cgroup *cur_cgroup;
- struct dev_cgroup *devcgroup, *cur_devcgroup;
- int filetype = cft->private;
- char *buffer, *b;
+static int devcgroup_update_access(struct dev_cgroup *devcgroup,
+ int filetype, const char *buffer)
+{
+ struct dev_cgroup *cur_devcgroup;
+ const char *b;
+ int retval = 0, count;
+ struct dev_whitelist_item wh;

+ if (!capable(CAP_SYS_ADMIN))
+ return -EPERM;

- devcgroup = cgroup_to_devcgroup(cgroup);
- cur_cgroup = task_cgroup(current, devices_subsys.subsys_id);
- cur_devcgroup = cgroup_to_devcgroup(cur_cgroup);
-
- buffer = kmalloc(nbytes+1, GFP_KERNEL);
- if (!buffer)
- return -ENOMEM;
-
- if (copy_from_user(buffer, userbuf, nbytes)) {
- retval = -EFAULT;
- goto out1;
- }
- buffer[nbytes] = 0; /* nul-terminate */
-
- cgroup_lock();
- if (cgroup_is_removed(cgroup)) {
- retval = -ENODEV;
- goto out2;
- }
+ cur_devcgroup = task_devcgroup(current);

+ memset(&wh, 0, sizeof(wh));
+ b = buffer;
@@ -390,14 +374,11 @@ static ssize_t devcgroup_access_write(st
+ wh.type = DEV_CHAR;
+ break;
+ default:

```

```

- retval = -EINVAL;
- goto out2;
+ return -EINVAL;
}
b++;
- if (!isspace(*b)) {
- retval = -EINVAL;
- goto out2;
- }
+ if (!isspace(*b))
+ return -EINVAL;
b++;
if (*b == '*') {
wh.major = ~0;
@@ -409,13 +390,10 @@ static ssize_t devcgroup_access_write(st
b++;
}
} else {
- retval = -EINVAL;
- goto out2;
- }
- if (*b != ':') {
- retval = -EINVAL;
- goto out2;
+ return -EINVAL;
}
+ if (*b != ':')
+ return -EINVAL;
b++;

/* read minor */
@@ -429,13 +407,10 @@ static ssize_t devcgroup_access_write(st
b++;
}
} else {
- retval = -EINVAL;
- goto out2;
- }
- if (!isspace(*b)) {
- retval = -EINVAL;
- goto out2;
+ return -EINVAL;
}
+ if (!isspace(*b))
+ return -EINVAL;
for (b++, count = 0; count < 3; count++, b++) {
switch (*b) {
case 'r':

```

```

@@ -452,8 +427,7 @@ static ssize_t devcgroup_access_write(st
    count = 3;
    break;
    default:
-   retval = -EINVAL;
-   goto out2;
+   return -EINVAL;
    }
}

```

```

@@ -461,38 +435,39 @@ handle:
    retval = 0;
    switch (filetype) {
    case DEVCG_ALLOW:
-   if (!parent_has_perm(cgroup, &wh))
-   retval = -EPERM;
-   else
-   retval = dev_whitelist_add(devcgroup, &wh);
-   break;
+   if (!parent_has_perm(devcgroup, &wh))
+   return -EPERM;
+   return dev_whitelist_add(devcgroup, &wh);
    case DEVCG_DENY:
        dev_whitelist_rm(devcgroup, &wh);
        break;
    default:
-   retval = -EINVAL;
-   goto out2;
+   return -EINVAL;
    }
+   return 0;
+}

```

```

-   if (retval == 0)
-   retval = nbytes;
-
-out2:
+static int devcgroup_access_write(struct cgroup *cgrp, struct cftype *cft,
+    const char *buffer)
+{
+   int retval;
+   if (!cgroup_lock_live_group(cgrp))
+   return -ENODEV;
+   retval = devcgroup_update_access(cgroup_to_devcgroup(cgrp),
+       cft->private, buffer);
+   cgroup_unlock();
-out1:
-   kfree(buffer);

```

```
return retval;
}

static struct cftype dev_cgroup_files[] = {
{
.name = "allow",
- .write = devcgroup_access_write,
+ .write_string = devcgroup_access_write,
.private = DEVCG_ALLOW,
},
{
.name = "deny",
- .write = devcgroup_access_write,
+ .write_string = devcgroup_access_write,
.private = DEVCG_DENY,
},
{
--

```

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>
