
Subject: [PATCH 6/9] user namespaces: convert all capable checks in kernel/sys.c
Posted by [serge](#) on Thu, 17 Feb 2011 15:03:42 GMT

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

This allows setuid/setgid in containers. It also fixes some corner cases where kernel logic foregoes capability checks when uids are equivalent. The latter will need to be done throughout the whole kernel.

Changelog:

Jan 11: Use nsown_capable() as suggested by Bastian Blank.
Jan 11: Fix logic errors in uid checks pointed out by Bastian.
Feb 15: allow prlimit to current (was regression in previous version)

Signed-off-by: Serge E. Hallyn <serge.hallyn@canonical.com>

```
kernel/sys.c | 74 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++-----  
1 files changed, 47 insertions(+), 27 deletions(-)
```

```
diff --git a/kernel/sys.c b/kernel/sys.c
```

```
index 7a1bbad..075370d 100644
```

```
--- a/kernel/sys.c
```

```
+++ b/kernel/sys.c
```

```
@@ -118,17 +118,29 @@ EXPORT_SYMBOL(cad_pid);
```

```
void (*pm_power_off_prepare)(void);
```

```
+/ * called with rcu_read_lock, creds are safe */
```

```
+static inline int set_one_prio_perm(struct task_struct *p)
```

```
+{
```

```
+ const struct cred *cred = current_cred(), *pcred = __task_cred(p);
```

```
+
```

```
+ if (pcred->user->user_ns == cred->user->user_ns &&
```

```
+ (pcred->uid == cred->euid ||
```

```
+ pcred->euid == cred->euid))
```

```
+ return 1;
```

```
+ if (ns_capable(pcred->user->user_ns, CAP_SYS_NICE))
```

```
+ return 1;
```

```
+ return 0;
```

```
+}
```

```
+
```

```
/*
```

```
 * set the priority of a task
```

```
 * - the caller must hold the RCU read lock
```

```
*/
```

```
static int set_one_prio(struct task_struct *p, int niceval, int error)
```

```
{
```

```
- const struct cred *cred = current_cred(), *pcred = __task_cred(p);
```

```

int no_nice;

- if (pcred->uid != cred->euid &&
-   pcred->euid != cred->euid && !capable(CAP_SYS_NICE)) {
+ if (!set_one_prio_perm(p)) {
    error = -EPERM;
    goto out;
}
@@ -502,7 +514,7 @@ SYSCALL_DEFINE2(setregid, gid_t, rgid, gid_t, egid)
if (rgid != (gid_t) -1) {
    if (old->gid == rgid ||
        old->egid == rgid ||
-   capable(CAP_SETGID))
+   nsown_capable(CAP_SETGID))
    new->gid = rgid;
    else
        goto error;
@@ -511,7 +523,7 @@ SYSCALL_DEFINE2(setregid, gid_t, rgid, gid_t, egid)
if (old->gid == egid ||
    old->egid == egid ||
    old->sgid == egid ||
-   capable(CAP_SETGID))
+   nsown_capable(CAP_SETGID))
    new->egid = egid;
    else
        goto error;
@@ -546,7 +558,7 @@ SYSCALL_DEFINE1(setgid, gid_t, gid)
old = current_cred();

    retval = -EPERM;
- if (capable(CAP_SETGID))
+ if (nsown_capable(CAP_SETGID))
    new->gid = new->egid = new->sgid = new->fsgid = gid;
    else if (gid == old->gid || gid == old->sgid)
        new->egid = new->fsgid = gid;
@@ -613,7 +625,7 @@ SYSCALL_DEFINE2(setreuid, uid_t, ruid, uid_t, euid)
new->uid = ruid;
if (old->uid != ruid &&
    old->euid != ruid &&
-   !capable(CAP_SETUID))
+   !nsown_capable(CAP_SETUID))
    goto error;
}

@@ -622,7 +634,7 @@ SYSCALL_DEFINE2(setreuid, uid_t, ruid, uid_t, euid)
if (old->uid != euid &&
    old->euid != euid &&
    old->suid != euid &&

```

```

- !capable(CAP_SETUID))
+ !nsown_capable(CAP_SETUID))
  goto error;
}

@@ -670,7 +682,7 @@ SYSCALL_DEFINE1(setuid, uid_t, uid)
  old = current_cred();

  retval = -EPERM;
- if (capable(CAP_SETUID)) {
+ if (nsown_capable(CAP_SETUID)) {
  new->suid = new->uid = uid;
  if (uid != old->uid) {
    retval = set_user(new);
@@ -712,7 +724,7 @@ SYSCALL_DEFINE3(setresuid, uid_t, ruid, uid_t, euid, uid_t, suid)
  old = current_cred();

  retval = -EPERM;
- if (!capable(CAP_SETUID)) {
+ if (!nsown_capable(CAP_SETUID)) {
  if (ruid != (uid_t) -1 && ruid != old->uid &&
      ruid != old->euid && ruid != old->suid)
    goto error;
@@ -776,7 +788,7 @@ SYSCALL_DEFINE3(setresgid, gid_t, rgid, gid_t, egid, gid_t, sgid)
  old = current_cred();

  retval = -EPERM;
- if (!capable(CAP_SETGID)) {
+ if (!nsown_capable(CAP_SETGID)) {
  if (rgid != (gid_t) -1 && rgid != old->gid &&
      rgid != old->egid && rgid != old->sgid)
    goto error;
@@ -836,7 +848,7 @@ SYSCALL_DEFINE1(setfsuid, uid_t, uid)

  if (uid == old->uid || uid == old->euid ||
      uid == old->suid || uid == old->fsuid ||
-   capable(CAP_SETUID)) {
+   nsown_capable(CAP_SETUID)) {
  if (uid != old_fsuid) {
    new->fsuid = uid;
    if (security_task_fix_setuid(new, old, LSM_SETID_FS) == 0)
@@ -869,7 +881,7 @@ SYSCALL_DEFINE1(setfsgid, gid_t, gid)

  if (gid == old->gid || gid == old->egid ||
      gid == old->sgid || gid == old->fsgid ||
-   capable(CAP_SETGID)) {
+   nsown_capable(CAP_SETGID)) {
  if (gid != old_fsgid) {

```

```

    new->fsgid = gid;
    goto change_okay;
@@ -1177,8 +1189,11 @@ SYSCALL_DEFINE2(sethostname, char __user *, name, int, len)
    int errno;
    char tmp[__NEW_UTS_LEN];

- if (!ns_capable(current->nsproxy->uts_ns->user_ns, CAP_SYS_ADMIN))
+ if (!ns_capable(current->nsproxy->uts_ns->user_ns, CAP_SYS_ADMIN)) {
+ printk(KERN_NOTICE "%s: did not have CAP_SYS_ADMIN\n", __func__);
    return -EPERM;
+ }
+ printk(KERN_NOTICE "%s: did have CAP_SYS_ADMIN\n", __func__);
    if (len < 0 || len > __NEW_UTS_LEN)
        return -EINVAL;
    down_write(&uts_sem);
@@ -1226,7 +1241,7 @@ SYSCALL_DEFINE2(setdomainname, char __user *, name, int, len)
    int errno;
    char tmp[__NEW_UTS_LEN];

- if (!capable(CAP_SYS_ADMIN))
+ if (!ns_capable(current->nsproxy->uts_ns->user_ns, CAP_SYS_ADMIN))
    return -EPERM;
    if (len < 0 || len > __NEW_UTS_LEN)
        return -EINVAL;
@@ -1341,6 +1356,8 @@ int do_prlimit(struct task_struct *tsk, unsigned int resource,
    rlim = tsk->signal->rlim + resource;
    task_lock(tsk->group_leader);
    if (new_rlim) {
+ /* Keep the capable check against init_user_ns until
+  cgroups can contain all limits */
    if (new_rlim->rlim_max > rlim->rlim_max &&
        !capable(CAP_SYS_RESOURCE))
        retval = -EPERM;
@@ -1384,19 +1401,22 @@ static int check_prlimit_permission(struct task_struct *task)
    {
    const struct cred *cred = current_cred(), *tcred;

- tcred = __task_cred(task);
- if (current != task &&
-     (cred->uid != tcred->euid ||
-      cred->uid != tcred->suid ||
-      cred->uid != tcred->uid ||
-      cred->gid != tcred->egid ||
-      cred->gid != tcred->sgid ||
-      cred->gid != tcred->gid) &&
-     !capable(CAP_SYS_RESOURCE)) {
- return -EPERM;
- }

```

```
+ if (current == task)
+ return 0;

- return 0;
+ tcred = __task_cred(task);
+ if (cred->user->user_ns == tcred->user->user_ns &&
+     (cred->uid == tcred->euid &&
+      cred->uid == tcred->suid &&
+      cred->uid == tcred->uid &&
+      cred->gid == tcred->egid &&
+      cred->gid == tcred->sgid &&
+      cred->gid == tcred->gid))
+ return 0;
+ if (ns_capable(tcred->user->user_ns, CAP_SYS_RESOURCE))
+ return 0;
+
+ return -EPERM;
}
```

```
SYSCALL_DEFINE4(prlimit64, pid_t, pid, unsigned int, resource,
```

```
--
```

```
1.7.0.4
```

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