Subject: Re: [PATCH RFC] cgroup_clone: use pid of newly created task for new cgroup

Posted by Paul Menage on Wed, 11 Jun 2008 07:24:31 GMT

View Forum Message <> Reply to Message

On Tue, Jun 10, 2008 at 2:23 PM, Serge E. Hallyn <serue@us.ibm.com> wrote:

```
> From faa707a44b971f5f3bf24e6a0c760ccb4ad278e6 Mon Sep 17 00:00:00 2001
> From: Serge Hallyn <serge@us.ibm.com>
> Date: Tue, 10 Jun 2008 15:57:32 -0500
> Subject: [PATCH 1/1] cgroup_clone: use pid of newly created task for new cgroup
>
> cgroup_clone creates a new cgroup with the pid of the task.  This works
> correctly for unshare, but for clone cgroup_clone is called from
> copy_namespaces inside copy_process, which happens before the new pid
> is created.  As a result, the new cgroup was created with current's pid.
> This patch:
>
>       1. Moves the call inside copy_process to after the new pid
>          is created
>       2. Passes the struct pid into ns_cgroup_clone (as it is not
>          yet attached to the task)
>       3. Passes a name from ns_cgroup_clone() into cgroup_clone()
>          so as to keep cgroup_clone() itself simpler
>       4. Uses pid_vnr() to get the process id value, so that the
>          pid used to name the new cgroup is always the pid as it
>          would be known to the task which did the cloning or
>          unsharing.  I think that is the most intuitive thing to
>          do.  This way, task t1 does clone(CLONE_NEWPID) to get
>          t2, which does clone(CLONE_NEWPID) to get t3, then the
>          cgroup for t3 will be named for the pid by which t2 knows
>          t3.
>
> This hasn't been tested enough to request inclusion, but I'd like to
> get feedback especially from Paul Menage on whether the semantics
> make sense.
```

Seems like a reasonable idea. It represents yet another change to the
userspace API following the 2.6.25.x one, but I guess that again it's
not one that anyone is seriously relying on yet (in particular since
it's not usable more than once from the same parent currently).

```
> -int cgroup_clone(struct task_struct *tsk, struct cgroup_subsys *subsys)
> +int cgroup_clone(struct task_struct *tsk, struct cgroup_subsys *subsys,
> +                                                       char *name)
```

You could reduce the patch churn by naming this parameter nodename.

```
> -       return cgroup_clone(task, &ns_subsys);
> +       struct pid *pid = (inpid ? inpid : task_pid(task));
> +      char name[MAX_CGROUP_TYPE_NAMELEN];
```

We should probably stop using MAX_CGROUP_TYPE_NAMELEN for this buffer
length and use something that explicitly sized to fit a pid_t.

```
> +
> +      snprintf(name, MAX_CGROUP_TYPE_NAMELEN, "%d", pid_vnr(pid));
> +      return cgroup_clone(task, &ns_subsys, name);
>  }
>
>  /*
> diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
> index adc7851..5ca106d 100644
> --- a/kernel/nsproxy.c
> +++ b/kernel/nsproxy.c
> @@ -157,12 +157,6 @@ int copy_namespaces(unsigned long flags, struct task_struct *tsk)
>               goto out;
>       }
>
> -      err = ns_cgroup_clone(tsk);
> -      if (err) {
> -              put_nsproxy(new_ns);
> -              goto out;
> -      }
> -
>       tsk->nsproxy = new_ns;
>
>  out:
> @@ -209,7 +203,7 @@ int unshare_nsproxy_namespaces(unsigned long unshare_flags,
>               goto out;
>       }
>
> -      err = ns_cgroup_clone(current);
> +      err = ns_cgroup_clone(current, NULL);
```

Maybe pass task_pid(current) here rather than doing the ?: in
ns_cgroup_clone() ?

Paul