Subject: Re: [PATCH RFC] cgroup_clone: use pid of newly created task for new cgroup
Posted by Paul Menage on Wed, 11 Jun 2008 15:59:12 GMT
View Forum Message <> Reply to Message

On Wed, Jun 11, 2008 at 8:46 AM, Serge E. Hallyn <serue@us.ibm.com> wrote:
>
> From f0635c20e9e9643fa9a90dd7e29b7855ff32ad40 Mon Sep 17 00:00:00 2001
> From: Serge Hallyn <serge@us.ibm.com>
> Date: Wed, 11 Jun 2008 10:41:37 -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.
>
> (Thanks to Dan Smith for finding the main bug)
>
> Changelog:
> June 11: Incorporate Paul Menage's feedback: don't pass
> NULL to ns_cgroup_clone from unshare, and reduce
> patch size by using 'nodename' in cgroup_clone.
> June 10: Original version
>
> Signed-off-by: Serge Hallyn <serge@us.ibm.com>

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

> ---
> include/linux/cgroup.h | 3 ++-
> include/linux/nsproxy.h | 7 +++++--

> kernel/cgroup.c | 7 ++++----
> kernel/fork.c | 4 ++++
> kernel/ns_cgroup.c | 7 +++++--
> kernel/nsproxy.c | 8 +-------
> 6 files changed, 20 insertions(+), 16 deletions(-)
>
> diff --git a/include/linux/cgroup.h b/include/linux/cgroup.h
> index 3690c3b..72cb6ca 100644
> --- a/include/linux/cgroup.h
> +++ b/include/linux/cgroup.h
> @@ -349,7 +349,8 @@ static inline struct cgroup* task_cgroup(struct task_struct *task,
>   return task_subsys_state(task, subsys_id)->cgroup;
> }
>
> -int cgroup_clone(struct task_struct *tsk, struct cgroup_subsys *ss);
> +int cgroup_clone(struct task_struct *tsk, struct cgroup_subsys *ss,
> +   char *nodename);
>
> /* A cgroup_iter should be treated as an opaque object */
> struct cgroup_iter {
> diff --git a/include/linux/nsproxy.h b/include/linux/nsproxy.h
> index 0e66b57..c8a768e 100644
> --- a/include/linux/nsproxy.h
> +++ b/include/linux/nsproxy.h
> @@ -82,9 +82,12 @@ static inline void get_nsproxy(struct nsproxy *ns)
> }
>
> #ifdef CONFIG_CGROUP_NS
> -int ns_cgroup_clone(struct task_struct *tsk);
> +int ns_cgroup_clone(struct task_struct *tsk, struct pid *pid);
> #else
> -static inline int ns_cgroup_clone(struct task_struct *tsk) { return 0; }
> +static inline int ns_cgroup_clone(struct task_struct *tsk, struct pid *pid)
> +{
> +   return 0;
> +}
> #endif
>
> #endif
> diff --git a/kernel/cgroup.c b/kernel/cgroup.c
> index 79fa060..b4c4b75 100644
> --- a/kernel/cgroup.c
> +++ b/kernel/cgroup.c
> @@ -2869,16 +2869,17 @@ void cgroup_exit(struct task_struct *tsk, int run_callbacks)
> * cgroup_clone - clone the cgroup the given subsystem is attached to
> * @tsk: the task to be moved
> * @subsys: the given subsystem
> + * @nodename: the name for the new cgroup

```
>  *
>  * Duplicate the current cgroup in the hierarchy that the given
>  * subsystem is attached to, and move this task into the new
>  * child.
>  */
> -int cgroup_clone(struct task_struct *tsk, struct cgroup_subsys *subsys)
> +int cgroup_clone(struct task_struct *tsk, struct cgroup_subsys *subsys,
> +                                          char *nodename)
>  {
>         struct dentry *dentry;
>         int ret = 0;
> -       char nodename[MAX_CGROUP_TYPE_NAMELEN];
>         struct cgroup *parent, *child;
>         struct inode *inode;
>         struct css_set *cg;
> @@ -2903,8 +2904,6 @@ int cgroup_clone(struct task_struct *tsk, struct cgroup_subsys
*subsys)
>         cg = tsk->cgroups;
>         parent = task_cgroup(tsk, subsys->subsys_id);
>
> -       snprintf(nodename, MAX_CGROUP_TYPE_NAMELEN, "%d", tsk->pid);
> -
>         /* Pin the hierarchy */
>         atomic_inc(&parent->root->sb->s_active);
>
> diff --git a/kernel/fork.c b/kernel/fork.c
> index f0e7767..47c0a97 100644
> --- a/kernel/fork.c
> +++ b/kernel/fork.c
> @@ -1125,6 +1125,10 @@ static struct task_struct *copy_process(unsigned long clone_flags,
>         if (clone_flags & CLONE_THREAD)
>                 p->tgid = current->tgid;
>
> +       if (current->nsproxy != p->nsproxy)
> +               if ((retval = ns_cgroup_clone(p, pid)))
> +                       goto bad_fork_free_pid;
> +
>         p->set_child_tid = (clone_flags & CLONE_CHILD_SETTID) ? child_tidptr : NULL;
>         /*
>          * Clear TID on mm_release()?
> diff --git a/kernel/ns_cgroup.c b/kernel/ns_cgroup.c
> index 48d7ed6..6431fb7 100644
> --- a/kernel/ns_cgroup.c
> +++ b/kernel/ns_cgroup.c
> @@ -24,9 +24,12 @@ static inline struct ns_cgroup *cgroup_to_ns(
>                         struct ns_cgroup, css);
>  }
>
```

```
> -int ns_cgroup_clone(struct task_struct *task)
> +int ns_cgroup_clone(struct task_struct *task, struct pid *pid)
>  {
> -       return cgroup_clone(task, &ns_subsys);
> +       char name[PROC_NUMBUF];
> +
> +       snprintf(name, PROC_NUMBUF, "%d", pid_vnr(pid));
> +       return cgroup_clone(task, &ns_subsys, name);
>  }
>
>  /*
> diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
> index adc7851..21575fc 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, task_pid(current));
>         if (err)
>                 put_nsproxy(*new_nsp);
>
> --
> 1.5.4.3
>
>
```