
Subject: Re: [PATCH 5/5] Move alloc_pid call to copy_process
Posted by [Sukadev Bhattiprolu](#) on Mon, 16 Jul 2007 22:54:11 GMT
[View Forum Message](#) <> [Reply to Message](#)

Oleg Nesterov [oleg@tv-sign.ru] wrote:

| Sukadev Bhattiprolu wrote:

| >
| > --- lx26-22-rc6-mm1.orig/kernel/pid.c 2007-07-13 18:23:55.000000000 -0700
| > +++ lx26-22-rc6-mm1/kernel/pid.c 2007-07-13 18:23:55.000000000 -0700
| > @@ -206,6 +206,10 @@ fastcall void free_pid(struct pid *pid)
| > /* We can be called with write_lock_irq(&tasklist_lock) held */
| > unsigned long flags;
| >
| > + /* check this here to keep copy_process() cleaner */
| > + if (unlikely(pid == &init_struct_pid))
| > + return;
| > +
| > spin_lock_irqsave(&pidmap_lock, flags);
| > hlist_del_rcu(&pid->pid_chain);
| > spin_unlock_irqrestore(&pidmap_lock, flags);
| > @@ -214,13 +218,17 @@ fastcall void free_pid(struct pid *pid)
| > call_rcu(&pid->rcu, delayed_put_pid);
| > }
| >
| > -struct pid *alloc_pid(void)
| > +struct pid *alloc_pid(enum copy_process_type copy_src)
| > {
| > struct pid *pid;
| > enum pid_type type;
| > int nr = -1;
| > struct pid_namespace *ns;
| >
| > + /* check this here to keep copy_process() cleaner */
| > + if (unlikely(copy_src == COPY_IDLE_PROCESS))
| > + return &init_struct_pid;
| > +
| > ns = task_active_pid_ns(current);
| > pid = kmem_cache_alloc(ns->pid_cachep, GFP_KERNEL);
| > if (!pid)

| Ugh. I am sorry! but imho this is so ugly :(

Ok :-)

| Could you please give more details why we need this change?

Well, with multiple pid namespaces, we may need to allocate a new

'struct pid_namespace' if the CLONE_NEWPID flag is specified. And as a part of initializing this pid_namespace, we need the 'task_struct' that will be the reaper of the new pid namespace.

And this task_struct is allocated in copy_process(). So we could still alloc_pid() in do_fork(), as we are doing currently and set the reaper of the new pid_namespace later in copy_process(). But that seemed to complicate error handling and add checks again in copy_process() for the CLONE_NEWPID.

| Even if we really need this, can't we do these checks in copy_process() ?

We could and I did have a check in copy_process() in one of my earlier versions to Containers@ list. We thought it cluttered copy_process() a bit. I can't seem to find that patch now, but will use Pavel Emelianov's more recent idea and even remove 'copy_src'

Pavel pls Sign-off (or at least ack) this patch.

| free_pid(&init_struct_pid) can only happen when fork fails, this is a slow
| path, and copy_process() knows copy_src.

Yep.

|
| Oleg.

Subject: [PATCH 5/5] Move alloc_pid call to copy_process

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>

Move alloc_pid() into copy_process(). This will keep all pid and pid namespace code together and simplify error handling when we support multiple pid namespaces.

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>

Cc: Pavel Emelianov <xemul@openvz.org>
Cc: Eric W. Biederman <ebiederm@xmission.com>
Cc: Cedric Le Goater <clg@fr.ibm.com>
Cc: Dave Hansen <haveblue@us.ibm.com>
Cc: Serge Hallyn <serue@us.ibm.com>
Cc: Herbert Poetzel <herbert@13thfloor.at>
Cc: Oleg Nesterov <oleg@tv-sign.ru>

kernel/fork.c | 18 ++++++++-----
kernel/pid.c | 4 ++++
2 files changed, 16 insertions(+), 6 deletions(-)

Index: lx26-22-rc6-mm1a/kernel/fork.c

```
=====
--- lx26-22-rc6-mm1a.orig/kernel/fork.c 2007-07-16 12:55:13.000000000 -0700
+++ lx26-22-rc6-mm1a/kernel/fork.c 2007-07-16 14:36:48.000000000 -0700
@@ -1029,6 +1029,12 @@ static struct task_struct *copy_process(
     if (p->binfmt && !try_module_get(p->binfmt->module))
         goto bad_fork_cleanup_put_domain;

+ if (!pid) {
+     pid = alloc_pid();
+     if (!pid)
+         goto bad_fork_put_binfmt_module;
+ }
+
     p->did_exec = 0;
     delayacct_tsk_init(p); /* Must remain after dup_task_struct() */
     copy_flags(clone_flags, p);
@@ -1316,6 +1322,8 @@ bad_fork_cleanup_container:
 #endif
     container_exit(p, container_callbacks_done);
     delayacct_tsk_free(p);
+ free_pid(pid);
+bad_fork_put_binfmt_module:
     if (p->binfmt)
         module_put(p->binfmt->module);
 bad_fork_cleanup_put_domain:
@@ -1380,19 +1388,16 @@ long do_fork(unsigned long clone_flags,
 {
     struct task_struct *p;
     int trace = 0;
- struct pid *pid = alloc_pid();
     long nr;

- if (!pid)
-     return -EAGAIN;
- nr = pid->nr;
     if (unlikely(current->ptrace)) {
         trace = fork_traceflag (clone_flags);
         if (trace)
             clone_flags |= CLONE_PTRACE;
     }

- p = copy_process(clone_flags, stack_start, regs, stack_size, parent_tidptr, child_tidptr, pid);
+ p = copy_process(clone_flags, stack_start, regs, stack_size,
+     parent_tidptr, child_tidptr, NULL);
     /*
      * Do this prior waking up the new thread - the thread pointer
```

```

* might get invalid after that point, if the thread exits quickly.
@@ -1400,6 +1405,8 @@ long do_fork(unsigned long clone_flags,
if (!IS_ERR(p)) {
    struct completion vfork;

+ nr = pid_nr(task_pid(p));
+
    if (clone_flags & CLONE_VFORK) {
        p->vfork_done = &vfork;
        init_completion(&vfork);
@@ -1433,7 +1440,6 @@ long do_fork(unsigned long clone_flags,
    }
    }
} else {
- free_pid(pid);
    nr = PTR_ERR(p);
}
return nr;

```

Index: lx26-22-rc6-mm1a/kernel/pid.c

```

=====
--- lx26-22-rc6-mm1a.orig/kernel/pid.c 2007-07-16 13:10:48.000000000 -0700
+++ lx26-22-rc6-mm1a/kernel/pid.c 2007-07-16 14:36:00.000000000 -0700
@@ -206,6 +206,10 @@ fastcall void free_pid(struct pid *pid)
/* We can be called with write_lock_irq(&tasklist_lock) held */
unsigned long flags;

+ /* check this here to keep copy_process() cleaner */
+ if (unlikely(pid == &init_struct_pid))
+ return;
+
    spin_lock_irqsave(&pidmap_lock, flags);
    hlist_del_rcu(&pid->pid_chain);
    spin_unlock_irqrestore(&pidmap_lock, flags);

```

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>
