Subject: Re: [PATCH 14/15] Destroy pid namespace on init's death
Posted by Oleg Nesterov on Thu, 02 Aug 2007 17:08:20 GMT
View Forum Message <> Reply to Message

On 08/02, Oleg Nesterov wrote:
>
> On 08/02, Kirill Korotaev wrote:
> >
> > Oleg Nesterov wrote:
> > >
> > > As it was already discussed, the current code is buggy, and should be
> > > fixed.
> >
> > I'm not that sure it MUST be fixed. There are no multi-threaded init's anywhere.
> > Oleg, does it worth changing without reasons?
>
> I don't know. But the kernel already tries to support multi-threaded init's.
> Look at de_thread(), it could be simplified a bit (and we don't need tasklist
> lock for zap_other_threads()) if we forbid them.
>
> Still. A non-root user does clone(CLONE_PIDNS), then clone(CLONE_THREAD),
> and sys_exit() from the main thread, then proceeds with fork()s. Now this
> ns has the global init as a child reaper, and admin can't kill entire pid_ns
> by killing its init. Worse, (see the reply to Sukadev' message), we should
> not reset pid_ns->child_reaper before zap_pid_ns_processes(). In that case
> ->child_reaper points to the freed task when the last thread exits, this
> means the non-root user can crash the kernel.
>
> Or, some embedded system uses multi-threaded init, and the kernel panics
> when the main thread exits.
>
> Perhaps this is just a "quality of implementation" question. sys_exit()
> from the main thread should be OK, why /sbin/init should be special?
>
> That said, I personally do not think that multi-threaded init is terribly
> useful.

So I think the patch below makes sense for now. Note that it removes the
games with pid_ns->child_reaper: this doesn't work currently, and this
has to be modified when we actually support pid namespaces anyway.

Oleg.

```
--- t/kernel/exit.c~MTINIT 2007-07-28 16:58:17.000000000 +0400
+++ t/kernel/exit.c 2007-08-02 20:59:59.000000000 +0400
@@ -895,6 +895,14 @@ static void check_stack_usage(void)
 static inline void check_stack_usage(void) {}
 #endif
```

```
+static inline void exit_child_reaper(struct task_struct *tsk)
+{
+ if (likely(tsk->group_leader != child_reaper(tsk)))
+  return;
+
+ panic("Attempted to kill init!");
+}
+
 fastcall NORET_TYPE void do_exit(long code)
 {
  struct task_struct *tsk = current;
@@ -908,13 +916,6 @@ fastcall NORET_TYPE void do_exit(long co
   panic("Aiee, killing interrupt handler!");
  if (unlikely(!tsk->pid))
   panic("Attempted to kill the idle task!");
- if (unlikely(tsk == child_reaper(tsk))) {
-  if (tsk->nsproxy->pid_ns != &init_pid_ns)
-   tsk->nsproxy->pid_ns->child_reaper = init_pid_ns.child_reaper;
-  else
-   panic("Attempted to kill init!");
- }
-

  if (unlikely(current->ptrace & PT_TRACE_EXIT)) {
   current->ptrace_message = code;
@@ -964,6 +965,7 @@ fastcall NORET_TYPE void do_exit(long co
  }
  group_dead = atomic_dec_and_test(&tsk->signal->live);
  if (group_dead) {
+  exit_child_reaper(tsk);
   hrtimer_cancel(&tsk->signal->real_timer);
   exit_itimers(tsk->signal);
  }
```