Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Louis Rilling on Thu, 17 Feb 2011 15:21:16 GMT
View Forum Message <> Reply to Message

On 16/02/11 12:10 -0800, Sukadev Bhattiprolu wrote:
> Oren Laadan [orenl@cs.columbia.edu] wrote:
> | So instead, we can call __wake_up_parent() from exit_checkpoint()
> | if indeed we are already reaped there:
> |
> | exit_checkpoint()
> | {
> | ...
> | if (current->flags & PF_RESTARTING) {
> | ...
> | /* either zombie or reaped ghost/dead */
> | if (current->exit_state = EXIT_DEAD)
> | __wake_up_parent(...); /* probably need lock */
> | ...
> | }
> | ...
> | }
> |
> | and to avoid userspace misuse, disallow non-thread-group-leader ghosts.
> |
> | ?
>
> Well, I don't see a problem as such, but notice one inconsistency.
>
> By the time the ghost task calls exit_checkpoint() it would have
> gone through release_task()/__exit_signal()/__unhash_process() so
> it is no longer on the parent's ->children list. We will be accessing
> the task's ->parent pointer after this.
>
> I am looking to see if anything prevents the parent from itself going
> through release_task(), after the child does the release_task() but before
> the child does the exit_checkpoint().
>
> In 2.6.38, I don't see specifically where a task's ->parent pointer is
> invalidated. The task->parent and task->parent->signal are freed in the
> final __put_task_struct(). So its probably safe to access them, even if the
> parent itself is exiting and has gone through release_task().
>
> But in 2.6.32 i.e RHEL5, tsk->signal is set to NULL in __exit_signal().
> So, I am trying to rule out the following scenario:
>
> Child (may not be a ghost) Parent
> ------------------------ ------
> - exit_notify(): is EXIT_DEAD

> - release_task():
>   - drops task_list_lock
>   - itself proceeds to exit.
>   - enters release_task()
>   - sets own->signal = NULL
>   (in 2.6.32, __exit_signal())
>
> - enters exit_checkpoint()
> - __wake_up_parent()
> access parents->signal NULL ptr
>
> Not sure if holding task_list_lock here is needed or will help.

Giving my 2 cents since I've been Cc'ed.

AFAICS, holding tasklist_lock prevents __exit_signal() from setting
parent->signal to NULL in your back. So something like this should be safe:

```
 read_lock(&tasklist_lock);
 if (current->parent->signal)
  __wake_up_parent(...);
 read_unlock(&tasklist_lock);
```

I haven't looked at the context, but of course this also requires that some
get_task_struct() on current->parent has been done somewhere else before current
has passed __exit_signal().

By the way, instead of checking current->parent->signal,
current->parent->exit_state would look cleaner to me. current->parent is not
supposed to wait on ->wait_chldexit after calling do_exit(), right?

Louis

--
Dr Louis Rilling   Kerlabs
Skype: louis.rilling   Batiment Germanium
Phone: (+33|0) 6 80 89 08 23 80 avenue des Buttes de Coesmes
http://www.kerlabs.com/   35700 Rennes