Subject: Re: [PATCH 14/15] Destroy pid namespace on init's death
Posted by Oleg Nesterov on Thu, 02 Aug 2007 16:08:51 GMT

View Forum Message <> Reply to Message

On 08/02, Kirill Korotaev wrote:
>
> Oleg Nesterov wrote:
> > On 08/01, Dave Hansen wrote:
> >
> >>> If the main thread is exiting, but is not the last thread in the
> >>> group, should we let it exit and let the next thread in the group
> >>> the reaper of the pid ns ?
> >>
> >>Well, what happens with a multithreaded init today?
> >
> >
> > 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.

Oleg.

____
Containers mailing list
Containers@lists.linux-foundation.org

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