Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Oren Laadan on Thu, 10 Feb 2011 03:53:56 GMT
View Forum Message <> Reply to Message

On 02/09/2011 09:44 PM, Sukadev Bhattiprolu wrote:
> Oren Laadan [orenl@cs.columbia.edu] wrote:
> |
> |
> | > But if that is true, I need to investigate why Louis Rilling was getting
> | > the crash in Jun 2010 - which he tried to fix here:
> | >
> | > http://lkml.org/lkml/2010/6/16/295
> |
> | I see. So basically there is a kerenl bug that can be potentially
> | exposed by the c/r code. Therefore, we need to fix the kernel bug...
> | (and until such a fix makes it to mainline, we'll add it as part of
> | the linux-cr patchset).
>
> Yes, but there is more than one problem (at least in our C/R kernel).
>
> There is the bug that Louis Rilling reported and Eric has a fix for.
> Even if we apply Eric's fix to the C/R kernel, we still will have
> another problem if do_ghost_task() sets ->exit_signal to -1.
>
> Consider this in 2.6.37:
>
> Container-init: Ghost child of container-init
>
>   do_ghost_task()
> zap_pid_ns..()
>   Send SIGKILL
>
>   do_wait()
>   - adds self to ->wait_chldexit queue
>   - goes through do_wait_thread() - finds that
>   it has at least one child (on tsk->children),
>   but it has not yet exited
>   - so waits for the child to exit
>   wakes up for SIGKILL
>   ->exit_signal = -1
>   do_exit()
>
> Note that exit_notify() does not notify parent when the ghost process
> exits, because ->exit_signal is -1.

I see.. nice catch :)

To address this, initially I thought that we could make ghosts take

the tasklist_lock (write) when they change their ->exit_signal.

But that's not enough because the parent may already be blocked in
wait (so it's too late). Therefore, we also need to have ghosts
wake-up their parent through __wake_up_parent().

so something like:

```
void ghost_auto_reapable()
{
 write_lock(&tasklist_lock);
 current->exit_signal = -1;
 __wake_up_sync_key(current, current->parent);
 write_unlock(&tasklist_lock);
}
```

If the parent wasn't at all waiting for us, no harm done...

>
> So you may ask how did the container-init have a ghost child. That was
> due to a bug in usercr :-).

You don't need a bug: the ghost flag is used for both ghost and dead
tasks (the former used to instantiate specific pids, the latter to
make other tasks orphans). So restarting a container that had orphan
tasks is guaranteed to do this.

Oren.

>
> But my point is such a userspace bug can leave the above container init
> unkillable.
>
> Note that this does not happen with normal threads which set ->exit_signal
> to -1 . That is because of the following two pieces of code in copy_process():
>
>   /* ok, now we should be set up.. */
>   p->exit_signal = (clone_flags & CLONE_THREAD) ? -1 : (clone_flags & CSIGNAL);
>
> and
>
>   /* CLONE_PARENT re-uses the old parent */
>   if (clone_flags & (CLONE_PARENT|CLONE_THREAD)) {
>   p->real_parent = current->real_parent;
>   p->parent_exec_id = current->parent_exec_id;
>
> With this our container-init above will not have any children to wait
> for in do_wait_thread().

>
> Sukadev
>

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containe rs