Subject: Re: [RFC][PATCH 06/16] Define is_global_init()
Posted by Sukadev Bhattiprolu on Fri, 25 May 2007 20:44:45 GMT
View Forum Message <> Reply to Message

Dave Hansen [hansendc@us.ibm.com] wrote:
| On Thu, 2007-05-24 at 13:24 +0400, Pavel Emelianov wrote:
| > > | > +int is_global_init(struct task_struct *tsk)
| > > | > +{
| > > | > + return (task_active_pid_ns(tsk) == &init_pid_ns && tsk->pid == 1);
| > > |
| > > | This can OOPS if you pass arbitrary task to this call...
| > > | tsk->nsproxy can already be NULL.
| > >
| > > Hmm. You are right. btw, this could be a bisect issue. Patch 9 of uses
| > > pid_ns from pid->upid_list and removes nsproxy->pid_ns.
| >
| > Yes, but that patch is not good either.
| > task_pid(tsk) may become NULL as well and this will oops.
|
| Have you reviewed the call paths to make sure this can actually happen
| in practice?

task_pid() can be NULL when we are tearing down the task structure in release_task() and in the tiny window between detach_pid() and attach_pid() in de_thread().

I think task_pid() is safe as long as it is called for 'current'. (we should probably add some comments)

I will double check my code, but I think all my calls to task_pid() and hence, to task_active_pid_ns() are safe, except for two cases:

a) is_global_init(). There are a few calls to process other than current, but not sure if they are a problem.

   For instance in current code, unhandled_signal() checks tsk->pid == 1 and proceeds to derefernce tsk->sighand.

   If task_pid() is NULL because the task was in release_task(), then so is tsk->sighand.

b) the temporary check I added in check_kill_permissions(). (I need to address Serge's comment here anyway).

To make is_global_init() more efficient and independent of task_pid(), can we steal a bit from task_struct->flags ? Like PF_KSWAPD, and there are unused bits :-)

|
| This just seems like another one of those racing-with-task-exit races.
| Shouldn't be too invasive to solve.

A little invasive approach for the release_task() case could be to remove
the 'struct pid' from the hash table, but leave it attached to the
'task_struct' till the 'task_struct' itself is freed.

Removing from hash table ensures no one finds this process anymore, but
keeping it attached allows those who have already found the 'task_struct'
to also use the 'struct pid' as long as they have the task_struct.

Of course, needs investigation and micro surgery.

|
| -- Dave

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers