
Subject: Re: [RFC][PATCH 06/16] Define is_global_init()
Posted by [Sukadev Bhattiprolu](#) on Wed, 30 May 2007 00:29:27 GMT
[View Forum Message](#) <> [Reply to Message](#)

Dave Hansen [hansendc@us.ibm.com] wrote:

```
| On Fri, 2007-05-25 at 13:44 -0700, sukadev@us.ibm.com wrote:  
| > 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)  
|  
| If we only call it for "current", then perhaps we should just change the  
| function so that it doesn't take any arguments. That way nobody can  
| screw it up.
```

Well, if the caller can confirm that the tsk passed in to task_pid() is not exiting, then it is ok to use. We have one such usage in do_fork() where we use the task struct we just initialized. The task has not yet been woken up.

```
|  
| > 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 dereference tsk->sigband.
| >
| > If task_pid() is NULL because the task was in release_task(),
| > then so is tsk->sigband.
|
| Really? Are there barriers or locks to make this happen? Can you be
| sure that compiler or cpu re-ordered code will keep this true?

I don't understand. Here is unhandled_signal() from 2.6.21-mm2.

```
int unhandled_signal(struct task_struct *tsk, int sig)
{
    if (is_init(tsk))
        return 1;
    if (tsk->ptrace & PT_PTRACED)
        return 0;
    return (tsk->sigband->action[sig-1].sa.sa_handler == SIG_IGN) ||
        (tsk->sigband->action[sig-1].sa.sa_handler == SIG_DFL);
}
```

My patch changed the is_init() to is_global_init() but the theory is that is_global_init() is not safe since it uses task_pid() which can return NULL if @tsk is exiting.

But if @tsk is exiting, tsk->sigband will also be NULL. So either the current callers have somehow ensured @tsk is not exiting or they are risking accessing tsk->sigband.

Not sure how compiler/cpu-reordering changes things. Can you elaborate ?

|
| > 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 :-)
|
| It feels to me like we're adding too many hacks on hacks here. Let's
| define the problem, because it sounds to me like we don't even know what
| it really is. What is the problem here, again?

We need an interface is_global_init() that tells us if the given process is /sbin/init (which is the process with pid_t == 1 in init_pid_ns).

Why we need is_global_init(): to allow/deny some facilities (eg: you cannot send arbitrary signals to the process).

How to implement `is_global_init()`: One simple/efficient way is to use a bit in `task->flags`.

Another way is to:

- ensure `tsk->pid == 1` and
- active pid ns of `@tsk` is `init_pid_ns`.

To check active pid ns of a process we currently need `task_pid()` which *may* not be safe for all processes at all times.

A related interface is `is_container_init()` which can be defined as the process with `pid_t == 1` in its active pid namespace.

|

| -- Dave

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