
Subject: Re: Re: pspace child_reaper
Posted by [rkagan](#) on Wed, 30 Aug 2006 12:42:16 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Tue, Aug 29, 2006 at 12:20:45PM -0500, Serge E. Hallyn wrote:
> However for an application container, since there was no real init to
> begin with, it seems valid to simply recognize that there is no pid for
> current->real_parent in the current pidspace, and that therefore we
> should not show it, treating ourselves as the root of the process tree.

And if you have a process that starts a new pidspace, forks a couple of children and exits, you end up with two roots of the process tree?

> > Having
> > processes escape the pid namespace when their parents exit is not
> > desirable.

Indeed.

> Clearly any process without a struct pid for
> (container=current_container, pid_t) shouldn't be presented to a process
> in current_container.
>
> As for the per-container init process, the alternative to always
> enforcing a separate init process for every container is to allow an
> option of making the process which did the pidspace unshare (or is it
> the parent of that process) masquerade as (pidspace=new_container, pid=1).

There's no point enforcing a separate 'init' process in every container.
The root of the process tree in a namespace has to be the child reaper
for that namespace meaning that

- it is immune to signals, ptracing, etc. from within the pidspace
- every process in the pidspace is reparented to it once that process' parent dies
- when it dies the whole pidspace is terminated

These are the standard properties of pid == 1 in UNIX. If it happens to be (or execs) /sbin/init then indeed it'll sit in the background spawning the usual user processes when necessary, but it doesn't have to be. E.g. I've just run an FC5 machine with init=/usr/bin/python which is how your application container would probably look like (the result of 'import os; os.system("ps axf")' in python prompt):

PID	TTY	STAT	TIME	COMMAND
1 ?		S	0:00	/usr/bin/python
2 ?		SN	0:00	[ksoftirqd/0]
3 ?		S	0:00	[watchdog/0]

```

4 ?    S<    0:00 [events/0]
5 ?    S<    0:00 [khelper]
6 ?    S<    0:00 [kthread]
8 ?    S<    0:00 \_ [kblockd/0]
9 ?    S<    0:00 \_ [kacpid]
67 ?   S<    0:00 \_ [khubd]
122 ?   S     0:00 \_ [pdflush]
123 ?   S     0:00 \_ [pdflush]
125 ?   S<    0:00 \_ [aio/0]
212 ?   S<    0:00 \_ [kseriod]
282 ?   S<    0:00 \_ [kpsmoused]
303 ?   S<    0:00 \_ [scsi_eh_0]
124 ?   S     0:00 [kswapd0]
290 ?   Ss    0:00 /bin/nash /init
317 ?   S     0:00 [kjournald]
329 ?   R     0:00 sh -c ps axf
330 ?   R     0:00 \_ ps axf

```

so there's no fundamental difference between "system containers" and "application containers".

Roman.

Containers mailing list

Containers@lists.osdl.org

<https://lists.osdl.org/mailman/listinfo/containers>
