Subject: Re: pspace child_reaper
Posted by ebiederm on Tue, 29 Aug 2006 18:27:38 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:

>> > But don't confuse /sbin/init with the reaper. /sbin/init will only know
>> > about pids in it's own container, so if we do need to call to userspace
>> > for every task which the reaper hears about, then we will need a
>> > per-container reaper. But if that's not the case (and i haven't seen
>> > where it is), then userspace is simply not involved, and so one global
>> > reaper suffices, since it will know not about pid_ts but about struct
>> > pids == (container,pid_t).
>>
>> First I don't remember the details but there was at least one
>> case where the vserver guys hit an application that cared.
>
> If that's the case, then per-container reaper it is.
>
> Herbert, do you recall which software that was?
>
>> Second the child_reaper is in some sense badly named. It is the
>> process that becomes your parent when your parent dies.
>
> For a system container, clearly we'll want to reparent to the
> container->init instead.
>
> 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.

Which would normally make us pid == 1. Which might be valid for
application containers.

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

> By masquerade, of course, I mean define a struct pid for it, so it's not
> actually masquerading - it really is that process.

Yes. That is reasonable, and actually what I would expect.
Although unshare is always weird.

But yes this does sound like sane set of semantics. Setup a
parent that lives in a different pid namespace, but has pid == 1
in our pid namespace. That parent can be our child reaper.
That parent will see the children when they die with pids mapped into
it's pid namespace.

This is at least reasonable as we now have the infrastructure that
can allow a struct pid to show up in multiple pid namespaces with
different pid values.

The question here is what is the user space interface to distinguish
between making the child of a clone pid == 1 or pid == 2?

>>From a user perspective that seems nicer, so long as it doesn't
> complicate the kernel side.
>
> But in any case, what you are suggesting in effect is that we first
> implement only system containers (offering a tiny userspace init process
> for small system containers), and worry about application containers
> later. That's a valid approach, of course.

Mostly what I'm suggesting is implement something that is stupid and
correct. Then we optimize.

The effect of my suggestion seems to be system containers and then
application containers but that isn't the point, and we can generalize
one namespace for application containers before we do all of the rest.

Eric

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers