Subject: Re: pspace child_reaper
Posted by ebiederm on Tue, 29 Aug 2006 16:46:44 GMT
View Forum Message <> Reply to Message

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

> Quoting Cedric Le Goater (clg@fr.ibm.com):
>> Eric W. Biederman wrote:
>> > Cedric Le Goater <clg@fr.ibm.com> writes:
>>
>> >> Any completely different idea on the topic ?
>> > Init reaps the children, and I believe there are parts of user space
>> > that depend on this. We shouldn't change that semantic.
>
> What parts of userspace actually depend on this?
>
> As far as I can tell, the closest we get is /sbin/init wanting to know
> when children it spawned die so it can restart this. Clearly that will
> happen anyway, even if there is one global reaper.

There is some process accounting in the kernel.

>> IMHO, the only semantic i see is in the kernel, which needs someone to take
>> care of sigchld. /sbin/init is a very good candidate bc it collects sigchld
>> anyway and discards the ones it doesn't know about.
>
> 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.

Second the child_reaper is in some sense badly named. It is the
process that becomes your parent when your parent dies. Having
processes escape the pid namespace when their parents exit is not
desirable.

Serge you are asking the question from the wrong direction. For
behavior that is well documented and standardized we need to prove
that user space won't care if we break it, when we depart from the
standard, the burden is on us.

My gut feel is that having insisting on an init in a pid namespace

really isn't that much of a burden, and if we have that there are
no problems providing the standard semantics.

Personally I would prefer to provide the standard semantics first
and then if there are optimizations that are worth while we can
see if it is worth doing. Instead of starting with the optmization
of having no init and then figuring out how to rework the unix
semantics to correct for that.

The core unix semantics are tied together in some really interesting
ways. One of the things that surprised me the other day was that
the way process groups and sessions are defined with respect to ttys
it is impossible for the kernel to send a signal to a session or
a process group that has exited, making the interface pid rollover
safe. Yet all of the newer interfaces that use sessions and process
groups are not pid rollover safe.

Eric

_____