Subject: Re: [RFC][PATCH] fork: Don't special case CLONE_NEWPID for process or sessions
Posted by ebiederm on Thu, 01 Nov 2007 17:03:42 GMT
View Forum Message <> Reply to Message

Pavel Emelyanov <xemul@openvz.org> writes:

>> As this is the main reason for this I don't see any reason to keep
>> the current clone behavior.
>
> Are you talking about keeping the ability to kill the outer processes?

I am talking about keeping the session and pgrp from the outer pid namespace.
Signals going out is less important.

>> Sending signals to our process group and our parent is an ability that
>> we allow even the most untrusted processes normally, and it is an
>> ability we can easily remove simply by calling setsid.
>
> You mix two things together - letting tasks send signals to their
> groups is good, but letting tasks send signals outside the namespace
> is bad.

If we have a case where we can send signals to the parent namespace
that makes some setting si_pid more difficult. So on those grounds
it is technically worth avoiding if we can.

I don't see any problem sharing a session and a process group
optionally with processes outside the pid namespace, and in fact
I find that desirable. So we don't always have to run pid
namespace leaders as daemons. To not be a daemon we need
the session, pgrp, and tty from the outside.

As for sending signals outside since setsid nicely closes that hole
in the cases we want to avoid, I am ambivalent about whether
we should look for a more robust solution to handling si_pid in which
case technical objections go away or if we should disallow the sending
altogether.

If the code can readily handle the full general case

Since si_pid is always task_pid(current) fixing the technical side may
not be much of a problem.

The way namespaces are defined sending signals outside the namespace
is almost impossible (because the pid lookup fails). If we get past
that sending signals is well defined and I don't have a problem with
it.

To support migration and strong virtual servers it is necessary that
we close all of the holes where we can get access to objects outside
of our namespace. In general that doesn't mean we should remove the
possibility for other users.

> Well, we allow a tiny possibility to have shared pids, but do we
> really want to support this possibility in the rest of the code?

There is essentially no cost if things are implemented properly. Just
use struct pid and it works. So I don't see a reason to avoid it.

Eric

____