Subject: Re: [PATCH 0/13] Pid namespaces (OpenVZ view)
Posted by serue on Fri, 25 May 2007 13:29:29 GMT
View Forum Message <> Reply to Message

Quoting Pavel Emelianov (xemul@openvz.org):
> Serge E. Hallyn wrote:
> > Quoting Eric W. Biederman (ebiederm@xmission.com):
> >> Pavel Emelianov <xemul@openvz.org> writes:
> >>
> >>> That's how OpenVZ sees the pid namespaces.
> >>>
> >>> The main idea is that kernel keeps operating with tasks pid
> >>> as it did before, but each task obtains one more pid for each
> >>> pid type - the virtual pid. When putting the pid to user or
> >>> getting the pid from it kernel operates with the virtual ones.
> >> Just a quick reaction.
> >>
> >> - I would very much like to see a minimum of 3 levels of pids,
> >>   being supported. Otherwise it is easy to overlook some of the
> >>   cases that are required to properly support nesting, which long
> >>   terms seems important.
> >
> > Pavel,
> >
> > If I wanted to start a virtual server and in there start some checkpoint
> > restart jobs, so I start a new pid namespace inside the c/r job, what
> > will happen?
>
> What will happen with this namespace on restore? What pids will
> you assign to it in the parent (but not that init) namespace?

No, no, my question is earlier. Maybe my use of the term
"checkpoint/restart job" is confusing, so let me call it a "batch job"
instead, with the understanding that it is started with the intent of
being safely checkpoint/restartable later on.

So in the original batch job, started in a vserver, what will the pids
look like in the checkpoint/restart job?

But I think I know the answer - you'll leave vpid == pid for these
tasks, and only set vpid differently when restarting a job, since that's
when you really care?

So the only situation where there might be a shortcoming is when
restarting a job in a vserver?

-serge

> a. arbitrary: that means that you don't care that subgroup
>   of tasks in the VS namespace. Thus why don't move them
>   into separate namespace
> b. try to hold them as they were: this way is likely to fail
>   and can work w/o namespaces at all.
>
> So what's your answer?
>
> > a. second pidns unshare is refused
> > b. second pidns unshare is allowed, but c/r job is not visible
> > from the virtual server (but is from the global pidns)
> > c. second pidns unshare is allowed, and somehow the c/r job
> > is visible from the virtual server
> >
> > If (a), is this a short-term shortcoming for simplicity of prototype and
> > code review, or do you think it's actually the right thing t do long
> > term?
> >
> > thanks,
> > -serge
> >
> >> - Semantically fork is easier then unshare. Unshare can mean
> >>   a lot of things, and it is easy to pick a meaning that has weird
> >>   side effects. Your implementation has a serious problem in that you
> >>   change the value of getpid() at runtime. Glibc does not know how to
> >>   cope with the value of getpid() changing.
> >>
> >> Eric
> >