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

Subject: Re: pspace child_reaper
Posted by serue on Tue, 29 Aug 2006 17:20:45 GMT
View Forum Message <> Reply to Message

> > 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.

> 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.

>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.

-serge

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

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

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 termiated

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

Subject: Re: pspace child_reaper
Posted by Cedric Le Goater on Wed, 30 Aug 2006 13:03:22 GMT
View Forum Message <> Reply to Message

Serge E. Hallyn wrote:
>>> 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?

I'm not sure herbert is on the list.

C.

_____

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

Subject: Re: pspace child_reaper
Posted by Herbert Poetzl on Wed, 30 Aug 2006 17:11:39 GMT
View Forum Message <> Reply to Message

On Wed, Aug 30, 2006 at 03:03:22PM +0200, Cedric Le Goater wrote:
> Serge E. Hallyn wrote:
> >>> 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.

first, to comment on that, here is what we (Linux-VServer)
do regarding init and reaper ...

the 'host' reaper is the original init process started
on the host. besides this one, we also know a guest
reaper, which 'usually' is identical with the guest's
init process, but, can be set to something differently.
most important, if that reaper is not present or defined
the kernel logic falls back to reparenting 'to-be-reaped'
children to the host init, which successfully reaps them

> > Herbert, do you recall which software that was?

no, I do not remember, but I will have a look in the
IRC logs, probably something will turn up

> I'm not sure herbert is on the list.

indeed, I am not (currently I'm in the progress of
subscribing to that mailing list too :)

best,
Herbert

> C.

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