Subject: [PATCH] Allow signalling container-init
Posted by Sukadev Bhattiprolu on Wed, 08 Aug 2007 23:47:37 GMT
View Forum Message <> Reply to Message

Pavel,

Should we include this in the patchset ?

Sukadev

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Subject: [PATCH] Allow signalling container-init

Only the global-init process must be special - any other container-init
process must be killable to prevent run-away processes in the system.

TODO: Ideally we should allow killing the container-init only from
ancestor containers and prevent it being killed from that or
descendant containers. But that is a more complex change and
will be addressed by a follow-on patch. For now allow the
container-init to be terminated by any process with sufficient
privileges.

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
---
kernel/signal.c | 6 ++----
1 file changed, 2 insertions(+), 4 deletions(-)

Index: lx26-23-rc1-mm1/kernel/signal.c
==========
--- lx26-23-rc1-mm1.orig/kernel/signal.c 2007-08-07 13:52:12.000000000 -0700
+++ lx26-23-rc1-mm1/kernel/signal.c 2007-08-08 15:09:27.000000000 -0700
@@ -1861,11 +1861,9 @@ relock:
continue;

/*
- * Init of a pid space gets no signals it doesn't want from
- * within that pid space. It can of course get signals from
- * its parent pid space.
+ * Global init gets no signals it doesn't want.
*/
- if (current == task_child_reaper(current))
+ if (is_global_init(current->group_leader))
continue;

if (sig_kernel_stop(signr)) {

Subject: Re: [PATCH] Allow signalling container-init
Posted by Oleg Nesterov on Thu, 09 Aug 2007 00:02:34 GMT
View Forum Message <> Reply to Message

On 08/08, sukadev@us.ibm.com wrote:
>
> From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> Subject: [PATCH] Allow signalling container-init
>
> Only the global-init process must be special - any other container-init
> process must be killable to prevent run-away processes in the system.

I think you are right, but....

> --- lx26-23-rc1-mm1.orig/kernel/signal.c 2007-08-07 13:52:12.000000000 -0700
> +++ lx26-23-rc1-mm1/kernel/signal.c 2007-08-08 15:09:27.000000000 -0700
> @@ -1861,11 +1861,9 @@ relock:
> continue;
>
> /*
> - * Init of a pid space gets no signals it doesn't want from
> - * within that pid space. It can of course get signals from
> - * its parent pid space.
> + * Global init gets no signals it doesn't want.
> */
> - if (current == task_child_reaper(current))
> + if (is_global_init(current->group_leader))
> continue;

...this breaks exec() from /sbin/init. Note that de_thread() kills other
sub-threads with SIGKILL. With this patch de_thread() will hang waiting
for other threads to die.

I think it is better to not change the current behaviour which is not
perfect (buggy), until we actually protect /sbin/init from unwanted
signals.

(That said, I am not sure what behaviour is better (worse :), with or
without this patch)

Oleg.

_____

Subject: Re: [PATCH] Allow signalling container-init
Posted by Daniel Pittman on Thu, 09 Aug 2007 00:46:33 GMT
View Forum Message <> Reply to Message

sukadev@us.ibm.com writes:

> Should we include this in the patchset ?

[...]

> Only the global-init process must be special - any other
> container-init process must be killable to prevent run-away processes
> in the system.

One problem I hit while using OpenVZ is that some init processes --
notable upstart used by Ubuntu -- depend on the special signal processing
extended to init by the kernel.

The problem here was that a signal the kernel would otherwise have
blocked was sent to upstart, the default handler was invoked and init
was terminated.

> TODO: Ideally we should allow killing the container-init only from
> ancestor containers and prevent it being killed from that or
> descendant containers. But that is a more complex change and
> will be addressed by a follow-on patch. For now allow the
> container-init to be terminated by any process with sufficient
> privileges.

This will break, as far as I can see, by allowing the container root to
send signals to init that it doesn't expect.

Regards,
Daniel
--
Digital Infrastructure Solutions -- making IT simple, stable and secure
Phone: 0401 155 707 email: contact@digital-infrastructure.com.au
http://digital-infrastructure.com.au/

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: [PATCH] Allow signalling container-init
Posted by serue on Thu, 09 Aug 2007 01:21:28 GMT
View Forum Message <> Reply to Message

Quoting Daniel Pittman (daniel@rimspace.net):
> sukadev@us.ibm.com writes:
>
> > Should we include this in the patchset ?
>
> [...]
>
> > Only the global-init process must be special - any other
> > container-init process must be killable to prevent run-away processes
> > in the system.
>
> One problem I hit while using OpenVZ is that some init processes --
> notable upstart used by Ubuntu -- depend on the special signal processing
> extended to init by the kernel.
>
> The problem here was that a signal the kernel would otherwise have
> blocked was sent to upstart, the default handler was invoked and init
> was terminated.
>
> > TODO: Ideally we should allow killing the container-init only from
> > ancestor containers and prevent it being killed from that or
> > descendant containers. But that is a more complex change and
> > will be addressed by a follow-on patch. For now allow the
> > container-init to be terminated by any process with sufficient
> > privileges.
>
> This will break, as far as I can see, by allowing the container root to
> send signals to init that it doesn't expect.

Yes, in the end what we want is for a container init to receive

1. all signals from a (authorized) process in a parent
   pid namespace.
2. for signals sent from inside it's pid namespace, only
   exactly those signals for which it has installed a
   custom signal handler, no others.

In other words to a process in an ancestor pid namespace, the init of a
container is like any other process. To a process inside the namespace
for which it is init, it is as /sbin/init is to the system now.

Actually achieving that without affecting performance for all signalers
is nontrivial. The current patchset is complex enough that I'd like to
see us settle on non-optimal semantics for now, and once these patches
have settled implement the ideal signaling.

-serge

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH] Allow signalling container-init
Posted by Daniel Pittman on Thu, 09 Aug 2007 01:29:06 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:
> Quoting Daniel Pittman (daniel@rimspace.net):
>> sukadev@us.ibm.com writes:

[...]

>> > TODO: Ideally we should allow killing the container-init only from
>> > ancestor containers and prevent it being killed from that or
>> > descendant containers. But that is a more complex change and
>> > will be addressed by a follow-on patch. For now allow the
>> > container-init to be terminated by any process with sufficient
>> > privileges.
>>
>> This will break, as far as I can see, by allowing the container root to
>> send signals to init that it doesn't expect.
>
> Yes, in the end what we want is for a container init to receive
>
> 1. all signals from a (authorized) process in a parent
> pid namespace.
> 2. for signals sent from inside it's pid namespace, only
> exactly those signals for which it has installed a
> custom signal handler, no others.
>
> In other words to a process in an ancestor pid namespace, the init of a
> container is like any other process. To a process inside the namespace
> for which it is init, it is as /sbin/init is to the system now.

That makes sense.

> Actually achieving that without affecting performance for all
> signalers is nontrivial. The current patchset is complex enough that
> I'd like to see us settle on non-optimal semantics for now, and once
> these patches have settled implement the ideal signaling.

I appreciate that. I figured to make you aware that this will make it

impossible to run upstart and, probably, other versions of init in your container as expected.

Since this was a somewhat subtle bug to track down it is, I think, work documenting so that people trying to use this code are aware of the limitation.

Regards,
Daniel
--
Digital Infrastructure Solutions -- making IT simple, stable and secure
Phone: 0401 155 707 email: contact@digital-infrastructure.com.au
http://digital-infrastructure.com.au/

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH] Allow signalling container-init
Posted by Sukadev Bhattiprolu on Thu, 09 Aug 2007 07:29:33 GMT
View Forum Message <> Reply to Message

Oleg Nesterov [oleg@tv-sign.ru] wrote:
| On 08/08, sukadev@us.ibm.com wrote:
| >
| > From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
| > Subject: [PATCH] Allow signalling container-init
| >
| > Only the global-init process must be special - any other container-init
| > process must be killable to prevent run-away processes in the system.
|
| I think you are right, but....
|
| > --- lx26-23-rc1-mm1.orig/kernel/signal.c 2007-08-07 13:52:12.000000000 -0700
| > +++ lx26-23-rc1-mm1/kernel/signal.c 2007-08-08 15:09:27.000000000 -0700
| > @@ -1861,11 +1861,9 @@ relock:
| > continue;
| >
| > /*
| > - * Init of a pid space gets no signals it doesn't want from
| > - * within that pid space. It can of course get signals from
| > - * its parent pid space.
| > + * Global init gets no signals it doesn't want.
| > */
| > - if (current == task_child_reaper(current))
| > + if (is_global_init(current->group_leader))
| > continue;

|
| ...this breaks exec() from /sbin/init. Note that de_thread() kills other
| sub-threads with SIGKILL. With this patch de_thread() will hang waiting
| for other threads to die.

Again for threaded-init I guess :-(

Well, we discussed last week about allowing non-root users to clone their
pid namespace. The user can then create a container-init and this
process would become immune to signal even by a root user ?

|
| I think it is better to not change the current behaviour which is not
| perfect (buggy), until we actually protect /sbin/init from unwanted
| signals.

Can we preserve the existing behavior by checking only the main thread
of global init (i.e pass in 'current' rather than 'current->group_leader'
to is_global_init()) ?

|
| (That said, I am not sure what behaviour is better (worse :), with or
| without this patch)
|
| Oleg.

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH] Allow signalling container-init
Posted by Oleg Nesterov on Thu, 09 Aug 2007 07:55:35 GMT
View Forum Message <> Reply to Message

On 08/09, sukadev@us.ibm.com wrote:
>
> Oleg Nesterov [oleg@tv-sign.ru] wrote:
> | On 08/08, sukadev@us.ibm.com wrote:
> | >
> | > From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> | > Subject: [PATCH] Allow signalling container-init
> | >
> | > Only the global-init process must be special - any other container-init
> | > process must be killable to prevent run-away processes in the system.
> |
> | I think you are right, but....
> |

> | > --- lx26-23-rc1-mm1.orig/kernel/signal.c 2007-08-07 13:52:12.000000000 -0700
> | > +++ lx26-23-rc1-mm1/kernel/signal.c 2007-08-08 15:09:27.000000000 -0700
> | > @@ -1861,11 +1861,9 @@ relock:
> | > 	continue;
> | >
> | > 	/*
> | > -	 * Init of a pid space gets no signals it doesn't want from
> | > -	 * within that pid space. It can of course get signals from
> | > -	 * its parent pid space.
> | > +	 * Global init gets no signals it doesn't want.
> | > 	 */
> | > -	if (current == task_child_reaper(current))
> | > +	if (is_global_init(current->group_leader))
> | > 	continue;
> |
> | ...this breaks exec() from /sbin/init. Note that de_thread() kills other
> | sub-threads with SIGKILL. With this patch de_thread() will hang waiting
> | for other threads to die.
>
> Again for threaded-init I guess :-(
>
> Well, we discussed last week about allowing non-root users to clone their
> pid namespace. The user can then create a container-init and this
> process would become immune to signal even by a root user ?

please see below,

> |
> | I think it is better to not change the current behaviour which is not
> | perfect (buggy), until we actually protect /sbin/init from unwanted
> | signals.
>
> Can we preserve the existing behavior by checking only the main thread
> of global init (i.e pass in 'current' rather than 'current->group_leader'
> to is_global_init()) ?

Yes, this is what I meant, this is what we have in Linus's tree.
This way a container-init could be killed. This all is not correct,
but we shouldn't replace one bug with another.

Oleg.

Subject: Re: [PATCH] Allow signalling container-init
Posted by dev on Thu, 09 Aug 2007 08:16:07 GMT
View Forum Message <> Reply to Message

Daniel Pittman wrote:
> sukadev@us.ibm.com writes:
>
>
>>Should we include this in the patchset ?
>
>
> [...]
>
>
>>Only the global-init process must be special - any other
>>container-init process must be killable to prevent run-away processes
>>in the system.
>
>
> One problem I hit while using OpenVZ is that some init processes --
> notable upstart used by Ubuntu -- depend on the special signal processing
> extended to init by the kernel.
>
> The problem here was that a signal the kernel would otherwise have
> blocked was sent to upstart, the default handler was invoked and init
> was terminated.
>
>
>>TODO: Ideally we should allow killing the container-init only from
>> ancestor containers and prevent it being killed from that or
>> descendant containers. But that is a more complex change and
>> will be addressed by a follow-on patch. For now allow the
>> container-init to be terminated by any process with sufficient
>> privileges.
>
>
> This will break, as far as I can see, by allowing the container root to
> send signals to init that it doesn't expect.

This was fixed in OpenVZ in recent kernels and Pavel tried to address this in
pid namespace patches as well, but since no beatiful solution was found
it was decided to postpone this issue.

NOTE: parent can still send signals to child container' init. This
  is convinient since you can terminate the whole container quickly from
  the host node.

Thanks,
Kirill

Subject: Re: [PATCH] Allow signalling container-init
Posted by Pavel Emelianov on Thu, 09 Aug 2007 10:47:39 GMT
View Forum Message <> Reply to Message

Oleg Nesterov wrote:
> On 08/09, sukadev@us.ibm.com wrote:
>> Oleg Nesterov [oleg@tv-sign.ru] wrote:
>> | On 08/08, sukadev@us.ibm.com wrote:
>> | >
>> | > From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
>> | > Subject: [PATCH] Allow signalling container-init
>> | >
>> | > Only the global-init process must be special - any other container-init
>> | > process must be killable to prevent run-away processes in the system.
>> |
>> | I think you are right, but....
>> |
>> | > --- lx26-23-rc1-mm1.orig/kernel/signal.c 2007-08-07 13:52:12.000000000 -0700
>> | > +++ lx26-23-rc1-mm1/kernel/signal.c 2007-08-08 15:09:27.000000000 -0700
>> | > @@ -1861,11 +1861,9 @@ relock:
>> | > continue;
>> | >
>> | > /*
>> | > - * Init of a pid space gets no signals it doesn't want from
>> | > - * within that pid space. It can of course get signals from
>> | > - * its parent pid space.
>> | > + * Global init gets no signals it doesn't want.
>> | > */
>> | > - if (current == task_child_reaper(current))
>> | > + if (is_global_init(current->group_leader))
>> | > continue;
>> |
>> | ...this breaks exec() from /sbin/init. Note that de_thread() kills other
>> | sub-threads with SIGKILL. With this patch de_thread() will hang waiting
>> | for other threads to die.
>>
>> Again for threaded-init I guess :-(
>>
>> Well, we discussed last week about allowing non-root users to clone their
>> pid namespace. The user can then create a container-init and this
>> process would become immune to signal even by a root user ?

>
> please see below,
>
>> |
>> | I think it is better to not change the current behaviour which is not
>> | perfect (buggy), until we actually protect /sbin/init from unwanted
>> | signals.
>>
>> Can we preserve the existing behavior by checking only the main thread
>> of global init (i.e pass in 'current' rather than 'current->group_leader'
>> to is_global_init()) ?
>
> Yes, this is what I meant, this is what we have in Linus's tree.
> This way a container-init could be killed. This all is not correct,
> but we shouldn't replace one bug with another.

Well, I agree with Oleg. I think that we should keep the patches
without the signal handling until this set is in (at least) -mm.
init pid namespace will work without it as used to do, and we'll
develop a better signal handling and fix existing BUGs.

I know that this creates a hole for creating unkillable process,
but since this is for root user only (CAP_SYS_ADMIN) this is OK.

> Oleg.

Thanks,
Pavel

Subject: Re: [PATCH] Allow signalling container-init
Posted by serue on Thu, 09 Aug 2007 14:42:14 GMT
View Forum Message <> Reply to Message

Quoting Daniel Pittman (daniel@rimspace.net):
> "Serge E. Hallyn" <serue@us.ibm.com> writes:
> > Quoting Daniel Pittman (daniel@rimspace.net):
> >> sukadev@us.ibm.com writes:
>
> [...]
>
> >> > TODO: Ideally we should allow killing the container-init only from
> >> > ancestor containers and prevent it being killed from that or
> >> > descendant containers. But that is a more complex change and

> >> > will be addressed by a follow-on patch. For now allow the
> >> > container-init to be terminated by any process with sufficient
> >> > privileges.
> >>
> >> This will break, as far as I can see, by allowing the container root to
> >> send signals to init that it doesn't expect.
> >
> > Yes, in the end what we want is for a container init to receive
> >
> > 1. all signals from a (authorized) process in a parent
> >   pid namespace.
> > 2. for signals sent from inside it's pid namespace, only
> >   exactly those signals for which it has installed a
> >   custom signal handler, no others.
> >
> > In other words to a process in an ancestor pid namespace, the init of a
> > container is like any other process. To a process inside the namespace
> > for which it is init, it is as /sbin/init is to the system now.
>
> That makes sense.
>
> > Actually achieving that without affecting performance for all
> > signalers is nontrivial. The current patchset is complex enough that
> > I'd like to see us settle on non-optimal semantics for now, and once
> > these patches have settled implement the ideal signaling.
>
> I appreciate that. I figured to make you aware that this will make it
> impossible to run upstart and, probably, other versions of init in your
> container as expected.
>
> Since this was a somewhat subtle bug to track down it is, I think, work
> documenting so that people trying to use this code are aware of the
> limitation.

Agreed. I do think it is documented in the code and in changelogs.
Maybe it's worth adding a Documentation/ file describing how to use the
pid namespaces, ideal semantics, and current shortcomings, for people
who want to use+test the feature rather than work with the kernel code.

-serge

Subject: Re: [PATCH] Allow signalling container-init

Pavel Emelianov [xemul@openvz.org] wrote:
| Oleg Nesterov wrote:
| >On 08/09, sukadev@us.ibm.com wrote:
| >>Oleg Nesterov [oleg@tv-sign.ru] wrote:
| >>| On 08/08, sukadev@us.ibm.com wrote:
| >>| >
| >>| > From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
| >>| > Subject: [PATCH] Allow signalling container-init
| >>| >
| >>| > Only the global-init process must be special - any other
| >>container-init
| >>| > process must be killable to prevent run-away processes in the system.
| >>|
| >>| I think you are right, but....
| >>|
| >>| > --- lx26-23-rc1-mm1.orig/kernel/signal.c 2007-08-07
| >>13:52:12.000000000 -0700
| >>| > +++ lx26-23-rc1-mm1/kernel/signal.c 2007-08-08
| >>15:09:27.000000000 -0700
| >>| > @@ -1861,11 +1861,9 @@ relock:
| >>| >   continue;
| >>| >
| >>| >   /*
| >>| > -   * Init of a pid space gets no signals it doesn't
| >>want from
| >>| > -   * within that pid space. It can of course get
| >>signals from
| >>| > -   * its parent pid space.
| >>| > +   * Global init gets no signals it doesn't want.
| >>| >   */
| >>| > - if (current == task_child_reaper(current))
| >>| > + if (is_global_init(current->group_leader))
| >>| >   continue;
| >>|
| >>| ...this breaks exec() from /sbin/init. Note that de_thread() kills other
| >>| sub-threads with SIGKILL. With this patch de_thread() will hang waiting
| >>| for other threads to die.
| >>
| >>Again for threaded-init I guess :-(
| >>
| >>Well, we discussed last week about allowing non-root users to clone their
| >>pid namespace. The user can then create a container-init and this
| >>process would become immune to signal even by a root user ?
| >
| >please see below,
| >

| >>|
| >>| I think it is better to not change the current behaviour which is not
| >>| perfect (buggy), until we actually protect /sbin/init from unwanted
| >>| signals.
| >>
| >>Can we preserve the existing behavior by checking only the main thread
| >>of global init (i.e pass in 'current' rather than 'current->group_leader'
| >>to is_global_init()) ?
| >
| >Yes, this is what I meant, this is what we have in Linus's tree.
| >This way a container-init could be killed. This all is not correct,
| >but we shouldn't replace one bug with another.
|
| Well, I agree with Oleg. I think that we should keep the patches
| without the signal handling until this set is in (at least) -mm.
| init pid namespace will work without it as used to do, and we'll
| develop a better signal handling and fix existing BUGs.
|
| I know that this creates a hole for creating unkillable process,
| but since this is for root user only (CAP_SYS_ADMIN) this is OK.

But I think there is a difference bw what you are saying and what
Oleg is saying.

Oleg pls correct me if I am wrong, but from what I understand, we
just need modify my earlier fix so we can still terminate the
container from a parent namespace but preserve the existing behavior
w.r.t threaded-inits.

Here is the modified patch for this.

Suka

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Subject: [PATCH] Allow signalling container-init

Only the global-init process must be special - any other container-init
process must be killable to prevent run-away processes in the system.

TODO: Ideally we should allow killing the container-init only from parent
 container and prevent it being killed from within the container.
 But that is a more complex change and will be addressed by a follow-on
 patch. For now allow the container-init to be terminated by any process
 with sufficient privileges.

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
---
 kernel/signal.c | 6 ++----

1 file changed, 2 insertions(+), 4 deletions(-)

Index: lx26-23-rc1-mm1/kernel/signal.c
===
--- lx26-23-rc1-mm1.orig/kernel/signal.c 2007-08-07 13:52:12.000000000 -0700
+++ lx26-23-rc1-mm1/kernel/signal.c 2007-08-09 17:22:19.000000000 -0700
@@ -1861,11 +1861,9 @@ relock:
 continue;

 /*
- * Init of a pid space gets no signals it doesn't want from
- * within that pid space. It can of course get signals from
- * its parent pid space.
+ * Global init gets no signals it doesn't want.
 */
- if (current == task_child_reaper(current))
+ if (is_global_init(current))
 continue;

 if (sig_kernel_stop(signr)) {

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH] Allow signalling container-init
Posted by Oleg Nesterov on Fri, 10 Aug 2007 10:53:59 GMT
View Forum Message <> Reply to Message

On 08/09, sukadev@us.ibm.com wrote:
>
> Pavel Emelianov [xemul@openvz.org] wrote:
> | Oleg Nesterov wrote:
> | >>|
> | >>| I think it is better to not change the current behaviour which is not
> | >>| perfect (buggy), until we actually protect /sbin/init from unwanted
> | >>| signals.
> | >>
> | >>Can we preserve the existing behavior by checking only the main thread
> | >>of global init (i.e pass in 'current' rather than 'current->group_leader'
> | >>to is_global_init()) ?
> | >
> | >Yes, this is what I meant, this is what we have in Linus's tree.
> | >This way a container-init could be killed. This all is not correct,
> | >but we shouldn't replace one bug with another.
> |
> | Well, I agree with Oleg. I think that we should keep the patches

> | without the signal handling until this set is in (at least) -mm.
> | init pid namespace will work without it as used to do, and we'll
> | develop a better signal handling and fix existing BUGs.
> |
> | I know that this creates a hole for creating unkillable process,
> | but since this is for root user only (CAP_SYS_ADMIN) this is OK.
>
> But I think there is a difference bw what you are saying and what
> Oleg is saying.

Hm. It seems to me there is no difference, but perhaps I misunderstood
Pavel?

> Only the global-init process must be special - any other container-init
> process must be killable to prevent run-away processes in the system.
>
> TODO: Ideally we should allow killing the container-init only from parent
> container and prevent it being killed from within the container.
> But that is a more complex change and will be addressed by a follow-on
> patch. For now allow the container-init to be terminated by any process
> with sufficient privileges.
>
> Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> ---
> kernel/signal.c | 6 ++----
> 1 file changed, 2 insertions(+), 4 deletions(-)
>
> Index: lx26-23-rc1-mm1/kernel/signal.c
> ==
> --- lx26-23-rc1-mm1.orig/kernel/signal.c 2007-08-07 13:52:12.000000000 -0700
> +++ lx26-23-rc1-mm1/kernel/signal.c 2007-08-09 17:22:19.000000000 -0700
> @@ -1861,11 +1861,9 @@ relock:
>   continue;
>
>   /*
> -   * Init of a pid space gets no signals it doesn't want from
> -   * within that pid space. It can of course get signals from
> -   * its parent pid space.
> +   * Global init gets no signals it doesn't want.
>   */
> - if (current == task_child_reaper(current))
> + if (is_global_init(current))
>   continue;

Imho, this is a right change for now. The Linus's tree does the same:

```
  if (current == child_reaper(current))
   continue;
```

Because child_reaper() === init_pid_ns.child_reaper.

Oleg.

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers