Subject: Re: [PATCH 1/3] Signal semantics for /sbin/init
Posted by serue on Mon, 01 Oct 2007 18:08:49 GMT
View Forum Message <> Reply to Message

Quoting sukadev@us.ibm.com (sukadev@us.ibm.com):
> Serge E. Hallyn [serue@us.ibm.com] wrote:
> | Quoting sukadev@us.ibm.com (sukadev@us.ibm.com):
> | > Oleg Nesterov [oleg@tv-sign.ru] wrote:
> | > | On 09/13, sukadev@us.ibm.com wrote:
> | > | >
> | > | > Oleg Nesterov [oleg@tv-sign.ru] wrote:
> | > | > | > >
> | > | > | > >> Notes:
> | > | > | > >>
> | > | > | > >> - Blocked signals are never ignored, so init still can receive
> | > | > | > >>   a pending blocked signal after sigprocmask(SIG_UNBLOCK).
> | > | > | > >>   Easy to fix, but probably we can ignore this issue.
> | > | > | > >
> | > | > | > > I was wrong. This should be fixed right now. I _think_ this is easy,
> | > | > | > > and I was going to finish this patch yesterday, but - sorry! - I just
> | > | > | > > can't switch to "kernel mode" these days, I am fighting with some urgent
> | > | > | > > tasks on my paid job.
> | > | > | > >
> | > | > | > To respect the current init semantic,
> | > | > | >
> | > | > | The current init semantic is broken in many ways ;)
> | > | > | >
> | > | > | > shouldn't we discard any unblockable
> | > | > | > signal (STOP and KILL) sent by a process to its pid namespace init process ?
> | > | > |
> | > | > Yes. And Patch 1/3 (Oleg's patch) in the set I sent, handles this already
> | > | > (since STOP and KILL are never in the task->blocked list)
> | > | >
> | > | >
> | > | > | Then, all other signals should be handled appropriately by the pid namespace
> | > | > | init.
> | > | >
> | > | > |
> | > | > | Yes, I think you are probably right, this should be enough in practice. After all,
> | > | > | only root can send the signal to /sbin/init.
> | > | >
> | > | > I agree - the assumption that the container-init will handle these
> | > | > other signals, simplifies the kernel implementation for now.
> | > | >
> | > | >
> | > | > | On my machine, /proc/1/status shows that init doesn't have a handler for
> | > | > | non-ignored SIGUNUSED == 31, though.
> | > | > |

> | > | > | But who knows? The kernel promises some guarantees, it is not good to break them.
> | > | > | Perhaps some strange non-standard environment may suffer.
> | > | > |
> | > | > | > We are assuming that the pid namespace init is not doing anything silly and
> | > | > | > I guess it's OK if the consequences are only on the its pid namespace and
> | > | > | > not the whole system.
> | > | > |
> | > | > | The sub-namespace case is very easy afaics, we only need the "signal comes from
> | > | > | the parent namespace" check, not a problem if we make the decision on the sender's
> | > | > | path, like this patch does.
> | > | >
> | > | > Yes, patches 2 and 3 of the set already do the ancestor-ns check. no ?
> | > |
> | > | Yes, I think patches 2-3 are good. But this patch is not. I thought that we
> | > | can ignore the "Blocked signals are never ignored" problem, now I am not sure.
> | > | It is possible that init temporary blocks a signal which it is not going to
> | > | handle.
> | > |
> | > | Perhaps we can do something like the patch below, but I don't like it. With
> | > | this patch, we check the signal handler even if /sbin/init blocks the signal.
> | > | This makes the semantics a bit strange for /sbin/init. Hopefully not a problem
> | > | in practice, but still not good.
> | >
> | > I think this is one step ahead of what we were discussing last week.
> | > A container-init that does not have a handler for a fatal signal would
> | > survive even if the signal is posted when it is blocked.
> | >
> | > |
> | > | Unfortunately, I don't know how to make it better. The problem with blocked
> | > | signals is that we don't know who is the sender of the signal at the time
> | > | when the signal is unblocked.
> | >
> | > One solution I was thinking of was to possibly queue pending blocked
> | > signals to a container init seperately and then requeue them on the
> | > normal queue when signals are unblocked. Its definitely not an easier
> | > solution, but might be less intrusive than the "signal from parent ns
> | > flag" solution.
> |
> | I personally prefer the flag solution just because it will remain
> | clear why it is there, whereas understanding why the separate queue
> | is there will be harder unless it is named something like
> | "child_contaienr_init_blocked_pending_queue".
>
> In my first version of the "flag" solution, I stored the flag in the
> sigqueue structure. The problem with that approach was that if the
> allocation of the sigqueue failed, we would not know if the sender
> was in parent ns. Note that we post a signal to the process (add to
> signals->signal set) even if this allocation fails.

>
> By storing the signal info in 'struct pid_namespace' we avoid having
> to allocate when posting the signal.
>
> I agree that a descriptive name is needed. but since the fields are
> in 'struct pid_namespace' I was thinking 'child' was not necessary.
> Maybe 'cinit' instead of 'container init'. Also 'pending' somehow
> implies a 'queue' - no ?

Sure. So cinit_blocked_pending? Sorry, I see now that that was what
you had :)

Please send a patch when you can. It sounds promising.

> | But still it may be the way to go. Have you coded up a version of this?
>
> I was playing with a slightly different solution that I could modify
> for this. But I can code that up in a couple of days. Just wanted to
> see if it was interesting approach at all.

There is interest in getting this issue solved :)

thanks,
-serge

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