Subject: Re: [PATCH 8/9] signal: Drop signals before sending them to init.
Posted by ebiederm on Tue, 18 Dec 2007 04:06:52 GMT
View Forum Message <> Reply to Message

Oleg Nesterov <oleg@tv-sign.ru> writes:

> On 12/13, Eric W. Biederman wrote:
>>
>> Oleg Nesterov <oleg@tv-sign.ru> writes:
>>
>> > OK, if we change the semantics for /sbin/init signals we can avoid
>> > a lot of problems,
>>
>> Yes. Otherwise we must track the source of the signals.
>
> Yes.

Good. We have fundamental agreement that we should slightly
change the definition of how signals to init are dropped.

For me the really important part is that we have a clean
definition so that we can fix bugs in the implementation.
Rather then declaring a particular implementation is the
definition and getting lost there.

>> > Well, I am not sure about "explain" though. Unless I missed something
>> > this makes the semantics a bit special.
>>
>> Well the semantics are a bit special for init period. I just
>> make them special in a slightly different way.
>
> Yes.
>
> But still I personally can't agree with the new behaviour.

You argument that a program that blocks signals wants them is
compelling, especially since blocking signals is not a common
thing to do.

So I would have no problem with a definition said signals
will be dropped when sent to init if at the time they are
sent the signal is SIG_DFL and unblocked.

>> > Suppose that init does sigtimedwait() but the handler == SIG_DFL.
>>
>> Yes that is a bit surprising. However it is still easy to explain.
>> The signal is never enqueued so sigtimedwait never gets the chance
>> to do anything with it.

>
> But this precisely means that sigtimedwait() is just broken for init.
>
> Another example. /sbin/init execs, and tries to make sure it doesn't
> miss the important signal (say, SIGCHLD). So it blocks SIGCHLD, execs,
> and the new program installs the handler. However, the handler == SIG_DFL
> right after exec, so signal could be missed.
>
> Eric, I think the blocked signal should be enqueued. If application
> blocks the signal, this is the strong indication it does care about
> it, we shouldn't throw it out.

A reasonable argument.

> Yes, this also has surprises. If init unblocks the signal without
> installing the handler, it could be killed. But this can happen with
> your patch as well. sig_init_drop() returns false if we have a handler,
> but this races with sys_rt_sigaction() which can set SIG_DFL, so init
> could be killed.

I am checking under the sighand lock so we should not race,
at least not internally to the kernel.

> IOW, I still have a strong feeling that this patch
>
> http://marc.info/?l=linux-kernel&m=118753610515859
>
> is better, and more correct. That said, this all is very subjective,
> I can't "prove" this of course.

My fundamental problem with that patch is that it drops signals
after we have started processing them, and it modifies the code
of an optimization.

To have a clean definition and clean semantics I think we need
to drop the signal earlier in the path. Which is what I
really object to in your patch.

> Yes sure, the "good" init shouldn't have any problems with any approach.

So the only day to day case we care about are programs that are not
built to be init like /bin/bash running as init. For cases like
this I can see a real advantage in not dropping blocked signals going
to init, they might even use sigtimedwait.

For day to day usage I don't see any practical difference in
dropping or not dropping blocked signals, because the default
is unblocked.

Eric

_____

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