Subject: Re: [PATCH 8/9] signal: Drop signals before sending them to init.
Posted by Oleg Nesterov on Wed, 19 Dec 2007 13:42:46 GMT
View Forum Message <> Reply to Message

On 12/18, Eric W. Biederman wrote:
>
> Oleg Nesterov <oleg@tv-sign.ru> writes:
> >
> >> In the namespace case we can not look at a pending signal and decide
> >> if we should drop it or not. So changing sigaction is impossible.
> >
> > You mean that it is possible that this signal has come from the parent
> > namespace, and so we should die but not just discard the signal.
>
> Yes.
>
> > I think we can ignore this problem. If we had a handler before (when
> > the signal was sent), this is - imho - the correct behaviour. If not
> > then yes, /sbin/init can "accidentally" survive. But the parent namespace
> > can always use SIGKILL to really kill us.
>
> Only because we can't change SIGKILL to SIG_DFL.
>
> Think of force_siginfo and what happens when we stop dropping signals
> on that path. We send the signal and then before we process it
> user space does signal(SIG_DFL), and we drop SIGSEGV. Ouch!

Not a problem.

First of all, this has nothing to do with init's problems, any application
can can do this with signal(SIG_IGN).

The most important case is SIGSEGV sent from do_trap/do_page_fault/etc.
Another sub-thread can "steal" the signal, but this is harmless. The signal
will be re-generated when application returns from the exception and restarts
the faulting instruction.

> > But yes I agree, this changes one corner case to another. And let me
> > repeat, I don't claim that "I am right and you are not", and I can't
> > really prove that my approach is "technically" better. Just a personal
> > feeling about the "better" tradeoff. And I already said my taste is
> > perverted ;)
>
> In this instance I can prove that my choice is better.
>
> When the code is called into question and we must decided if
> a code behavior is a bug or not we require a definition of
> what the code is supposed to do.

>
> Given our technical constraints of not being able to track
> the source of the signal, and needing to appear as a normal
> process to signal senders outside of the pid namespace we
> don't have many choices of definition. The definition that
> I can see is:
>
>   Signals sent to init will be silently dropped without
>   ever being sent to init, when init has the signal
>   handler set to SIG_DFL.
>
> With that definition then any time we process a signal
> in handle_stop_signal or allow the signal to be processed
> in because of ptrace or anything else. We are doing the
> wrong thing.

I never argued, you propose the very simple and understandable definition.

But this simple rule leads to non-obvious and not good consequences.
Imho, of course.

sigtimedwait() is broken, init can lost the signal during exec,
signal(sighandler) is safe but signal(SIG_DFL) is not.

And speaking about ptrace, it is very special anyway. Just look at
get_signal_to_deliver() which re-sends the signal after ptrace_stop().

> That is why I drop the signal earlier.

I can't understand this part of the discussion. What do you mean "earlier" ?
Note that the patch I showed changes handle_stop_signal(). Because I believe
it should be changed anyway to filter out kernel threads at least.

Regardless of which rules we use to drop the signal, I think it is more
natural to modify sig_ignored(), this also makes the patch smaller.

> Oleg if you can show me a definition that permits the behavior
> in your patch we can look at it. Currently I don't believe
> that is possible.
>
> My basic contention is that without a solid definition the code
> is unmaintainable, because we can't tell bugs from features.

No, I can't show.

Eric, let's stop here. I don't believe we can convince each other.
This happens, and of course it is OK to have different opinions.
And in any case, I am happy with this discussion ;)

Let's go with your approach. In any case it solves the real problems
we have.

Oleg.