Subject: Re: [PATCH 8/9] signal: Drop signals before sending them to init.
Posted by ebiederm on Tue, 18 Dec 2007 21:34:26 GMT
View Forum Message <> Reply to Message

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

```
>
>> > @@ -2303,8 +2316,7 @@ int do_sigaction(int sig, struct k_sigac
>> >     *   (for example, SIGCHLD), shall cause the pending signal to
>> >     *   be discarded, whether or not it is blocked"
>> >     */
>> > -  if (act->sa.sa_handler == SIG_IGN ||
>> > -     (act->sa.sa_handler == SIG_DFL && sig_kernel_ignore(sig))) {
>> > +  if (sig_task_ignore(current, sig)) {
>> >     struct task_struct *t = current;
>> >     sigemptyset(&mask);
>> >     sigaddset(&mask, sig);
>>
```

>> Any time you start ignoring the signal here you are not thinking
>> in terms of never sending the signal or not. Once a signal is sent
>> we must treat it like normal (to have a clean definition).
>
> We don't agree here.

Which seems to be the heart of the matter.

>> 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!

> 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.

That is why I drop the signal earlier. That is why I don't
do things in sigaction.

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.

The behavior of your code seems to be about picking convenient spots
in the code today and throwing away the signal there. Doing what is
most convenient now tends to fails miserably when you try for a
completely different implementation.

My basic contention is that without a solid definition the code
is unmaintainable, because we can't tell bugs from features.

Oleg does picking a definition and sticking to it make sense to you?
Do you at least see where I am coming from?

Eric
____