Subject: Re: [PATCH 8/9] signal: Drop signals before sending them to init.
Posted by ebiederm on Thu, 13 Dec 2007 17:50:59 GMT
View Forum Message <> Reply to Message

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

> On 12/12, Eric W. Biederman wrote:
>>
>> By making the rule (for init dropping signals):
>> When sending a signal to init, the presence of a signal handler that
>> is not SIG_DFL allows the signal to be sent to init. If the signal
>> is not sent it is silently dropped without becoming pending.
>
> But isn't it better to modify sig_ignore() and handle_stop_signal()
> instead? This way we seem to need less changes,
>
> http://marc.info/?l=linux-kernel&m=118753610515859
>
> (the patch above itself is not complete and a bit obsolete)

No. That way leads to semantic disaster.

In particular if it is traced or blocked your patch did not
ignore the signal, the signal would be queued up. We would
not know who the sender is and thus not know the proper
disposition for the signal.

For a clean definition that we can maintain into the future
either we must either drop the signal before it is placed on the
queue or otherwise processed. Or we must track the sender.

Tracking the sender is expensive.

>> The only noticeable user space difference from todays init is that it
>> no longer needs to worry about signals becoming pending when it has
>> them marked as SIG_DFL and blocked.
>
> Ugh. I have to apologize again. I got a fever, and it turns out I just
> can't read English.
>
> So, do you mean we can ignore the problems with the signals which are
> currently blocked by /sbin/init?

Yes. Further I am saying those signals will never become pending if
we do not have a signal handler installed.

> I personally agree, but I'm not sure I understand this right.
>

>> +static int sig_init_drop(struct task_struct *tsk, int sig)
>> +{
>> + /* All signals for which init has a SIG_DFL handler are
>> + * silently dropped without being sent.
>> + */
>> + if (!is_sig_init(tsk))
>> + return 0;
>> +
>> + return (tsk->sighand->action[sig-1].sa.sa_handler == SIG_DFL);
>> +}
>
> What if /sbin/init has a handler, but before this signal is delivered
> /sbin/init does signal(SIG_DFL) ? We should modify so_sigaction() to
> prevent this. Note again the patch above.

No. We should treat signals that we process for /sbin/init completely
normally.

If we try and do the right thing with pending signals we have the question
which pid namespaces sent the signal. If the signal came from a decedent
of /sbin/init we should drop the signal. If the signal came from outside
the pid namespace of init we should keep the signal. There is no right
correct if you define what happens that way.

Therefore we make a small change to the definition of where we drop
signals to init. We always unconditionally drop them in the sender
if init has the signal handler set to SIG_DFL. Otherwise the signals
are processed normally.

This gives /sbin/init completely normal signal handling if the signal is
ever enqueued. Something trivial to implement and explain.

Theoretically the new definition can deliver a few more signals to
/sbin/init and see them processed. In practice I do not expect this
to matter. Especially since /sbin/init did ask for the signals at
one time.

Eric

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