Subject: Re: [PATCH 11/15] Signal semantics
Posted by Oleg Nesterov on Sun, 29 Jul 2007 11:23:34 GMT
View Forum Message <> Reply to Message

On 07/27, sukadev@us.ibm.com wrote:
>
> Pavel Emelianov [xemul@openvz.org] wrote:
> | Oleg Nesterov wrote:
> | >>
> | >>@@ -1852,7 +1950,7 @@ relock:
> | >>   * within that pid space. It can of course get signals from
> | >>   * its parent pid space.
> | >>   */
> | >>- if (current == task_child_reaper(current))
> | >>+ if (kinfo.flags & KERN_SIGINFO_CINIT)
> | >>   continue;
> | >
> | >I think the whole idea is broken, it assumes the sender put something into
> | >"struct sigqueue".
> |
> | Yup. That's the problem. It seems to me that the only way to handle init's
> | signals is to check for permissions in the sending path.
>
> We can check permissions in the sending path - and in fact we do check for
> SIGKILL case (deny_signal_to_container_init() below).
>
> But the receiver knows/decides whether or not the signal is wanted/not. No ?

I can't understand your question. Yes, this is what we are doing currently,
but this is broken by this patch.

> Are you saying we should check/special case all fatal signals ?
>
> |
> | >Suppose that /sbin/init has no handler for (say) SIGTERM, and we send this
> | >signal from the same namespace. send_signal() sets SIGQUEUE_CINIT, but it
> | >is lost because __group_complete_signal() silently "converts" sig_fatal()
> | >signals to SIGKILL using sigaddset().
>
> Yes, I should have called it out, but this patch currently assumes /sbin/init
> (or container-init) has a handler for the fatal signals like SIGTERM

Changelog says nothing about that. And in that case we don't need any complications
except a) deny_signal_to_container_init() (should be named deny_SIGKILL_to_container_init)
and b) "cross-namespace signals must have si_code == SI_KERNEL".

I don't know whether this limitation (/sbin/init must install the handler
for each fatal signal) acceptable or not.

However, we should also take care about sig_kernel_stop() signals, and please
note that it is not possible to install a handler for SIGSTOP.

> | >>+static void encode_sender_info(struct task_struct *t, struct sigqueue *q)
> | >>+{
> | >>+ if (pid_ns_equal(t)) {
> | >>+ if (is_container_init(t)) {
> | >>+   q->flags |= SIGQUEUE_CINIT;
> | >
> | >Ironically, this change carefully preserves the bug we already have :)
> | >
> | >This doesn't protect init from "bad" signal if we send it to sub-thread
> | >of init. Actually, this make the behaviour a bit worse compared to what
> | >we currently have. Currently, at least the main init's thread survives
> | >if we send SIGKILL to sub-thread.
>
> Do you mean "init's main thread" ?

Yes.

> But doesn't SIGKILL to any thread kill
> the entire process ?

It should, but it doesn't if it was sent to init's sub-thread, exactly
because of child_reaper() check in get_signal_to_deliver().

> | >>+   error = deny_signal_to_container_init(t, sig);
> | >>+   if (error)
> | >>+   return error;
> | >
> | >Hm. Could you explain this change? Why do we need a special check for
> | >SIGKILL?
>
> As you pointed out above, SIGKILL goes through the __group_complete_signal()/
> sigaddset() path and bypasses/loses the KERN_SIGINFO_CINIT flag. Other
> sig_fatal() signals take this path too, but we assume for now, container-init
> has a handler.

No, SIGKILL doesn't bypasses send_signal(). IOW, if other parts of this patch
were correct, we don't need this change. If init has a handler, we don't neeed
other parts.

> | >(What about ptrace_attach() btw? If it is possible to send a signal to init
> | > from the "parent" namespace, perhaps it makes sense to allow ptracing as
> | > well).
> |
> | ptracing of tasks fro different namespaces is not possible at all, since

> | strace utility determines the fork()-ed child pid from the parent's eax
> | register, which would contain the pid value as this parent sees his child.
> | But if the strace is in different namespace - it won't be able to find
> | this child with the pid value from parent's eax.
> |
> | Maybe it's worth disabling cross-namespaces ptracing...
>
> I think so too. Its probably not a serious limitation ?

My question was not clear, sorry. And I was confused because I had a false impression that ptrace_attach() was already changed to use is_container_init().

Afaics, the cross-namespaces ptracing should work (modulo fork() problem pointed out by Pavel), and probably it is useful.

But we should fix ptrace_attach(), it should not be possible to do PTRACE_ATTACH to /sbin/init from the _same_ namespace.

Oleg.
