Subject: Re: [PATCH 11/15] Signal semantics
Posted by Oleg Nesterov on Fri, 27 Jul 2007 12:30:12 GMT
View Forum Message <> Reply to Message

Damn. I don't have time to read these patches today (will try tomorrow),
but when I glanced at this patch yesterday I had some suspicions...

On 07/26, Pavel Emelyanov wrote:

```
>
> +++ linux-2.6.23-rc1-mm1-7/kernel/signal.c 2007-07-26
> 16:36:37.000000000 +0400
> @@ -323,6 +325,9 @@ static int collect_signal(int sig, struc
>  if (first) {
>   list_del_init(&first->list);
>   copy_siginfo(info, &first->info);
> +  if (first->flags & SIGQUEUE_CINIT)
> +   kinfo->flags |= KERN_SIGINFO_CINIT;
> +
>
> [...snip...]
>
> @@ -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".

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().

```
> +static void encode_sender_info(struct task_struct *t, struct sigqueue *q)
> +{
> + /*
> +  * If sender (i.e 'current') and receiver have the same active
> +  * pid namespace and the receiver is the container-init, set the
> +  * SIGQUEUE_CINIT flag. This tells the container-init that the
> +  * signal originated in its own namespace and so it can choose
> +  * to ignore the signal.
> +  *
> +  * If the receiver is the container-init of a pid namespace,
> +  * but the sender is from an ancestor pid namespace, the
> +  * container-init cannot ignore the signal. So clear the
> +  * SIGQUEUE_CINIT flag in this case.
> +  *
> +  * Also, if the sender does not have a pid_t in the receiver's
> +  * active pid namespace, set si_pid to 0 and pretend it originated
> +  * from the kernel.
> +  */
> + 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.

```
> static int send_signal(int sig, struct siginfo *info, struct task_struct *t,
>    struct sigpending *signals)
> {
> @@ -710,6 +781,7 @@ static int send_signal(int sig, struct s
>    copy_siginfo(&q->info, info);
>    break;
>   }
> +  encode_sender_info(t, q);
```

We still send the signal if __sigqueue_alloc() fails. In that case, the
dequeued siginfo won't have SIGQUEUE_CINIT/KERN_SIGINFO_CINIT, not good.

```
> @@ -1158,6 +1232,13 @@ static int kill_something_info(int sig,
>
>   read_lock(&tasklist_lock);
>   for_each_process(p) {
> +   /*
> +    * System-wide signals apply only to the sender's
> +    * pid namespace, unless issued from init_pid_ns.
> +    */
> +   if (!task_visible_in_pid_ns(p, my_ns))
> +    continue;
> +
>    if (p->pid > 1 && p->tgid != current->tgid) {
```

This "p->pid > 1" check should die.

```
> +static int deny_signal_to_container_init(struct task_struct *tsk, int sig)
> +{
> + /*
> +  * If receiver is the container-init of sender and signal is SIGKILL
> +  * reject it right-away. If signal is any other one, let the container
> +  * init decide (in get_signal_to_deliver()) whether to handle it or
> +  * ignore it.
> +  */
> + if (is_container_init(tsk) && (sig == SIGKILL) && pid_ns_equal(tsk))
> +  return -EPERM;
> +
> + return 0;
> +}
> +
> /*
>  * Bad permissions for sending the signal
>  */
> @@ -545,6 +584,10 @@ static int check_kill_permission(int sig
>      && !capable(CAP_KILL))
>   return error;
>
> + 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?

(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).

Oleg.