
Subject: Re: [PATCH 8/9] signal: Drop signals before sending them to init.
Posted by [Oleg Nesterov](#) on Tue, 18 Dec 2007 12:22:41 GMT
[View Forum Message](#) <> [Reply to Message](#)

On 12/17, Eric W. Biederman wrote:

>
> So I would have no problem with a definition said signals
> will be dropped when sent to init if at the time they are
> sent the signal is SIG_DFL and unblocked.

Great!

> > But this can happen with
> > your patch as well. sig_init_drop() returns false if we have a handler,
> > but this races with sys_rt_sigaction() which can set SIG_DFL, so init
> > could be killed.
>
> I am checking under the sighand lock so we should not race,
> at least not internally to the kernel.

Yes, but as soon as we drop ->siglock /sbin/init can set SIG_DFL before noticing the signal.

> > IOW, I still have a strong feeling that this patch
> >
> > <http://marc.info/?l=linux-kernel&m=118753610515859>
> >
> > is better, and more correct. That said, this all is very subjective,
> > I can't "prove" this of course.
>
> My fundamental problem with that patch is that it drops signals
> after we have started processing them, and it modifies the code
> of an optimization.
>
> To have a clean definition and clean semantics I think we need
> to drop the signal earlier in the path. Which is what I
> really object to in your patch.

Hmm. Could you look at this patch again? I'm attaching it at the end.
(re-diffed against the current code)

It modifies sig_ignored(), so we drop the signal before we started processing. And in fact it is more "optimized", because we don't need to check sa_handler twice.

Btw. I don't think we should change force_sig_info(). Suppose that init blocks/ignores SIGSEGV and do_page_fault() does force_sig_info_fault(). In that case it is better to die explicately than go into the endless

loop.

Oleg.

--- t/kernel/signal.c~INITSIGS 2007-08-19 14:39:35.000000000 +0400

+++ t/kernel/signal.c 2007-08-19 19:00:27.000000000 +0400

@ @ -39,11 +39,33 @ @

```
static struct kmem_cache *sigqueue_cache;

+static int sig_init_ignore(struct task_struct *tsk)
+{
+ if (likely(!is_container_init(tsk->group_leader)))
+ return 0;
+
+ // ----- Multiple pid namespaces -----
+ // if (current is from tsk's parent pid_ns && !in_interrupt())
+ // return 0;
+
+ return 1;
+}
+
+static int sig_task_ignore(struct task_struct *tsk, int sig)
+{
+ void __user * handler = tsk->sigand->action[sig-1].sa.sa_handler;
+
+ if (handler == SIG_IGN)
+ return 1;
+
+ if (handler != SIG_DFL)
+ return 0;
+
+ return sig_kernel_ignore(sig) || sig_init_ignore(tsk);
+}

static int sig_ignored(struct task_struct *t, int sig)
{
- void __user * handler;
-
- /*
-  * Tracers always want to know about signals..
-  */
@@ -58,10 +82,7 @@ static int sig_ignored(struct task_struct
if (sigismember(&t->blocked, sig) || sigismember(&t->real_blocked, sig))
return 0;

- /* Is it explicitly or implicitly ignored? */
- handler = t->sigand->action[sig-1].sa.sa_handler;
```

```

- return handler == SIG_IGN ||
- (handler == SIG_DFL && sig_kernel_ignore(sig));
+ return sig_task_ignore(t, sig);
}

/*
@@ -566,6 +587,9 @@ static void handle_stop_signal(int sig,
*/
return;

+ if (sig_init_ignore(p))
+ return;
+
+ if (sig_kernel_stop(sig)) {
/*
* This is a stop signal. Remove SIGCONT from all queues.
@@ -1786,12 +1810,6 @@ relock:
if (sig_kernel_ignore(signr)) /* Default is nothing. */
continue;

- /*
- * Global init gets no signals it doesn't want.
- */
- if (is_global_init(current))
- continue;
-
+ if (sig_kernel_stop(signr)) {
/*
* The default action is to stop all threads in
@@ -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);

```

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>
