Subject: [PATCH 9/9] signal: Ignore signals sent to the pid namespace init
Posted by ebiederm on Wed, 12 Dec 2007 12:58:27 GMT
View Forum Message <> Reply to Message

This patch extends sit_init_drop to take a sender pid so
it can make decisions about dropping signals based on
the sender of the signal.

With knowledge of the signal sender sig_init_drop is extended
to drop signals meant for the pid namespace init processes
as well as the global init process. However only signals
with senders contained in the pid namespace of the init
process are dropped. Ensuring that outside of the pid
namespace the pid namespace init process continues to look
like an ordinary process.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 kernel/signal.c | 24 ++++++++++++++++++---------
 1 files changed, 15 insertions(+), 9 deletions(-)

diff --git a/kernel/signal.c b/kernel/signal.c
index 029a45d..074905f 100644
--- a/kernel/signal.c
+++ b/kernel/signal.c
@@ -64,20 +64,26 @@ static int sig_ignored(struct task_struct *t, int sig)
   (handler == SIG_DFL && sig_kernel_ignore(sig));
 }

-static int is_sig_init(struct task_struct *tsk)
+static int is_sig_init(struct task_struct *init, struct pid *sender)
 {
- if (likely(!is_global_init(tsk->group_leader)))
+ if (!is_container_init(init))
+ return 0;
+
+ if (!sender)
+ sender = task_tgid(current);
+
+ if (!pid_in_pid_ns(sender, task_active_pid_ns(init)))
   return 0;

  return 1;
 }

-static int sig_init_drop(struct task_struct *tsk, int sig)
+static int sig_init_drop(struct task_struct *tsk, int sig, struct pid *sender)
 {

```
  /* All signals for which init has a SIG_DFL handler are
   * silently dropped without being sent.
   */
- if (!is_sig_init(tsk))
+ if (!is_sig_init(tsk, sender))
   return 0;

  return (tsk->sighand->action[sig-1].sa.sa_handler == SIG_DFL);
@@ -854,7 +860,7 @@ force_sig_info(int sig, struct siginfo *info, struct task_struct *t)

  spin_lock_irqsave(&t->sighand->siglock, flags);
  ret = 0;
- if (sig_init_drop(t, sig))
+ if (sig_init_drop(t, sig, NULL))
   goto out;
  action = &t->sighand->action[sig-1];
  ignored = action->sa.sa_handler == SIG_IGN;
@@ -985,7 +991,7 @@ __group_send_sig_info(int sig, struct siginfo *info, struct task_struct *p,
  int ret = 0;

  assert_spin_locked(&p->sighand->siglock);
- if (sig_init_drop(p, sig))
+ if (sig_init_drop(p, sig, sender))
   return ret;

  handle_stop_signal(sig, p);
@@ -1251,7 +1257,7 @@ send_sig_info(int sig, struct siginfo *info, struct task_struct *p)
  read_lock(&tasklist_lock);
  spin_lock_irqsave(&p->sighand->siglock, flags);
  ret = 0;
- if (!sig_init_drop(p, sig))
+ if (!sig_init_drop(p, sig, NULL))
   ret = specific_send_sig_info(sig, info, p);
  spin_unlock_irqrestore(&p->sighand->siglock, flags);
  read_unlock(&tasklist_lock);
@@ -1420,7 +1426,7 @@ send_group_sigqueue(int sig, struct sigqueue *q, struct task_struct *p)
  read_lock(&tasklist_lock);
  /* Since it_lock is held, p->sighand cannot be NULL. */
  spin_lock_irqsave(&p->sighand->siglock, flags);
- if (sig_init_drop(p, sig)) {
+ if (sig_init_drop(p, sig, NULL)) {
   ret = 1;
   goto out;
  }
@@ -2299,7 +2305,7 @@ static int do_tkill(int tgid, int pid, int sig)
    */
   if (!error && sig && p->sighand) {
    spin_lock_irq(&p->sighand->siglock);
```

```
-   if (!sig_init_drop(p, sig)) {
+   if (!sig_init_drop(p, sig, NULL)) {
     handle_stop_signal(sig, p);
     error = specific_send_sig_info(sig, &info, p);
    }
--
1.5.3.rc6.17.g1911
```