Subject: Re: [PATCH 8/9] signal: Drop signals before sending them to init.
Posted by serue on Wed, 12 Dec 2007 19:00:42 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
>
> Currently init drops all signals including SIGKILL that are sent to it
> that it does not ignore and does have a handler for. Extending this to

Description is crucial in this patchset, so should the above not be:

"does not have a handler for"

?

Otherwise, this whole patchset looks good to me (deferring to Pavel for
his NAK on patch 5 of course).

Acked-by: Serge Hallyn <serue@us.ibm.com>

thanks,
-serge

> pid namespaces where we want to maintain this semantic for signals sent
> from inside the pid namespace (descendents of init) but we want the
> namespace init to appear as a normal process is problematic, as a
> naive approach requires always tracking the sender of a signal.
>
> 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.
>
> 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.
>
> This change by making the presence of a signal handler effectively
> a permission check allows us to do all of the work before we enqueue
> the signal, and there is no need for any fancy tracking of the
> signal sender.
>
> Which means that we can now allow force_sig_info to send signals to
> init, that panic the kernel instead of going into an infinite busy
> loop taking an exception sending a signal and then retaking the same
> exception, eating all of the cpu time but accomplishing nothing.
>
> This change also makes it possible to easily implement the desired

> semantics of /sbin/init for pid namespaces where outer processes can
> kill init but processes inside the pid namespace can not.
>
> While it is now easy to remove the dropping of signals from individual
> code paths such as force_sig_info this patch does not implement that,
> to retain as much of the current behavior as possible. The only
> behavioral difference besides not queuing blocked SIG_DFL signals
> are signals directly added with sigaddset. In practice a threaded init
> now receives SIGKILL sent from a core dump, a thread group exit, or an
> exec shutting down extraneous threads.
>
> This patch was inspired by a patch from Oleg and initial refined
> in conversation with Suka and others on the containers list.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
> ---
> kernel/signal.c | 47 ++++++++++++++++++++++++++++++++++++++---------
> 1 files changed, 38 insertions(+), 9 deletions(-)
>
> diff --git a/kernel/signal.c b/kernel/signal.c
> index c01e3cd..029a45d 100644
> --- a/kernel/signal.c
> +++ b/kernel/signal.c
> @@ -64,6 +64,25 @@ 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)
> +{
> + if (likely(!is_global_init(tsk->group_leader)))
> + return 0;
> +
> + return 1;
> +}
> +
> +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);
> +}
> +
> /*
>   * Re-calculate pending state from the set of locally pending

>   * signals, globally pending signals, and blocked signals.
> @@ -834,6 +853,9 @@ force_sig_info(int sig, struct siginfo *info, struct task_struct *t)
>   struct k_sigaction *action;
>
>   spin_lock_irqsave(&t->sighand->siglock, flags);
> + ret = 0;
> + if (sig_init_drop(t, sig))
> + goto out;
>   action = &t->sighand->action[sig-1];
>   ignored = action->sa.sa_handler == SIG_IGN;
>   blocked = sigismember(&t->blocked, sig);
> @@ -845,6 +867,7 @@ force_sig_info(int sig, struct siginfo *info, struct task_struct *t)
>   }
>   }
>   ret = specific_send_sig_info(sig, info, t);
> +out:
>   spin_unlock_irqrestore(&t->sighand->siglock, flags);
>
>   return ret;
> @@ -962,6 +985,9 @@ __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))
> + return ret;
> +
>   handle_stop_signal(sig, p);
>
>   /* Short-circuit ignored signals. */
> @@ -1224,7 +1250,9 @@ send_sig_info(int sig, struct siginfo *info, struct task_struct *p)
>   */
>   read_lock(&tasklist_lock);
>   spin_lock_irqsave(&p->sighand->siglock, flags);
> - ret = specific_send_sig_info(sig, info, p);
> + ret = 0;
> + if (!sig_init_drop(p, sig))
> + ret = specific_send_sig_info(sig, info, p);
>   spin_unlock_irqrestore(&p->sighand->siglock, flags);
>   read_unlock(&tasklist_lock);
>   return ret;
> @@ -1392,6 +1420,11 @@ 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)) {
> + ret = 1;
> + goto out;

```
> + }
> +
>   handle_stop_signal(sig, p);
>
>   /* Short-circuit ignored signals.  */
> @@ -1844,12 +1877,6 @@ 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
> @@ -2272,8 +2299,10 @@ static int do_tkill(int tgid, int pid, int sig)
>     */
>    if (!error && sig && p->sighand) {
>     spin_lock_irq(&p->sighand->siglock);
> -   handle_stop_signal(sig, p);
> -   error = specific_send_sig_info(sig, &info, p);
> +   if (!sig_init_drop(p, sig)) {
> +    handle_stop_signal(sig, p);
> +    error = specific_send_sig_info(sig, &info, p);
> +   }
>     spin_unlock_irq(&p->sighand->siglock);
>    }
>   }
> --
> 1.5.3.rc6.17.g1911
```