Subject: Re: [PATCH 14/15] Destroy pid namespace on init's death
Posted by Sukadev Bhattiprolu on Wed, 01 Aug 2007 06:16:16 GMT
View Forum Message <> Reply to Message

Oleg Nesterov [oleg@tv-sign.ru] wrote:
| On 07/30, sukadev@us.ibm.com wrote:
| >
| > --- lx26-23-rc1-mm1.orig/kernel/exit.c 2007-07-26 20:08:16.000000000 -0700
| > +++ lx26-23-rc1-mm1/kernel/exit.c 2007-07-30 23:10:30.000000000 -0700
| > @@ -915,6 +915,7 @@ fastcall NORET_TYPE void do_exit(long co
| > {
| > struct task_struct *tsk = current;
| > int group_dead;
| > + struct pid_namespace *pid_ns = tsk->nsproxy->pid_ns;
| >
| > profile_task_exit(tsk);
| >
| > @@ -925,9 +926,10 @@ fastcall NORET_TYPE void do_exit(long co
| > if (unlikely(!tsk->pid))
| > panic("Attempted to kill the idle task!");
| > if (unlikely(tsk == task_child_reaper(tsk))) {
| > - if (task_active_pid_ns(tsk) != &init_pid_ns)
| > - task_active_pid_ns(tsk)->child_reaper =
| > - init_pid_ns.child_reaper;
| > + if (pid_ns != &init_pid_ns) {
| > + zap_pid_ns_processes(pid_ns);
| > + pid_ns->child_reaper = init_pid_ns.child_reaper;
| > + }
| > else
| > panic("Attempted to kill init!");
| > }
|
| Just to remind you, this is not right when init is multi-threaded,
| we should do this only when the last thread exits.

Sorry, I needed to clarify somethings about the multi-threaded init. I
got the impresssion that you were sending a patch for the existing bug,
and meant to review/clarify in the context of the patch.

Anyways, re: requirements for multi-threaded init:

Our current definition of is_container_init() and task_child_reaper()
refer only to the main-thread of the container-init (since they check
for pid_t == 1)

If the main-thread is exiting and is the last thread in the group,
we want terminate other processes in the pid ns (simple case).

If the main thread is exiting, but is not the last thread in the
group, should we let it exit and let the next thread in the group
the reaper of the pid ns ?

Then we would have the pid ns w/o a container-init (i.e reaper
does not have a pid_t == 1, but probably does not matter).

And, when this last thread is exiting, we want to terminate other
processes in the ns right ?

|
| > -static long do_wait(pid_t pid, int options, struct siginfo __user *infop,
| > +long do_wait(pid_t pid, int options, struct siginfo __user *infop,
| > int __user *stat_addr, struct rusage __user *ru)
|
| Small nit, other in-kernel reapers use sys_wait4(), perhaps we can use
| it too, in that case we don't need to export do_wait().

Ok.

|
| > +void zap_pid_ns_processes(struct pid_namespace *pid_ns)
| > +{
| > + int nr;
| > + int rc;
| > + int options = WEXITED|__WALL;
| > +
| > + /*
| > + * We know pid == 1 is terminating. Find remaining pid_ts
| > + * in the namespace, signal them and then wait for them
| > + * exit.
| > + */
| > + nr = next_pidmap(pid_ns, 1);
| > + while (nr > 0) {
| > + kill_proc_info(SIGKILL, SEND_SIG_PRIV, nr);
| > + nr = next_pidmap(pid_ns, nr);
| > + }
|
| Without tasklist_lock held this is not reliable.

Ok. BTW, find_ge_pid() also walks the pidmap, but does not seem to hold
the tasklist_lock. Is that bc its only used in /proc ?

|
| Oleg.