Subject: Re: [PATCH 14/15] Destroy pid namespace on init's death
Posted by Oleg Nesterov on Fri, 03 Aug 2007 10:55:57 GMT
View Forum Message <> Reply to Message

On 08/02, sukadev@us.ibm.com wrote:

```
>
> --- lx26-23-rc1-mm1.orig/kernel/exit.c 2007-08-02 11:06:36.000000000 -0700
> +++ lx26-23-rc1-mm1/kernel/exit.c 2007-08-02 23:06:47.000000000 -0700
> @@ -916,7 +916,32 @@ static inline void exit_child_reaper(str
>   if (likely(tsk->group_leader != task_child_reaper(tsk)))
>    return;
>
> - panic("Attempted to kill init!");
> + if (tsk->nsproxy->pid_ns == &init_pid_ns)
> +  panic("Attempted to kill init!");
> +
> + /*
> +  * @tsk is the last thread in the 'container-init' and is exiting.
> +  * Terminate all remaining processes in the namespace and reap them
> +  * before exiting @tsk.
> +  *
> +  * Note that @tsk (last thread of container-init) may not necessarily
> +  * be the child-reaper (i.e main thread of container-init) of the
> +  * namespace i.e the child_reaper may have already exited.
> +   *
> +  * Even after a child_reaper exits, we let it inherit orphaned children,
> +  * because, pid_ns->child_reaper remains valid as long as there is
> +  * at least one living sub-thread in the container init.
> +
> +  * This living sub-thread of the container-init will be notified when
> +  * a child inherited by the 'child-reaper' exits (do_notify_parent()
> +  * uses __group_send_sig_info()). Further, when reaping child processes,
> +  * do_wait() iterates over children of all living sub threads.
> +
> +  * i.e even though 'child_reaper' thread is listed as the parent of the
> +  * orphaned children, any living sub-thread in the  container-init can
> +  * receive notification of the child exiting and reap the child.
> +  */
```

Great, thanks.

```
> + zap_pid_ns_processes(tsk->nsproxy->pid_ns, tsk);
>  }
>
> +void zap_pid_ns_processes(struct pid_namespace *pid_ns,
> +   struct task_struct *reaper)
> +{
> + int nr;
> + int rc;
> + pid_t reaper_pid = pid_nr_ns(task_pid(reaper), pid_ns);
```

I personally dislike these paramaters. reaper == current, and it is /sbin/init
always. Not because zap_pid_ns_processes() is always called with reaper == current,
but because zap_pid_ns_processes() can't work otherwise: we are using do_wait()
and assume that forget_original_parent() will re-parent threads to use.

And we use it just to figure out reaper_pid, it is used to avoid sending
SIGKILL to us,

```
> + read_lock(&tasklist_lock);
> + nr = next_pidmap(pid_ns, 0);
> + while (nr > 0) {
> +  if (reaper_pid != nr)
> +   kill_proc_info(SIGKILL, SEND_SIG_PRIV, nr);
> +  nr = next_pidmap(pid_ns, nr);
> + }
> + read_unlock(&tasklist_lock);
```

But this doesn't work if we are not ->group_leader (iow, when reaper_pid != 1).
Because in that case we are doing kill_proc_info(SIGKILL, SEND_SIG_PRIV, 1),
which sends the signal to entire thread group, and thus to us (because we are
the last alive thread).

This is harmless (and note that it is possible that current was actually killed
with SIGKILL from the parent namespace), but the code imho looks confusing.

I'd suggest to make zap_pid_ns_processes(void), and start the loop from nr == 1.
Or zap_pid_ns_processes(struct pid_namespace *pid_ns).

Oleg.