
Subject: Re: [PATCH 1/5] Define and use task_active_pid_ns() wrapper
Posted by [Sukadev Bhattiprolu](#) on Mon, 16 Jul 2007 23:02:37 GMT
[View Forum Message](#) <> [Reply to Message](#)

Serge E. Hallyn [serue@us.ibm.com] wrote:

| Quoting sukadev@us.ibm.com (sukadev@us.ibm.com):

| >
| > Subject: [PATCH 1/5] Define and use task_active_pid_ns() wrapper
| >
| > From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
| >
| > With multiple pid namespaces, a process is known by some pid_t in
| > every ancestor pid namespace. Every time the process forks, the
| > child process also gets a pid_t in every ancestor pid namespace.
| >
| > While a process is visible in >=1 pid namespaces, it can see pid_t's
| > in only one pid namespace. We call this pid namespace it's "active
| > pid namespace", and it is always the youngest pid namespace in which
| > the process is known.
| >
| > This patch defines and uses a wrapper to find the active pid namespace
| > of a process. The implementation of the wrapper will be changed in
| > when support for multiple pid namespaces are added.
| >
| > Changelog:
| > 2.6.22-rc4-mm2-pidns1:
| > - [Pavel Emelianov, Alexey Dobriyan] Back out the change to use
| > task_active_pid_ns() in child_reaper() since task->nsproxy
| > can be NULL during task exit (so child_reaper() continues to
| > use init_pid_ns).
| >
| > to implement child_reaper() since init_pid_ns.child_reaper to
| > implement child_reaper() since tsk->nsproxy can be NULL during exit.
| >
| > 2.6.21-rc6-mm1:
| > - Rename task_pid_ns() to task_active_pid_ns() to reflect that a
| > process can have multiple pid namespaces.
| >
| > Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
| > Aacked-by: Pavel Emelianov <xemul@openvz.org>
| >
| > Cc: Eric W. Biederman <ebiederm@xmission.com>
| > Cc: Cedric Le Goater <clg@fr.ibm.com>
| > Cc: Dave Hansen <haveblue@us.ibm.com>
| > Cc: Serge Hallyn <serue@us.ibm.com>
| > Cc: Herbert Poetzel <herbert@13thfloor.at>
| > ---
| > fs/exec.c | 2 +-
| >

```

| > fs/proc/proc_misc.c      | 3 ++-
| > include/linux/pid_namespace.h | 7 ++++++-
| > kernel/exit.c            | 5 +++--
| > kernel/nsproxy.c         | 2 +-
| > kernel/pid.c             | 4 ++--
| > 6 files changed, 15 insertions(+), 8 deletions(-)
| >
| > Index: lx26-22-rc6-mm1/include/linux/pid_namespace.h
| > =====
| > --- lx26-22-rc6-mm1.orig/include/linux/pid_namespace.h 2007-07-13 13:07:01.000000000
-0700
| > +++ lx26-22-rc6-mm1/include/linux/pid_namespace.h 2007-07-13 18:22:49.000000000 -0700
| > @@ -20,7 +20,7 @@ struct pid_namespace {
| > struct pidmap pidmap[PIDMAP_ENTRIES];
| > int last_pid;
| > struct task_struct *child_reaper;
| > - struct kmem_cache_t *pid_cachep;
| > + struct kmem_cache *pid_cachep;
|
| This change is, of course, unrelated to the description.

```

Yes. It fixes a warning.

I had sent a mail earlier to this list about the warning, but I guess that mail did not make it due to our mail server being down on Fri/Sat.

```

|
| > };
| >
| > extern struct pid_namespace init_pid_ns;
| > @@ -39,6 +39,11 @@ static inline void put_pid_ns(struct pid
| > kref_put(&ns->kref, free_pid_ns);
| > }
| >
| > +static inline struct pid_namespace *task_active_pid_ns(struct task_struct *tsk)
| > +{
| > + return tsk->nsproxy->pid_ns;
| > +}
| > +
|
| I trust you've tested this for the NFS oops?

```

The NFS problem we got was when `exit_task_namespaces()` and `exit_notify()` were swapped in `do_exit()`. That change was in a separate patch and Pavel's has a fix for it.

And that patch is not in -mm yet I think.

This patch is a simple short-cut to what the mainline code is already doing.

```
|
| Taking the pid_ns out of the nsproxy was the trigger for the original
| bug, right, to which the solutions were to either take it from struct
| pid, or, so long as pid namespaces couldn't yet be unshared, use
| init_pid_ns?
|
| thanks,
| -serge
|
| > static inline struct task_struct *child_reaper(struct task_struct *tsk)
| > {
| >   return init_pid_ns.child_reaper;
| > Index: lx26-22-rc6-mm1/fs/exec.c
| > =====
| > --- lx26-22-rc6-mm1.orig/fs/exec.c 2007-07-13 13:05:38.000000000 -0700
| > +++ lx26-22-rc6-mm1/fs/exec.c 2007-07-13 18:13:39.000000000 -0700
| > @@ -827,7 +827,7 @@ static int de_thread(struct task_struct
| >   * so it is safe to do it under read_lock.
| >   */
| >   if (unlikely(tsk->group_leader == child_reaper(tsk)))
| > - tsk->nsproxy->pid_ns->child_reaper = tsk;
| > + task_active_pid_ns(tsk)->child_reaper = tsk;
| >
| >   zap_other_threads(tsk);
| >   read_unlock(&tasklist_lock);
| > Index: lx26-22-rc6-mm1/fs/proc/proc_misc.c
| > =====
| > --- lx26-22-rc6-mm1.orig/fs/proc/proc_misc.c 2007-07-13 13:05:38.000000000 -0700
| > +++ lx26-22-rc6-mm1/fs/proc/proc_misc.c 2007-07-13 13:07:48.000000000 -0700
| > @@ -94,7 +94,8 @@ static int loadavg_read_proc(char *page,
| >   LOAD_INT(a), LOAD_FRAC(a),
| >   LOAD_INT(b), LOAD_FRAC(b),
| >   LOAD_INT(c), LOAD_FRAC(c),
| > - nr_running(), nr_threads, current->nsproxy->pid_ns->last_pid);
| > + nr_running(), nr_threads,
| > + task_active_pid_ns(current)->last_pid);
| >   return proc_calc_metrics(page, start, off, count, eof, len);
| > }
| >
| > Index: lx26-22-rc6-mm1/kernel/exit.c
| > =====
| > --- lx26-22-rc6-mm1.orig/kernel/exit.c 2007-07-13 13:06:52.000000000 -0700
| > +++ lx26-22-rc6-mm1/kernel/exit.c 2007-07-13 18:13:39.000000000 -0700
| > @@ -909,8 +909,9 @@ fastcall NORET_TYPE void do_exit(long co
| >   if (unlikely(!tsk->pid))
```

```

|> panic("Attempted to kill the idle task!");
|> if (unlikely(tsk == child_reaper(tsk))) {
|> - if (tsk->nsproxy->pid_ns != &init_pid_ns)
|> - tsk->nsproxy->pid_ns->child_reaper = init_pid_ns.child_reaper;
|> + if (task_active_pid_ns(tsk) != &init_pid_ns)
|> + task_active_pid_ns(tsk)->child_reaper =
|> + init_pid_ns.child_reaper;
|> else
|> panic("Attempted to kill init!");
|> }
|> Index: lx26-22-rc6-mm1/kernel/pid.c
|> =====
|> --- lx26-22-rc6-mm1.orig/kernel/pid.c 2007-07-13 13:07:01.000000000 -0700
|> +++ lx26-22-rc6-mm1/kernel/pid.c 2007-07-13 18:13:38.000000000 -0700
|> @@ -214,7 +214,7 @@ struct pid *alloc_pid(void)
|> int nr = -1;
|> struct pid_namespace *ns;
|>
|> - ns = current->nsproxy->pid_ns;
|> + ns = task_active_pid_ns(current);
|> pid = kmem_cache_alloc(ns->pid_cachep, GFP_KERNEL);
|> if (!pid)
|> goto out;
|> @@ -364,7 +364,7 @@ struct pid *find_ge_pid(int nr)
|> pid = find_pid(nr);
|> if (pid)
|> break;
|> - nr = next_pidmap(current->nsproxy->pid_ns, nr);
|> + nr = next_pidmap(task_active_pid_ns(current), nr);
|> } while (nr > 0);
|>
|> return pid;
|> Index: lx26-22-rc6-mm1/kernel/nsproxy.c
|> =====
|> --- lx26-22-rc6-mm1.orig/kernel/nsproxy.c 2007-07-13 13:05:38.000000000 -0700
|> +++ lx26-22-rc6-mm1/kernel/nsproxy.c 2007-07-13 13:07:48.000000000 -0700
|> @@ -86,7 +86,7 @@ static struct nsproxy *create_new_namesp
|> goto out_ipc;
|> }
|>
|> - new_nsp->pid_ns = copy_pid_ns(flags, tsk->nsproxy->pid_ns);
|> + new_nsp->pid_ns = copy_pid_ns(flags, task_active_pid_ns(tsk));
|> if (IS_ERR(new_nsp->pid_ns)) {
|> err = PTR_ERR(new_nsp->pid_ns);
|> goto out_pid;

```

Containers mailing list
Containers@lists.linux-foundation.org

