Subject: [PATCH 4/9] pid: Generalize task_active_pid_ns
Posted by ebiederm on Wed, 12 Dec 2007 12:46:55 GMT
View Forum Message <> Reply to Message

Currently task_active_pid_ns is not safe to call after a
task becomes a zombie and exit_task_namespaces is called,
as nsproxy becomes NULL. By reading the pid namespace from
the pid of the task we can trivially solve this problem at
the cost of one extra memory read in what should be the
same cacheline as we read the namespace from.

When moving things around I have made task_active_pid_ns
out of line because keeping it in pid_namespace.h would
require adding includes of pid.h and sched.h that I
don't think we want.

This change does make task_active_pid_ns unsafe to call during
copy_process until we attach a pid on the task_struct which
seems to be a reasonable trade off.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
include/linux/pid_namespace.h | 5 +----
kernel/fork.c | 4 ++--
kernel/pid.c | 6 ++++++
3 files changed, 9 insertions(+), 6 deletions(-)

diff --git a/include/linux/pid_namespace.h b/include/linux/pid_namespace.h
index fcd61fa..ab13faa 100644
--- a/include/linux/pid_namespace.h
+++ b/include/linux/pid_namespace.h
@@ -74,10 +74,7 @@ static inline void zap_pid_ns_processes(struct pid_namespace *ns)
}
#endif /* CONFIG_PID_NS */

-static inline struct pid_namespace *task_active_pid_ns(struct task_struct *tsk)
-{
- return tsk->nsproxy->pid_ns;
-}
+extern struct pid_namespace *task_active_pid_ns(struct task_struct *tsk);

static inline struct task_struct *task_child_reaper(struct task_struct *tsk)
{
diff --git a/kernel/fork.c b/kernel/fork.c
index 32d75d8..a210860 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -1164,12 +1164,12 @@ static struct task_struct *copy_process(unsigned long clone_flags,

```
  if (pid != &init_struct_pid) {
   retval = -ENOMEM;
-  pid = alloc_pid(task_active_pid_ns(p));
+  pid = alloc_pid(p->nsproxy->pid_ns);
   if (!pid)
    goto bad_fork_cleanup_namespaces;

   if (clone_flags & CLONE_NEWPID) {
-   retval = pid_ns_prepare_proc(task_active_pid_ns(p));
+   retval = pid_ns_prepare_proc(p->nsproxy->pid_ns);
    if (retval < 0)
     goto bad_fork_free_pid;
   }
diff --git a/kernel/pid.c b/kernel/pid.c
index c507ca7..54d7634 100644
--- a/kernel/pid.c
+++ b/kernel/pid.c
@@ -472,6 +472,12 @@ pid_t task_session_nr_ns(struct task_struct *tsk, struct pid_namespace
*ns)
 }
 EXPORT_SYMBOL(task_session_nr_ns);

+struct pid_namespace *task_active_pid_ns(struct task_struct *tsk)
+{
+ return ns_of_pid(task_pid(tsk));
+}
+EXPORT_SYMBOL_GPL(task_active_pid_ns);
+
 /*
  * Used by proc to find the first pid that is greater then or equal to nr.
  *
--
1.5.3.rc6.17.g1911
```

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers