
Subject: [PATCH] attach_pid() with struct pid parameter
Posted by [Sukadev Bhattiprolu](#) on Thu, 22 Feb 2007 23:23:40 GMT
[View Forum Message](#) <> [Reply to Message](#)

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Subject: [PATCH] attach_pid() with struct pid parameter

attach_pid() currently takes a pid_t and then uses find_pid() to find the corresponding struct pid. Sometimes we already have the struct pid. We can then skip find_pid() if attach_pid() were to take a struct pid parameter.

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Cc: Cedric Le Goater <clg@fr.ibm.com>
Cc: Dave Hansen <haveblue@us.ibm.com>
Cc: Serge Hallyn <serue@us.ibm.com>
Cc: containers@lists.osdl.org

fs/exec.c | 2 +-
include/linux/pid.h | 3 +--
kernel/exit.c | 4 +++-
kernel/fork.c | 11 ++++++-----
kernel/pid.c | 9 ++++++---
kernel/sys.c | 2 +-
6 files changed, 18 insertions(+), 13 deletions(-)

Index: lx26-20-mm2/fs/exec.c

```
=====
--- lx26-20-mm2.orig/fs/exec.c 2007-02-21 14:31:17.000000000 -0800
+++ lx26-20-mm2/fs/exec.c 2007-02-21 15:33:57.000000000 -0800
@@ -701,7 +701,7 @@ static int de_thread(struct task_struct
 */
 detach_pid(tsk, PIDTYPE_PID);
 tsk->pid = leader->pid;
- attach_pid(tsk, PIDTYPE_PID, tsk->pid);
+ attach_pid(tsk, PIDTYPE_PID, find_pid(tsk->pid));
 transfer_pid(leader, tsk, PIDTYPE_PGID);
 transfer_pid(leader, tsk, PIDTYPE_SID);
 list_replace_rcu(&leader->tasks, &tsk->tasks);
Index: lx26-20-mm2/include/linux/pid.h
```

```
=====
--- lx26-20-mm2.orig/include/linux/pid.h 2007-02-21 14:31:25.000000000 -0800
+++ lx26-20-mm2/include/linux/pid.h 2007-02-21 15:33:57.000000000 -0800
@@ -76,8 +76,7 @@ extern struct pid *get_task_pid(struct t
 * write-held.
 */
 extern int FASTCALL(attach_pid(struct task_struct *task,
- enum pid_type type, int nr));
```

```
-
+ enum pid_type type, struct pid *pid));
extern void FASTCALL(detach_pid(struct task_struct *task, enum pid_type));
extern void FASTCALL(transfer_pid(struct task_struct *old,
    struct task_struct *new, enum pid_type));
```

Index: lx26-20-mm2/kernel/exit.c

```
=====
--- lx26-20-mm2.orig/kernel/exit.c 2007-02-21 14:31:26.000000000 -0800
+++ lx26-20-mm2/kernel/exit.c 2007-02-21 15:33:57.000000000 -0800
@@ -299,12 +299,12 @@ void __set_special_pids(pid_t session, p
    if (process_session(curr) != session) {
        detach_pid(curr, PIDTYPE_SID);
        set_signal_session(curr->signal, session);
-   attach_pid(curr, PIDTYPE_SID, session);
+   attach_pid(curr, PIDTYPE_SID, find_pid(session));
    }
    if (process_group(curr) != pgrp) {
        detach_pid(curr, PIDTYPE_PGID);
        curr->signal->pgrp = pgrp;
-   attach_pid(curr, PIDTYPE_PGID, pgrp);
+   attach_pid(curr, PIDTYPE_PGID, find_pid(pgrp));
    }
}
```

Index: lx26-20-mm2/kernel/fork.c

```
=====
--- lx26-20-mm2.orig/kernel/fork.c 2007-02-21 14:31:26.000000000 -0800
+++ lx26-20-mm2/kernel/fork.c 2007-02-21 15:33:57.000000000 -0800
@@ -1254,16 +1254,19 @@ static struct task_struct *copy_process(
    tracehook_init_task(p);

    if (thread_group_leader(p)) {
+   pid_t pgid = process_group(current);
+   pid_t sid = process_session(current);
+
        p->signal->tty = current->signal->tty;
-   p->signal->pgrp = process_group(current);
+   p->signal->pgrp = pgid;
        set_signal_session(p->signal, process_session(current));
-   attach_pid(p, PIDTYPE_PGID, process_group(p));
-   attach_pid(p, PIDTYPE_SID, process_session(p));
+   attach_pid(p, PIDTYPE_PGID, find_pid(pgid));
+   attach_pid(p, PIDTYPE_SID, find_pid(sid));

        list_add_tail_rcu(&p->tasks, &init_task.tasks);
        __get_cpu_var(process_counts)++;
    }
-   attach_pid(p, PIDTYPE_PID, p->pid);
```

```
+ attach_pid(p, PIDTYPE_PID, find_pid(p->pid));
  nr_threads++;
}
```

Index: lx26-20-mm2/kernel/pid.c

```
=====
--- lx26-20-mm2.orig/kernel/pid.c 2007-02-04 10:44:54.000000000 -0800
+++ lx26-20-mm2/kernel/pid.c 2007-02-21 15:33:57.000000000 -0800
@@ -247,13 +247,16 @@ struct pid * fastcall find_pid(int nr)
}
EXPORT_SYMBOL_GPL(find_pid);

-int fastcall attach_pid(struct task_struct *task, enum pid_type type, int nr)
+/*
+ * attach_pid() must be called with the tasklist_lock write-held.
+ */
+int fastcall attach_pid(struct task_struct *task, enum pid_type type,
+ struct pid *pid)
{
  struct pid_link *link;
- struct pid *pid;

  link = &task->pids[type];
- link->pid = pid = find_pid(nr);
+ link->pid = pid;
  hlist_add_head_rcu(&link->node, &pid->tasks[type]);

  return 0;

```

Index: lx26-20-mm2/kernel/sys.c

```
=====
--- lx26-20-mm2.orig/kernel/sys.c 2007-02-21 14:31:27.000000000 -0800
+++ lx26-20-mm2/kernel/sys.c 2007-02-21 15:33:57.000000000 -0800
@@ -1486,7 +1486,7 @@ asmlinkage long sys_setpgid(pid_t pid, p
  if (process_group(p) != pgid) {
    detach_pid(p, PIDTYPE_PGID);
    p->signal->pgrp = pgid;
-   attach_pid(p, PIDTYPE_PGID, pgid);
+   attach_pid(p, PIDTYPE_PGID, find_pid(pgid));
  }

  err = 0;

```

Containers mailing list

Containers@lists.osdl.org

<https://lists.osdl.org/mailman/listinfo/containers>
