
Subject: [PATCH] Rename attach_pid() to find_attach_pid()
Posted by [Sukadev Bhattiprolu](#) on Thu, 11 Jan 2007 13:02:35 GMT
[View Forum Message](#) <> [Reply to Message](#)

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>

attach_pid() currently takes a pid_t parameter and uses find_pid() to find the struct pid associated with the pid_t. With containers, we sometimes already have the struct pid and could skip the find_pid() - if we have a version of attach_pid() that takes a struct pid parameter.

This patch renames the attach_pid() to find_attach_pid(). My next patch defines attach_pid() to take a struct pid.

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 | 4 ++--
kernel/exit.c | 4 ++--
kernel/fork.c | 6 +++--
kernel/pid.c | 3 ++-
kernel/sys.c | 2 +-
6 files changed, 11 insertions(+), 10 deletions(-)

Index: lx26-20-rc2-mm1/fs/exec.c

=====

```
--- lx26-20-rc2-mm1.orig/fs/exec.c 2007-01-10 14:16:17.000000000 -0800
+++ lx26-20-rc2-mm1/fs/exec.c 2007-01-10 21:42:43.000000000 -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);
+ find_attach_pid(tsk, PIDTYPE_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-rc2-mm1/include/linux/pid.h

=====

```
--- lx26-20-rc2-mm1.orig/include/linux/pid.h 2007-01-10 14:16:18.000000000 -0800
+++ lx26-20-rc2-mm1/include/linux/pid.h 2007-01-11 04:44:06.674046656 -0800
@@ -72,10 +72,10 @@ extern struct task_struct *FASTCALL(get_
 extern struct pid *get_task_pid(struct task_struct *task, enum pid_type type);
```

```

/*
- * attach_pid() and detach_pid() must be called with the tasklist_lock
+ * find_attach_pid() and detach_pid() must be called with the tasklist_lock
  * write-held.
 */
-extern int FASTCALL(attach_pid(struct task_struct *task,
+extern int FASTCALL(find_attach_pid(struct task_struct *task,
    enum pid_type type, int nr));

```

```

extern void FASTCALL(detach_pid(struct task_struct *task, enum pid_type));
Index: lx26-20-rc2-mm1/kernel/exit.c

```

```

=====
--- lx26-20-rc2-mm1.orig/kernel/exit.c 2007-01-10 14:16:18.000000000 -0800
+++ lx26-20-rc2-mm1/kernel/exit.c 2007-01-10 21:43:09.000000000 -0800
@@ -301,12 +301,12 @@ 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);
+    find_attach_pid(curr, PIDTYPE_SID, session);
    }
    if (process_group(curr) != pgrp) {
        detach_pid(curr, PIDTYPE_PGID);
        curr->signal->pgrp = pgrp;
-    attach_pid(curr, PIDTYPE_PGID, pgrp);
+    find_attach_pid(curr, PIDTYPE_PGID, pgrp);
    }
}

```

```

Index: lx26-20-rc2-mm1/kernel/fork.c

```

```

=====
--- lx26-20-rc2-mm1.orig/kernel/fork.c 2007-01-10 14:16:18.000000000 -0800
+++ lx26-20-rc2-mm1/kernel/fork.c 2007-01-11 04:43:33.602074360 -0800
@@ -1248,13 +1248,13 @@ static struct task_struct *copy_process(
    p->signal->tty = current->signal->tty;
    p->signal->pgrp = process_group(current);
    set_signal_session(p->signal, process_session(current));
-    attach_pid(p, PIDTYPE_PGID, process_group(p));
-    attach_pid(p, PIDTYPE_SID, process_session(p));
+    find_attach_pid(p, PIDTYPE_PGID, process_group(p));
+    find_attach_pid(p, PIDTYPE_SID, process_session(p));

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

```

```
nr_threads++;  
}
```

Index: lx26-20-rc2-mm1/kernel/pid.c

```
=====
--- lx26-20-rc2-mm1.orig/kernel/pid.c 2007-01-10 14:15:49.000000000 -0800
+++ lx26-20-rc2-mm1/kernel/pid.c 2007-01-11 04:44:06.674046656 -0800
@@ -247,7 +247,8 @@ 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)
+int fastcall find_attach_pid(struct task_struct *task, enum pid_type type,
+ int nr)
{
    struct pid_link *link;
    struct pid *pid;
```

Index: lx26-20-rc2-mm1/kernel/sys.c

```
=====
--- lx26-20-rc2-mm1.orig/kernel/sys.c 2007-01-10 14:16:18.000000000 -0800
+++ lx26-20-rc2-mm1/kernel/sys.c 2007-01-10 21:45:01.000000000 -0800
@@ -1480,7 +1480,7 @@ 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);
+    find_attach_pid(p, PIDTYPE_PGID, pgid);
    }
```

```
err = 0;
```

Containers mailing list

Containers@lists.osdl.org

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

Subject: Re: [PATCH] Rename attach_pid() to find_attach_pid()

Posted by [serue](#) on Thu, 11 Jan 2007 15:31:24 GMT

[View Forum Message](#) <> [Reply to Message](#)

Looks straightforward enough.

thanks,
-serge

Quoting Sukadev Bhattiprolu (sukadev@us.ibm.com):

>

> From: Sukadev Bhattiprolu <sukadev@us.ibm.com>

```

>
> attach_pid() currently takes a pid_t parameter and uses find_pid() to find
> the struct pid associated with the pid_t. With containers, we sometimes
> already have the struct pid and could skip the find_pid() - if we have a
> version of attach_pid() that takes a struct pid parameter.
>
> This patch renames the attach_pid() to find_attach_pid(). My next patch
> defines attach_pid() to take a struct pid.
>
> 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 | 4 +++
> kernel/exit.c   | 4 +++
> kernel/fork.c   | 6 +++
> kernel/pid.c    | 3 ++
> kernel/sys.c    | 2 +-
> 6 files changed, 11 insertions(+), 10 deletions(-)
>
> Index: lx26-20-rc2-mm1/fs/exec.c
> =====
> --- lx26-20-rc2-mm1.orig/fs/exec.c 2007-01-10 14:16:17.000000000 -0800
> +++ lx26-20-rc2-mm1/fs/exec.c 2007-01-10 21:42:43.000000000 -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);
> + find_attach_pid(tsk, PIDTYPE_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-rc2-mm1/include/linux/pid.h
> =====
> --- lx26-20-rc2-mm1.orig/include/linux/pid.h 2007-01-10 14:16:18.000000000 -0800
> +++ lx26-20-rc2-mm1/include/linux/pid.h 2007-01-11 04:44:06.674046656 -0800
> @@ -72,10 +72,10 @@ extern struct task_struct *FASTCALL(get_
> extern struct pid *get_task_pid(struct task_struct *task, enum pid_type type);
>
> /*
> - * attach_pid() and detach_pid() must be called with the tasklist_lock
> + * find_attach_pid() and detach_pid() must be called with the tasklist_lock
>  * write-held.

```

```

> */
> -extern int FASTCALL(attach_pid(struct task_struct *task,
> +extern int FASTCALL(find_attach_pid(struct task_struct *task,
>     enum pid_type type, int nr));
>
> extern void FASTCALL(detach_pid(struct task_struct *task, enum pid_type));
> Index: lx26-20-rc2-mm1/kernel/exit.c
> =====
> --- lx26-20-rc2-mm1.orig/kernel/exit.c 2007-01-10 14:16:18.000000000 -0800
> +++ lx26-20-rc2-mm1/kernel/exit.c 2007-01-10 21:43:09.000000000 -0800
> @@ -301,12 +301,12 @@ 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);
> + find_attach_pid(curr, PIDTYPE_SID, session);
> }
> if (process_group(curr) != pgrp) {
> detach_pid(curr, PIDTYPE_PGID);
> curr->signal->pgrp = pgrp;
> - attach_pid(curr, PIDTYPE_PGID, pgrp);
> + find_attach_pid(curr, PIDTYPE_PGID, pgrp);
> }
> }
>
> Index: lx26-20-rc2-mm1/kernel/fork.c
> =====
> --- lx26-20-rc2-mm1.orig/kernel/fork.c 2007-01-10 14:16:18.000000000 -0800
> +++ lx26-20-rc2-mm1/kernel/fork.c 2007-01-11 04:43:33.602074360 -0800
> @@ -1248,13 +1248,13 @@ static struct task_struct *copy_process(
> p->signal->tty = current->signal->tty;
> p->signal->pgrp = process_group(current);
> set_signal_session(p->signal, process_session(current));
> - attach_pid(p, PIDTYPE_PGID, process_group(p));
> - attach_pid(p, PIDTYPE_SID, process_session(p));
> + find_attach_pid(p, PIDTYPE_PGID, process_group(p));
> + find_attach_pid(p, PIDTYPE_SID, process_session(p));
>
> list_add_tail_rcu(&p->tasks, &init_task.tasks);
> __get_cpu_var(process_counts)++;
> }
> - attach_pid(p, PIDTYPE_PID, p->pid);
> + find_attach_pid(p, PIDTYPE_PID, p->pid);
> nr_threads++;
> }
>
> Index: lx26-20-rc2-mm1/kernel/pid.c
> =====

```

```

> --- lx26-20-rc2-mm1.orig/kernel/pid.c 2007-01-10 14:15:49.000000000 -0800
> +++ lx26-20-rc2-mm1/kernel/pid.c 2007-01-11 04:44:06.674046656 -0800
> @@ -247,7 +247,8 @@ 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)
> +int fastcall find_attach_pid(struct task_struct *task, enum pid_type type,
> +    int nr)
> {
>     struct pid_link *link;
>     struct pid *pid;
> Index: lx26-20-rc2-mm1/kernel/sys.c
> =====
> --- lx26-20-rc2-mm1.orig/kernel/sys.c 2007-01-10 14:16:18.000000000 -0800
> +++ lx26-20-rc2-mm1/kernel/sys.c 2007-01-10 21:45:01.000000000 -0800
> @@ -1480,7 +1480,7 @@ 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);
>     + find_attach_pid(p, PIDTYPE_PGID, pgid);
>     }
>
>     err = 0;

```

Containers mailing list
Containers@lists.osdl.org
<https://lists.osdl.org/mailman/listinfo/containers>
