Subject: [PATCH] allow a task to join a pid namespace
Posted by Glauber Costa on Mon, 04 Jun 2012 13:33:48 GMT

View Forum Message <> Reply to Message

Currently, it is possible for a process to join existing
net, uts and ipc namespaces. This patch allows a process to join an
existing pid namespace as well.

For that to remain sane, some restrictions are made in the calling process:

* It needs to be in the parent namespace of the namespace it wants to jump to
* It needs to sit in its own session and group as a leader.

The rationale for that, is that people want to trigger actions in a Container
from the outside. For instance, mainstream linux recently gained the ability
to safely reboot a container. It would be desirable, however, that this
action is triggered from an admin in the outside world, very much like a
power switch in a physical box.

This would also allow us to connect a console to the container, provide a
repair mode for setups without networking (or with a broken one), etc.

Signed-off-by: Glauber Costa <glommer@parallels.com>
CC: Serge Hallyn <serge.hallyn@canonical.com>
CC: Oleg Nesterov <oleg@redhat.com>
CC: Michael Kerrisk <mtk.manpages@gmail.com>
CC: "Eric W. Biederman" <ebiederm@xmission.com>
CC: Tejun Heo <tj@kernel.org>
CC: Daniel Lezcano <daniel.lezcano@linaro.org>
---
 fs/proc/namespaces.c | 3 ++
 include/linux/proc_fs.h | 1 +
 kernel/pid_namespace.c | 76 +++
 3 files changed, 80 insertions(+)

diff --git a/fs/proc/namespaces.c b/fs/proc/namespaces.c
index 0d9e23a..6b52af5 100644
--- a/fs/proc/namespaces.c
+++ b/fs/proc/namespaces.c
@@ -24,6 +24,9 @@ static const struct proc_ns_operations *ns_entries[] = {
 #ifdef CONFIG_IPC_NS
  &ipcns_operations,
 #endif
+#ifdef CONFIG_PID_NS
+ &pidns_operations,
+#endif
 };

static const struct file_operations ns_file_operations = {
diff --git a/include/linux/proc_fs.h b/include/linux/proc_fs.h
index 3fd2e87..acaafcd 100644
--- a/include/linux/proc_fs.h
+++ b/include/linux/proc_fs.h
@@ -251,6 +251,7 @@ struct proc_ns_operations {
 extern const struct proc_ns_operations netns_operations;
 extern const struct proc_ns_operations utsns_operations;
 extern const struct proc_ns_operations ipcns_operations;
+extern const struct proc_ns_operations pidns_operations;

 union proc_op {
  int (*proc_get_link)(struct dentry *, struct path *);
diff --git a/kernel/pid_namespace.c b/kernel/pid_namespace.c
index 57bc1fd..c4555b9d 100644
--- a/kernel/pid_namespace.c
+++ b/kernel/pid_namespace.c
@@ -258,3 +258,79 @@ static __init int pid_namespaces_init(void)
 }

 __initcall(pid_namespaces_init);
+
+static void *pidns_get(struct task_struct *task)
+{
+ struct pid_namespace *pid = NULL;
+ struct nsproxy *nsproxy;
+
+ rcu_read_lock();
+ nsproxy = task_nsproxy(task);
+ if (nsproxy)
+ pid = get_pid_ns(nsproxy->pid_ns);
+ rcu_read_unlock();
+
+ return pid;
+}
+
+static void pidns_put(void *ns)
+{
+ put_pid_ns(ns);
+}
+
+/*
+ * pid_ns' callback for setns
+ *
+ * this call switches current's pid_ns from nsproxy to ns.
+ * In order to do that successfully, we need to create a new pid living
+ * in the new namespace, and attach_pid() it.
+ *

```
+ * Because we don't want to deal with processes leaving their current
+ * namespace or being duplicate, it is mandatory that the namespace
+ * we're switching from is the parent of the namespace we are switching to.
+ * This is because in this scenario, a view of the pid exists there anyway.
+ *
+ * Caller must be group and session leader. This restriction guarantees
+ * that we won't mess with more than we should, like the controlling terminal
+ * in our host namespace, and ambiguities about who is the child reaper.
+ */
+static int pidns_install(struct nsproxy *nsproxy, void *_ns)
+{
+ struct pid *newpid;
+ struct pid_namespace *ns = _ns;
+
+ if (is_container_init(current))
+  return -EINVAL;
+
+ if (nsproxy->pid_ns != ns->parent)
+  return -EPERM;
+
+ if (task_pgrp(current) != task_pid(current))
+  return -EPERM;
+
+ if (task_session(current) != task_pid(current))
+  return -EPERM;
+
+ newpid = alloc_pid(ns);
+ if (!newpid)
+  return -ENOMEM;
+
+ put_pid_ns(nsproxy->pid_ns);
+ nsproxy->pid_ns = get_pid_ns(ns);
+
+ write_lock_irq(&tasklist_lock);
+ change_pid(current, PIDTYPE_PID, newpid);
+ change_pid(current, PIDTYPE_PGID, newpid);
+ change_pid(current, PIDTYPE_SID, newpid);
+ write_unlock_irq(&tasklist_lock);
+
+ return 0;
+}
+
+const struct proc_ns_operations pidns_operations = {
+ .name  = "pid",
+ .type  = CLONE_NEWPID,
+ .get  = pidns_get,
+ .put  = pidns_put,
+ .install = pidns_install,
```

```
+};
--
1.7.10.2
```