Subject: Re: [PATCH 3/9] pid: Implement ns_of_pid.
Posted by Sukadev Bhattiprolu on Thu, 13 Dec 2007 03:28:27 GMT
View Forum Message <> Reply to Message

Eric W. Biederman [ebiederm@xmission.com] wrote:
| sukadev@us.ibm.com writes:
|
| >
| > My patch refers to this function as pid_active_pid_ns() - I have
| > been meaning to send that out on top of your signals patch.
| > Since a pid has many namespaces, we have been using 'active pid ns'
| > to refer to this ns.
|
| Currently we don't ask for any of the others, and the namespace
| the pid came from is special. That fundamentally is the namespace
| of the pid. The rest byproducts of being in that pid namespace,
| as we could derive them by walking the namespace's parent list.
|
| > Even your next patch modifies task_active_pid_ns() to use this.
| > So can we rename this functio to pid_active_pid_ns() ?
|
| I'd be more inclined to rename task_active_pid_ns to task_pid_ns.
|
| And to rename pid_in_pid_ns that Pavel has issues with to pid_in_ns.
|
| When I read active_pid_ns I wonder what the other namespaces are
| that we are distinguishing this from. They do exist in the
| implementation but so far it is a complete don't care.

Well, there are interfaces like pid_nr_ns() and pid_in_ns() and
task_in_pid_ns() that imply existence of other namespaces and
for that reason we added 'active' in the name.

But I am fine with the terse name and of course we should remove
the the 'active' in task_active_pid_ns() also.

|
| So I expect being as terse as we can while still conveying all of the
| relevant information is the most maintainable long term.
|
| Eric

I did some initial testing on your patchset (minus patch 5) and noticed
that it seems to be missing the patch to address kill -1 semantics
(here is the earlier version).

This patch implements task_in_pid_ns and uses it to limit cap_set_all
and sys_kill(-1,) to only those tasks in the current pid namespace.

Without this we have a setup for a very nasty surprise.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 include/linux/pid_namespace.h | 2 ++
 kernel/capability.c | 3 +++
 kernel/pid.c | 11 +++++++++++
 kernel/signal.c | 5 ++++-
 4 files changed, 20 insertions(+), 1 deletions(-)

diff --git a/include/linux/pid_namespace.h b/include/linux/pid_namespace.h
index 0227e68..b454678 100644
--- a/include/linux/pid_namespace.h
+++ b/include/linux/pid_namespace.h
@@ -78,4 +78,6 @@ static inline struct task_struct *task_child_reaper(struct task_struct *tsk)
  return tsk->nsproxy->pid_ns->child_reaper;
 }

+extern int task_in_pid_ns(struct task_struct *tsk, struct pid_namespace *ns);
+
 #endif /* _LINUX_PID_NS_H */
diff --git a/kernel/capability.c b/kernel/capability.c
index efbd9cd..a801016 100644
--- a/kernel/capability.c
+++ b/kernel/capability.c
@@ -125,6 +125,7 @@ static inline int cap_set_all(kernel_cap_t *effective,
          kernel_cap_t *inheritable,
          kernel_cap_t *permitted)
 {
+   struct pid_namespace *pid_ns = current->nsproxy->pid_ns;
      struct task_struct *g, *target;
      int ret = -EPERM;
      int found = 0;
@@ -132,6 +133,8 @@ static inline int cap_set_all(kernel_cap_t *effective,
      do_each_thread(g, target) {
           if (target == current || is_container_init(target->group_leader))
                continue;
+   if (!task_in_pid_ns(target, pid_ns))
+   continue;
           found = 1;
       if (security_capset_check(target, effective, inheritable,
       permitted))
diff --git a/kernel/pid.c b/kernel/pid.c
index f815455..1c332ca 100644
--- a/kernel/pid.c

```
+++ b/kernel/pid.c
@@ -430,6 +430,17 @@ struct pid *find_get_pid(pid_t nr)
  return pid;
 }

+static int pid_in_pid_ns(struct pid *pid, struct pid_namespace *ns)
+{
+ return pid && (ns->level <= pid->level) &&
+  pid->numbers[ns->level].ns == ns;
+}
+
+int task_in_pid_ns(struct task_struct *task, struct pid_namespace *ns)
+{
+ return pid_in_pid_ns(task_pid(task), ns);
+}
+
 pid_t pid_nr_ns(struct pid *pid, struct pid_namespace *ns)
 {
  struct upid *upid;
diff --git a/kernel/signal.c b/kernel/signal.c
index 1200630..8f5a31f 100644
--- a/kernel/signal.c
+++ b/kernel/signal.c
@@ -1147,10 +1147,13 @@ static int kill_something_info(int sig, struct siginfo *info, int pid)
  } else if (pid == -1) {
   int retval = 0, count = 0;
   struct task_struct * p;
+  struct pid_namespace *ns = current->nsproxy->pid_ns;

   read_lock(&tasklist_lock);
   for_each_process(p) {
-   if (p->pid > 1 && !same_thread_group(p, current)) {
+   if (!is_container_init(p) &&
+       !same_thread_group(p, current) &&
+       task_in_pid_ns(p, ns)) {
     int err = group_send_sig_info(sig, info, p);
     ++count;
     if (err != -EPERM)
--
1.5.3.rc6.17.g1911
```