Subject: [PATCH 1/2] pidns: Don't allow new pids after the namespace is dead.
Posted by Daniel Lezcano on Tue, 15 Feb 2011 16:53:43 GMT
View Forum Message <> Reply to Message

From: Eric W. Biederman <ebiederm@xmission.com>

In the case of unsharing or joining a pid namespace, it becomes
possible to attempt to allocate a pid after zap_pid_namespace has
killed everything in the namespace. Close the hole for now by simply
not allowing any of those pid allocations to succeed. At least for
now it is too strange to think about.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
Signed-off-by: Daniel Lezcano <daniel.lezcano@free.fr>
---
include/linux/pid_namespace.h | 1 +
kernel/pid.c | 4 ++++
kernel/pid_namespace.c | 2 ++
3 files changed, 7 insertions(+), 0 deletions(-)

```
diff --git a/include/linux/pid_namespace.h b/include/linux/pid_namespace.h
index 38d1032..b447d37 100644
--- a/include/linux/pid_namespace.h
+++ b/include/linux/pid_namespace.h
@@ -20,6 +20,7 @@ struct pid_namespace {
  struct kref kref;
  struct pidmap pidmap[PIDMAP_ENTRIES];
  int last_pid;
+ atomic_t dead;
  struct task_struct *child_reaper;
  struct kmem_cache *pid_cachep;
  unsigned int level;
diff --git a/kernel/pid.c b/kernel/pid.c
index 39b65b6..e996950 100644
--- a/kernel/pid.c
+++ b/kernel/pid.c
@@ -282,6 +282,10 @@ struct pid *alloc_pid(struct pid_namespace *ns)
  struct pid_namespace *tmp;
  struct upid *upid;

+ pid = NULL;
+ if (atomic_read(&ns->dead))
+  goto out;
+
  pid = kmem_cache_alloc(ns->pid_cachep, GFP_KERNEL);
  if (!pid)
   goto out;
diff --git a/kernel/pid_namespace.c b/kernel/pid_namespace.c
```

```
index e9c9adc..e8ea25d 100644
--- a/kernel/pid_namespace.c
+++ b/kernel/pid_namespace.c
@@ -90,6 +90,7 @@ static struct pid_namespace *create_pid_namespace(struct
pid_namespace *parent_p
  kref_init(&ns->kref);
  ns->level = level;
  ns->parent = get_pid_ns(parent_pid_ns);
+ atomic_set(&ns->dead, 0);

  set_bit(0, ns->pidmap[0].page);
  atomic_set(&ns->pidmap[0].nr_free, BITS_PER_PAGE - 1);
@@ -164,6 +165,7 @@ void zap_pid_ns_processes(struct pid_namespace *pid_ns)
   *
   */
  read_lock(&tasklist_lock);
+ atomic_set(&pid_ns->dead, 1);
  nr = next_pidmap(pid_ns, 1);
  while (nr > 0) {
   rcu_read_lock();
--
1.7.1
```

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containe rs