
Subject: [PATCH 0/4] Properly handle talking to all processes in a pid namespace
Posted by [ebiederm](#) on Wed, 12 Dec 2007 13:46:20 GMT

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

Currently when using -1 to mean all processes or all threads we don't localize that to the pid namespace. This patchset corrects the instances I know of.

Eric

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: [PATCH 1/4] signal: Introduce kill_pid_ns_info
Posted by [ebiederm](#) on Wed, 12 Dec 2007 13:49:47 GMT

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

Implement the basic helper function that walks all of the processes in a pid namespace and sends them all a signal.

Both locations that could use this functions are also updated to use this function.

I use find_ge_pid instead of for_each_process because it has a chance of not touching every process in the system.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

include/linux/sched.h | 2 +
kernel/pid_namespace.c | 17 +-----
kernel/signal.c | 54 ++++++-----
3 files changed, 43 insertions(+), 30 deletions(-)

```
diff --git a/include/linux/sched.h b/include/linux/sched.h
index cda9ee9..eef3c22 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -1665,6 +1665,8 @@ extern void release_task(struct task_struct * p);
extern int send_sig_info(int, struct siginfo *, struct task_struct *);
extern int force_sigsegv(int, struct task_struct *);
extern int force_sig_info(int, struct siginfo *, struct task_struct *);
+extern int __kill_pid_ns_info(int sig, struct siginfo *info, struct pid_namespace *ns);
+extern int kill_pid_ns_info(int sig, struct siginfo *info, struct pid_namespace *ns);
extern int __kill_pgrp_info(int sig, struct siginfo *info, struct pid *pgrp);
extern int kill_pgrp_info(int sig, struct siginfo *info, struct pid *pgrp);
extern int kill_pid_info(int sig, struct siginfo *info, struct pid *pid);
```

```

diff --git a/kernel/pid_namespace.c b/kernel/pid_namespace.c
index 739a72b..67ba069 100644
--- a/kernel/pid_namespace.c
+++ b/kernel/pid_namespace.c
@@ -178,29 +178,14 @@ void free_pid_ns(struct kref *kref)

void zap_pid_ns_processes(struct pid_namespace *pid_ns)
{
- int nr;
  int rc;

  /*
   * The last thread in the cgroup-init thread group is terminating.
   * Find remaining pid_ts in the namespace, signal and wait for them
   * to exit.
-  *
-  * Note: This signals each threads in the namespace - even those that
-  * belong to the same thread group, To avoid this, we would have
-  * to walk the entire tasklist looking a processes in this
-  * namespace, but that could be unnecessarily expensive if the
-  * pid namespace has just a few processes. Or we need to
-  * maintain a tasklist for each pid namespace.
-  */
- read_lock(&tasklist_lock);
- nr = next_pidmap(pid_ns, 1);
- while (nr > 0) {
- kill_proc_info(SIGKILL, SEND_SIG_PRIV, nr);
- nr = next_pidmap(pid_ns, nr);
- }
- read_unlock(&tasklist_lock);
+ kill_pid_ns_info(SIGKILL, SEND_SIG_PRIV, pid_ns);

  do {
    clear_thread_flag(TIF_SIGPENDING);
diff --git a/kernel/signal.c b/kernel/signal.c
index 074905f..1eb0661 100644
--- a/kernel/signal.c
+++ b/kernel/signal.c
@@ -1083,6 +1083,45 @@ int group_send_sig_info(int sig, struct siginfo *info, struct task_struct
 *p)
  return ret;
 }

+int __kill_pid_ns_info(int sig, struct siginfo *info, struct pid_namespace *ns)
+{
+ int retval = 0, count = 0;
+ struct task_struct *p;

```

```

+ struct pid *pid;
+ int nr;
+
+ /* Since there isn't a pid namespace list of tasks use the closet
+  * approximation we have: find_ge_pid.
+  */
+ nr = 0;
+ while ((pid = find_ge_pid(nr + 1, ns))) {
+   int err;
+
+   nr = pid_nr_ns(pid, ns);
+   p = pid_task(pid, PIDTYPE_PID);
+   if (!p || (nr <= 1) || !thread_group_leader(p) ||
+       same_thread_group(p, current))
+     continue;
+
+   err = group_send_sig_info(sig, info, p);
+   ++count;
+   if (err != -EPERM)
+     retval = err;
+ }
+ return count ? retval : -ESRCH;
+}
+
+int kill_pid_ns_info(int sig, struct siginfo *info, struct pid_namespace *ns)
+{
+ int retval;
+
+ read_lock(&tasklist_lock);
+ retval = __kill_pid_ns_info(sig, info, ns);
+ read_unlock(&tasklist_lock);
+
+ return retval;
+}
+
+/*
+ * kill_pgrp_info() sends a signal to a process group: this is what the tty
+ * control characters do (^C, ^Z etc)
+@@ -1204,20 +1243,7 @@ static int kill_something_info(int sig, struct siginfo *info, int pid)
+ if (!pid) {
+   ret = kill_pgrp_info(sig, info, task_pgrp(current));
+ } else if (pid == -1) {
+ - int retval = 0, count = 0;
+ - struct task_struct * p;
+ -
+ - read_lock(&tasklist_lock);
+ - for_each_process(p) {
+ -   if (p->pid > 1 && !same_thread_group(p, current)) {

```

```
- int err = group_send_sig_info(sig, info, p);
- ++count;
- if (err != -EPERM)
-     retval = err;
- }
- }
- read_unlock(&tasklist_lock);
- ret = count ? retval : -ESRCH;
+ ret = kill_pid_ns_info(sig, info, task_active_pid_ns(current));
  } else if (pid < 0) {
    ret = kill_pgrp_info(sig, info, find_vpid(-pid));
  } else {
--
```

1.5.3.rc6.17.g1911

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: Re: [PATCH 1/4] signal: Introduce kill_pid_ns_info
Posted by [Pavel Emelianov](#) on Wed, 12 Dec 2007 16:09:03 GMT
[View Forum Message](#) <> [Reply to Message](#)

Eric W. Biederman wrote:

```
> Implement the basic helper function that walks all of the processes in
> a pid namespace and sends them all a signal.
>
> Both locations that could use this functions are also updated to use
> this function.
>
> I use find_ge_pid instead of for_each_process because it has a chance
> of not touching every process in the system.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
```

Looks sane.

Acked-by: Pavel Emelyanov <xemul@openvz.org>

but one comment inside.

[snip]

```
> diff --git a/kernel/signal.c b/kernel/signal.c
> index 074905f..1eb0661 100644
> --- a/kernel/signal.c
```

```

> +++ b/kernel/signal.c
> @@ -1083,6 +1083,45 @@ int group_send_sig_info(int sig, struct siginfo *info, struct
task_struct *p)
>     return ret;
> }
>
> +int __kill_pid_ns_info(int sig, struct siginfo *info, struct pid_namespace *ns)

```

Shouldn't it be static?

```

> +{
> + int retval = 0, count = 0;
> + struct task_struct *p;
> + struct pid *pid;
> + int nr;
> +
> + /* Since there isn't a pid namespace list of tasks use the closet
> + * approximation we have: find_ge_pid.
> + */
> + nr = 0;
> + while ((pid = find_ge_pid(nr + 1, ns))) {
> +     int err;
> +
> +     nr = pid_nr_ns(pid, ns);
> +     p = pid_task(pid, PIDTYPE_PID);
> +     if (!p || (nr <= 1) || !thread_group_leader(p) ||
> +         same_thread_group(p, current))
> +         continue;
> +
> +     err = group_send_sig_info(sig, info, p);
> +     ++count;
> +     if (err != -EPERM)
> +         retval = err;
> + }
> + return count ? retval : -ESRCH;
> +}
> +

```

[snip]

Thanks,
Pavel

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>
