Subject: [RFC][PATCH] fork: Don't special case CLONE_NEWPID for process or sessions
Posted by ebiederm on Sat, 27 Oct 2007 01:49:48 GMT
View Forum Message <> Reply to Message

Given that the kernel supports sys_setsid we don't need a special case
in fork if we want to set: session == pgrp == pid.

The historical (although not 2.6) linux behavior has been to start the
init with session == pgrp == 0 which is effectively what removing this
special case will do.

Is there any reason why we want/need this special case in fork? Or
can we remove it and save some code, make copy_process easier to read
easier to maintain, and possibly a little faster?

I know it is a little weird belong to a process groups that isn't
visible in your pid namespace, but it there are no good reasons
why it shouldn't work.

I think making this change makes the interface more flexible,
and general.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 kernel/fork.c | 18 +++++----------
 1 files changed, 5 insertions(+), 13 deletions(-)

diff --git a/kernel/fork.c b/kernel/fork.c
index ddafdfa..b0de799 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -1292,20 +1292,12 @@ static struct task_struct *copy_process(unsigned long clone_flags,
   if (thread_group_leader(p)) {
    if (clone_flags & CLONE_NEWPID) {
     p->nsproxy->pid_ns->child_reaper = p;
-    p->signal->tty = NULL;
-    set_task_pgrp(p, p->pid);
-    set_task_session(p, p->pid);
-    attach_pid(p, PIDTYPE_PGID, pid);
-    attach_pid(p, PIDTYPE_SID, pid);
-   } else {
-    p->signal->tty = current->signal->tty;
-    set_task_pgrp(p, task_pgrp_nr(current));
-    set_task_session(p, task_session_nr(current));
-    attach_pid(p, PIDTYPE_PGID,
-      task_pgrp(current));
-    attach_pid(p, PIDTYPE_SID,

```
-      task_session(current));
    }
+   p->signal->tty = current->signal->tty;
+   set_task_pgrp(p, task_pgrp_nr(current));
+   set_task_session(p, task_session_nr(current));
+   attach_pid(p, PIDTYPE_PGID, task_pgrp(current));
+   attach_pid(p, PIDTYPE_SID, task_session(current));

    list_add_tail_rcu(&p->tasks, &init_task.tasks);
    __get_cpu_var(process_counts)++;
--
1.5.3.rc6.17.g1911
```

Subject: Re: [RFC][PATCH] fork: Don't special case CLONE_NEWPID for process or sessions
Posted by Pavel Emelianov on Thu, 01 Nov 2007 09:28:23 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:

Sorry for the late answer, I have just noticed that I forgot to
answer on this patch.

> Given that the kernel supports sys_setsid we don't need a special case
> in fork if we want to set: session == pgrp == pid.
>
> The historical (although not 2.6) linux behavior has been to start the
> init with session == pgrp == 0 which is effectively what removing this
> special case will do.

Hm... I overlooked this fact. Looks like the namespace's init will
have them set to 1.

> Is there any reason why we want/need this special case in fork? Or

Mainly to address the issue I describe below.

> can we remove it and save some code, make copy_process easier to read
> easier to maintain, and possibly a little faster?
>
> I know it is a little weird belong to a process groups that isn't
> visible in your pid namespace, but it there are no good reasons

> why it shouldn't work.

This is not good to have such a situation as the init will have
the ability to kill the tasks from the namespace he can't see,
e.g. his parent and the processes in that group.

> I think making this change makes the interface more flexible,
> and general.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
> ---
> kernel/fork.c | 18 +++++----------
> 1 files changed, 5 insertions(+), 13 deletions(-)
>
> diff --git a/kernel/fork.c b/kernel/fork.c
> index ddafdfa..b0de799 100644
> --- a/kernel/fork.c
> +++ b/kernel/fork.c
> @@ -1292,20 +1292,12 @@ static struct task_struct *copy_process(unsigned long clone_flags,
>   if (thread_group_leader(p)) {
>   if (clone_flags & CLONE_NEWPID) {
>   p->nsproxy->pid_ns->child_reaper = p;
> -   p->signal->tty = NULL;
> -   set_task_pgrp(p, p->pid);
> -   set_task_session(p, p->pid);
> -   attach_pid(p, PIDTYPE_PGID, pid);
> -   attach_pid(p, PIDTYPE_SID, pid);
> -   } else {
> -   p->signal->tty = current->signal->tty;
> -   set_task_pgrp(p, task_pgrp_nr(current));
> -   set_task_session(p, task_session_nr(current));
> -   attach_pid(p, PIDTYPE_PGID,
> -   task_pgrp(current));
> -   attach_pid(p, PIDTYPE_SID,
> -   task_session(current));
>   }
> +   p->signal->tty = current->signal->tty;
> +   set_task_pgrp(p, task_pgrp_nr(current));
> +   set_task_session(p, task_session_nr(current));
> +   attach_pid(p, PIDTYPE_PGID, task_pgrp(current));
> +   attach_pid(p, PIDTYPE_SID, task_session(current));
>
>   list_add_tail_rcu(&p->tasks, &init_task.tasks);
>   __get_cpu_var(process_counts)++;

_____

Containers mailing list
Containers@lists.linux-foundation.org

Subject: Re: [RFC][PATCH] fork: Don't special case CLONE_NEWPID for process or sessions
Posted by ebiederm on Thu, 01 Nov 2007 15:14:09 GMT
View Forum Message <> Reply to Message

Pavel Emelyanov <xemul@openvz.org> writes:

> Eric W. Biederman wrote:
>
> Sorry for the late answer, I have just noticed that I forgot to
> answer on this patch.

Thanks for answering.

>> Given that the kernel supports sys_setsid we don't need a special case
>> in fork if we want to set: session == pgrp == pid.
>>
>> The historical (although not 2.6) linux behavior has been to start the
>> init with session == pgrp == 0 which is effectively what removing this
>> special case will do.
>
> Hm... I overlooked this fact. Looks like the namespace's init will
> have them set to 1.

Yes. It is not a big difference as init can handle being exec'd by
something else, thus is expected to be able to handle the case where
setsid has already been called.

So we are good but your current code makes it impossible to set
tsk->signal->leader and become a proper session leader which is
painful.

>> can we remove it and save some code, make copy_process easier to read
>> easier to maintain, and possibly a little faster?
>>
>> I know it is a little weird belong to a process groups that isn't
>> visible in your pid namespace, but it there are no good reasons
>> why it shouldn't work.
>
> This is not good to have such a situation as the init will have
> the ability to kill the tasks from the namespace he can't see,
> e.g. his parent and the processes in that group.

Yes. sys_kill(0, SIGXXX) will allow this.

As this is the main reason for this I don't see any reason to keep
the current clone behavior.

Sending signals to our process group and our parent is an ability that
we allow even the most untrusted processes normally, and it is an
ability we can easily remove simply by calling setsid.

Not doing magic with the session and the process group allows init
to properly become a session leader when setsid is called.

Starting with a shared session and process group makes it more likely
kernel implementors will look closely to ensure they handle strange
cases like this properly and that developers using CLONE_NEWPID will
look closely to ensure there are not other pid gotchas the need to
deal with.

Sharing the process group, session and controlling tty of our parent
can be an advantage in small scenarios where using an existing
controlling tty is an advantage. Think of a chroot build root or a
chroot rpm install. Not letting processes escape and become deaemons
is an advantage, but it really doesn't matter if they send signals to
their parent.

When isolation is important we do not want the ability to send signals
to outside of the pid namespace. Currently except for the child death
signal of init it appears that simply calling setsid is enough.

So short of any other objections I think I will brush up this patch and
send it along to Andrew.

Eric

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

---

Subject: Re: [RFC][PATCH] fork: Don't special case CLONE_NEWPID for process or sessions
Posted by Pavel Emelianov on Thu, 01 Nov 2007 15:37:51 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> Pavel Emelyanov <xemul@openvz.org> writes:
>
>> Eric W. Biederman wrote:
>>
>> Sorry for the late answer, I have just noticed that I forgot to

>> answer on this patch.
>
> Thanks for answering.
>
>>> Given that the kernel supports sys_setsid we don't need a special case
>>> in fork if we want to set: session == pgrp == pid.
>>>
>>> The historical (although not 2.6) linux behavior has been to start the
>>> init with session == pgrp == 0 which is effectively what removing this
>>> special case will do.
>> Hm... I overlooked this fact. Looks like the namespace's init will
>> have them set to 1.
>
> Yes. It is not a big difference as init can handle being exec'd by
> something else, thus is expected to be able to handle the case where
> setsid has already been called.
>
> So we are good but your current code makes it impossible to set
> tsk->signal->leader and become a proper session leader which is
> painful.
>
>>> can we remove it and save some code, make copy_process easier to read
>>> easier to maintain, and possibly a little faster?
>>>
>>> I know it is a little weird belong to a process groups that isn't
>>> visible in your pid namespace, but it there are no good reasons
>>> why it shouldn't work.
>> This is not good to have such a situation as the init will have
>> the ability to kill the tasks from the namespace he can't see,
>> e.g. his parent and the processes in that group.
>
> Yes. sys_kill(0, SIGXXX) will allow this.
>
> As this is the main reason for this I don't see any reason to keep
> the current clone behavior.

Are you talking about keeping the ability to kill the outer processes?

> Sending signals to our process group and our parent is an ability that
> we allow even the most untrusted processes normally, and it is an
> ability we can easily remove simply by calling setsid.

You mix two things together - letting tasks send signals to their
groups is good, but letting tasks send signals outside the namespace
is bad.

> Not doing magic with the session and the process group allows init
> to properly become a session leader when setsid is called.

>
> Starting with a shared session and process group makes it more likely
> kernel implementors will look closely to ensure they handle strange
> cases like this properly and that developers using CLONE_NEWPID will
> look closely to ensure there are not other pid gotchas the need to
> deal with.
>
> Sharing the process group, session and controlling tty of our parent
> can be an advantage in small scenarios where using an existing
> controlling tty is an advantage. Think of a chroot build root or a
> chroot rpm install. Not letting processes escape and become deaemons
> is an advantage, but it really doesn't matter if they send signals to
> their parent.

Well, we allow a tiny possibility to have shared pids, but do we
really want to support this possibility in the rest of the code?

> When isolation is important we do not want the ability to send signals
> to outside of the pid namespace. Currently except for the child death
> signal of init it appears that simply calling setsid is enough.
>
> So short of any other objections I think I will brush up this patch and
> send it along to Andrew.

Hm... Could you please send it for pre-rfc before then?

> Eric
>

Subject: Re: [RFC][PATCH] fork: Don't special case CLONE_NEWPID for process or sessions
Posted by ebiederm on Thu, 01 Nov 2007 17:03:42 GMT
View Forum Message <> Reply to Message

Pavel Emelyanov <xemul@openvz.org> writes:

>> As this is the main reason for this I don't see any reason to keep
>> the current clone behavior.
>
> Are you talking about keeping the ability to kill the outer processes?

I am talking about keeping the session and pgrp from the outer pid namespace.

Signals going out is less important.

>> Sending signals to our process group and our parent is an ability that
>> we allow even the most untrusted processes normally, and it is an
>> ability we can easily remove simply by calling setsid.
>
> You mix two things together - letting tasks send signals to their
> groups is good, but letting tasks send signals outside the namespace
> is bad.

If we have a case where we can send signals to the parent namespace
that makes some setting si_pid more difficult. So on those grounds
it is technically worth avoiding if we can.

I don't see any problem sharing a session and a process group
optionally with processes outside the pid namespace, and in fact
I find that desirable. So we don't always have to run pid
namespace leaders as daemons. To not be a daemon we need
the session, pgrp, and tty from the outside.

As for sending signals outside since setsid nicely closes that hole
in the cases we want to avoid, I am ambivalent about whether
we should look for a more robust solution to handling si_pid in which
case technical objections go away or if we should disallow the sending
altogether.

If the code can readily handle the full general case

Since si_pid is always task_pid(current) fixing the technical side may
not be much of a problem.

The way namespaces are defined sending signals outside the namespace
is almost impossible (because the pid lookup fails). If we get past
that sending signals is well defined and I don't have a problem with
it.

To support migration and strong virtual servers it is necessary that
we close all of the holes where we can get access to objects outside
of our namespace. In general that doesn't mean we should remove the
possibility for other users.

> Well, we allow a tiny possibility to have shared pids, but do we
> really want to support this possibility in the rest of the code?

There is essentially no cost if things are implemented properly. Just
use struct pid and it works. So I don't see a reason to avoid it.

Eric

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