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

Subject: Re: [PATCH] allow a task to join a pid namespace
Posted by Oleg Nesterov on Mon, 04 Jun 2012 16:51:17 GMT
View Forum Message <> Reply to Message

On 06/04, Glauber Costa wrote:
>
> 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.

I can't understand this patch... but probably I missed something,
I never really understood setns.

> +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;

At least you should also check that current is single-threaded,
I guess.

> +
> + if (task_pgrp(current) != task_pid(current))
> + return -EPERM;
> +
> + if (task_session(current) != task_pid(current))
> + return -EPERM;

Both checks are obviously racy without tasklist.

> + newpid = alloc_pid(ns);
> + if (!newpid)
> + return -ENOMEM;

Hmm. Doesn't this mean that pid_nr of this task (as it seen
in its current namespace) will be changed? This doesn't look
sane.

> + 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);

Hmm. So, until the caller does switch_task_namespaces()
task_active_pid_ns(current) != current->nsproxy->pid_ns,
doesn't look very nice too.

I don't think this can be right. If nothing else, this breaks
it_real_fn().

Oleg.

Subject: Re: [PATCH] allow a task to join a pid namespace
Posted by Daniel Lezcano on Tue, 05 Jun 2012 09:30:08 GMT
View Forum Message <> Reply to Message

On 06/04/2012 06:51 PM, Oleg Nesterov wrote:
> On 06/04, Glauber Costa wrote:
>>
>> 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.
>
> I can't understand this patch... but probably I missed something,
> I never really understood setns.

Hi Oleg,

let me clarify why is needed setns. In the world of container, setns
allows to administrate the container from outside. One good example is
to shutdown the container. The users setup their hosts with the init's
services to startup the containers when the system starts, but they have
no way to invoke 'shutdown' from inside the container when the system
goes down except doing some trick with the signals. The setns syscall
with the pid namespace support will allow to do that.

Also a complete setns support will allow to write some administrative
tools to have a global view of the different separated resources running
in several containers.

For example, if you are the administrator of the host and you have
hundred of containers running on it, you can use setns to run netstat
within each container and build a view of the different network stack.
The same applies for 'ps' or 'top'.

Without setns, things are much more complicated and in some cases,
impossible. For instance, you can run a daemon inside the container,
send command to it and redirect its output to the fifo but that
increase the number of processes and has some limitations. Also that
means the command you want to run is present in the container's FS.

The setns syscall is highly needed for the VRF, where a single process
can handle thousand of network namespaces and switch from a network
namespace to another network namespace with one syscall. The usage of
the file descriptors pins the namespace and prevent it from being
destroyed when switching from one namespace to another.

In other words, +1 for pid ns support with setns :)

-- Daniel

Subject: Re: [PATCH] allow a task to join a pid namespace
Posted by Daniel Lezcano on Tue, 05 Jun 2012 09:36:21 GMT
View Forum Message <> Reply to Message

On 06/04/2012 03:33 PM, Glauber Costa wrote:
> 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.

Hi Glauber,

I am in favor of this patch but I think the pidns support won't be

complete and some corner-cases are not handled.

May be you can look at Eric's patchset [1] where, IMO, everything is
taken into account. Some of the patches may be already upstream.

Thanks
  -- Daniel

[1]
 http://git.kernel.org/?p=linux/kernel/git/ebiederm/linux-nam espace-control-devel.git;a=summary

---

Subject: Re: [PATCH] allow a task to join a pid namespace
Posted by Glauber Costa on Tue, 05 Jun 2012 09:37:10 GMT
View Forum Message <> Reply to Message

On 06/05/2012 01:36 PM, Daniel Lezcano wrote:
> On 06/04/2012 03:33 PM, Glauber Costa wrote:
>> 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.
>
> Hi Glauber,
>
> I am in favor of this patch but I think the pidns support won't be
> complete and some corner-cases are not handled.
>
> May be you can look at Eric's patchset [1] where, IMO, everything is
> taken into account. Some of the patches may be already upstream.
>
> Thanks
>   -- Daniel

I don't remember seeing such patchset in the mailing lists, but that

might be my fault, due to traffic...

I'll take a look. If it does what I need, I can just drop this.

Thanks

Subject: Re: Re: [PATCH] allow a task to join a pid namespace
Posted by Glauber Costa on Tue, 05 Jun 2012 10:00:00 GMT
View Forum Message <> Reply to Message

On 06/05/2012 01:37 PM, Glauber Costa wrote:
> On 06/05/2012 01:36 PM, Daniel Lezcano wrote:
>> On 06/04/2012 03:33 PM, Glauber Costa wrote:
>>> 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.
>>
>> Hi Glauber,
>>
>> I am in favor of this patch but I think the pidns support won't be
>> complete and some corner-cases are not handled.
>>
>> May be you can look at Eric's patchset [1] where, IMO, everything is
>> taken into account. Some of the patches may be already upstream.
>>
>> Thanks
>> -- Daniel
>

> I don't remember seeing such patchset in the mailing lists, but that
> might be my fault, due to traffic...
>
> I'll take a look. If it does what I need, I can just drop this.
>

Ok. In a quick look, it does not seem to go all the way. This is just by reading, but your reboot patch, for instance, is unlikely to work with that, since if it doesn't alter pid->level, things like task ns_of_pid won't work.

Running the test scripts I wrote for my testing of that patch also doesn't seem to produce the expected result:

after doing setns, the pid won't show up in that namespace.

But I can work on top of that tree, may save a lot of work.

Eric, what are your plans for merging the content of that tree ?

Subject: Re: [PATCH] allow a task to join a pid namespace
Posted by Glauber Costa on Tue, 05 Jun 2012 11:33:34 GMT
View Forum Message <> Reply to Message

On 06/05/2012 01:36 PM, Daniel Lezcano wrote:
> On 06/04/2012 03:33 PM, Glauber Costa wrote:
>> 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.
>
> Hi Glauber,
>
> I am in favor of this patch but I think the pidns support won't be

> complete and some corner-cases are not handled.
>
> May be you can look at Eric's patchset [1] where, IMO, everything is
> taken into account. Some of the patches may be already upstream.
>
> Thanks
> -- Daniel
>

Daniel,

Please let me know what you think of the attached patch. It is ontop of Eric's tree that you pointed me to, but I am not really using any of its functionality, so this would be equally doable in current mainline kernel - but I wanted to make sure it integrates well with what Eric is doing as well.

It is a bit wasteful space-wise, but this approach pretty much guarantees we don't need to update pointers anywhere - therefore, totally lock-free. pid->level is only updated after switch_task_namespaces(), so every call before that will see correct information up to the previous level.

File Attachments

1) 0001-pid_ns-expand-the-current-pid-namespace-to-a-new-nam.patch, downloaded 492 times

---

Subject: Re: Re: [PATCH] allow a task to join a pid namespace
Posted by Daniel Lezcano on Tue, 05 Jun 2012 12:52:17 GMT
View Forum Message <> Reply to Message

On 06/05/2012 12:00 PM, Glauber Costa wrote:
> On 06/05/2012 01:37 PM, Glauber Costa wrote:
>> On 06/05/2012 01:36 PM, Daniel Lezcano wrote:
>>> On 06/04/2012 03:33 PM, Glauber Costa wrote:
>>>> 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.
>>>
>>> Hi Glauber,
>>>
>>> I am in favor of this patch but I think the pidns support won't be
>>> complete and some corner-cases are not handled.
>>>
>>> May be you can look at Eric's patchset [1] where, IMO, everything is
>>> taken into account. Some of the patches may be already upstream.
>>>
>>> Thanks
>>> -- Daniel
>>
>> I don't remember seeing such patchset in the mailing lists, but that
>> might be my fault, due to traffic...
>>
>> I'll take a look. If it does what I need, I can just drop this.
>>
>
> Ok. In a quick look, it does not seem to go all the way. This is just
> by reading, but your reboot patch, for instance, is unlikely to work
> with that, since if it doesn't alter pid->level, things like task
> ns_of_pid won't work.
>
> Running the test scripts I wrote for my testing of that patch also
> doesn't seem to produce the expected result:
>
> after doing setns, the pid won't show up in that namespace.

Yes, AFAIR, pid won't show up, you have to do fork-exec.

Subject: Re: Re: [PATCH] allow a task to join a pid namespace
Posted by Glauber Costa on Tue, 05 Jun 2012 12:53:43 GMT
View Forum Message <> Reply to Message

On 06/05/2012 04:52 PM, Daniel Lezcano wrote:
> On 06/05/2012 12:00 PM, Glauber Costa wrote:
>> On 06/05/2012 01:37 PM, Glauber Costa wrote:
>>> On 06/05/2012 01:36 PM, Daniel Lezcano wrote:
>>>> On 06/04/2012 03:33 PM, Glauber Costa wrote:
>>>>> 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.
>>>>
>>>> Hi Glauber,
>>>>
>>>> I am in favor of this patch but I think the pidns support won't be
>>>> complete and some corner-cases are not handled.
>>>>
>>>> May be you can look at Eric's patchset [1] where, IMO, everything is
>>>> taken into account. Some of the patches may be already upstream.
>>>>
>>>> Thanks
>>>> -- Daniel
>>>
>>> I don't remember seeing such patchset in the mailing lists, but that
>>> might be my fault, due to traffic...
>>>
>>> I'll take a look. If it does what I need, I can just drop this.
>>>
>>
>> Ok. In a quick look, it does not seem to go all the way. This is just

>> by reading, but your reboot patch, for instance, is unlikely to work
>> with that, since if it doesn't alter pid->level, things like task
>> ns_of_pid won't work.
>>
>> Running the test scripts I wrote for my testing of that patch also
>> doesn't seem to produce the expected result:
>>
>> after doing setns, the pid won't show up in that namespace.
>
> Yes, AFAIR, pid won't show up, you have to do fork-exec.

Ah, so you mean the kid will show up... Well, ok.

That's acceptable, but how about the behavior I am proposing ? (in the
patch I sent as a reply to this thread).

I believe it to be saner, even though there is a price tag attached to
it. None of the other setns calls require you to do any such trickery...

Subject: Re: Re: [PATCH] allow a task to join a pid namespace
Posted by Daniel Lezcano on Tue, 05 Jun 2012 13:18:59 GMT
View Forum Message <> Reply to Message

On 06/05/2012 02:53 PM, Glauber Costa wrote:
> On 06/05/2012 04:52 PM, Daniel Lezcano wrote:
>> On 06/05/2012 12:00 PM, Glauber Costa wrote:
>>> On 06/05/2012 01:37 PM, Glauber Costa wrote:
>>>> On 06/05/2012 01:36 PM, Daniel Lezcano wrote:
>>>>> On 06/04/2012 03:33 PM, Glauber Costa wrote:
>>>>>> 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.
>>>>>
>>>>> Hi Glauber,
>>>>>
>>>>> I am in favor of this patch but I think the pidns support won't be
>>>>> complete and some corner-cases are not handled.
>>>>>
>>>>> May be you can look at Eric's patchset [1] where, IMO, everything is
>>>>> taken into account. Some of the patches may be already upstream.
>>>>>
>>>>> Thanks
>>>>> -- Daniel
>>>>
>>>> I don't remember seeing such patchset in the mailing lists, but that
>>>> might be my fault, due to traffic...
>>>>
>>>> I'll take a look. If it does what I need, I can just drop this.
>>>>
>>>
>>> Ok. In a quick look, it does not seem to go all the way. This is just
>>> by reading, but your reboot patch, for instance, is unlikely to work
>>> with that, since if it doesn't alter pid->level, things like task
>>> ns_of_pid won't work.
>>>
>>> Running the test scripts I wrote for my testing of that patch also
>>> doesn't seem to produce the expected result:
>>>
>>> after doing setns, the pid won't show up in that namespace.
>>
>> Yes, AFAIR, pid won't show up, you have to do fork-exec.
>
> Ah, so you mean the kid will show up... Well, ok.
>
> That's acceptable, but how about the behavior I am proposing ? (in the
> patch I sent as a reply to this thread).

Let me look at the patch closely.
>
> I believe it to be saner, even though there is a price tag attached to
> it. None of the other setns calls require you to do any such trickery...

Yeah, but the pidns is different from the other namespace, it is not
supposed to be unshared.
I remember we had a discussion about this and Eric had some good reasons
to do it this way. One of them of the pid cached by the glibc. Also, we
don't want to have our pid changing in our application.

You may find more informations in there [1]

Thanks
  -- Daniel

[1] http://thread.gmane.org/gmane.linux.network/153200

Subject: Re: [PATCH] allow a task to join a pid namespace
Posted by ebiederm on Tue, 05 Jun 2012 16:49:33 GMT
View Forum Message <> Reply to Message

Glauber Costa <glommer@parallels.com> writes:

> 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);

This part where you are changing the pid on an existing process
to something different is flat out wrong.

The practical issues include processes sending signals will no longer
find this process, not handling multithreaded processes and glibc
caching getpid and getting totally confusedl, and those are just
off the top of my head.

We do need to figure out a practical way of entering a pid namespace.

Eric

Subject: Re: [PATCH] allow a task to join a pid namespace
Posted by ebiederm on Tue, 05 Jun 2012 17:18:52 GMT
View Forum Message <> Reply to Message

Oleg Nesterov <oleg@redhat.com> writes:

> On 06/04, Glauber Costa wrote:
>>
>> 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.
>
> I can't understand this patch... but probably I missed something,
> I never really understood setns.

The idea with setns is akin to callusermodehelper in the kernel.

>From outside a container we want to allow an appropriately privileged
user to create a process inside the container.

We run into all kinds of interesting gotchas with entering the
pid namespace:
 - Disjoint process trees.
 - Ensuring all processes are gone when we exit a pid namespace.
 - Not letting an empty pid namespace accept more processes.

We really only have two possbilities here.
- Allocate a new struct pid that is a superset of our current struct

pid but having additional processes ids inside a new pid namespace.

Along with all of the appropriate sanity checks to make that safe.

- Just modify the pid namespace the child processes of setns will use.

I lean towards the second option as that seems to have the best semantic
match to practical applications, and fewer kernel races to contend with,
but I might be persuadable.

However we do this we need to fix the bugs in pid namespace cleanup,
and deal with the issues that disjoint process trees bring to waiting
for all processes in a pid namespace to exit.

Ugh. Getting the waking up of zap_pid_ns_processes right and handling
the reaping of zombines in the cases of disjoint process trees is going
to be interesting.

Eric

Subject: Re: Re: [PATCH] allow a task to join a pid namespace
Posted by ebiederm on Tue, 05 Jun 2012 17:39:09 GMT
View Forum Message <> Reply to Message

> Eric, what are your plans for merging the content of that tree ?

Hopefully this cycle.

The issue with the pid namespace is primarily that we need to fix
all of the life cycle issues with the pid namespace and make certain
that setns adding an additional pid in a pid namespace does not break
those pid namespace life cycle fixes.

So I have been working on the hard tricky fixes off and on with Oleg.

Eric

Subject: Re: [PATCH] allow a task to join a pid namespace
Posted by Glauber Costa on Wed, 06 Jun 2012 08:54:23 GMT
View Forum Message <> Reply to Message

On 06/05/2012 08:49 PM, Eric W. Biederman wrote:
> Glauber Costa<glommer@parallels.com> writes:
>
>> 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);
>
> This part where you are changing the pid on an existing process
> to something different is flat out wrong.
>
> The practical issues include processes sending signals will no longer
> find this process, not handling multithreaded processes and glibc
> caching getpid and getting totally confusedl, and those are just
> off the top of my head.
>
> We do need to figure out a practical way of entering a pid namespace.
>
```

Yes, I realize that. So, I posted a new version of this patchset here,
as a reply to one of Daniel's message, but I can post it again
separately to avoid confusion.

The main idea is as follows:

* We only allow transitions to a namespace that has the current
  namespace as its parent. While doing that by keeping track of the
  original allocation level.

* I leave an extra upid slot at the end of the pid structure. That's
  the main drawback, I think, because it waste space even if you're not
  using setns. But since it is a small structure, the final result won't
  be that big.

* When calling setns, after a bunch of safety checks, I use the extra
  pid to allocate an extra level.

* When freeing, I use the original level to find the correct cache to
  free from.

This basically means that the task continue existing the same way as
before. Its pid won't change, we won't mess with pointers, and the task
is equally visible in the parent namespace. We can go one step further
with this and have getpid return based on the original level information.

What I personally like about this approach, is that we never have a
ghost process in a namespace. You can always see from the process tree
which processes are in each namespace.

Since one of the requirements I make is to have the calling process be
the group and session leader, and be single-threaded, there is no
pre-existing process tree to preserve, and the one to be created will be
sane.

I'd very much like to hear your thoughts on this approach. I do realize
that you have been hacking for a lot longer than me on this, so I may be
overlooking something.

Subject: Re: [PATCH] allow a task to join a pid namespace
Posted by ebiederm on Wed, 06 Jun 2012 18:29:16 GMT
View Forum Message <> Reply to Message

Glauber Costa <glommer@parallels.com> writes:
>
> Please let me know what you think of the attached patch. It is ontop of Eric's
> tree that you pointed me to, but I am not really using any of its functionality,
> so this would be equally doable in current mainline kernel - but I wanted to
> make sure it integrates well with what Eric is doing as well.

The trees I have out there for pid namespace work are for historical
reference at this point.

> It is a bit wasteful space-wise, but this approach pretty much guarantees we
> don't need to update pointers anywhere - therefore, totally
> lock-free. pid->level is only updated after switch_task_namespaces(), so every
> call before that will see correct information up to the previous
> level.

This solves none of the pid namespace life cycle problems and the issues
you solve by extending a pid can probably be solved better by allocating
a new pid and setting it's low pids the same as the original pid and
then atomically swapping them.

So I do not find the patch below at all interesting.

Eric

> From 1192e0ff3c854f4690d44fadccbc575fff653578 Mon Sep 17 00:00:00 2001
> From: Glauber Costa <glommer@parallels.com>
> Date: Tue, 5 Jun 2012 15:23:12 +0400
> Subject: [PATCH] pid_ns: expand the current pid namespace to a new namespace
>
> install pid_ns doesn't go as far as needed. Things like
> ns_of_pid() will rely on the current level to get to the
> right task. So when attaching a task to a new pid namespace,
> we need to make sure the upid vector grows as needed, and the
> level updated accordingly
>
> To do that, this patch leaves room for one more upid at the end
> of the structure at all times. It also stores the original level so
> we know afterwards which cache to free from.
>
> Although this is, of course, a bit wasteful, it has the advantage
> that we never need to touch the existing struct pid, nor the existing
> upid vectors. That would be racy in nature, and may require us to pick
> locks here and there - which seems to me like a big no.
>
> Signed-off-by: Glauber Costa <glommer@parallels.com>
> ---
> include/linux/pid.h | 3 +++
> kernel/nsproxy.c | 2 ++
> kernel/pid.c | 54 ++-
> kernel/pid_namespace.c | 30 ++++++++++++++++++++++++++--
> 4 files changed, 86 insertions(+), 3 deletions(-)
>
> diff --git a/include/linux/pid.h b/include/linux/pid.h

```
> index b152d44..2f01763 100644
> --- a/include/linux/pid.h
> +++ b/include/linux/pid.h
> @@ -58,6 +58,7 @@ struct pid
>  {
>   atomic_t count;
>   unsigned int level;
> + unsigned int orig_level;
>   /* lists of tasks that use this pid */
>   struct hlist_head tasks[PIDTYPE_MAX];
>   struct rcu_head rcu;
> @@ -121,6 +122,8 @@ int next_pidmap(struct pid_namespace *pid_ns, unsigned int last);
>
>  extern struct pid *alloc_pid(struct pid_namespace *ns);
>  extern void free_pid(struct pid *pid);
> +extern int expand_pid(struct pid *pid, struct pid_namespace *ns);
> +extern void  update_expanded_pid(struct task_struct *task);
>
>  /*
>   * ns_of_pid() returns the pid namespace in which the specified pid was
> diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
> index b956079..9851f11 100644
> --- a/kernel/nsproxy.c
> +++ b/kernel/nsproxy.c
> @@ -267,6 +267,8 @@ SYSCALL_DEFINE2(setns, int, fd, int, nstype)
>    goto out;
>   }
>   switch_task_namespaces(tsk, new_nsproxy);
> + if (nstype == CLONE_NEWPID)
> +  update_expanded_pid(current);
>  out:
>   fput(file);
>   return err;
> diff --git a/kernel/pid.c b/kernel/pid.c
> index 8c51c26..f2b1bd7 100644
> --- a/kernel/pid.c
> +++ b/kernel/pid.c
> @@ -247,7 +247,7 @@ void put_pid(struct pid *pid)
>   if (!pid)
>    return;
>
> - ns = pid->numbers[pid->level].ns;
> + ns = pid->numbers[pid->orig_level].ns;
>   if ((atomic_read(&pid->count) == 1) ||
>       atomic_dec_and_test(&pid->count)) {
>    kmem_cache_free(ns->pid_cachep, pid);
> @@ -306,6 +306,7 @@ struct pid *alloc_pid(struct pid_namespace *ns)
>
```

```
>   get_pid_ns(ns);
>   pid->level = ns->level;
> + pid->orig_level = ns->level;
>   atomic_set(&pid->count, 1);
>   for (type = 0; type < PIDTYPE_MAX; ++type)
>    INIT_HLIST_HEAD(&pid->tasks[type]);
> @@ -329,6 +330,57 @@ out_free:
>   goto out;
>  }
>
> +int expand_pid(struct pid *pid, struct pid_namespace *ns)
> +{
> + int i, nr;
> + struct pid_namespace *tmp;
> + struct upid *upid;
> +
> + if  ((ns->level - pid->orig_level) > 1)
> +  return -EINVAL;
> +
> + if (ns->parent != pid->numbers[pid->orig_level].ns)
> +  return -EINVAL;
> +
> + tmp = ns;
> + for (i = ns->level; i > pid->level; i--) {
> +  if (ns->dead)
> +   goto out_free;
> +  nr = alloc_pidmap(tmp);
> +  if (nr < 0)
> +   goto out_free;
> +
> +  pid->numbers[i].nr = nr;
> +  pid->numbers[i].ns = tmp;
> +  tmp = tmp->parent;
> + }
> +
> + upid = pid->numbers + ns->level;
> + spin_lock_irq(&pidmap_lock);
> + for (i = ns->level; i > pid->level; i--) {
> +  upid = &pid->numbers[i];
> +  hlist_add_head_rcu(&upid->pid_chain,
> +    &pid_hash[pid_hashfn(upid->nr, upid->ns)]);
> + }
> + spin_unlock_irq(&pidmap_lock);
> +
> + return 0;
> +
> +out_free:
> + while (++i <= ns->level)
```

```
> +  free_pidmap(pid->numbers + i);
> +
> + return -ENOMEM;
> +}
> +
> +void update_expanded_pid(struct task_struct *task)
> +{
> + struct pid *pid;
> + pid = get_task_pid(task, PIDTYPE_PID);
> +
> + pid->level++;
> +}
> +
>  struct pid *find_pid_ns(int nr, struct pid_namespace *ns)
>  {
>   struct hlist_node *elem;
> diff --git a/kernel/pid_namespace.c b/kernel/pid_namespace.c
> index 79bf459..0f3a521 100644
> --- a/kernel/pid_namespace.c
> +++ b/kernel/pid_namespace.c
> @@ -49,8 +49,12 @@ static struct kmem_cache *create_pid_cachep(int nr_ids)
>    goto err_alloc;
>
>   snprintf(pcache->name, sizeof(pcache->name), "pid_%d", nr_ids);
> + /*
> +  * nr_ids - 1 would provide room for upids up to the right level.
> +  * We leave room for one in the end for usage with setns.
> +  */
>   cachep = kmem_cache_create(pcache->name,
> -   sizeof(struct pid) + (nr_ids - 1) * sizeof(struct upid),
> +   sizeof(struct pid) + (nr_ids) * sizeof(struct upid),
>     0, SLAB_HWCACHE_ALIGN, NULL);
>   if (cachep == NULL)
>    goto err_cachep;
> @@ -244,8 +248,30 @@ static void pidns_put(void *ns)
>   put_pid_ns(ns);
>  }
>
> -static int pidns_install(struct nsproxy *nsproxy, void *ns)
> +static int pidns_install(struct nsproxy *nsproxy, void *_ns)
>  {
> + int ret = 0;
> + struct pid *pid;
> + struct pid_namespace *ns = _ns;
> +
> + rcu_read_lock();
> + pid = task_pid(current);
> + rcu_read_unlock();
```

```
> +
> + if (is_container_init(current) || (get_nr_threads(current) > 1))
> +  return -EINVAL;
> +
> + read_lock(&tasklist_lock);
> + if ((task_pgrp(current) != pid) || task_session(current) != pid)
> +  ret = -EPERM;
> + read_unlock(&tasklist_lock);
> + if (ret)
> +  return ret;
> +
> + ret = expand_pid(pid, ns);
> + if (ret)
> +  return ret;
> +
>   put_pid_ns(nsproxy->pid_ns);
>   nsproxy->pid_ns = get_pid_ns(ns);
>   return 0;
```