Subject: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by Dave Hansen on Mon, 19 Mar 2007 22:27:47 GMT
View Forum Message <> Reply to Message

I was tracking down why we need find_get_pid(1) in
proc_get_sb(), when I realized that we apparently
don't need a pid at all in the non-pid parts of /proc.

Anyone see any problems with this approach?

For what I would imagine are historical reasons, we set
all struct proc_inode->pid fields. We use the init
process for all non-/proc/<pid> inodes.

We get a handle to the init process in proc_get_sb()
then fetch it out in proc_pid_readdir():

```
 struct task_struct *reaper = get_proc_task(filp->f_path.dentry->d_inode);
```

The filp in that case is always the root inode on which
someone is doing a readdir. This reaper variable gets
passed down into proc_base_instantiate() and eventually
set in the new inode's ->pid field.

The problem is that I don't see anywhere that we
actually go and use this, outside of the /proc/<pid>
directories. Just referencing the init process like
this is a pain for containers because our init process
(pid == 1) can actually go away.

So, this patch removes all non-pid-dir use of
proc_inode->pid. It puts a WARN_ON() in case anyone
tries to instantiate a proc inode with a pid in a place
we don't expect there to be one.

```
 lxc-dave/fs//proc/inode.c |    6 ------
 lxc-dave/fs/proc/base.c   |   31 ++++++++++----------------------
 2 files changed, 10 insertions(+), 27 deletions(-)

diff -puN fs//proc/inode.c~funny-proc-patch fs//proc/inode.c
--- lxc/fs//proc/inode.c~funny-proc-patch 2007-03-19 15:10:50.000000000 -0700
+++ lxc-dave/fs//proc/inode.c 2007-03-19 15:10:50.000000000 -0700
@@ -184,7 +184,6 @@ out_mod:
 int proc_fill_super(struct super_block *s, void *data, int silent)
```

```
 {
  struct pid_namespace *pid_ns = data;
- struct proc_inode *ei;
  struct inode * root_inode;

  s->s_flags |= MS_NODIRATIME | MS_NOSUID | MS_NOEXEC;
@@ -204,11 +203,6 @@ int proc_fill_super(struct super_block *
  s->s_root = d_alloc_root(root_inode);
  if (!s->s_root)
   goto out_no_root;
- /* Seed the root directory with a pid so it doesn't need
-  * to be special in base.c.
-  */
- ei = PROC_I(root_inode);
- ei->pid = find_get_pid(1);
  return 0;

 out_no_root:
diff -puN fs//proc/internal.h~funny-proc-patch fs//proc/internal.h
diff -puN fs/proc/base.c~funny-proc-patch fs/proc/base.c
--- lxc/fs/proc/base.c~funny-proc-patch 2007-03-19 15:10:50.000000000 -0700
+++ lxc-dave/fs/proc/base.c 2007-03-19 15:11:40.000000000 -0700
@@ -1171,11 +1171,15 @@ static int pid_revalidate(struct dentry

 static int pid_delete_dentry(struct dentry * dentry)
 {
+ struct pid *pid;
  /* Is the task we represent dead?
   * If so, then don't put the dentry on the lru list,
   * kill it immediately.
   */
- return !proc_pid(dentry->d_inode)->tasks[PIDTYPE_PID].first;
+ pid = proc_pid(dentry->d_inode);
+ if (!pid)
+  return 0;
+ return !pid->tasks[PIDTYPE_PID].first;
 }

 static struct dentry_operations pid_dentry_operations =
@@ -1813,6 +1817,7 @@ static struct dentry *proc_base_instanti
  struct proc_inode *ei;
  struct dentry *error = ERR_PTR(-EINVAL);

+ WARN_ON(task);
  /* Allocate the inode */
  error = ERR_PTR(-ENOMEM);
  inode = new_inode(dir->i_sb);
@@ -1823,13 +1828,6 @@ static struct dentry *proc_base_instanti
```

```
  ei = PROC_I(inode);
  inode->i_mtime = inode->i_atime = inode->i_ctime = CURRENT_TIME;

- /*
-  * grab the reference to the task.
-  */
- ei->pid = get_task_pid(task, PIDTYPE_PID);
- if (!ei->pid)
-  goto out_iput;
-
  inode->i_uid = 0;
  inode->i_gid = 0;
  inode->i_mode = p->mode;
@@ -1847,9 +1845,6 @@ static struct dentry *proc_base_instanti
  error = NULL;
 out:
  return error;
-out_iput:
- iput(inode);
- goto out;
 }

 static struct dentry *proc_base_lookup(struct inode *dir, struct dentry *dentry)
@@ -1874,7 +1869,7 @@ static struct dentry *proc_base_lookup(s
  if (p > last)
   goto out;

- error = proc_base_instantiate(dir, dentry, task, p);
+ error = proc_base_instantiate(dir, dentry, NULL, p);

 out:
  put_task_struct(task);
@@ -1883,10 +1878,10 @@ out_no_task:
 }

 static int proc_base_fill_cache(struct file *filp, void *dirent, filldir_t filldir,
- struct task_struct *task, struct pid_entry *p)
+ struct pid_entry *p)
 {
  return proc_fill_cache(filp, dirent, filldir, p->name, p->len,
-    proc_base_instantiate, task, p);
+    proc_base_instantiate, NULL, p);
 }

 #ifdef CONFIG_TASK_IO_ACCOUNTING
@@ -2197,16 +2192,12 @@ static int proc_pid_fill_cache(struct fi
 int proc_pid_readdir(struct file * filp, void * dirent, filldir_t filldir)
 {
```

```
  unsigned int nr = filp->f_pos - FIRST_PROCESS_ENTRY;
- struct task_struct *reaper = get_proc_task(filp->f_path.dentry->d_inode);
  struct task_struct *task;
  int tgid;

- if (!reaper)
-  goto out_no_task;
-
  for (; nr < ARRAY_SIZE(proc_base_stuff); filp->f_pos++, nr++) {
   struct pid_entry *p = &proc_base_stuff[nr];
-  if (proc_base_fill_cache(filp, dirent, filldir, reaper, p) < 0)
+  if (proc_base_fill_cache(filp, dirent, filldir, p) < 0)
    goto out;
  }

@@ -2223,8 +2214,6 @@ int proc_pid_readdir(struct file * filp,
  }
  filp->f_pos = PID_MAX_LIMIT + TGID_OFFSET;
 out:
- put_task_struct(reaper);
-out_no_task:
  return 0;
 }

diff -puN fs/proc/root.c~funny-proc-patch fs/proc/root.c
_
```

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by ebiederm on Tue, 20 Mar 2007 02:04:12 GMT
View Forum Message <> Reply to Message

Dave Hansen <hansendc@us.ibm.com> writes:

> I was tracking down why we need find_get_pid(1) in
> proc_get_sb(), when I realized that we apparently
> don't need a pid at all in the non-pid parts of /proc.
>
> Anyone see any problems with this approach?

The thing is these are pid related parts of /proc you are
working with.

I'm trying to remember what the actual semantics were.

I do know doing this means if our pid namespace goes away these
functions do the right thing.

This may have been how I was getting the pid namespace in originally
so this code may be obsolete.

Partly I think doing this made the code a little more symmetric.

Regardless I would like to see a little farther down on
how we test to see if the pid namespace is alive and how we
make these functions do nothing if it has died. I would also
like to see how we perform the appropriate lookups by pid namespace.

Basically I want to see how we finish up multiple namespace support
for /proc before we start with the micro optimizations.

I'm fairly certain this patch causes us to do the wrong thing when
the pid namespace exits, and I don't see much gain except for the
death of find_get_pid.

> For what I would imagine are historical reasons, we set
> all struct proc_inode->pid fields. We use the init
> process for all non-/proc/<pid> inodes.
>
> We get a handle to the init process in proc_get_sb()
> then fetch it out in proc_pid_readdir():
>
> struct task_struct *reaper =
> get_proc_task(filp->f_path.dentry->d_inode);
>
> The filp in that case is always the root inode on which
> someone is doing a readdir. This reaper variable gets
> passed down into proc_base_instantiate() and eventually
> set in the new inode's ->pid field.
>
> The problem is that I don't see anywhere that we
> actually go and use this, outside of the /proc/<pid>
> directories. Just referencing the init process like
> this is a pain for containers because our init process
> (pid == 1) can actually go away.

Which as far as can recall is part of the point. If you have a pid
namespace with normal semantics the child reaper pid == 1 is the last
pid in the pid namespace to exit. Therefore when it exists the pid

namespace exists and with it doesn't the pid namespace does not exist.

Eric

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

---

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by Dave Hansen on Tue, 20 Mar 2007 02:30:32 GMT
View Forum Message <> Reply to Message

On Mon, 2007-03-19 at 20:04 -0600, Eric W. Biederman wrote:
> Dave Hansen <hansendc@us.ibm.com> writes:
> Regardless I would like to see a little farther down on
> how we test to see if the pid namespace is alive and how we
> make these functions do nothing if it has died.

That shouldn't be too hard. We have access to the superblock pretty much everywhere, and we now store the pid_namespace in there (with some patches I posted earlier).

> I would also
> like to see how we perform the appropriate lookups by pid namespace.

What do you mean?

> Basically I want to see how we finish up multiple namespace support
> for /proc before we start with the micro optimizations.

Serge was tracking down some weird /proc issues and noticed that we expect a pid_nr==1 for the pid namespace as long as it has a /proc around. That is an assumption doesn't always hold now.

> I'm fairly certain this patch causes us to do the wrong thing when
> the pid namespace exits, and I don't see much gain except for the
> death of find_get_pid.

In the default, mainline case, it shouldn't be a problem at all. We don't have the init pid namespace exiting.

Shouldn't the lifetime of things under a /proc mount be tied to the life of the mount, and not to the pid_namespace it is tied to? It seems relatively sane to me to have a /proc empty of all processes, but still have /proc/cpuinfo even if all of its processes are gone.

> > For what I would imagine are historical reasons, we set

> > all struct proc_inode->pid fields. We use the init
> > process for all non-/proc/<pid> inodes.
> >
> > We get a handle to the init process in proc_get_sb()
> > then fetch it out in proc_pid_readdir():
> >
> > struct task_struct *reaper =
> > get_proc_task(filp->f_path.dentry->d_inode);
> >
> > The filp in that case is always the root inode on which
> > someone is doing a readdir. This reaper variable gets
> > passed down into proc_base_instantiate() and eventually
> > set in the new inode's ->pid field.
> >
> > The problem is that I don't see anywhere that we
> > actually go and use this, outside of the /proc/<pid>
> > directories. Just referencing the init process like
> > this is a pain for containers because our init process
> > (pid == 1) can actually go away.
>
> Which as far as can recall is part of the point. If you have a pid
> namespace with normal semantics the child reaper pid == 1 is the last
> pid in the pid namespace to exit. Therefore when it exists the pid
> namespace exists and with it doesn't the pid namespace does not exist.

pid_delete_dentry() looks like the remaining place that really cares.
It would be pretty easy to have it check the pid namespace.

-- Dave

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

---

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by ebiederm on Tue, 20 Mar 2007 04:07:32 GMT
View Forum Message <> Reply to Message

Dave Hansen <hansendc@us.ibm.com> writes:

> On Mon, 2007-03-19 at 20:04 -0600, Eric W. Biederman wrote:
>> Dave Hansen <hansendc@us.ibm.com> writes:
>> Regardless I would like to see a little farther down on
>> how we test to see if the pid namespace is alive and how we
>> make these functions do nothing if it has died.
>

> That shouldn't be too hard. We have access to the superblock pretty
> much everywhere, and we now store the pid_namespace in there (with some
> patches I posted earlier).

Sounds right. I don't think my original version had that. Which
changes the rules a little bit.

>> I would also
>> like to see how we perform the appropriate lookups by pid namespace.
>
> What do you mean?

proc_pid_readdir ... next_tgid().

>> Basically I want to see how we finish up multiple namespace support
>> for /proc before we start with the micro optimizations.
>
> Serge was tracking down some weird /proc issues and noticed that we
> expect a pid_nr==1 for the pid namespace as long as it has a /proc
> around. That is an assumption doesn't always hold now.

Maybe. It really depends on how we define a namespace exiting.
That must be in the lxc tree.

There should be no code in the -mm or in Linus's tree that has
that property.

While I'm not categorically opposed to supporting things like that it
but it is something for which we need to tread very carefully because
it is an extension of current semantics. I can't think of any weird
semantics right now but for something user visible we will have to
support indefinitely I don't see a reason to rush into it either.

>> I'm fairly certain this patch causes us to do the wrong thing when
>> the pid namespace exits, and I don't see much gain except for the
>> death of find_get_pid.
>
> In the default, mainline case, it shouldn't be a problem at all. We
> don't have the init pid namespace exiting.

True but we are getting close. And it is about time we worked up
patches for that so our conversations can become less theoretical.

> Shouldn't the lifetime of things under a /proc mount be tied to the life
> of the mount, and not to the pid_namespace it is tied to? It seems
> relatively sane to me to have a /proc empty of all processes, but still
> have /proc/cpuinfo even if all of its processes are gone.

That is what is implemented. When the pid namespace goes away there
are no more pid directories, and the /proc/self symlink goes away.
But everything else remains.

If you look proc_root_readdir is not affected when the pid namespace
goes away. Just proc_pid_readdir.

Everything in fs/proc/base.c is about pid files in one way or another.

> pid_delete_dentry() looks like the remaining place that really cares.
> It would be pretty easy to have it check the pid namespace.

Sure although it also needs the pid check for files that have it as
the process can go away sooner.

Eric

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

---

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by serue on Tue, 20 Mar 2007 14:58:12 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> Dave Hansen <hansendc@us.ibm.com> writes:
>
> > On Mon, 2007-03-19 at 20:04 -0600, Eric W. Biederman wrote:
> >> Dave Hansen <hansendc@us.ibm.com> writes:
> >> Regardless I would like to see a little farther down on
> >> how we test to see if the pid namespace is alive and how we
> >> make these functions do nothing if it has died.
> >
> > That shouldn't be too hard. We have access to the superblock pretty
> > much everywhere, and we now store the pid_namespace in there (with some
> > patches I posted earlier).
>
> Sounds right. I don't think my original version had that. Which
> changes the rules a little bit.
>
> >> I would also
> >> like to see how we perform the appropriate lookups by pid namespace.
> >
> > What do you mean?
>
> proc_pid_readdir ... next_tgid().

next_tgid() is simple enough - we can always use current->pid_ns to find
the next pidnr.

The only hitch, as mentioned earlier, is how do we find the first task.
Currently task 1 is statically stored as the first inode, and as Dave
mentioned we can't do that now, because we dont' know of any one task
which will outlive the pid_ns.

> >> Basically I want to see how we finish up multiple namespace support
> >> for /proc before we start with the micro optimizations.
> >
> > Serge was tracking down some weird /proc issues and noticed that we
> > expect a pid_nr==1 for the pid namespace as long as it has a /proc
> > around. That is an assumption doesn't always hold now.
>
> Maybe. It really depends on how we define a namespace exiting.
> That must be in the lxc tree.
>
> There should be no code in the -mm or in Linus's tree that has
> that property.

True.

> While I'm not categorically opposed to supporting things like that it
> but it is something for which we need to tread very carefully because
> it is an extension of current semantics. I can't think of any weird
> semantics right now but for something user visible we will have to
> support indefinitely I don't see a reason to rush into it either.

Except that unless we mandate that pid1 in any namespace can't exit, and
put that feature off until later, we can't not address it.

> >> I'm fairly certain this patch causes us to do the wrong thing when
> >> the pid namespace exits, and I don't see much gain except for the
> >> death of find_get_pid.
> >
> > In the default, mainline case, it shouldn't be a problem at all. We
> > don't have the init pid namespace exiting.
>
> True but we are getting close. And it is about time we worked up
> patches for that so our conversations can become less theoretical.

Yes I really hope a patchset goes out today.

> > Shouldn't the lifetime of things under a /proc mount be tied to the life
> > of the mount, and not to the pid_namespace it is tied to? It seems
> > relatively sane to me to have a /proc empty of all processes, but still

> > have /proc/cpuinfo even if all of its processes are gone.
>
> That is what is implemented. When the pid namespace goes away there
> are no more pid directories, and the /proc/self symlink goes away.
> But everything else remains.
>
> If you look proc_root_readdir is not affected when the pid namespace
> goes away. Just proc_pid_readdir.
>
> Everything in fs/proc/base.c is about pid files in one way or another.
>
> > pid_delete_dentry() looks like the remaining place that really cares.
> > It would be pretty easy to have it check the pid namespace.
>
> Sure although it also needs the pid check for files that have it as
> the process can go away sooner.
>
> Eric
> _____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containers

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

---

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by ebiederm on Tue, 20 Mar 2007 15:51:42 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:

> Quoting Eric W. Biederman (ebiederm@xmission.com):
>> Dave Hansen <hansendc@us.ibm.com> writes:
>> > On Mon, 2007-03-19 at 20:04 -0600, Eric W. Biederman wrote:

>> >> I would also
>> >> like to see how we perform the appropriate lookups by pid namespace.
>> >
>> > What do you mean?
>>
>> proc_pid_readdir ... next_tgid().
>
> next_tgid() is simple enough - we can always use current->pid_ns to find
> the next pidnr.

No. We cannot use current->pid_ns. We must get it from the mount or
something in the mount.

Using current to set the default pid_ns to mount is fine. But if
we use current to select our files we have a moderately serious problem.

> The only hitch, as mentioned earlier, is how do we find the first task.
> Currently task 1 is statically stored as the first inode, and as Dave
> mentioned we can't do that now, because we dont' know of any one task
> which will outlive the pid_ns.

Outlive is the wrong concept. Ideally we want something that will
live as long as there are processes in the pid_ns.

As I thought about this some more there are some problems for holding
a reference to a pid_ns for a long period of time. Currently struct_pid
is designed so you can hang onto it forever. struct pid_namespace isn't.
So we have some very interesting semantic questions of what happens when
the pid namespace exits.

Since we distinguish mounts by their pid namespace this looks like
something we need to sort through.

>> While I'm not categorically opposed to supporting things like that it
>> but it is something for which we need to tread very carefully because
>> it is an extension of current semantics. I can't think of any weird
>> semantics right now but for something user visible we will have to
>> support indefinitely I don't see a reason to rush into it either.
>
> Except that unless we mandate that pid1 in any namespace can't exit, and
> put that feature off until later, we can't not address it.

What if we mandate that pid1 is the last process to exit?

Problems actually only show up in this context if other pids live
substantially longer than pid1.

>> True but we are getting close. And it is about time we worked up
>> patches for that so our conversations can become less theoretical.
>
> Yes I really hope a patchset goes out today.

Sounds good. I expect it will take a couple of rounds of review,
before we have all of the little things nailed down but starting that
process is a hopeful sign.

Eric

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

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by Dave Hansen on Tue, 20 Mar 2007 15:55:08 GMT
View Forum Message <> Reply to Message

On Tue, 2007-03-20 at 09:51 -0600, Eric W. Biederman wrote:
> No. We cannot use current->pid_ns. We must get it from the mount or
> something in the mount.

Ugh. I just thought about /proc/self :)

-- Dave

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

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by serue on Tue, 20 Mar 2007 16:00:57 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> "Serge E. Hallyn" <serue@us.ibm.com> writes:
>
> > Quoting Eric W. Biederman (ebiederm@xmission.com):
> >> Dave Hansen <hansendc@us.ibm.com> writes:
> >> > On Mon, 2007-03-19 at 20:04 -0600, Eric W. Biederman wrote:
>
> >> >> I would also
> >> >> like to see how we perform the appropriate lookups by pid namespace.
> >> >
> >> > What do you mean?
> >>
> >> proc_pid_readdir ... next_tgid().
> >
> > next_tgid() is simple enough - we can always use current->pid_ns to find
> > the next pidnr.
>
> No. We cannot use current->pid_ns. We must get it from the mount or
> something in the mount.

Actually I think Dave has it coming from superblock data.

> Using current to set the default pid_ns to mount is fine. But if
> we use current to select our files we have a moderately serious problem.
>
> > The only hitch, as mentioned earlier, is how do we find the first task.
> > Currently task 1 is statically stored as the first inode, and as Dave
> > mentioned we can't do that now, because we dont' know of any one task
> > which will outlive the pid_ns.
>
> Outlive is the wrong concept. Ideally we want something that will
> live as long as there are processes in the pid_ns.

And there is no such thing.

> As I thought about this some more there are some problems for holding
> a reference to a pid_ns for a long period of time. Currently struct_pid
> is designed so you can hang onto it forever. struct pid_namespace isn't.
> So we have some very interesting semantic questions of what happens when
> the pid namespace exits.
>
> Since we distinguish mounts by their pid namespace this looks like
> something we need to sort through.

Yup.

> >> While I'm not categorically opposed to supporting things like that it
> >> but it is something for which we need to tread very carefully because
> >> it is an extension of current semantics. I can't think of any weird
> >> semantics right now but for something user visible we will have to
> >> support indefinitely I don't see a reason to rush into it either.
> >
> > Except that unless we mandate that pid1 in any namespace can't exit, and
> > put that feature off until later, we can't not address it.
>
> What if we mandate that pid1 is the last process to exit?

I think people have complained about that in the past for application
containers, but I really don't see where it hurts anything.

Cedric, Herbert, did one of you think it would be bad?

> Problems actually only show up in this context if other pids live
> substantially longer than pid1.
>
> >> True but we are getting close. And it is about time we worked up
> >> patches for that so our conversations can become less theoretical.

> >
> > Yes I really hope a patchset goes out today.
>
> Sounds good. I expect it will take a couple of rounds of review,
> before we have all of the little things nailed down but starting that
> process is a hopeful sign.

I'm hoping some of the earlier patches can be acked this time so we can
get to discussing the more interesting parts :)

But I'm afraid it might be no earlier than tomorrow that the patches go
out. Will try.

thanks,
-serge

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

---

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by Cedric Le Goater on Tue, 20 Mar 2007 17:42:32 GMT
View Forum Message <> Reply to Message

>>>> True but we are getting close. And it is about time we worked up
>>>> patches for that so our conversations can become less theoretical.
>>> Yes I really hope a patchset goes out today.
>> Sounds good. I expect it will take a couple of rounds of review,
>> before we have all of the little things nailed down but starting that
>> process is a hopeful sign.
>
> I'm hoping some of the earlier patches can be acked this time so we can
> get to discussing the more interesting parts :)
>
> But I'm afraid it might be no earlier than tomorrow that the patches go
> out. Will try.

suka is out but I think i can send his patchset this evening.

the first patches seem ackable. they used to be -mm and were dropped
because of compile issues. we'll give them another review, it can't hurt
them.

C.

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

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by ebiederm on Tue, 20 Mar 2007 22:04:22 GMT
View Forum Message <> Reply to Message

Dave Hansen <hansendc@us.ibm.com> writes:

> On Tue, 2007-03-20 at 09:51 -0600, Eric W. Biederman wrote:
>> Outlive is the wrong concept. Ideally we want something that will
>> live as long as there are processes in the pid_ns.
>
> How about they just live as long as there is a mount? Now that we _can_
> have multiple superblocks and meaningful vfsmounts, I think it's time to
> make it act like a normal filesystem.

Agreed.

My concern is that the mount will outlive the pid namespace. In which
case we need something that is safe to test when the pid namespace goes
away.

Eric

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

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by ebiederm on Tue, 20 Mar 2007 22:11:36 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:

>> >
>> > Except that unless we mandate that pid1 in any namespace can't exit, and
>> > put that feature off until later, we can't not address it.
>>
>> What if we mandate that pid1 is the last process to exit?
>
> I think people have complained about that in the past for application
> containers, but I really don't see where it hurts anything.
>
> Cedric, Herbert, did one of you think it would be bad?

Sure. As an extension I don't have a problem with the notion, of
allowing pid1 to exit before others. But if it makes things harder
on us I don't want to support it, at least not initially.

Eric

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

---

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by serue on Wed, 21 Mar 2007 01:02:49 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> Dave Hansen <hansendc@us.ibm.com> writes:
>
> > On Tue, 2007-03-20 at 09:51 -0600, Eric W. Biederman wrote:
> >> Outlive is the wrong concept. Ideally we want something that will
> >> live as long as there are processes in the pid_ns.
> >
> > How about they just live as long as there is a mount? Now that we _can_
> > have multiple superblocks and meaningful vfsmounts, I think it's time to
> > make it act like a normal filesystem.
>
> Agreed.
>
> My concern is that the mount will outlive the pid namespace. In which
> case we need something that is safe to test when the pid namespace goes
> away.

Offhand I would assume the mount would get a reference to the pidns.
pidns may be empty, but would exist.

-serge

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

---

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by serue on Wed, 21 Mar 2007 14:41:28 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):

> "Serge E. Hallyn" <serue@us.ibm.com> writes:
>
> >> >
> >> > Except that unless we mandate that pid1 in any namespace can't exit, and
> >> > put that feature off until later, we can't not address it.
> >>
> >> What if we mandate that pid1 is the last process to exit?
> >
> > I think people have complained about that in the past for application
> > containers, but I really don't see where it hurts anything.
> >
> > Cedric, Herbert, did one of you think it would be bad?
>
> Sure. As an extension I don't have a problem with the notion, of
> allowing pid1 to exit before others. But if it makes things harder
> on us I don't want to support it, at least not initially.

So how do you see us enforcing pid1's existance? Somehow keep it from
fully exiting, or just kill all the processes in it's namespace if it
exits?

-serge

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by ebiederm on Wed, 21 Mar 2007 16:30:38 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:

>
> So how do you see us enforcing pid1's existance? Somehow keep it from
> fully exiting, or just kill all the processes in it's namespace if it
> exits?

Killing all other processes in the namespace when pid1 exits is what
I implemented last time around.

Eric

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by Cedric Le Goater on Wed, 21 Mar 2007 16:35:34 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> "Serge E. Hallyn" <serue@us.ibm.com> writes:
>
>> So how do you see us enforcing pid1's existance? Somehow keep it from
>> fully exiting, or just kill all the processes in it's namespace if it
>> exits?

what about a kthread that would be spawned when a task is cloned in an
unshared pid namespace ? This is an extra cost in term of tasks.

> Killing all other processes in the namespace when pid1 exits is what
> I implemented last time around.

this looks like a sane thing to do.

C.

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

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by ebiederm on Wed, 21 Mar 2007 16:57:24 GMT
View Forum Message <> Reply to Message

Cedric Le Goater <clg@fr.ibm.com> writes:

> Eric W. Biederman wrote:
>> "Serge E. Hallyn" <serue@us.ibm.com> writes:
>>
>>> So how do you see us enforcing pid1's existance? Somehow keep it from
>>> fully exiting, or just kill all the processes in it's namespace if it
>>> exits?
>
> what about a kthread that would be spawned when a task is cloned in an
> unshared pid namespace ? This is an extra cost in term of tasks.

If you use kernel_thread this can happen. (Die kernel_thread).
If you use the kthread interface keventd will be the parent process and
we won't have problems. Thus most users of kernel_thread need to be fixed
to use the kthread interface.

Thanks for the reminder of this one, I had forgotten that bit of

reasoning for updating kernel_thread users.

>> Killing all other processes in the namespace when pid1 exits is what
>> I implemented last time around.
>
> this looks like a sane thing to do.

Eric

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

---

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by Cedric Le Goater on Wed, 21 Mar 2007 17:29:35 GMT
View Forum Message <> Reply to Message

>> what about a kthread that would be spawned when a task is cloned in an
>> unshared pid namespace ? This is an extra cost in term of tasks.
>
> If you use kernel_thread this can happen. (Die kernel_thread).
> If you use the kthread interface keventd will be the parent process and
> we won't have problems.

so is it something acceptable for mainline ? I think openvz has such
a thread doing the reaping.

> Thus most users of kernel_thread need to be fixed to use the kthread
> interface.
>
> Thanks for the reminder of this one, I had forgotten that bit of
> reasoning for updating kernel_thread users.

there are not much left. see below a quick and dirty survey on
2.6.21-rc4-mm1.

C.

./fs/jffs2/background.c: pid = kernel_thread(jffs2_garbage_collect_thread, c,
CLONE_FS|CLONE_FILES);
./fs/nfs/delegation.c: status = kernel_thread(recall_thread, &data, CLONE_KERNEL);
./fs/cifs/connect.c: rc = (int)kernel_thread((void *)(void *)cifs_demultiplex_thread, srvTcp,
./fs/lockd/clntlock.c: if (kernel_thread(reclaimer, host, CLONE_KERNEL) < 0)
./fs/afs/kafsasyncd.c: ret = kernel_thread(kafsasyncd, NULL, 0);
./fs/afs/kafstimod.c: ret = kernel_thread(kafstimod, NULL, 0);

./fs/afs/cmservice.c: ret = kernel_thread(kafscmd, NULL, 0);
./arch/powerpc/platforms/pseries/rtasd.c: if (kernel_thread(rtasd, NULL, CLONE_FS) < 0)
./arch/powerpc/platforms/pseries/eeh_event.c: if (kernel_thread(eeh_event_handler, NULL,
CLONE_KERNEL) < 0)
./arch/ia64/sn/kernel/xpc_main.c: pid = kernel_thread(xpc_activating, (void *) ((u64) partid), 0);
./arch/ia64/sn/kernel/xpc_main.c: pid = kernel_thread(xpc_daemonize_kthread, (void *) args, 0);
./arch/ia64/sn/kernel/xpc_main.c: pid = kernel_thread(xpc_hb_checker, NULL, 0);
./arch/ia64/sn/kernel/xpc_main.c: pid = kernel_thread(xpc_initiate_discovery, NULL, 0);
./arch/arm/kernel/ecard.c: ret = kernel_thread(ecard_task, NULL, CLONE_KERNEL);
./arch/sparc64/kernel/power.c: if (kernel_thread(powerd, NULL, CLONE_FS) < 0) {
./arch/i386/mach-voyager/voyager_thread.c: if(kernel_thread(thread, NULL, CLONE_KERNEL) <
0) {
./arch/i386/kernel/io_apic.c: if (kernel_thread(balanced_irq, NULL, CLONE_KERNEL) >= 0)
./arch/parisc/kernel/process.c: return __kernel_thread(fn, arg, flags);
./init/do_mounts_initrd.c: pid = kernel_thread(do_linuxrc, "/linuxrc", SIGCHLD);
./kernel/kmod.c: pid = kernel_thread(____call_usermodehelper, sub_info, SIGCHLD);
./kernel/kmod.c: pid = kernel_thread(wait_for_helper, sub_info,
./kernel/kmod.c: pid = kernel_thread(____call_usermodehelper, sub_info,
./kernel/stop_machine.c: ret = kernel_thread(stopmachine, (void *)(long)i,CLONE_KERNEL);
./net/ipv4/ipvs/ip_vs_sync.c: if ((pid = kernel_thread(sync_thread, startup, 0)) < 0) {
./net/ipv4/ipvs/ip_vs_sync.c: if ((pid = kernel_thread(fork_sync_thread, &startup, 0)) < 0) {
./net/sunrpc/svc.c: error = kernel_thread((int (*)(void *)) func, rqstp, 0);
./net/rxrpc/krxiod.c: return kernel_thread(rxrpc_krxiod, NULL, 0);
./net/rxrpc/krxsecd.c: return kernel_thread(rxrpc_krxsecd, NULL, 0);
./net/rxrpc/krxtimod.c: ret = kernel_thread(krxtimod, NULL, 0);
./net/bluetooth/bnep/core.c: err = kernel_thread(bnep_session, s, CLONE_KERNEL);
./net/bluetooth/hidp/core.c: err = kernel_thread(hidp_session, session, CLONE_KERNEL);
./net/bluetooth/cmtp/core.c: err = kernel_thread(cmtp_session, session, CLONE_KERNEL);
./net/bluetooth/rfcomm/core.c: kernel_thread(rfcomm_run, NULL, CLONE_KERNEL);
./drivers/media/video/saa7134/saa7134-tvaudio.c: dev->thread.pid =
kernel_thread(my_thread,dev,0);
./drivers/media/video/saa7134/saa7134-tvaudio.c: printk(KERN_WARNING "%s: kernel_thread()
failed\n",
./drivers/media/dvb/dvb-core/dvb_ca_en50221.c: ret = kernel_thread(dvb_ca_en50221_thread,
ca, 0);
./drivers/usb/atm/usbatm.c: int ret = kernel_thread(usbatm_do_heavy_init, instance,
CLONE_KERNEL);
./drivers/s390/scsi/zfcp_erp.c: retval = kernel_thread(zfcp_erp_thread, adapter, SIGCHLD);
./drivers/s390/net/lcs.c: kernel_thread(lcs_recovery, (void *) card, SIGCHLD);
./drivers/s390/net/lcs.c: kernel_thread(lcs_register_mc_addresses,
./drivers/s390/net/qeth_main.c: kernel_thread(qeth_recover, (void *) card, SIGCHLD);
./drivers/scsi/libsas/sas_scsi_host.c: res = kernel_thread(sas_queue_thread, sas_ha, 0);
./drivers/pnp/pnpbios/core.c: if (kernel_thread(pnp_dock_thread, NULL, CLONE_KERNEL) > 0)
./drivers/mtd/ubi/background.c: pid = kernel_thread(ubi_thread, ubi, CLONE_FS |
CLONE_FILES);
./drivers/mtd/mtd_blkdevs.c: ret = kernel_thread(mtd_blktrans_thread, tr, CLONE_KERNEL);
./drivers/pci/hotplug/cpci_hotplug_core.c: pid = kernel_thread(event_thread, NULL, 0);
./drivers/pci/hotplug/cpci_hotplug_core.c: pid = kernel_thread(poll_thread, NULL, 0);

./drivers/pci/hotplug/cpqphp_ctrl.c: pid = kernel_thread(event_thread, NULL, 0);
./drivers/macintosh/therm_windtunnel.c: x.poll_task = kernel_thread(control_loop, NULL, SIGCHLD | CLONE_KERNEL);
./drivers/macintosh/mediabay.c: kernel_thread(media_bay_task, NULL, CLONE_KERNEL);
./drivers/macintosh/adb.c: adb_probe_task_pid = kernel_thread(adb_probe_task, NULL, SIGCHLD | CLONE_KERNEL);
./drivers/macintosh/therm_pm72.c: ctrl_task = kernel_thread(main_control_loop, NULL, SIGCHLD | CLONE_KERNEL);

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

---

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by ebiederm on Wed, 21 Mar 2007 17:42:55 GMT
View Forum Message <> Reply to Message

Cedric Le Goater <clg@fr.ibm.com> writes:

>>> what about a kthread that would be spawned when a task is cloned in an
>>> unshared pid namespace ? This is an extra cost in term of tasks.
>>
>> If you use kernel_thread this can happen. (Die kernel_thread).
>> If you use the kthread interface keventd will be the parent process and
>> we won't have problems.
>
> so is it something acceptable for mainline ? I think openvz has such
> a thread doing the reaping.

Please clarify. Is what acceptable for mainline?

>> Thus most users of kernel_thread need to be fixed to use the kthread
>> interface.
>>
>> Thanks for the reminder of this one, I had forgotten that bit of
>> reasoning for updating kernel_thread users.
>
> there are not much left. see below a quick and dirty survey on
> 2.6.21-rc4-mm1.

Yep. We are almost there.
Of course with nfs still pending we have some of the nastier ones left.

A couple of the ones in arch/ and kernel/ we don't have to worry about
because they are either started early enough it doesn't matter or they

implement kthread...

Eric

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by serue on Wed, 21 Mar 2007 20:29:41 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> Cedric Le Goater <clg@fr.ibm.com> writes:
>
> >>> what about a kthread that would be spawned when a task is cloned in an
> >>> unshared pid namespace ? This is an extra cost in term of tasks.
> >>
> >> If you use kernel_thread this can happen. (Die kernel_thread).
> >> If you use the kthread interface keventd will be the parent process and
> >> we won't have problems.
> >
> > so is it something acceptable for mainline ? I think openvz has such
> > a thread doing the reaping.
>
> Please clarify. Is what acceptable for mainline?

I think Cedric is thinking about a per-pidnamespace reaper thread.

I think you and I are just thinking of walking a list of all the
processes with a pid in the namespace, and killing each.

-serge

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by Cedric Le Goater on Thu, 22 Mar 2007 10:44:24 GMT
View Forum Message <> Reply to Message

[long long thread]

Eric W. Biederman wrote:

> Cedric Le Goater <clg@fr.ibm.com> writes:
>
>>>> what about a kthread that would be spawned when a task is cloned in an
>>>> unshared pid namespace ? This is an extra cost in term of tasks.
>>> If you use kernel_thread this can happen. (Die kernel_thread).
>>> If you use the kthread interface keventd will be the parent process and
>>> we won't have problems.
>> so is it something acceptable for mainline ? I think openvz has such
>> a thread doing the reaping.
>
> Please clarify. Is what acceptable for mainline?

[As i kind of jumped in the thread, i did some digging in the thread to
  see where these comments were coming from.]

Correct me if i got something wrong : the initial question is how do we
handle the pid namespace exit and if we mandate task with pid == 1 to be
the last task to die ?

So I suggested to have a kthread be pid == 1 for each new pid namespace.
the kthread can do the killing of all tasks if needed and will die when
the refcount on the pid namespace == 0.

Would such a (rough) design be acceptable for mainline ?

C.

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

---

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by ebiederm on Thu, 22 Mar 2007 12:16:08 GMT
View Forum Message <> Reply to Message

Cedric Le Goater <clg@fr.ibm.com> writes:

> [long long thread]
>
> Eric W. Biederman wrote:
>> Cedric Le Goater <clg@fr.ibm.com> writes:
>>
>>>>> what about a kthread that would be spawned when a task is cloned in an
>>>>> unshared pid namespace ? This is an extra cost in term of tasks.
>>>> If you use kernel_thread this can happen. (Die kernel_thread).
>>>> If you use the kthread interface keventd will be the parent process and
>>>> we won't have problems.

>>> so is it something acceptable for mainline ? I think openvz has such
>>> a thread doing the reaping.
>>
>> Please clarify. Is what acceptable for mainline?
>
> [As i kind of jumped in the thread, i did some digging in the thread to
>   see where these comments were coming from.]
>
> Correct me if i got something wrong : the initial question is how do we
> handle the pid namespace exit and if we mandate task with pid == 1 to be
> the last task to die ?
>
> So I suggested to have a kthread be pid == 1 for each new pid namespace.
> the kthread can do the killing of all tasks if needed and will die when
> the refcount on the pid namespace == 0.
>
> Would such a (rough) design be acceptable for mainline ?

The case that preserves existing semantics requires us to be able to
run /sbin/init in a container. Therefore pid 1 should be a user space
process.

So I don't think a design that doesn't allow us to run /sbin/init as
in a container would be acceptable for mainline.

Eric

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

---

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by Cedric Le Goater on Thu, 22 Mar 2007 13:14:48 GMT
View Forum Message <> Reply to Message

>> So I suggested to have a kthread be pid == 1 for each new pid namespace.
>> the kthread can do the killing of all tasks if needed and will die when
>> the refcount on the pid namespace == 0.
>>
>> Would such a (rough) design be acceptable for mainline ?
>
> The case that preserves existing semantics requires us to be able to
> run /sbin/init in a container. Therefore pid 1 should be a user space
> process.

/sbin/init can't run without being pid == 1. hmm ? need to check. When we
have more of the pid namespace, it should be easier.

> So I don't think a design that doesn't allow us to run /sbin/init as
> in a container would be acceptable for mainline.

I agree that user space is assuming that /sbin/init has pid == 1 but don't
you think that's a strong assumption ?

on the kernel side we have is_init() so it shouldn't be an issue.

C.

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

---

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by ebiederm on Thu, 22 Mar 2007 14:16:41 GMT
View Forum Message <> Reply to Message

Cedric Le Goater <clg@fr.ibm.com> writes:

>>> So I suggested to have a kthread be pid == 1 for each new pid namespace.
>>> the kthread can do the killing of all tasks if needed and will die when
>>> the refcount on the pid namespace == 0.
>>>
>>> Would such a (rough) design be acceptable for mainline ?
>>
>> The case that preserves existing semantics requires us to be able to
>> run /sbin/init in a container. Therefore pid 1 should be a user space
>> process.
>
> /sbin/init can't run without being pid == 1. hmm ? need to check. When we
> have more of the pid namespace, it should be easier.

Correct.

>From sysvinit src/init.c:main
>	/*
>	 *	Is this telinit or init ?
>	 */
>	isinit = (getpid() == 1);
>	for (f = 1; f < argc; f++) {
>		if (!strcmp(argv[f], "-i") || !strcmp(argv[f], "--init"))
>			isinit = 1;
>			break;

> }
> if (!isinit) exit(telinit(p, argc, argv));
>

Plus there are the additional signal handling semantics of pid == 1
where signals are received unless pid == 1 has set up a signal
handler. This especially includes SIGKILL.

>> So I don't think a design that doesn't allow us to run /sbin/init as
>> in a container would be acceptable for mainline.
>
> I agree that user space is assuming that /sbin/init has pid == 1 but don't
> you think that's a strong assumption ?
>
> on the kernel side we have is_init() so it shouldn't be an issue.

Basically there are some of the semantics of pid == 1 that only apply to
the /sbin/init in the initial pid namespace. This is what is_init is for.

There are other semantics that should apply to every process that has
pid == 1, like dropping signals from other processes in it's pid namespace
or children of it's pid namespace that it doesn't have a handler for.

Back to the main subject I still don't understand the idea of running
a kernel daemon as pid == 1. What would that buy us?

Eric

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by serue on Thu, 22 Mar 2007 14:33:50 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
...
> Back to the main subject I still don't understand the idea of running
> a kernel daemon as pid == 1. What would that buy us?

I think the idea is that for lightweight application containers, where
there is no explicit /sbin/init process, the kthread would act as reaper
for the pid_ns so that the first userspace process could freely exit
while other processes continued.

I still prefer that we forego that kthread, and just work toward

allowing pid1 to exit. Really I think the crufty /proc/<pid> handling
is the only reason we were going to punt on that for now. So for our
first stab I think we should have pid=1 exiting cause all other
processes in the same pid_ns to be killed. Then when we get /proc fixed
up, we can change the semantics so that pid=1 exiting just switches the
pid_namespace's reaper to either the parent of the killed pid=1, or to
the global init.

-serge

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by Cedric Le Goater on Thu, 22 Mar 2007 14:49:54 GMT
View Forum Message <> Reply to Message

```
>>From sysvinit src/init.c:main
>>       /*
>>        *      Is this telinit or init ?
>>        */
>>       isinit = (getpid() == 1);
>>       for (f = 1; f < argc; f++) {
>>               if (!strcmp(argv[f], "-i") || !strcmp(argv[f], "--init"))
>>                       isinit = 1;
>>                       break;
>>       }
>>       if (!isinit) exit(telinit(p, argc, argv));
>>
```

ok thanks for looking. I guess this is the end of the discussion on
pid == 1 :)

> [snip]
>
> Back to the main subject I still don't understand the idea of running
> a kernel daemon as pid == 1. What would that buy us?

mostly a child reaper when there are no /sbin/init but its pid cannot
be 1.

C.

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by ebiederm on Thu, 22 Mar 2007 15:10:28 GMT
View Forum Message <> Reply to Message

Cedric Le Goater <clg@fr.ibm.com> writes:

>> Back to the main subject I still don't understand the idea of running
>> a kernel daemon as pid == 1. What would that buy us?
>
> mostly a child reaper when there are no /sbin/init but its pid cannot
> be 1.

Yes we should be able to assign just about any process as the child_reaper
for the pid namespace. That is an entirely distinct concept in the kernel.
Although we have to be careful we don't reap ourselves, so there are some
subtle cases but it isn't a pid issue.

If this is just an application container it might make sense to have a
kernel thread standing in for /sbin/init.

Basically the whole application container thing is something we need to
revisit when we have the basic pid namespace implemented and before we
finalize things, and remove the CONFIG_EXPERIMENTAL tag.

Eric

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

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by Dave Hansen on Thu, 22 Mar 2007 17:07:43 GMT
View Forum Message <> Reply to Message

On Tue, 2007-03-20 at 16:04 -0600, Eric W. Biederman wrote:
> Dave Hansen <hansendc@us.ibm.com> writes:
>
> > On Tue, 2007-03-20 at 09:51 -0600, Eric W. Biederman wrote:
> >> Outlive is the wrong concept. Ideally we want something that will
> >> live as long as there are processes in the pid_ns.
> >
> > How about they just live as long as there is a mount? Now that we
> _can_
> > have multiple superblocks and meaningful vfsmounts, I think it's
> time to

> > make it act like a normal filesystem.
>
> My concern is that the mount will outlive the pid namespace. In which
> case we need something that is safe to test when the pid namespace
> goes away.

So, doesn't that problem go away (or at least move to be umount's duty)
if we completely disconnect those inodes' lifetime from that of any
process or pid namespace?

-- Dave

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by Dave Hansen on Thu, 22 Mar 2007 17:21:19 GMT
View Forum Message <> Reply to Message

On Thu, 2007-03-22 at 09:33 -0500, Serge E. Hallyn wrote:
>
>
> I still prefer that we forego that kthread, and just work toward
> allowing pid1 to exit. Really I think the crufty /proc/<pid> handling
> is the only reason we were going to punt on that for now.

It's everything _but_ the /proc/<pid> stuff. /proc/{mem,cpu}info and
friends are the ones suspiciously tied to pid 1.

-- Dave

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by serue on Thu, 22 Mar 2007 17:26:43 GMT
View Forum Message <> Reply to Message

Quoting Dave Hansen (hansendc@us.ibm.com):
> On Thu, 2007-03-22 at 09:33 -0500, Serge E. Hallyn wrote:
> >

> >
> > I still prefer that we forego that kthread, and just work toward
> > allowing pid1 to exit. Really I think the crufty /proc/<pid> handling
> > is the only reason we were going to punt on that for now.
>
> It's everything _but_ the /proc/<pid> stuff. /proc/{mem,cpu}info and
> friends are the ones suspiciously tied to pid 1.

Oh, right. I was looking at the proc_mnt->mnt_sb->s_root->d_inode->pid
being tied to find_get_pid(1) and misreading it.

So you're right it may be even easier than I thought to fix.

-serge

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

---

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by ebiederm on Thu, 22 Mar 2007 19:40:53 GMT
View Forum Message <> Reply to Message

Dave Hansen <hansendc@us.ibm.com> writes:

> On Thu, 2007-03-22 at 09:33 -0500, Serge E. Hallyn wrote:
>>
>>
>> I still prefer that we forego that kthread, and just work toward
>> allowing pid1 to exit. Really I think the crufty /proc/<pid> handling
>> is the only reason we were going to punt on that for now.
>
> It's everything _but_ the /proc/<pid> stuff. /proc/{mem,cpu}info and
> friends are the ones suspiciously tied to pid 1.

The files and directories affected by the oddity you spotted are:
/proc/
/proc/self
/proc/<pid>

That is the complete list.

Only /proc/self and /proc/<pid> Actually test for the presence of pid == 1.

Eric

_____
Containers mailing list

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by ebiederm on Thu, 22 Mar 2007 20:17:40 GMT
View Forum Message <> Reply to Message

Dave Hansen <hansendc@us.ibm.com> writes:

> So, doesn't that problem go away (or at least move to be umount's duty)
> if we completely disconnect those inodes' lifetime from that of any
> process or pid namespace?

If the last process has exited the pid namespace I would like the
code to continue to behave as it currently does.

I would like readdir on /proc/ to not even try to show any pids when
there are no pids or pid related files in the pid namespace.

I would like /proc/self to completely disappear when the are not
any pids in the pid namespace.

I misspoke in when I said that /proc/<pid> was affected. The function
is proc_pid_readdir and it is a subset of /proc/ so it gets a little
confusing.

Eric

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

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by Herbert Poetzl on Fri, 23 Mar 2007 00:57:30 GMT
View Forum Message <> Reply to Message

On Mon, Mar 19, 2007 at 08:04:12PM -0600, Eric W. Biederman wrote:
> Dave Hansen <hansendc@us.ibm.com> writes:
>
> > I was tracking down why we need find_get_pid(1) in
> > proc_get_sb(), when I realized that we apparently
> > don't need a pid at all in the non-pid parts of /proc.
> >
> > Anyone see any problems with this approach?

>
> The thing is these are pid related parts of /proc you are
> working with.
>
>
> I'm trying to remember what the actual semantics were.
>
> I do know doing this means if our pid namespace goes away these
> functions do the right thing.
>
> This may have been how I was getting the pid namespace in originally
> so this code may be obsolete.
>
> Partly I think doing this made the code a little more symmetric.
>
> Regardless I would like to see a little farther down on
> how we test to see if the pid namespace is alive and how we
> make these functions do nothing if it has died. I would also
> like to see how we perform the appropriate lookups by pid namespace.
>
> Basically I want to see how we finish up multiple namespace support
> for /proc before we start with the micro optimizations.
>
>
> I'm fairly certain this patch causes us to do the wrong thing when
> the pid namespace exits, and I don't see much gain except for the
> death of find_get_pid.
>
>
> > For what I would imagine are historical reasons, we set
> > all struct proc_inode->pid fields. We use the init
> > process for all non-/proc/<pid> inodes.
> >
> > We get a handle to the init process in proc_get_sb()
> > then fetch it out in proc_pid_readdir():
> >
> > struct task_struct *reaper =
> > get_proc_task(filp->f_path.dentry->d_inode);
> >
> > The filp in that case is always the root inode on which
> > someone is doing a readdir. This reaper variable gets
> > passed down into proc_base_instantiate() and eventually
> > set in the new inode's ->pid field.
> >
> > The problem is that I don't see anywhere that we
> > actually go and use this, outside of the /proc/<pid>
> > directories. Just referencing the init process like
> > this is a pain for containers because our init process

> > (pid == 1) can actually go away.
>
> Which as far as can recall is part of the point. If you have a pid
> namespace with normal semantics the child reaper pid == 1 is the last
> pid in the pid namespace to exit. Therefore when it exists the pid
> namespace exists and with it doesn't the pid namespace does not exist.

what about lightweight pid spaces, which do not have
a real init process/pid?

IMHO we should define the pid namespace by the
processes and thus it would seize to exist when
the last process leaves the pid space

best,
Herbert

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

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

---

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by Herbert Poetzl on Fri, 23 Mar 2007 01:02:03 GMT
View Forum Message <> Reply to Message

On Tue, Mar 20, 2007 at 11:00:57AM -0500, Serge E. Hallyn wrote:
> Quoting Eric W. Biederman (ebiederm@xmission.com):
> > "Serge E. Hallyn" <serue@us.ibm.com> writes:
> >
> > > Quoting Eric W. Biederman (ebiederm@xmission.com):
> > >> Dave Hansen <hansendc@us.ibm.com> writes:
> > >> > On Mon, 2007-03-19 at 20:04 -0600, Eric W. Biederman wrote:
> >
> > >> >> I would also
> > >> >> like to see how we perform the appropriate lookups by pid
> > >> >> namespace.
> > >> >
> > >> > What do you mean?
> > >>
> > >> proc_pid_readdir ... next_tgid().
> > >

> > > next_tgid() is simple enough - we can always use current->pid_ns
> > > to find the next pidnr.
> >
> > No. We cannot use current->pid_ns. We must get it from the mount or
> > something in the mount.
>
> Actually I think Dave has it coming from superblock data.
>
> > Using current to set the default pid_ns to mount is fine. But if
> > we use current to select our files we have a moderately serious
> > problem.
> >
> > > The only hitch, as mentioned earlier, is how do we find the first
> > > task. Currently task 1 is statically stored as the first inode,
> > > and as Dave mentioned we can't do that now, because we dont' know
> > > of any one task which will outlive the pid_ns.
> >
> > Outlive is the wrong concept. Ideally we want something that will
> > live as long as there are processes in the pid_ns.
>
> And there is no such thing.
>
> > As I thought about this some more there are some problems for
> > holding a reference to a pid_ns for a long period of time. Currently
> > struct_pid is designed so you can hang onto it forever. struct
> > pid_namespace isn't. So we have some very interesting semantic
> > questions of what happens when the pid namespace exits.
> >
> > Since we distinguish mounts by their pid namespace this looks like
> > something we need to sort through.
>
> Yup.
>
> > >> While I'm not categorically opposed to supporting things like
> > >> that it but it is something for which we need to tread very
> > >> carefully because it is an extension of current semantics. I
> > >> can't think of any weird semantics right now but for something
> > >> user visible we will have to support indefinitely I don't see a
> > >> reason to rush into it either.
> > >
> > > Except that unless we mandate that pid1 in any namespace can't
> > > exit, and put that feature off until later, we can't not address
> > > it.
> >
> > What if we mandate that pid1 is the last process to exit?
>
> I think people have complained about that in the past for application
> containers, but I really don't see where it hurts anything.

>
> Cedric, Herbert, did one of you think it would be bad?

yes, we (Linux-VServer) consider that bad, because it
would not allow to have lightweight containers which
do not have a real init process ...

e.g. think: 'guest running sshd only'

thanks,
Herbert

> > Problems actually only show up in this context if other pids live
> > substantially longer than pid1.
> >
> > >> True but we are getting close. And it is about time we worked up
> > >> patches for that so our conversations can become less theoretical.
> > >
> > > Yes I really hope a patchset goes out today.
> >
> > Sounds good. I expect it will take a couple of rounds of review,
> > before we have all of the little things nailed down but starting that
> > process is a hopeful sign.
>
> I'm hoping some of the earlier patches can be acked this time so we can
> get to discussing the more interesting parts :)
>
> But I'm afraid it might be no earlier than tomorrow that the patches go
> out. Will try.
>
> thanks,
> -serge
> _____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containers

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

---

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by Herbert Poetzl on Fri, 23 Mar 2007 01:06:27 GMT
View Forum Message <> Reply to Message

On Thu, Mar 22, 2007 at 02:14:48PM +0100, Cedric Le Goater wrote:
>

> >> So I suggested to have a kthread be pid == 1 for each new pid
> >> namespace. the kthread can do the killing of all tasks if needed
> >> and will die when the refcount on the pid namespace == 0.
> >>
> >> Would such a (rough) design be acceptable for mainline ?
> >
> > The case that preserves existing semantics requires us to be able
> > to run /sbin/init in a container. Therefore pid 1 should be a user
> > space process.
>
> /sbin/init can't run without being pid == 1. hmm ? need to check. When
> we have more of the pid namespace, it should be easier.
>
> > So I don't think a design that doesn't allow us to run /sbin/init as
> > in a container would be acceptable for mainline.
>
> I agree that user space is assuming that /sbin/init has pid == 1 but
> don't you think that's a strong assumption ?

most inits around even act differently depending on
the pid, e.g. they act as telinit when pid != 1
so while it might be a wrong assumption, almost all
inits on Linux make that assumption and would need
to be changed ...

best,
Herbert

> on the kernel side we have is_init() so it shouldn't be an issue.
>
> C.
> _____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containers

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

---

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by Herbert Poetzl on Fri, 23 Mar 2007 01:10:52 GMT
View Forum Message <> Reply to Message

On Thu, Mar 22, 2007 at 09:33:50AM -0500, Serge E. Hallyn wrote:
> Quoting Eric W. Biederman (ebiederm@xmission.com):
> ...

> > Back to the main subject I still don't understand the idea of running
> > a kernel daemon as pid == 1. What would that buy us?
>
> I think the idea is that for lightweight application containers, where
> there is no explicit /sbin/init process, the kthread would act as
> reaper for the pid_ns so that the first userspace process could freely
> exit while other processes continued.

ah, that might actually work, but the question
remains, what resources would such a kernel thread
consume?

think 500 containers with

 a) one process running inside
 b) one process and a kernel thread

if the kernel thread uses up only half the amount
of resources the actual process does, it will
increase the overall resource consumption by 50%
(which is quite suboptimal)

best,
Herbert

> I still prefer that we forego that kthread, and just work toward
> allowing pid1 to exit. Really I think the crufty /proc/<pid> handling
> is the only reason we were going to punt on that for now. So for our
> first stab I think we should have pid=1 exiting cause all other
> processes in the same pid_ns to be killed. Then when we get /proc fixed
> up, we can change the semantics so that pid=1 exiting just switches the
> pid_namespace's reaper to either the parent of the killed pid=1, or to
> the global init.
>
> -serge
> _____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containers

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

---

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by serue on Mon, 26 Mar 2007 13:54:14 GMT

Quoting Herbert Poetzl (herbert@13thfloor.at):
> On Tue, Mar 20, 2007 at 11:00:57AM -0500, Serge E. Hallyn wrote:
> > Quoting Eric W. Biederman (ebiederm@xmission.com):
> > > "Serge E. Hallyn" <serue@us.ibm.com> writes:
> > >
> > > > Quoting Eric W. Biederman (ebiederm@xmission.com):
> > > >> Dave Hansen <hansendc@us.ibm.com> writes:
> > > >> > On Mon, 2007-03-19 at 20:04 -0600, Eric W. Biederman wrote:
> > >
> > > >> >> I would also
> > > >> >> like to see how we perform the appropriate lookups by pid
> > > >> >> namespace.
> > > >> >
> > > >> > What do you mean?
> > > >>
> > > >> proc_pid_readdir ... next_tgid().
> > > >
> > > > next_tgid() is simple enough - we can always use current->pid_ns
> > > > to find the next pidnr.
> > >
> > > No. We cannot use current->pid_ns. We must get it from the mount or
> > > something in the mount.
> >
> > Actually I think Dave has it coming from superblock data.
> >
> > > Using current to set the default pid_ns to mount is fine. But if
> > > we use current to select our files we have a moderately serious
> > > problem.
> > >
> > > > The only hitch, as mentioned earlier, is how do we find the first
> > > > task. Currently task 1 is statically stored as the first inode,
> > > > and as Dave mentioned we can't do that now, because we dont' know
> > > > of any one task which will outlive the pid_ns.
> > >
> > > Outlive is the wrong concept. Ideally we want something that will
> > > live as long as there are processes in the pid_ns.
> >
> > And there is no such thing.
> >
> > > As I thought about this some more there are some problems for
> > > holding a reference to a pid_ns for a long period of time. Currently
> > > struct_pid is designed so you can hang onto it forever. struct
> > > pid_namespace isn't. So we have some very interesting semantic
> > > questions of what happens when the pid namespace exits.
> > >
> > > Since we distinguish mounts by their pid namespace this looks like
> > > something we need to sort through.

> >
> > Yup.
> >
> > > >> While I'm not categorically opposed to supporting things like
> > > >> that it but it is something for which we need to tread very
> > > >> carefully because it is an extension of current semantics. I
> > > >> can't think of any weird semantics right now but for something
> > > >> user visible we will have to support indefinitely I don't see a
> > > >> reason to rush into it either.
> > > >
> > > > Except that unless we mandate that pid1 in any namespace can't
> > > > exit, and put that feature off until later, we can't not address
> > > > it.
> > >
> > > What if we mandate that pid1 is the last process to exit?
> >
> > I think people have complained about that in the past for application
> > containers, but I really don't see where it hurts anything.
> >
> > Cedric, Herbert, did one of you think it would be bad?
>
> yes, we (Linux-VServer) consider that bad, because it
> would not allow to have lightweight containers which
> do not have a real init process ...
>
> e.g. think: 'guest running sshd only'

The way I'm testing pidspaces right now is

 ns_exec -c -p /usr/sbin/sshd -p 9999

in which case sshd is pid1. Works fine...

Would it be very limiting to have the first process have to stick
around? (I'm asking, not criticizing - it's *my* preference that we
allow pid==1 to exit, but if that's really not advantageous then
maybe it's not worth fixing the ugly pieces that require it right
now - afaik right now that's only the fact that PROC_INODE(/proc)->pid
points to the struct pid for pidnr==1)

-serge

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

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by serue on Mon, 26 Mar 2007 14:13:48 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> Dave Hansen <hansendc@us.ibm.com> writes:
>
> > So, doesn't that problem go away (or at least move to be umount's duty)
> > if we completely disconnect those inodes' lifetime from that of any
> > process or pid namespace?
>
> If the last process has exited the pid namespace I would like the
> code to continue to behave as it currently does.
>
> I would like readdir on /proc/ to not even try to show any pids when
> there are no pids or pid related files in the pid namespace.

In (at least one version of) Dave's patches, the /proc your pidns is automatically used when you use /proc. In that case a /proc should just go away when the last task goes away, since noone else can use that /proc.

I like that behavior, because otherwise (a) we require every new pid_namespace to start by remounting /proc ere they get undefined behavior, and (b) to gain anything from it, we would need a way to refer to another pidspace for the sake of mounting it's proc, i.e.

```
 mount -t proc -o init_pid=7501 proc_vserver1 /vserver1/proc
```

-serge

> I would like /proc/self to completely disappear when the are not
> any pids in the pid namespace.
>
> I misspoke in when I said that /proc/<pid> was affected. The function
> is proc_pid_readdir and it is a subset of /proc/ so it gets a little
> confusing.

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

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by Herbert Poetzl on Mon, 26 Mar 2007 14:36:55 GMT
View Forum Message <> Reply to Message

On Mon, Mar 26, 2007 at 08:54:14AM -0500, Serge E. Hallyn wrote:
> Quoting Herbert Poetzl (herbert@13thfloor.at):
> > On Tue, Mar 20, 2007 at 11:00:57AM -0500, Serge E. Hallyn wrote:
> > > Quoting Eric W. Biederman (ebiederm@xmission.com):
> > > > "Serge E. Hallyn" <serue@us.ibm.com> writes:
> > > >
> > > > > Quoting Eric W. Biederman (ebiederm@xmission.com):
> > > > >> Dave Hansen <hansendc@us.ibm.com> writes:
> > > > >> > On Mon, 2007-03-19 at 20:04 -0600, Eric W. Biederman wrote:
> > > > >
> > > > >> >> I would also
> > > > >> >> like to see how we perform the appropriate lookups by pid
> > > > >> >> namespace.
> > > > >> >
> > > > >> > What do you mean?
> > > > >>
> > > > >> proc_pid_readdir ... next_tgid().
> > > > >
> > > > > next_tgid() is simple enough - we can always use current->pid_ns
> > > > > to find the next pidnr.
> > > > >
> > > > No. We cannot use current->pid_ns. We must get it from the mount or
> > > > something in the mount.
> > > >
> > > Actually I think Dave has it coming from superblock data.
> > >
> > > Using current to set the default pid_ns to mount is fine. But if
> > > we use current to select our files we have a moderately serious
> > > problem.
> > >
> > > > The only hitch, as mentioned earlier, is how do we find the first
> > > > task. Currently task 1 is statically stored as the first inode,
> > > > and as Dave mentioned we can't do that now, because we dont' know
> > > > of any one task which will outlive the pid_ns.
> > > >
> > > Outlive is the wrong concept. Ideally we want something that will
> > > live as long as there are processes in the pid_ns.
> > >
> > And there is no such thing.
> >
> > > As I thought about this some more there are some problems for
> > > holding a reference to a pid_ns for a long period of time. Currently
> > > struct_pid is designed so you can hang onto it forever. struct
> > > pid_namespace isn't. So we have some very interesting semantic
> > > questions of what happens when the pid namespace exits.
> > >
> > > Since we distinguish mounts by their pid namespace this looks like
> > > something we need to sort through.

> > >
> > > Yup.
> > >
> > > > >> While I'm not categorically opposed to supporting things like
> > > > >> that it but it is something for which we need to tread very
> > > > >> carefully because it is an extension of current semantics. I
> > > > >> can't think of any weird semantics right now but for something
> > > > >> user visible we will have to support indefinitely I don't see a
> > > > >> reason to rush into it either.
> > > > > >
> > > > > > Except that unless we mandate that pid1 in any namespace can't
> > > > > > exit, and put that feature off until later, we can't not address
> > > > > > it.
> > > > > >
> > > > > > What if we mandate that pid1 is the last process to exit?
> > > > >
> > > > > I think people have complained about that in the past for application
> > > > > containers, but I really don't see where it hurts anything.
> > > > >
> > > > > Cedric, Herbert, did one of you think it would be bad?
> > > >
> > > > yes, we (Linux-VServer) consider that bad, because it
> > > > would not allow to have lightweight containers which
> > > > do not have a real init process ...
> > > >
> > > > e.g. think: 'guest running sshd only'
> > >
> > > The way I'm testing pidspaces right now is
> > >
> > > ns_exec -c -p /usr/sbin/sshd -p 9999
> > >
> > > in which case sshd is pid1. Works fine...
> > >
> > > Would it be very limiting to have the first process have to stick
> > > around? (I'm asking, not criticizing - it's *my* preference that we
> > > allow pid==1 to exit, but if that's really not advantageous then
> > > maybe it's not worth fixing the ugly pieces that require it right now
> > > - afaik right now that's only the fact that PROC_INODE(/proc)->pid
> > > points to the struct pid for pidnr==1)

again, we basically support 3 different guest models
(regarding init) which probably can be best explained
with an example ...

1) blend through/fake init (from the host system)

```
  USER   PID %CPU %MEM   VSZ  RSS TTY   STAT START   TIME COMMAND
  root     1  6.0  1.9  2036 1096 ?     S    14:24   0:06 init
```

```
root    38  0.7  0.8  2832  448 ?     S    14:26   0:00 sleep 1000
root    43 50.0  1.2  2536  676 ?     R    14:26   0:00 ps auxwww
```

2) a real init process (running inside the guest with pid=1)

```
USER   PID %CPU %MEM   VSZ  RSS TTY   STAT START   TIME COMMAND
root     1  1.6  0.7  2832  444 ?     S    14:26   0:00 sleep 1000
root    44  0.0  1.2  2536  676 ?     R    14:26   0:00 ps auxwww
```

3) no init process (inside a guest)

```
USER   PID %CPU %MEM   VSZ  RSS TTY   STAT START   TIME COMMAND
root    42  0.4  0.7  2828  444 ?     S    14:26   0:00 sleep 1000
root    45 38.0  1.2  2536  676 ?     R    14:26   0:00 ps auxwww
```

in cases 1) and 3) the 'first' process is in no
way special for the Guest, and must not be treated
special .. it can also go away anytime without
affecting the other guest processes ...

case 2) could in theory handle the pid=1 process
(which might not be the first process, but usually
is a special init process) special, and it would
be acceptable to zap the context when this process
dies off ...

note that the cases 1) and 2) are the most commonly
used cases as most init processes do not handle case
3) yet. still case 3) is important for application
isolation too (which doesn't need any init)

HTC,
Herbert

> -serge

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

---

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by ebiederm on Mon, 26 Mar 2007 17:12:36 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:

> Quoting Eric W. Biederman (ebiederm@xmission.com):
>> Dave Hansen <hansendc@us.ibm.com> writes:
>>
>> > So, doesn't that problem go away (or at least move to be umount's duty)
>> > if we completely disconnect those inodes' lifetime from that of any
>> > process or pid namespace?
>>
>> If the last process has exited the pid namespace I would like the
>> code to continue to behave as it currently does.
>>
>> I would like readdir on /proc/ to not even try to show any pids when
>> there are no pids or pid related files in the pid namespace.
>
> In (at least one version of) Dave's patches, the /proc your pidns is
> automatically used when you use /proc. In that case a /proc should
> just go away when the last task goes away, since noone else can use
> that /proc.

Unless I am rather confused that does extremely nasty things to
the VFS dentry cache. Because a dentry can point at one process
one minute and another process the next. It is doable but only
at the cost of decreased performance.

> I like that behavior, because otherwise (a) we require every new
> pid_namespace to start by remounting /proc ere they get undefined
> behavior,

The behavior won't be undefined just unexpected. Given the way
the vfs caching works the requirement for mounting /proc after
an we create a new copy of the pid namespace is a hard requirement.

> and (b) to gain anything from it, we would need a way
> to refer to another pidspace for the sake of mounting it's proc,
> i.e.
>
> mount -t proc -o init_pid=7501 proc_vserver1 /vserver1/proc

First we gain by not thrashing the dcache, and destroying /proc
performance.

Second we can use it if we unshare the mount namespace after we
create a separate pid namespace.

Third an option that points at the pid of a child process to dig
out the mount namespace isn't that hard, and is a simple extension.

Fourth there is an additional issue. There is the process related
part of /proc that is in fs/proc/base.c and then there is the

non-process related part of /proc in fs/proc/generic.c that probably
should have different rules.

Eric

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

---

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by Dave Hansen on Mon, 26 Mar 2007 17:20:10 GMT
View Forum Message <> Reply to Message

On Mon, 2007-03-26 at 11:12 -0600, Eric W. Biederman wrote:
>
> > In (at least one version of) Dave's patches, the /proc your pidns is
> > automatically used when you use /proc. In that case a /proc should
> > just go away when the last task goes away, since noone else can use
> > that /proc.
>
> Unless I am rather confused that does extremely nasty things to
> the VFS dentry cache. Because a dentry can point at one process
> one minute and another process the next. It is doable but only
> at the cost of decreased performance.

By using, I think Serge meant "mounting". We're going to statically
assign a /proc mount to a namespace when the mount is created, not fudge
it around at runtime.

How does this thrash the dcache?

-- Dave

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

---

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by ebiederm on Mon, 26 Mar 2007 17:24:06 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:

> The way I'm testing pidspaces right now is

>
> ns_exec -c -p /usr/sbin/sshd -p 9999
>
> in which case sshd is pid1. Works fine...
>
> Would it be very limiting to have the first process have to stick
> around? (I'm asking, not criticizing - it's *my* preference that we
> allow pid==1 to exit, but if that's really not advantageous then
> maybe it's not worth fixing the ugly pieces that require it right
> now - afaik right now that's only the fact that PROC_INODE(/proc)->pid
> points to the struct pid for pidnr==1)

The /proc part is easy to fix. All I want to see there is that
we do the right thing with pid related files. When the pid namespace
is empty.

The practical reason for only allowing a pid namespace while pid == 1
exists, is something much more simple.

pid == 1 must exists today. We get into an extension of the semantics
if we allow the case where pid == 1 exists. Semantic extensions
can be very tricky, and we are way to early to see what the impact
of such a semantic extension would be.

Therefore I request that we get a correct and work pid namespace
before we try and extend things.

I also request that until questions like this are settles we leave the
whole thing CONFIG_EXPERIMENTAL.

I have yet to see how we are going to implement things such as
kill -1. And the other changes. There are huge chunks of
functionality that we haven't gotten to yet.

Eric

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

---

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by ebiederm on Mon, 26 Mar 2007 17:32:22 GMT
View Forum Message <> Reply to Message

Herbert Poetzl <herbert@13thfloor.at> writes:

>

> again, we basically support 3 different guest models
> (regarding init) which probably can be best explained
> with an example ...
>
> 1) blend through/fake init (from the host system)
>
> USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
> root 1 6.0 1.9 2036 1096 ? S 14:24 0:06 init
> root 38 0.7 0.8 2832 448 ? S 14:26 0:00 sleep 1000
> root 43 50.0 1.2 2536 676 ? R 14:26 0:00 ps auxwww
>
> 2) a real init process (running inside the guest with pid=1)
>
> USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
> root 1 1.6 0.7 2832 444 ? S 14:26 0:00 sleep 1000
> root 44 0.0 1.2 2536 676 ? R 14:26 0:00 ps auxwww
>
> 3) no init process (inside a guest)
>
> USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
> root 42 0.4 0.7 2828 444 ? S 14:26 0:00 sleep 1000
> root 45 38.0 1.2 2536 676 ? R 14:26 0:00 ps auxwww
>
>
> in cases 1) and 3) the 'first' process is in no
> way special for the Guest, and must not be treated
> special .. it can also go away anytime without
> affecting the other guest processes ...
>
> case 2) could in theory handle the pid=1 process
> (which might not be the first process, but usually
> is a special init process) special, and it would
> be acceptable to zap the context when this process
> dies off ...
>
> note that the cases 1) and 2) are the most commonly
> used cases as most init processes do not handle case
> 3) yet. still case 3) is important for application
> isolation too (which doesn't need any init)

>From a maintenance standpoint options like this can be horrible.

The practical question is this. For application isolation what
problems have you encountered with running an application as pid == 1?

Why do you need the no init process inside a guest case?

If you can answer this question when it comes time to optimize things

it will give us incentive to solve these cases.

Eric

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by ebiederm on Mon, 26 Mar 2007 17:39:03 GMT
View Forum Message <> Reply to Message

Dave Hansen <hansendc@us.ibm.com> writes:

> On Mon, 2007-03-26 at 11:12 -0600, Eric W. Biederman wrote:
>>
>> > In (at least one version of) Dave's patches, the /proc your pidns is
>> > automatically used when you use /proc. In that case a /proc should
>> > just go away when the last task goes away, since noone else can use
>> > that /proc.
>>
>> Unless I am rather confused that does extremely nasty things to
>> the VFS dentry cache. Because a dentry can point at one process
>> one minute and another process the next. It is doable but only
>> at the cost of decreased performance.
>
> By using, I think Serge meant "mounting". We're going to statically
> assign a /proc mount to a namespace when the mount is created, not fudge
> it around at runtime.

I think Serge misread that. He specifically talked about not having
to remount /proc. Therefore I believe he understood it did something
at runtime.

> How does this thrash the dcache?

Fudging at runtime is horrible.

Caching the pid_mnt in the pid_namespace appears fine, as long as
we get the reference counting correct.

Eric

Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by serue on Mon, 26 Mar 2007 18:36:13 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> Dave Hansen <hansendc@us.ibm.com> writes:
>
> > On Mon, 2007-03-26 at 11:12 -0600, Eric W. Biederman wrote:
> >>
> >> > In (at least one version of) Dave's patches, the /proc your pidns is
> >> > automatically used when you use /proc. In that case a /proc should
> >> > just go away when the last task goes away, since noone else can use
> >> > that /proc.
> >>
> >> Unless I am rather confused that does extremely nasty things to
> >> the VFS dentry cache. Because a dentry can point at one process
> >> one minute and another process the next. It is doable but only
> >> at the cost of decreased performance.
> >
> > By using, I think Serge meant "mounting". We're going to statically
> > assign a /proc mount to a namespace when the mount is created, not fudge
> > it around at runtime.
>
> I think Serge misread that. He specifically talked about not having
> to remount /proc. Therefore I believe he understood it did something
> at runtime.

Yes, looks like I misread the code.

I thought a new sb was created automatically if needed. Going back I
see this is only done at mount time.

> > How does this thrash the dcache?
>
> Fudging at runtime is horrible.
>
> Caching the pid_mnt in the pid_namespace appears fine, as long as
> we get the reference counting correct.
>
> Eric

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