Subject: [PATCH 2/3] Pid ns helpers for signals
Posted by Sukadev Bhattiprolu on Fri, 31 Aug 2007 20:36:34 GMT
View Forum Message <> Reply to Message

Define some helper functions that will be used to implement signal semantics
with multiple pid namespaces.

 is_current_in_ancestor_pid_ns(task)

  TRUE iff active pid namespace of 'current' is an ancestor of
  active pid namespace of @task.

 is_current_in_same_or_ancestor_pid_ns(task)

  TRUE iff active pid namespace of 'current' is either same as
  or an ancestor of active pid namespace of @task.

 pid_ns_equal(tsk)
  TRUE if active pid ns of @tsk is same as active pid ns of
  'current'.

These interfaces take into account the fact that the tasks they are operating
on may be exiting and may have already exited their namespaces (i.e their
pid namespaces may be NULL).

Changelog: [Oleg Nesterov]: Renamed helpers and grab a reference to 'struct
    pid' and 'struct pid_namespace'.
---
 include/linux/pid.h | 15 ++++++
 include/linux/pid_namespace.h | 4 -
 kernel/pid.c | 93 ++
 3 files changed, 110 insertions(+), 2 deletions(-)

Index: 2.6.23-rc3-mm1/include/linux/pid.h
===
--- 2.6.23-rc3-mm1.orig/include/linux/pid.h 2007-08-31 00:45:18.000000000 -0700
+++ 2.6.23-rc3-mm1/include/linux/pid.h 2007-08-31 12:38:08.000000000 -0700
@@ -125,6 +125,21 @@ extern void FASTCALL(free_pid(struct pid
 extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);

 /*
+ * Caller must hold a reference to @pid.
+ */
+static inline struct pid_namespace *pid_active_ns(struct pid *pid)
+{
+ if (!pid)
+ return NULL;
+

```
+ return pid->numbers[pid->level].ns;
+}
+
+extern int pid_ns_equal(struct task_struct *tsk);
+extern int is_current_in_ancestor_pid_ns(struct task_struct *tsk);
+extern int is_current_in_same_or_ancestor_pid_ns(struct task_struct *tsk);
+
+/*
  * the helpers to get the pid's id seen from different namespaces
  *
  * pid_nr()    : global id, i.e. the id seen from the init namespace;
Index: 2.6.23-rc3-mm1/kernel/pid.c
===================================================================
--- 2.6.23-rc3-mm1.orig/kernel/pid.c 2007-08-31 00:45:18.000000000 -0700
+++ 2.6.23-rc3-mm1/kernel/pid.c 2007-08-31 12:29:10.000000000 -0700
@@ -199,6 +199,99 @@ static int next_pidmap(struct pid_namesp
  return -1;
 }

+static struct pid_namespace *get_task_pid_ns(struct task_struct *tsk)
+{
+ struct pid *pid;
+ struct pid_namespace *ns;
+
+ pid = get_task_pid(tsk, PIDTYPE_PID);
+ ns = get_pid_ns(pid_active_ns(pid));
+ put_pid(pid);
+
+ return ns;
+}
+
+/*
+ * Return TRUE if the active pid namespace of @tsk is same as active
+ * pid namespace of 'current'.
+ */
+int pid_ns_equal(struct task_struct *tsk)
+{
+ int rc;
+ struct pid_namespace *my_ns = get_task_pid_ns(current);
+ struct pid_namespace *tsk_ns = get_task_pid_ns(tsk);
+
+ rc = (my_ns == tsk_ns);
+
+ put_pid_ns(my_ns);
+ put_pid_ns(tsk_ns);
+
+ return rc;
+}
```

```
+
+/*
+ * Return TRUE if pid namespace @ns1 is an ancestor of pid namespace @ns2.
+ * Return FALSE otherwise.
+ *
+ * Note: Callers must ensure @ns1 and @ns2 are stable.
+ */
+static int ancestor_pid_ns(struct pid_namespace *ns1, struct pid_namespace *ns2)
+{
+ int i;
+ struct pid_namespace *tmp;
+
+ if (ns1 == NULL || ns2 == NULL)
+  return 0;
+
+ if (ns1->level >= ns2->level)
+  return 0;
+
+ tmp = ns2->parent;
+ for (i = tmp->level; i >= ns1->level; i--) {
+  if (tmp == ns1)
+   return 1;
+  tmp = tmp->parent;
+ }
+
+ return 0;
+}
+
+/*
+ * Return TRUE if active pid namespace of 'current' is an ancestor of
+ * pid namespace of @tsk. Return FALSE otherwise.
+ */
+int is_current_in_ancestor_pid_ns(struct task_struct *tsk)
+{
+ int rc;
+ struct pid_namespace *my_ns = get_task_pid_ns(current);
+ struct pid_namespace *tsk_ns = get_task_pid_ns(tsk);
+
+ rc = ancestor_pid_ns(my_ns, tsk_ns);
+
+ put_pid_ns(my_ns);
+ put_pid_ns(tsk_ns);
+
+ return rc;
+}
+
+/*
+ * Return TRUE if active pid namespace of 'current' is either same as
```

```
+ * or an ancestor of active pid namespace of @tsk.
+ */
+int is_current_in_same_or_ancestor_pid_ns(struct task_struct *tsk)
+{
+ int rc;
+ struct pid_namespace *my_ns = get_task_pid_ns(current);
+ struct pid_namespace *tsk_ns = get_task_pid_ns(tsk);
+
+ rc = my_ns == tsk_ns || ancestor_pid_ns(my_ns, tsk_ns);
+
+ put_pid_ns(my_ns);
+ put_pid_ns(tsk_ns);
+
+ return rc;
+}
+
 fastcall void put_pid(struct pid *pid)
 {
  struct pid_namespace *ns;
Index: 2.6.23-rc3-mm1/include/linux/pid_namespace.h
===================================================================
--- 2.6.23-rc3-mm1.orig/include/linux/pid_namespace.h 2007-08-31 00:45:18.000000000 -0700
+++ 2.6.23-rc3-mm1/include/linux/pid_namespace.h 2007-08-31 12:15:09.000000000 -0700
@@ -31,7 +31,7 @@ extern struct pid_namespace init_pid_ns;

 static inline struct pid_namespace *get_pid_ns(struct pid_namespace *ns)
 {
- if (ns != &init_pid_ns)
+ if (ns && ns != &init_pid_ns)
   kref_get(&ns->kref);
  return ns;
 }
@@ -41,7 +41,7 @@ extern void free_pid_ns(struct kref *kre

 static inline void put_pid_ns(struct pid_namespace *ns)
 {
- if (ns != &init_pid_ns)
+ if (ns && ns != &init_pid_ns)
   kref_put(&ns->kref, free_pid_ns);
 }
```

Subject: Re: [PATCH 2/3] Pid ns helpers for signals
Posted by Oleg Nesterov on Sat, 01 Sep 2007 11:29:03 GMT
View Forum Message <> Reply to Message

On 08/31, sukadev@us.ibm.com wrote:
>
> Define some helper functions that will be used to implement signal semantics
> with multiple pid namespaces.
>
> is_current_in_ancestor_pid_ns(task)
>
>   TRUE iff active pid namespace of 'current' is an ancestor of
>   active pid namespace of @task.
>
> is_current_in_same_or_ancestor_pid_ns(task)
>
>   TRUE iff active pid namespace of 'current' is either same as
>   or an ancestor of active pid namespace of @task.

These names are awfull :) Yes, yes, it was me who suggested them... No, I can't
suggest something better.

> + * Caller must hold a reference to @pid.
> + */
> +static inline struct pid_namespace *pid_active_ns(struct pid *pid)
> +{
> + if (!pid)
> + return NULL;
> +
> + return pid->numbers[pid->level].ns;
> +}

Well, the comment is a bit misleading. Yes, my previous comment was not very
clear. Yes, the function itself is not safe unless you know what are you doing,
like, for example, get_pid(). I think it is better to just kill the comment.
Please see below.

> +static struct pid_namespace *get_task_pid_ns(struct task_struct *tsk)
> +{
> + struct pid *pid;
> + struct pid_namespace *ns;
> +
> + pid = get_task_pid(tsk, PIDTYPE_PID);
> + ns = get_pid_ns(pid_active_ns(pid));
> + put_pid(pid);
> +
> + return ns;
> +}

Hmm. Firstly, we don't need this for the "current", but all users of this func
also do get_task_pid_ns(current).

Also, we don't need get/put_pid. rcu locks are enough,

```
 rcu_read_lock();
 ns = get_pid_ns(pid_active_ns(task_pid(tks)));
 rcu_read_unlock();
```

However, do we really need this complications right now? Currently, we use
this "compare namespaces" helpers only when we know that "struct pid" is
stable. We are sending the signal to that task, it must be pid_alive(), and
we either locked the task itself, or we hold tasklist.

Oleg.

Subject: Re: [PATCH 2/3] Pid ns helpers for signals
Posted by Oleg Nesterov on Sat, 01 Sep 2007 11:56:01 GMT
View Forum Message <> Reply to Message

On 09/01, Oleg Nesterov wrote:
>
> On 08/31, sukadev@us.ibm.com wrote:
> >
> > +static struct pid_namespace *get_task_pid_ns(struct task_struct *tsk)
> > +{
> > + struct pid *pid;
> > + struct pid_namespace *ns;
> > +
> > + pid = get_task_pid(tsk, PIDTYPE_PID);
> > + ns = get_pid_ns(pid_active_ns(pid));
> > + put_pid(pid);
> > +
> > + return ns;
> > +}
>
> Hmm. Firstly, we don't need this for the "current", but all users of this func
> also do get_task_pid_ns(current).
>
> Also, we don't need get/put_pid. rcu locks are enough,
>
> rcu_read_lock();

> ns = get_pid_ns(pid_active_ns(task_pid(tks)));
> rcu_read_unlock();
>
> However, do we really need this complications right now? Currently, we use
> this "compare namespaces" helpers only when we know that "struct pid" is
> stable. We are sending the signal to that task, it must be pid_alive(), and
> we either locked the task itself, or we hold tasklist.

(forgot to mention)

Otherwise, it is not safe to use "tsk" in get_task_pid_ns(), so I don't think
these get/put pid/pid_ns games make too much sense.

Oleg.

Subject: Re: [PATCH 2/3] Pid ns helpers for signals
Posted by Sukadev Bhattiprolu on Mon, 03 Sep 2007 16:01:47 GMT
View Forum Message <> Reply to Message

Oleg Nesterov [oleg@tv-sign.ru] wrote:
| On 09/01, Oleg Nesterov wrote:
| >
| > On 08/31, sukadev@us.ibm.com wrote:
| > >
| > > +static struct pid_namespace *get_task_pid_ns(struct task_struct *tsk)
| > > +{
| > > + struct pid *pid;
| > > + struct pid_namespace *ns;
| > > +
| > > + pid = get_task_pid(tsk, PIDTYPE_PID);
| > > + ns = get_pid_ns(pid_active_ns(pid));
| > > + put_pid(pid);
| > > +
| > > + return ns;
| > > +}
| >
| > Hmm. Firstly, we don't need this for the "current", but all users of this func
| > also do get_task_pid_ns(current).
| >
| > Also, we don't need get/put_pid. rcu locks are enough,
| >
| > rcu_read_lock();

| > ns = get_pid_ns(pid_active_ns(task_pid(tks)));
| > rcu_read_unlock();
| >
| > However, do we really need this complications right now? Currently, we use
| > this "compare namespaces" helpers only when we know that "struct pid" is
| > stable. We are sending the signal to that task, it must be pid_alive(), and
| > we either locked the task itself, or we hold tasklist.
|
| (forgot to mention)
|
| Otherwise, it is not safe to use "tsk" in get_task_pid_ns(), so I don't think
| these get/put pid/pid_ns games make too much sense.

get_pid(), put_pid(), get_pid_ns(), put_pid_ns() all allow pid to be NULL.
You mean tsk itself can be NULL bc task is exiting ?

|
| Oleg.

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

---

Subject: Re: [PATCH 2/3] Pid ns helpers for signals
Posted by Oleg Nesterov on Mon, 03 Sep 2007 16:24:55 GMT
View Forum Message <> Reply to Message

On 09/03, sukadev@us.ibm.com wrote:
>
> Oleg Nesterov [oleg@tv-sign.ru] wrote:
> | On 09/01, Oleg Nesterov wrote:
> | >
> | > On 08/31, sukadev@us.ibm.com wrote:
> | > >
> | > > +static struct pid_namespace *get_task_pid_ns(struct task_struct *tsk)
> | > > +{
> | > > + struct pid *pid;
> | > > + struct pid_namespace *ns;
> | > > +
> | > > + pid = get_task_pid(tsk, PIDTYPE_PID);
> | > > + ns = get_pid_ns(pid_active_ns(pid));
> | > > + put_pid(pid);
> | > > +
> | > > + return ns;
> | > > +}
> | >
> | > Hmm. Firstly, we don't need this for the "current", but all users of this func

> | > also do get_task_pid_ns(current).
> | >
> | > Also, we don't need get/put_pid. rcu locks are enough,
> | >
> | > rcu_read_lock();
> | > ns = get_pid_ns(pid_active_ns(task_pid(tks)));
> | > rcu_read_unlock();
> | >
> | > However, do we really need this complications right now? Currently, we use
> | > this "compare namespaces" helpers only when we know that "struct pid" is
> | > stable. We are sending the signal to that task, it must be pid_alive(), and
> | > we either locked the task itself, or we hold tasklist.
> |
> | (forgot to mention)
> |
> | Otherwise, it is not safe to use "tsk" in get_task_pid_ns(), so I don't think
> | these get/put pid/pid_ns games make too much sense.
>
> get_pid(), put_pid(), get_pid_ns(), put_pid_ns() all allow pid to be NULL.
> You mean tsk itself can be NULL bc task is exiting ?

My apologies, I was unclear (as always ;)

No, tsk can't be NULL, and its memory can't be freed, because we use this func
from signal sending path (see above).

But this (signal sending path) also guarantees that its pid is also stable and
can't be NULL, no need to get/put pid, or even check it is not NULL.

Oleg.

Subject: Re: [PATCH 2/3] Pid ns helpers for signals
Posted by Sukadev Bhattiprolu on Mon, 03 Sep 2007 16:55:14 GMT
View Forum Message <> Reply to Message

Oleg Nesterov [oleg@tv-sign.ru] wrote:
| On 08/31, sukadev@us.ibm.com wrote:
| >
| > Define some helper functions that will be used to implement signal semantics
| > with multiple pid namespaces.
| >
| > is_current_in_ancestor_pid_ns(task)

| >
| >   TRUE iff active pid namespace of 'current' is an ancestor of
| >   active pid namespace of @task.
| >
| > is_current_in_same_or_ancestor_pid_ns(task)
| >
| >   TRUE iff active pid namespace of 'current' is either same as
| >   or an ancestor of active pid namespace of @task.
|
| These names are awfull :) Yes, yes, it was me who suggested them... No, I can't
| suggest something better.

I agree :-) I tried smaller names like task_ancestor_pid_ns() and passing in
'current' as a parameter so its not in the name :-) but the functionality was
not obvious from the names.

|
| > + * Caller must hold a reference to @pid.
| > + */
| > +static inline struct pid_namespace *pid_active_ns(struct pid *pid)
| > +{
| > + if (!pid)
| > + return NULL;
| > +
| > + return pid->numbers[pid->level].ns;
| > +}
|
| Well, the comment is a bit misleading. Yes, my previous comment was not very
| clear. Yes, the function itself is not safe unless you know what are you doing,
| like, for example, get_pid(). I think it is better to just kill the comment.
| Please see below.

Ok. will remove the comment.
|
| > +static struct pid_namespace *get_task_pid_ns(struct task_struct *tsk)
| > +{
| > + struct pid *pid;
| > + struct pid_namespace *ns;
| > +
| > + pid = get_task_pid(tsk, PIDTYPE_PID);
| > + ns = get_pid_ns(pid_active_ns(pid));
| > + put_pid(pid);
| > +
| > + return ns;
| > +}
|
| Hmm. Firstly, we don't need this for the "current", but all users of this func
| also do get_task_pid_ns(current).

|
| Also, we don't need get/put_pid. rcu locks are enough,
|
| rcu_read_lock();
| ns = get_pid_ns(pid_active_ns(task_pid(tks)));
| rcu_read_unlock();
|

Ok.

| However, do we really need this complications right now? Currently, we use
| this "compare namespaces" helpers only when we know that "struct pid" is
| stable. We are sending the signal to that task, it must be pid_alive(), and
| we either locked the task itself, or we hold tasklist.

My concern was that the task could detach and free its pid which in turn
would drop the last reference on a pid namespace and free it.

By trying to keep is_current_in_ancestor*() general, I guess it is more
complicated than it needs to be right now.

Would holding the rcu_read_lock() be enough or since our callers hold
it now, can we just drop that too ?

is_current_in_ancstor_pid_ns(tsk)

```
 rcu_read_lock();
 my_ns = pid_active_ns(current);
 tsk_ns = pid_active_ns(tsk)
 rc = is_ancestor_ns(my_ns, tsk_ns)
 rcu_read_unlock();

 return rc;
```

Thanks for the comments,

Suka

Subject: Re: [PATCH 2/3] Pid ns helpers for signals
Posted by Oleg Nesterov on Mon, 03 Sep 2007 17:14:20 GMT
View Forum Message <> Reply to Message

On 09/03, sukadev@us.ibm.com wrote:

>
> By trying to keep is_current_in_ancestor*() general, I guess it is more
> complicated than it needs to be right now.
>
> Would holding the rcu_read_lock() be enough or since our callers hold
> it now, can we just drop that too ?

I think rcu_read_lock() is not needed right now, and we can drop it.

Oleg.