Subject: [PATCH 1/9] sig: Fix mqueue pid
Posted by ebiederm on Wed, 12 Dec 2007 12:40:54 GMT
View Forum Message <> Reply to Message

Currently in the sig info for posix message queues we are reporting the
task id and not the task group id. Given that this is a posix interface
and that si_pid is defined by posix as returning the process id, we should
be reporting the task group id.

So fix this interface.

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

```
---
 ipc/mqueue.c |    2 +-
 1 files changed, 1 insertions(+), 1 deletions(-)

diff --git a/ipc/mqueue.c b/ipc/mqueue.c
index 7d1b8aa..d3feadf 100644
--- a/ipc/mqueue.c
+++ b/ipc/mqueue.c
@@ -510,7 +510,7 @@ static void __do_notify(struct mqueue_inode_info *info)
 	sig_i.si_errno = 0;
 	sig_i.si_code = SI_MESGQ;
 	sig_i.si_value = info->notify.sigev_value;
-	sig_i.si_pid = task_pid_vnr(current);
+	sig_i.si_pid = task_tgid_vnr(current);
 	sig_i.si_uid = current->uid;

 	kill_pid_info(info->notify.sigev_signo,
-- 
1.5.3.rc6.17.g1911
```

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

Subject: [PATCH 2/9] sig: Fix SI_USER si_pid
Posted by ebiederm on Wed, 12 Dec 2007 12:42:24 GMT
View Forum Message <> Reply to Message

Currently we have a few instances where we are generating signals
setting si_code to SI_USER and si_pid to the task id.

However in the case of SI_USER we are using a posix defined
interface, and posix defines si_pid as the sending process id. Which
in linux is equivalent to the task group id.

So fix SI_USER to fill in the proper si_pid value.

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

```
---
 kernel/signal.c |    4 ++--
 1 files changed, 2 insertions(+), 2 deletions(-)

diff --git a/kernel/signal.c b/kernel/signal.c
index 280bccb..694a643 100644
--- a/kernel/signal.c
+++ b/kernel/signal.c
@@ -694,7 +694,7 @@ static int send_signal(int sig, struct siginfo *info, struct task_struct *t,
    q->info.si_signo = sig;
    q->info.si_errno = 0;
    q->info.si_code = SI_USER;
-   q->info.si_pid = task_pid_vnr(current);
+   q->info.si_pid = task_tgid_vnr(current);
    q->info.si_uid = current->uid;
    break;
   case (unsigned long) SEND_SIG_PRIV:
@@ -1794,7 +1794,7 @@ relock:
     info->si_signo = signr;
     info->si_errno = 0;
     info->si_code = SI_USER;
-    info->si_pid = task_pid_vnr(current->parent);
+    info->si_pid = task_tgid_vnr(current->parent);
     info->si_uid = current->parent->uid;
    }

-- 
1.5.3.rc6.17.g1911
```

Subject: [PATCH 3/9] pid: Implement ns_of_pid.
Posted by ebiederm on Wed, 12 Dec 2007 12:44:01 GMT

View Forum Message <> Reply to Message

A current problem with the pid namespace is that it is
easy to do pid related work after exit_task_namespaces which
drops the nsproxy pointer.

However if we are doing pid namespace related work we are

always operating on some struct pid which retains the pid_namespace
pointer of the pid namespace it was allocated in.

So provide ns_of_pid which allows us to find the pid
namespace a pid was allocated in.

Using this we have the needed infrastructure to do pid
namespace related work at anytime we have a struct pid,
removing the chance of accidentally having a NULL
pointer dereference when accessing current->nsproxy.

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

```
---
 include/linux/pid.h |   11 +++++++++++
 1 files changed, 11 insertions(+), 0 deletions(-)

diff --git a/include/linux/pid.h b/include/linux/pid.h
index f84d532..c4b56c0 100644
--- a/include/linux/pid.h
+++ b/include/linux/pid.h
@@ -123,6 +123,17 @@ int next_pidmap(struct pid_namespace *pid_ns, int last);
 extern struct pid *alloc_pid(struct pid_namespace *ns);
 extern void FASTCALL(free_pid(struct pid *pid));

+/* ns_of_pid returns the pid namespace in which the specified
+ * pid was allocated.
+ */
+static inline struct pid_namespace *ns_of_pid(struct pid *pid)
+{
+ struct pid_namespace *ns = NULL;
+ if (pid)
+  ns = pid->numbers[pid->level].ns;
+ return ns;
+}
+
 /*
  * the helpers to get the pid's id seen from different namespaces
  *
--
1.5.3.rc6.17.g1911
```

Subject: [PATCH 4/9] pid: Generalize task_active_pid_ns
Posted by ebiederm on Wed, 12 Dec 2007 12:46:55 GMT
View Forum Message <> Reply to Message

Currently task_active_pid_ns is not safe to call after a
task becomes a zombie and exit_task_namespaces is called,
as nsproxy becomes NULL. By reading the pid namespace from
the pid of the task we can trivially solve this problem at
the cost of one extra memory read in what should be the
same cacheline as we read the namespace from.

When moving things around I have made task_active_pid_ns
out of line because keeping it in pid_namespace.h would
require adding includes of pid.h and sched.h that I
don't think we want.

This change does make task_active_pid_ns unsafe to call during
copy_process until we attach a pid on the task_struct which
seems to be a reasonable trade off.

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

```
---
 include/linux/pid_namespace.h |    5 +----
 kernel/fork.c                 |    4 ++--
 kernel/pid.c                  |    6 ++++++
 3 files changed, 9 insertions(+), 6 deletions(-)

diff --git a/include/linux/pid_namespace.h b/include/linux/pid_namespace.h
index fcd61fa..ab13faa 100644
--- a/include/linux/pid_namespace.h
+++ b/include/linux/pid_namespace.h
@@ -74,10 +74,7 @@ static inline void zap_pid_ns_processes(struct pid_namespace *ns)
 }
 #endif /* CONFIG_PID_NS */

-static inline struct pid_namespace *task_active_pid_ns(struct task_struct *tsk)
-{
- return tsk->nsproxy->pid_ns;
-}
+extern struct pid_namespace *task_active_pid_ns(struct task_struct *tsk);

 static inline struct task_struct *task_child_reaper(struct task_struct *tsk)
 {
diff --git a/kernel/fork.c b/kernel/fork.c
index 32d75d8..a210860 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -1164,12 +1164,12 @@ static struct task_struct *copy_process(unsigned long clone_flags,
  if (pid != &init_struct_pid) {
   retval = -ENOMEM;
-  pid = alloc_pid(task_active_pid_ns(p));
+  pid = alloc_pid(p->nsproxy->pid_ns);
   if (!pid)
    goto bad_fork_cleanup_namespaces;

   if (clone_flags & CLONE_NEWPID) {
-   retval = pid_ns_prepare_proc(task_active_pid_ns(p));
+   retval = pid_ns_prepare_proc(p->nsproxy->pid_ns);
    if (retval < 0)
     goto bad_fork_free_pid;
   }
diff --git a/kernel/pid.c b/kernel/pid.c
index c507ca7..54d7634 100644
--- a/kernel/pid.c
+++ b/kernel/pid.c
@@ -472,6 +472,12 @@ pid_t task_session_nr_ns(struct task_struct *tsk, struct pid_namespace
*ns)
 }
 EXPORT_SYMBOL(task_session_nr_ns);

+struct pid_namespace *task_active_pid_ns(struct task_struct *tsk)
+{
+ return ns_of_pid(task_pid(tsk));
+}
+EXPORT_SYMBOL_GPL(task_active_pid_ns);
+
 /*
  * Used by proc to find the first pid that is greater then or equal to nr.
  *
--
1.5.3.rc6.17.g1911
```

____

Subject: Re: [PATCH 1/9] sig: Fix mqueue pid
Posted by Pavel Emelianov on Wed, 12 Dec 2007 13:24:47 GMT

View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> Currently in the sig info for posix message queues we are reporting the
> task id and not the task group id. Given that this is a posix interface
> and that si_pid is defined by posix as returning the process id, we should

```
> be reporting the task group id.
>
> So fix this interface.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
```

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

```
> ---
>  ipc/mqueue.c |    2 +-
>  1 files changed, 1 insertions(+), 1 deletions(-)
>
> diff --git a/ipc/mqueue.c b/ipc/mqueue.c
> index 7d1b8aa..d3feadf 100644
> --- a/ipc/mqueue.c
> +++ b/ipc/mqueue.c
> @@ -510,7 +510,7 @@ static void __do_notify(struct mqueue_inode_info *info)
>     sig_i.si_errno = 0;
>     sig_i.si_code = SI_MESGQ;
>     sig_i.si_value = info->notify.sigev_value;
> -   sig_i.si_pid = task_pid_vnr(current);
> +   sig_i.si_pid = task_tgid_vnr(current);
>     sig_i.si_uid = current->uid;
>
>     kill_pid_info(info->notify.sigev_signo,
```

Subject: Re: [PATCH 3/9] pid: Implement ns_of_pid.
Posted by Sukadev Bhattiprolu on Thu, 13 Dec 2007 00:59:45 GMT

View Forum Message <> Reply to Message

Eric W. Biederman [ebiederm@xmission.com] wrote:

|
| A current problem with the pid namespace is that it is
| easy to do pid related work after exit_task_namespaces which
| drops the nsproxy pointer.
|
| However if we are doing pid namespace related work we are
| always operating on some struct pid which retains the pid_namespace
| pointer of the pid namespace it was allocated in.
|
| So provide ns_of_pid which allows us to find the pid
| namespace a pid was allocated in.

|
| Using this we have the needed infrastructure to do pid
| namespace related work at anytime we have a struct pid,
| removing the chance of accidentally having a NULL
| pointer dereference when accessing current->nsproxy.

Yep.

|
| Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
| ---
| include/linux/pid.h | 11 +++++++++++
| 1 files changed, 11 insertions(+), 0 deletions(-)
|
| diff --git a/include/linux/pid.h b/include/linux/pid.h
| index f84d532..c4b56c0 100644
| --- a/include/linux/pid.h
| +++ b/include/linux/pid.h
| @@ -123,6 +123,17 @@ int next_pidmap(struct pid_namespace *pid_ns, int last);
| extern struct pid *alloc_pid(struct pid_namespace *ns);
| extern void FASTCALL(free_pid(struct pid *pid));
|
| +/* ns_of_pid returns the pid namespace in which the specified
| + * pid was allocated.
| + */
| +static inline struct pid_namespace *ns_of_pid(struct pid *pid)

My patch refers to this function as pid_active_pid_ns() - I have
been meaning to send that out on top of your signals patch.
Since a pid has many namespaces, we have been using 'active pid ns'
to refer to this ns.

Even your next patch modifies task_active_pid_ns() to use this.
So can we rename this functio to pid_active_pid_ns() ?

| +{
| + struct pid_namespace *ns = NULL;
| + if (pid)
| + ns = pid->numbers[pid->level].ns;
| + return ns;
| +}
| +
| /*
| * the helpers to get the pid's id seen from different namespaces
| *
| --
| 1.5.3.rc6.17.g1911

Subject: Re: [PATCH 3/9] pid: Implement ns_of_pid.
Posted by ebiederm on Thu, 13 Dec 2007 01:25:44 GMT
View Forum Message <> Reply to Message

sukadev@us.ibm.com writes:

>
> My patch refers to this function as pid_active_pid_ns() - I have
> been meaning to send that out on top of your signals patch.
> Since a pid has many namespaces, we have been using 'active pid ns'
> to refer to this ns.

Currently we don't ask for any of the others, and the namespace
the pid came from is special. That fundamentally is the namespace
of the pid. The rest byproducts of being in that pid namespace,
as we could derive them by walking the namespace's parent list.

> Even your next patch modifies task_active_pid_ns() to use this.
> So can we rename this functio to pid_active_pid_ns() ?

I'd be more inclined to rename task_active_pid_ns to task_pid_ns.

And to rename pid_in_pid_ns that Pavel has issues with to pid_in_ns.

When I read active_pid_ns I wonder what the other namespaces are
that we are distinguishing this from. They do exist in the
implementation but so far it is a complete don't care.

So I expect being as terse as we can while still conveying all of the
relevant information is the most maintainable long term.

Eric

Subject: Re: [PATCH 3/9] pid: Implement ns_of_pid.
Posted by Sukadev Bhattiprolu on Thu, 13 Dec 2007 03:28:27 GMT
View Forum Message <> Reply to Message

Eric W. Biederman [ebiederm@xmission.com] wrote:
| sukadev@us.ibm.com writes:
|
| >
| > My patch refers to this function as pid_active_pid_ns() - I have
| > been meaning to send that out on top of your signals patch.
| > Since a pid has many namespaces, we have been using 'active pid ns'
| > to refer to this ns.
|
| Currently we don't ask for any of the others, and the namespace
| the pid came from is special. That fundamentally is the namespace
| of the pid. The rest byproducts of being in that pid namespace,
| as we could derive them by walking the namespace's parent list.
|
| > Even your next patch modifies task_active_pid_ns() to use this.
| > So can we rename this functio to pid_active_pid_ns() ?
|
| I'd be more inclined to rename task_active_pid_ns to task_pid_ns.
|
| And to rename pid_in_pid_ns that Pavel has issues with to pid_in_ns.
|
| When I read active_pid_ns I wonder what the other namespaces are
| that we are distinguishing this from. They do exist in the
| implementation but so far it is a complete don't care.

Well, there are interfaces like pid_nr_ns() and pid_in_ns() and
task_in_pid_ns() that imply existence of other namespaces and
for that reason we added 'active' in the name.

But I am fine with the terse name and of course we should remove
the the 'active' in task_active_pid_ns() also.

|
| So I expect being as terse as we can while still conveying all of the
| relevant information is the most maintainable long term.
|
| Eric

I did some initial testing on your patchset (minus patch 5) and noticed
that it seems to be missing the patch to address kill -1 semantics
(here is the earlier version).

This patch implements task_in_pid_ns and uses it to limit cap_set_all
and sys_kill(-1,) to only those tasks in the current pid namespace.

Without this we have a setup for a very nasty surprise.

```
Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 include/linux/pid_namespace.h |    2 ++
 kernel/capability.c           |    3 +++
 kernel/pid.c                  |   11 +++++++++++
 kernel/signal.c               |    5 ++++-
 4 files changed, 20 insertions(+), 1 deletions(-)

diff --git a/include/linux/pid_namespace.h b/include/linux/pid_namespace.h
index 0227e68..b454678 100644
--- a/include/linux/pid_namespace.h
+++ b/include/linux/pid_namespace.h
@@ -78,4 +78,6 @@ static inline struct task_struct *task_child_reaper(struct task_struct *tsk)
  return tsk->nsproxy->pid_ns->child_reaper;
 }

+extern int task_in_pid_ns(struct task_struct *tsk, struct pid_namespace *ns);
+
 #endif /* _LINUX_PID_NS_H */
diff --git a/kernel/capability.c b/kernel/capability.c
index efbd9cd..a801016 100644
--- a/kernel/capability.c
+++ b/kernel/capability.c
@@ -125,6 +125,7 @@ static inline int cap_set_all(kernel_cap_t *effective,
          kernel_cap_t *inheritable,
          kernel_cap_t *permitted)
 {
+     struct pid_namespace *pid_ns = current->nsproxy->pid_ns;
      struct task_struct *g, *target;
      int ret = -EPERM;
      int found = 0;
@@ -132,6 +133,8 @@ static inline int cap_set_all(kernel_cap_t *effective,
      do_each_thread(g, target) {
             if (target == current || is_container_init(target->group_leader))
                    continue;
+             if (!task_in_pid_ns(target, pid_ns))
+      continue;
             found = 1;
       if (security_capset_check(target, effective, inheritable,
       permitted))
diff --git a/kernel/pid.c b/kernel/pid.c
index f815455..1c332ca 100644
--- a/kernel/pid.c
+++ b/kernel/pid.c
@@ -430,6 +430,17 @@ struct pid *find_get_pid(pid_t nr)
  return pid;
 }
```

```
+static int pid_in_pid_ns(struct pid *pid, struct pid_namespace *ns)
+{
+ return pid && (ns->level <= pid->level) &&
+  pid->numbers[ns->level].ns == ns;
+}
+
+int task_in_pid_ns(struct task_struct *task, struct pid_namespace *ns)
+{
+ return pid_in_pid_ns(task_pid(task), ns);
+}
+
 pid_t pid_nr_ns(struct pid *pid, struct pid_namespace *ns)
 {
  struct upid *upid;
diff --git a/kernel/signal.c b/kernel/signal.c
index 1200630..8f5a31f 100644
--- a/kernel/signal.c
+++ b/kernel/signal.c
@@ -1147,10 +1147,13 @@ static int kill_something_info(int sig, struct siginfo *info, int pid)
  } else if (pid == -1) {
   int retval = 0, count = 0;
   struct task_struct * p;
+  struct pid_namespace *ns = current->nsproxy->pid_ns;

   read_lock(&tasklist_lock);
   for_each_process(p) {
-   if (p->pid > 1 && !same_thread_group(p, current)) {
+   if (!is_container_init(p) &&
+       !same_thread_group(p, current) &&
+       task_in_pid_ns(p, ns)) {
     int err = group_send_sig_info(sig, info, p);
     ++count;
     if (err != -EPERM)
--
1.5.3.rc6.17.g1911
```

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

Subject: Re: [PATCH 4/9] pid: Generalize task_active_pid_ns
Posted by Oleg Nesterov on Thu, 13 Dec 2007 16:01:28 GMT

View Forum Message <> Reply to Message

Sorry for the delay, and sorry, can't read this series carefully now.
A couple of question though.

On 12/12, Eric W. Biederman wrote:
>
> Currently task_active_pid_ns is not safe to call after a
> task becomes a zombie and exit_task_namespaces is called,
> as nsproxy becomes NULL. By reading the pid namespace from
> the pid of the task we can trivially solve this problem

Confused. If the task becomes a zombie, we can't assume it has a valid
->pids[].pid. The parent can release us as soon as exit_notify() drops
tasklist.

?

Oleg.

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

---

Subject: Re: [PATCH 4/9] pid: Generalize task_active_pid_ns
Posted by ebiederm on Thu, 13 Dec 2007 16:22:39 GMT

View Forum Message <> Reply to Message

Oleg Nesterov <oleg@tv-sign.ru> writes:

> Sorry for the delay, and sorry, can't read this series carefully now.
> A couple of question though.
>
> On 12/12, Eric W. Biederman wrote:
>>
>> Currently task_active_pid_ns is not safe to call after a
>> task becomes a zombie and exit_task_namespaces is called,
>> as nsproxy becomes NULL. By reading the pid namespace from
>> the pid of the task we can trivially solve this problem
>
> Confused. If the task becomes a zombie, we can't assume it has a valid
> ->pids[].pid. The parent can release us as soon as exit_notify() drops
> tasklist.

Where this really matters is in the signal sending code. By the time
I have acquired sighand lock I know release_task has executed and
thus my ->pids[].pid is valid, because __exit_signal has not completed
and thus __unhash_process has not yet run.

When release_task_gets called we are EXIT_DEAD unhashed and unfindable

and I don't care. I do however care about finding my pid namespace
as long as the task is on hashed.

So as long as we have tasklist_lock or sighand lock and we can find
the task we are good.

What this allows me to do (as seen later in the patchset) is to send
to call pid_nr_ns and deliver a signal to a task group without caring
if the element of the task group I am talking to is a zombie or not.
Which is rather important when the task group leader has exited
and a zombie, but yet it is the task all of the signals are sent
to and group_send_siginfo is called on.

Eric

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

---

Subject: Re: [PATCH 4/9] pid: Generalize task_active_pid_ns
Posted by Oleg Nesterov on Thu, 13 Dec 2007 17:07:27 GMT

View Forum Message <> Reply to Message

On 12/13, Eric W. Biederman wrote:
>
> Oleg Nesterov <oleg@tv-sign.ru> writes:
>
> > On 12/12, Eric W. Biederman wrote:
> >>
> >> Currently task_active_pid_ns is not safe to call after a
> >> task becomes a zombie and exit_task_namespaces is called,
> >> as nsproxy becomes NULL. By reading the pid namespace from
> >> the pid of the task we can trivially solve this problem
> >
> > Confused. If the task becomes a zombie, we can't assume it has a valid
> > ->pids[].pid. The parent can release us as soon as exit_notify() drops
> > tasklist.
>
>
> What this allows me to do (as seen later in the patchset) is to send
> to call pid_nr_ns and deliver a signal to a task group without caring
> if the element of the task group I am talking to is a zombie or not.

Yes I see, thanks Eric. I was confused by changelog, and I missed the
subsequent changes which need task_active_pid_ns() of the reciever.

Oleg.

Subject: Re: [PATCH 3/9] pid: Implement ns_of_pid.
Posted by Sukadev Bhattiprolu on Sat, 15 Dec 2007 00:35:38 GMT

View Forum Message <> Reply to Message

sukadev@us.ibm.com [sukadev@us.ibm.com] wrote:

| I did some initial testing on your patchset (minus patch 5) and noticed
| that it seems to be missing the patch to address kill -1 semantics
| (here is the earlier version).

I thought I double checked, but still missed the other patchset that
addressed this. Sorry. Will review/test with that also.