Subject: [RFC][PATCH] Add child reaper to struct pspace
Posted by Sukadev Bhattiprolu on Thu, 07 Sep 2006 00:48:52 GMT
View Forum Message <> Reply to Message

Add per-pid-space child-reaper. This is needed so processes are reaped within
the same pid space and do not spill over to the parent pid space. Its also
needed so containers preserve existing semantic that pid == 1 would reap
orphaned children.

This is based on Eric Biederman's patch: http://lkml.org/lkml/2006/2/6/285

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Cc: Eric Biederman <ebiederm@xmission.com>
Cc: Cedric Le Goater <clg@fr.ibm.com>
Cc: Dave Hansen <haveblue@us.ibm.com>
Cc: Serge Hallyn <serue@us.ibm.com>
Cc: containers@lists.osdl.org

```
 fs/exec.c             |    4 ++--
 include/linux/pid.h   |    5 +++--
 include/linux/pspace.h |    1 +
 include/linux/sched.h  |    2 +-
 init/main.c           |    5 ++---
 kernel/exit.c         |   15 ++++++++-------
 kernel/signal.c       |    8 ++++++--
 7 files changed, 23 insertions(+), 17 deletions(-)

Index: lx26-18-rc5/include/linux/pspace.h
===================================================================
--- lx26-18-rc5.orig/include/linux/pspace.h 2006-09-06 17:04:03.000000000 -0700
+++ lx26-18-rc5/include/linux/pspace.h 2006-09-06 17:39:37.000000000 -0700
@@ -16,6 +16,7 @@ typedef struct pidmap
 struct pspace {
        struct pidmap pidmap[PIDMAP_ENTRIES];
        int last_pid;
+       struct task_struct * child_reaper;
 };

 extern struct pspace init_pspace;
Index: lx26-18-rc5/init/main.c
===================================================================
--- lx26-18-rc5.orig/init/main.c 2006-09-06 17:04:03.000000000 -0700
+++ lx26-18-rc5/init/main.c 2006-09-06 17:05:00.000000000 -0700
@@ -51,6 +51,7 @@
 #include <linux/buffer_head.h>
 #include <linux/debug_locks.h>
 #include <linux/lockdep.h>
+#include <linux/pspace.h>
```

```
 #include <asm/io.h>
 #include <asm/bugs.h>
@@ -596,8 +597,6 @@ static int __init initcall_debug_setup(c
 }
 __setup("initcall_debug", initcall_debug_setup);

-struct task_struct *child_reaper = &init_task;
-
 extern initcall_t __initcall_start[], __initcall_end[];

 static void __init do_initcalls(void)
@@ -697,7 +696,7 @@ static int init(void * unused)
   * assumptions about where in the task array this
   * can be found.
   */
- child_reaper = current;
+ init_pspace.child_reaper = current;

  smp_prepare_cpus(max_cpus);

Index: lx26-18-rc5/fs/exec.c
===================================================================
--- lx26-18-rc5.orig/fs/exec.c 2006-09-06 17:04:03.000000000 -0700
+++ lx26-18-rc5/fs/exec.c 2006-09-06 17:39:29.000000000 -0700
@@ -620,8 +620,8 @@ static int de_thread(struct task_struct
   * Reparenting needs write_lock on tasklist_lock,
   * so it is safe to do it under read_lock.
   */
- if (unlikely(current->group_leader == child_reaper))
-  child_reaper = current;
+ if (unlikely(current->group->leader == current->pspace->child_reaper)
+  current->pspace->child_reaper = current;

  zap_other_threads(current);
  read_unlock(&tasklist_lock);
Index: lx26-18-rc5/include/linux/pid.h
===================================================================
--- lx26-18-rc5.orig/include/linux/pid.h 2006-09-06 17:04:03.000000000 -0700
+++ lx26-18-rc5/include/linux/pid.h 2006-09-06 17:05:00.000000000 -0700
@@ -35,8 +35,9 @@ enum pid_type
  *
  * Holding a reference to struct pid solves both of these problems.
  * It is small so holding a reference does not consume a lot of
- * resources, and since a new struct pid is allocated when the numeric
- * pid value is reused we don't mistakenly refer to new processes.
+ * resources, and since a new struct pid is allocated when the numeric pid
+ * value is reused (when pids wrap around) we don't mistakenly refer to new
```

```
+ * processes.
  */

 struct pid
Index: lx26-18-rc5/include/linux/sched.h
===================================================================
--- lx26-18-rc5.orig/include/linux/sched.h 2006-09-06 17:04:03.000000000 -0700
+++ lx26-18-rc5/include/linux/sched.h 2006-09-06 17:05:00.000000000 -0700
@@ -80,6 +80,7 @@ struct sched_param {
 #include <linux/resource.h>
 #include <linux/timer.h>
 #include <linux/hrtimer.h>
+#include <linux/pspace.h>

 #include <asm/processor.h>

@@ -1297,7 +1298,6 @@ extern NORET_TYPE void do_group_exit(int
 extern void daemonize(const char *, ...);
 extern int allow_signal(int);
 extern int disallow_signal(int);
-extern struct task_struct *child_reaper;

 extern int do_execve(char *, char __user * __user *, char __user * __user *, struct pt_regs *);
 extern long do_fork(unsigned long, unsigned long, struct pt_regs *, unsigned long, int __user *,
int __user *);
Index: lx26-18-rc5/kernel/exit.c
===================================================================
--- lx26-18-rc5.orig/kernel/exit.c 2006-09-06 17:04:03.000000000 -0700
+++ lx26-18-rc5/kernel/exit.c 2006-09-06 17:18:47.000000000 -0700
@@ -45,7 +45,6 @@
 #include <asm/mmu_context.h>

 extern void sem_exit (void);
-extern struct task_struct *child_reaper;

 static void exit_mm(struct task_struct * tsk);

@@ -267,7 +266,8 @@ static int has_stopped_jobs(int pgrp)
 }

 /**
- * reparent_to_init - Reparent the calling kernel thread to the init task.
+ * reparent_to_init - Reparent the calling kernel thread to the init task
+ * of the pid space that the thread belongs to.
  *
  * If a kernel thread is launched as a result of a system call, or if
  * it ever exits, it should generally reparent itself to init so that
@@ -285,8 +285,8 @@ static void reparent_to_init(void)
```

```
  ptrace_unlink(current);
  /* Reparent to init */
  remove_parent(current);
- current->parent = child_reaper;
- current->real_parent = child_reaper;
+ current->parent = current->pspace->child_reaper;
+ current->real_parent = current->pspace->child_reaper;
  add_parent(current);

  /* Set the exit signal to SIGCHLD so we signal init on exit */
@@ -658,7 +658,8 @@ reparent_thread(struct task_struct *p, s
  * When we die, we re-parent all our children.
  * Try to give them to another thread in our thread
  * group, and if no such member exists, give it to
- * the global child reaper process (ie "init")
+ * the child reaper process (ie "init") in our pid
+ * space.
  */
 static void
 forget_original_parent(struct task_struct *father, struct list_head *to_release)
@@ -669,7 +670,7 @@ forget_original_parent(struct task_struc
  do {
   reaper = next_thread(reaper);
   if (reaper == father) {
-   reaper = child_reaper;
+   reaper = father->pspace->child_reaper;
    break;
   }
  } while (reaper->exit_state);
@@ -857,7 +858,7 @@ fastcall NORET_TYPE void do_exit(long co
   panic("Aiee, killing interrupt handler!");
  if (unlikely(!tsk->pid))
   panic("Attempted to kill the idle task!");
- if (unlikely(tsk == child_reaper))
+ if (unlikely(tsk == tsk->pspace->child_reaper))
   panic("Attempted to kill init!");

  if (unlikely(current->ptrace & PT_TRACE_EXIT)) {
Index: lx26-18-rc5/kernel/signal.c
===================================================================
--- lx26-18-rc5.orig/kernel/signal.c 2006-09-06 17:04:03.000000000 -0700
+++ lx26-18-rc5/kernel/signal.c 2006-09-06 17:05:00.000000000 -0700
@@ -1835,8 +1835,12 @@ relock:
   if (sig_kernel_ignore(signr)) /* Default is nothing. */
    continue;

-  /* Init gets no signals it doesn't want.  */
-  if (current == child_reaper)
```

```
+  /*
+   * Init of a pid space gets no signals it doesn't want from
+   * within that pid space. It can of course get signals from
+   * its parent pid space.
+   */
+  if (current == current->pspace->child_reaper)
    continue;

   if (sig_kernel_stop(signr)) {
```

_____

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [RFC][PATCH] Add child reaper to struct pspace
Posted by serue on Thu, 07 Sep 2006 01:39:54 GMT
View Forum Message <> Reply to Message

Quoting Sukadev Bhattiprolu (sukadev@us.ibm.com):
...
> Index: lx26-18-rc5/kernel/exit.c
> ===
> --- lx26-18-rc5.orig/kernel/exit.c 2006-09-06 17:04:03.000000000 -0700
> +++ lx26-18-rc5/kernel/exit.c 2006-09-06 17:18:47.000000000 -0700
> @@ -45,7 +45,6 @@
> #include <asm/mmu_context.h>
>
> extern void sem_exit (void);
> -extern struct task_struct *child_reaper;
>
> static void exit_mm(struct task_struct * tsk);
>
> @@ -267,7 +266,8 @@ static int has_stopped_jobs(int pgrp)
> }
>
> /**
> - * reparent_to_init - Reparent the calling kernel thread to the init task.
> + * reparent_to_init - Reparent the calling kernel thread to the init task
> + * of the pid space that the thread belongs to.
>   *
>   * If a kernel thread is launched as a result of a system call, or if
>   * it ever exits, it should generally reparent itself to init so that
> @@ -285,8 +285,8 @@ static void reparent_to_init(void)
>   ptrace_unlink(current);
>   /* Reparent to init */
>   remove_parent(current);
> - current->parent = child_reaper;

> - current->real_parent = child_reaper;
> + current->parent = current->pspace->child_reaper;
> + current->real_parent = current->pspace->child_reaper;
>   add_parent(current);
>
>   /* Set the exit signal to SIGCHLD so we signal init on exit */
> @@ -658,7 +658,8 @@ reparent_thread(struct task_struct *p, s
>   * When we die, we re-parent all our children.
>   * Try to give them to another thread in our thread
>   * group, and if no such member exists, give it to
> - * the global child reaper process (ie "init")
> + * the child reaper process (ie "init") in our pid
> + * space.
>   */
> static void
> forget_original_parent(struct task_struct *father, struct list_head *to_release)
> @@ -669,7 +670,7 @@ forget_original_parent(struct task_struc
>   do {
>   reaper = next_thread(reaper);
>   if (reaper == father) {
> -   reaper = child_reaper;
> +   reaper = father->pspace->child_reaper;
>   break;
>   }
>   } while (reaper->exit_state);
> @@ -857,7 +858,7 @@ fastcall NORET_TYPE void do_exit(long co
>   panic("Aiee, killing interrupt handler!");
>   if (unlikely(!tsk->pid))
>   panic("Attempted to kill the idle task!");
> - if (unlikely(tsk == child_reaper))
> + if (unlikely(tsk == tsk->pspace->child_reaper))
>   panic("Attempted to kill init!");

That becomes a little uncalled for now :)

Perhaps something more like

```
 if (unlikely(tsk == tsk->pspace->child_reaper)) {
  if (tsk->pspace != &init_pspace)
   tsk->pspace->child_reaper = init_pspace.child_reaper;
  else
   panic("Attempted to kill init!");
 }
```

It might be better yet to walk up the ancestor pspace tree, but I'm not sure it's worth it at that point. There's no more hope of userspace wanting to be informed, so we really just want task cleanup by the kernel, so really any init task will do.

-serge

_____

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [RFC][PATCH] Add child reaper to struct pspace
Posted by Cedric Le Goater on Thu, 07 Sep 2006 09:41:17 GMT
View Forum Message <> Reply to Message

Sukadev Bhattiprolu wrote:

<snip>

> @@ -620,8 +620,8 @@ static int de_thread(struct task_struct
> * Reparenting needs write_lock on tasklist_lock,
> * so it is safe to do it under read_lock.
> */
> - if (unlikely(current->group_leader == child_reaper))
> - child_reaper = current;
> + if (unlikely(current->group->leader == current->pspace->child_reaper)
> + current->pspace->child_reaper = current;
>
> zap_other_threads(current);
> read_unlock(&tasklist_lock);

I'm unsure about this one ?

C.

_____

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [RFC][PATCH] Add child reaper to struct pspace
Posted by dev on Thu, 07 Sep 2006 12:25:41 GMT
View Forum Message <> Reply to Message

Sukadev Bhattiprolu wrote:

[...skip...]

> @@ -857,7 +858,7 @@ fastcall NORET_TYPE void do_exit(long co
> panic("Aiee, killing interrupt handler!");

> if (unlikely(!tsk->pid))
>   panic("Attempted to kill the idle task!");
> - if (unlikely(tsk == child_reaper))
> + if (unlikely(tsk == tsk->pspace->child_reaper))
>   panic("Attempted to kill init!");
this panic is wrong here.

My HO is that termination of init task
should terminate all the other tasks in pspace.
Otherwise you have too select some one else to be child_reaper
and actually there is no good choice here as no one except init
is going to wait() for children.

p.s. can you enumerate these patches please?
since I failed to find where struct pspace was introduced...
imho in previous patch set? was that patch set already commited to -mm?

Thanks,
Kirill

Subject: Re: [RFC][PATCH] Add child reaper to struct pspace
Posted by Cedric Le Goater on Thu, 07 Sep 2006 12:33:10 GMT
View Forum Message <> Reply to Message

<snip>

> */
> static void
> forget_original_parent(struct task_struct *father, struct list_head *to_release)
> @@ -669,7 +670,7 @@ forget_original_parent(struct task_struc
> do {
>   reaper = next_thread(reaper);
>   if (reaper == father) {
> - reaper = child_reaper;
> + reaper = father->pspace->child_reaper;
>   break;
>   }
> } while (reaper->exit_state);
> @@ -857,7 +858,7 @@ fastcall NORET_TYPE void do_exit(long co

what about killing all the task in that pid space if child_reaper == init
dies ?

> panic("Aiee, killing interrupt handler!");
> if (unlikely(!tsk->pid))
> panic("Attempted to kill the idle task!");
> - if (unlikely(tsk == child_reaper))
> + if (unlikely(tsk == tsk->pspace->child_reaper))
> panic("Attempted to kill init!");
>
> if (unlikely(current->ptrace & PT_TRACE_EXIT)) {

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [RFC][PATCH] Add child reaper to struct pspace
Posted by Cedric Le Goater on Thu, 07 Sep 2006 13:01:59 GMT
View Forum Message <> Reply to Message

Kirill Korotaev wrote:
> Sukadev Bhattiprolu wrote:
>
> [...skip...]
>
>> @@ -857,7 +858,7 @@ fastcall NORET_TYPE void do_exit(long co
>> panic("Aiee, killing interrupt handler!");
>> if (unlikely(!tsk->pid))
>> panic("Attempted to kill the idle task!");
>> - if (unlikely(tsk == child_reaper))
>> + if (unlikely(tsk == tsk->pspace->child_reaper))
>> panic("Attempted to kill init!");
> this panic is wrong here.
>
> My HO is that termination of init task
> should terminate all the other tasks in pspace.

yes this should be done, in the complete patchset.

> Otherwise you have too select some one else to be child_reaper
> and actually there is no good choice here as no one except init
> is going to wait() for children.

serge pointed that out also :

http://lists.osdl.org/pipermail/containers/2006-September/000158.html

> p.s. can you enumerate these patches please?
> since I failed to find where struct pspace was introduced...
> imho in previous patch set? was that patch set already commited to -mm?

yes it is in -mm.

C.

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [RFC][PATCH] Add child reaper to struct pspace
Posted by Herbert Poetzl on Thu, 07 Sep 2006 15:41:54 GMT
View Forum Message <> Reply to Message

On Thu, Sep 07, 2006 at 04:25:41PM +0400, Kirill Korotaev wrote:
> Sukadev Bhattiprolu wrote:
>
> [...skip...]
>
> > @@ -857,7 +858,7 @@ fastcall NORET_TYPE void do_exit(long co
> >   panic("Aiee, killing interrupt handler!");
> >   if (unlikely(!tsk->pid))
> >   panic("Attempted to kill the idle task!");
> > - if (unlikely(tsk == child_reaper))
> > + if (unlikely(tsk == tsk->pspace->child_reaper))
> >   panic("Attempted to kill init!");
> this panic is wrong here.
>
> My HO is that termination of init task
> should terminate all the other tasks in pspace.

well, this is one way to handle it, another way
(also implemented in Linux-VServer) is to switch
to the 'main' child_reaper of the host context
and/or to allow to select/start a new reaper
inside the guest, still another option (which I
do not really favor) is to have a kernel thread
which poses as reaper inside the guest (either
once init is gone or in general)

the problem I see here is that once the guest
child_reaper (note: this is not necessary the
init process, but let's assume it is for now :)
terminates, it will not be able to do the reaping
at all, so terminating the other guest processes

might not be as simple as it sounds ....

> Otherwise you have too select some one else to be child_reaper
> and actually there is no good choice here as no one except init
> is going to wait() for children.

except for the host/master init :)

best,
Herbert

> p.s. can you enumerate these patches please?
> since I failed to find where struct pspace was introduced...
> imho in previous patch set? was that patch set already commited to -mm?
>
> Thanks,
> Kirill
>
> _____
> Containers mailing list
> Containers@lists.osdl.org
> https://lists.osdl.org/mailman/listinfo/containers

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [RFC][PATCH] Add child reaper to struct pspace
Posted by Sukadev Bhattiprolu on Thu, 07 Sep 2006 19:33:41 GMT
View Forum Message <> Reply to Message

Cedric Le Goater [clg@fr.ibm.com] wrote:
|
| <snip>
|
| >   */
| > static void
| > forget_original_parent(struct task_struct *father, struct list_head *to_release)
| > @@ -669,7 +670,7 @@ forget_original_parent(struct task_struc
| >   do {
| >   reaper = next_thread(reaper);
| >   if (reaper == father) {
| > -   reaper = child_reaper;
| > +   reaper = father->pspace->child_reaper;
| >   break;
| >   }
| >   } while (reaper->exit_state);

| > @@ -857,7 +858,7 @@ fastcall NORET_TYPE void do_exit(long co
|
| what about killing all the task in that pid space if child_reaper == init
| dies ?
|

We probably need that for instance when a process in the parent pspace
kills the init of a child pspace, we should destroy the child pspace
by killing all the tasks in the child pspace including the child reaper.

I guess we need to maintain a list of task_structs in the pspace and walk
that list. Will work on that as a separate patch.

|
| > panic("Aiee, killing interrupt handler!");
| > if (unlikely(!tsk->pid))
| > panic("Attempted to kill the idle task!");
| > - if (unlikely(tsk == child_reaper))
| > + if (unlikely(tsk == tsk->pspace->child_reaper))
| > panic("Attempted to kill init!");
| >
| > if (unlikely(current->ptrace & PT_TRACE_EXIT)) {
|
| _____
| Containers mailing list
| Containers@lists.osdl.org
| https://lists.osdl.org/mailman/listinfo/containers

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [RFC][PATCH] Add child reaper to struct pspace
Posted by ebiederm on Thu, 07 Sep 2006 20:02:59 GMT
View Forum Message <> Reply to Message

Sukadev Bhattiprolu <sukadev@us.ibm.com> writes:

> Cedric Le Goater [clg@fr.ibm.com] wrote:
> |
> | <snip>
> |
> | > */
> | > static void
> | > forget_original_parent(struct task_struct *father, struct list_head
> *to_release)
> | > @@ -669,7 +670,7 @@ forget_original_parent(struct task_struc

> | >   do {
> | >   reaper = next_thread(reaper);
> | >   if (reaper == father) {
> | > -   reaper = child_reaper;
> | > +   reaper = father->pspace->child_reaper;
> | >   break;
> | >   }
> | >   } while (reaper->exit_state);
> | > @@ -857,7 +858,7 @@ fastcall NORET_TYPE void do_exit(long co
> |
> | what about killing all the task in that pid space if child_reaper == init
> | dies ?
> |
>
> We probably need that for instance when a process in the parent pspace
> kills the init of a child pspace, we should destroy the child pspace
> by killing all the tasks in the child pspace including the child reaper.
>
> I guess we need to maintain a list of task_structs in the pspace and walk
> that list. Will work on that as a separate patch.

Yes. We all so need something like that list to support kill -1.
Although walking the list of all processes may be sufficient for a first
pass.

The real trick is handing nested pid namespaces, properly.

Eric

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [RFC][PATCH] Add child reaper to struct pspace
Posted by Cedric Le Goater on Fri, 08 Sep 2006 10:29:54 GMT
View Forum Message <> Reply to Message

Sukadev Bhattiprolu wrote:
> Cedric Le Goater [clg@fr.ibm.com] wrote:
> |
> | <snip>
> |
> | >   */
> | > static void
> | > forget_original_parent(struct task_struct *father, struct list_head *to_release)
> | > @@ -669,7 +670,7 @@ forget_original_parent(struct task_struc
> | >   do {

> | >   reaper = next_thread(reaper);
> | >   if (reaper == father) {
> | > -   reaper = child_reaper;
> | > +   reaper = father->pspace->child_reaper;
> | >   break;
> | >   }
> | >   } while (reaper->exit_state);
> | > @@ -857,7 +858,7 @@ fastcall NORET_TYPE void do_exit(long co
> |
> | what about killing all the task in that pid space if child_reaper == init
> | dies ?
> |
>
> We probably need that for instance when a process in the parent pspace
> kills the init of a child pspace, we should destroy the child pspace
> by killing all the tasks in the child pspace including the child reaper.
>
> I guess we need to maintain a list of task_structs in the pspace and walk
> that list. Will work on that as a separate patch.

checkout the openvz kernel. it adds a do_initproc_exit() routine in
kernel/exit.c which is interesting to study for this purpose

http://git.openvz.org/

this is not the method followed by vserver, though, which reparents to the
reall init process.

C.

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [RFC][PATCH] Add child reaper to struct pspace
Posted by serue on Fri, 08 Sep 2006 13:25:00 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> Sukadev Bhattiprolu <sukadev@us.ibm.com> writes:
>
> > Cedric Le Goater [clg@fr.ibm.com] wrote:
> > |
> > | <snip>
> > |
> > | >   */
> > | > static void

> > | > forget_original_parent(struct task_struct *father, struct list_head
> > *to_release)
> > | > @@ -669,7 +670,7 @@ forget_original_parent(struct task_struc
> > | >   do {
> > | >   reaper = next_thread(reaper);
> > | >   if (reaper == father) {
> > | > -   reaper = child_reaper;
> > | > +   reaper = father->pspace->child_reaper;
> > | >   break;
> > | >   }
> > | >   } while (reaper->exit_state);
> > | > @@ -857,7 +858,7 @@ fastcall NORET_TYPE void do_exit(long co
> > |
> > | what about killing all the task in that pid space if child_reaper == init
> > | dies ?
> > |
> >
> > We probably need that for instance when a process in the parent pspace
> > kills the init of a child pspace, we should destroy the child pspace
> > by killing all the tasks in the child pspace including the child reaper.
> >
> > I guess we need to maintain a list of task_structs in the pspace and walk
> > that list. Will work on that as a separate patch.
>
> Yes. We all so need something like that list to support kill -1.
> Although walking the list of all processes may be sufficient for a first
> pass.
>
> The real trick is handing nested pid namespaces, properly.

Not if, as you've suggested in the past, pid_ns 5 has valid pids in its
own pid_ns for every process in pid_namespaces nested under it.

It should be simple to implement, should not impact the non-container
cases, and should only start to impact performance as the nesting gets
deep, which AFAIK we all believe won't happen (max nesting of 2 AFAICS,
one checkpointable application container under one vserver-thingie)

And it makes kill -1 trivial, as in pid_ns 5 we just kill all processes
in pid_ns 5, without worrying about finding the ones in it's decendent
pid namespaces.

-serge

Subject: Re: [RFC][PATCH] Add child reaper to struct pspace
Posted by ebiederm on Fri, 08 Sep 2006 14:02:09 GMT
View Forum Message <> Reply to Message

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

> Quoting Eric W. Biederman (ebiederm@xmission.com):
>> Yes. We all so need something like that list to support kill -1.
>> Although walking the list of all processes may be sufficient for a first
>> pass.
>>
>> The real trick is handing nested pid namespaces, properly.
>
> Not if, as you've suggested in the past, pid_ns 5 has valid pids in its
> own pid_ns for every process in pid_namespaces nested under it.
>
> It should be simple to implement, should not impact the non-container
> cases, and should only start to impact performance as the nesting gets
> deep, which AFAIK we all believe won't happen (max nesting of 2 AFAICS,
> one checkpointable application container under one vserver-thingie)
>
> And it makes kill -1 trivial, as in pid_ns 5 we just kill all processes
> in pid_ns 5, without worrying about finding the ones in it's decendent
> pid namespaces.

If you do it correctly I agree. But you have to be very careful where
you put the list.

My point being not that we can't get this correct with simple code, but
that it is easy to get it wrong.

Eric

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [RFC][PATCH] Add child reaper to struct pspace
Posted by serue on Fri, 08 Sep 2006 14:36:08 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):
> >> Yes. We all so need something like that list to support kill -1.
> >> Although walking the list of all processes may be sufficient for a first

> >> pass.
> >>
> >> The real trick is handing nested pid namespaces, properly.
> >
> > Not if, as you've suggested in the past, pid_ns 5 has valid pids in its
> > own pid_ns for every process in pid_namespaces nested under it.
> >
> > It should be simple to implement, should not impact the non-container
> > cases, and should only start to impact performance as the nesting gets
> > deep, which AFAIK we all believe won't happen (max nesting of 2 AFAICS,
> > one checkpointable application container under one vserver-thingie)
> >
> > And it makes kill -1 trivial, as in pid_ns 5 we just kill all processes
> > in pid_ns 5, without worrying about finding the ones in it's decendent
> > pid namespaces.
>
> If you do it correctly I agree. But you have to be very careful where
> you put the list.
>
> My point being not that we can't get this correct with simple code, but
> that it is easy to get it wrong.

True.

I should think the list has to go into the struct pid, not the
task_struct. There is one struct pid per (pid_ns, pid), so we can keep
just one simple list to walk the pid_ns processes. If we were to put it
in the task_struct, well there's really no clean way to go about it then :)

thanks,
-serge

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [RFC][PATCH] Add child reaper to struct pspace
Posted by ebiederm on Fri, 08 Sep 2006 14:59:43 GMT
View Forum Message <> Reply to Message

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

> Quoting Eric W. Biederman (ebiederm@xmission.com):
>>
>> My point being not that we can't get this correct with simple code, but
>> that it is easy to get it wrong.
>

> True.
>
> I should think the list has to go into the struct pid, not the
> task_struct. There is one struct pid per (pid_ns, pid), so we can keep
> just one simple list to walk the pid_ns processes. If we were to put it
> in the task_struct, well there's really no clean way to go about it then :)

Exactly.

Add to this that all of the extra struct pids are really something akin
to symlinks and you are redirected to the primary one on lookup and
you see an even bigger slice of the puzzle. So I'm not really certain
we have additional pids to work with.

readdir in /proc has the same problem (except for the atomic requirement)
and I think I just handled it.

So this is definitely going to be doable but it is something we need
to handle very carefully.

Eric

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers