Subject: [PATCH 12/17] Pid-NS(V3) Terminate processes in a ns when reaper is exiting.
Posted by Sukadev Bhattiprolu on Sat, 16 Jun 2007 23:03:02 GMT
View Forum Message <> Reply to Message

Subject: [PATCH 12/17] Pid-NS(V3) Terminate processes in a ns when reaper is exiting.

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

This should actually be considered a part of the previous patch which
enables cloning of pid namespace. Its been separated out for easier
review.

Terminate all processes in a namespace when the reaper of the namespace
is exiting. We do this by walking the pidmap of the namespace and sending
SIGKILL to all processes.

TODO:
 - Consider maintaining a per-pid namespace tasklist. Use that list
   to terminate processes in the namespace more efficiently. Such a
   tasklist may also be useful to freeze or checkpoint an application.

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
---
 include/linux/pid.h | 1 +
 include/linux/pid_namespace.h | 1 +
 kernel/exit.c | 5 +++--
 kernel/fork.c | 19 ++++++++++++++++++
 kernel/pid.c | 42 ++
 5 files changed, 66 insertions(+), 2 deletions(-)

Index: lx26-22-rc4-mm2/include/linux/pid.h
==========
--- lx26-22-rc4-mm2.orig/include/linux/pid.h 2007-06-15 18:52:19.000000000 -0700
+++ lx26-22-rc4-mm2/include/linux/pid.h 2007-06-15 18:52:19.000000000 -0700
@@ -123,6 +123,7 @@ extern struct pid *dup_struct_pid(enum c
 extern void FASTCALL(free_pid(struct pid *pid));

 extern pid_t pid_to_nr(struct pid *pid);
+extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);

 #define do_each_pid_task(pid, type, task)   \
  do {   \
Index: lx26-22-rc4-mm2/include/linux/pid_namespace.h
==========
--- lx26-22-rc4-mm2.orig/include/linux/pid_namespace.h 2007-06-15 18:52:19.000000000 -0700
+++ lx26-22-rc4-mm2/include/linux/pid_namespace.h 2007-06-15 18:52:19.000000000 -0700
@@ -32,6 +32,7 @@ struct pid_namespace {

```
  struct pidmap pidmap[PIDMAP_ENTRIES];
  int last_pid;
  struct task_struct *child_reaper;
+ atomic_t terminating;
 };

 extern struct pid_namespace init_pid_ns;
Index: lx26-22-rc4-mm2/kernel/exit.c
===================================================================
--- lx26-22-rc4-mm2.orig/kernel/exit.c 2007-06-15 18:52:19.000000000 -0700
+++ lx26-22-rc4-mm2/kernel/exit.c 2007-06-15 18:52:19.000000000 -0700
@@ -888,9 +888,10 @@ fastcall NORET_TYPE void do_exit(long co
    * the reaper of the active pid namespace.
    */
  if (unlikely(tsk == pid_ns->child_reaper)) {
-  if (pid_ns != &init_pid_ns)
+  if (pid_ns != &init_pid_ns) {
+   zap_pid_ns_processes(pid_ns);
    pid_ns->child_reaper = init_pid_ns.child_reaper;
-  else
+  } else
    panic("Attempted to kill init!");
  }

Index: lx26-22-rc4-mm2/kernel/fork.c
===================================================================
--- lx26-22-rc4-mm2.orig/kernel/fork.c 2007-06-15 18:52:19.000000000 -0700
+++ lx26-22-rc4-mm2/kernel/fork.c 2007-06-15 18:52:19.000000000 -0700
@@ -1407,6 +1407,7 @@ long do_fork(unsigned long clone_flags,
   */
  if (!IS_ERR(p)) {
   struct completion vfork;
+  struct pid_namespace *pid_ns;

   nr = pid_to_nr(task_pid(p));

@@ -1415,6 +1416,24 @@ long do_fork(unsigned long clone_flags,
    init_completion(&vfork);
   }

+  /*
+   * If our pid namespace was asked to terminate while we were
+   * being created, exit immediately.
+   *
+   * We need the check here for the case where the termination
+   * request was received after a pid_t was allocated in
+   * alloc_pid_nr() and before the process was fully created
+   * ("old enough") to receive the signal sent from do_exit().
```

```
+   *
+   * TODO: Implement a task list per pid namespace to simplify
+   * terminating the pid namespace and remove this check.
+   */
+  pid_ns = task_active_pid_ns(p);
+  if (atomic_read(&pid_ns->terminating)) {
+   sigaddset(&p->pending.signal, SIGKILL);
+   set_tsk_thread_flag(p, TIF_SIGPENDING);
+  }
+
   if ((p->ptrace & PT_PTRACED) || (clone_flags & CLONE_STOPPED)) {
    /*
     * We'll start up with an immediate SIGSTOP.
Index: lx26-22-rc4-mm2/kernel/pid.c
===================================================================
--- lx26-22-rc4-mm2.orig/kernel/pid.c 2007-06-15 18:52:19.000000000 -0700
+++ lx26-22-rc4-mm2/kernel/pid.c 2007-06-15 18:52:19.000000000 -0700
@@ -145,6 +145,9 @@ static int alloc_pidmap(struct pid_names
  int i, offset, max_scan, pid, last = pid_ns->last_pid;
  struct pidmap *map;

+ if (atomic_read(&pid_ns->terminating))
+  return -1;
+
  pid = last + 1;
  if (pid >= pid_max)
   pid = RESERVED_PIDS;
@@ -314,6 +317,39 @@ static struct pid_namespace *alloc_pid_n
  return ns;
 }

+/*
+ * When child reaper of the pid namespace @pid_ns is itself terminating,
+ * we need to terminate all processes in the pid namespace since /proc
+ * has a reference to the child reaper of the pid namespace.
+ *
+ * Send SIGKILL to all processes in the pid namespace. Set the 'terminating'
+ * flag in pid_ns to prevent any new processes from getting created in the
+ * pid namespace.
+ *
+ * Note that we will also be terminating all our child pid namespaces
+ * (if any) since we send SIGKILL their reapers as well.
+ *
+ * TODO: It maybe more efficient to maintain a list of tasks in the
+ *       pid namespace and walk that list.
+ */
+void zap_pid_ns_processes(struct pid_namespace *pid_ns)
+{
```

```
+ int nr;
+
+ atomic_set(&pid_ns->terminating, 1);
+
+ /*
+  * We know pid == 1 is terminating. Find remaining pid_ts
+  * in the namespace and terminate them.
+  */
+ nr = next_pidmap(pid_ns, 1);
+ while (nr > 0) {
+  kill_proc(nr, SIGKILL, 1);
+  nr = next_pidmap(pid_ns, nr);
+ }
+ return;
+}
+
 #else

 static struct pid_namespace *alloc_pid_ns(void)
@@ -321,6 +357,12 @@ static struct pid_namespace *alloc_pid_n
  WARN_ON_ONCE(1);
  return ERR_PTR(-EINVAL);
 }
+
+void zap_pid_ns_processes(struct pid_namespace *pid_ns)
+{
+ /* Nothing to do when we don't have multiple pid namespaces */
+ return;
+}
 #endif  /*CONFIG_PID_NS*/

 static inline struct kmem_cache *select_pid_cache(int num_upids)

--

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

Subject: Re: [PATCH 12/17] Pid-NS(V3) Terminate processes in a ns when reaper is exiting.
Posted by Pavel Emelianov on Mon, 18 Jun 2007 09:02:48 GMT
View Forum Message <> Reply to Message

sukadev@us.ibm.com wrote:
> Subject: [PATCH 12/17] Pid-NS(V3) Terminate processes in a ns when reaper is exiting.
>

> From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
>
> This should actually be considered a part of the previous patch which
> enables cloning of pid namespace. Its been separated out for easier
> review.
>
> Terminate all processes in a namespace when the reaper of the namespace
> is exiting. We do this by walking the pidmap of the namespace and sending
> SIGKILL to all processes.
>
> TODO:
> - Consider maintaining a per-pid namespace tasklist. Use that list
>   to terminate processes in the namespace more efficiently. Such a
>   tasklist may also be useful to freeze or checkpoint an application.

Pid namespace of its own can happily live without this.
Why is this needed _for_the_namespace_?

> Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> ---
> include/linux/pid.h | 1 +
> include/linux/pid_namespace.h | 1 +
> kernel/exit.c | 5 +++--
> kernel/fork.c | 19 ++++++++++++++++++
> kernel/pid.c | 42 ++
> 5 files changed, 66 insertions(+), 2 deletions(-)
>

[snip]

> Index: lx26-22-rc4-mm2/kernel/pid.c
> ===
> --- lx26-22-rc4-mm2.orig/kernel/pid.c 2007-06-15 18:52:19.000000000 -0700
> +++ lx26-22-rc4-mm2/kernel/pid.c 2007-06-15 18:52:19.000000000 -0700
> @@ -145,6 +145,9 @@ static int alloc_pidmap(struct pid_names
>   int i, offset, max_scan, pid, last = pid_ns->last_pid;
>   struct pidmap *map;
>
> + if (atomic_read(&pid_ns->terminating))
> + return -1;
> +
>   pid = last + 1;
>   if (pid >= pid_max)
>   pid = RESERVED_PIDS;
> @@ -314,6 +317,39 @@ static struct pid_namespace *alloc_pid_n
>   return ns;
> }
>

> +/*
> + * When child reaper of the pid namespace @pid_ns is itself terminating,
> + * we need to terminate all processes in the pid namespace since /proc
> + * has a reference to the child reaper of the pid namespace.
> + *
> + * Send SIGKILL to all processes in the pid namespace. Set the 'terminating'
> + * flag in pid_ns to prevent any new processes from getting created in the
> + * pid namespace.
> + *
> + * Note that we will also be terminating all our child pid namespaces
> + * (if any) since we send SIGKILL their reapers as well.
> + *
> + * TODO: It maybe more efficient to maintain a list of tasks in the
> + *   pid namespace and walk that list.
> + */
> +void zap_pid_ns_processes(struct pid_namespace *pid_ns)
> +{
> + int nr;
> +
> + atomic_set(&pid_ns->terminating, 1);
> +
> + /*
> + * We know pid == 1 is terminating. Find remaining pid_ts
> + * in the namespace and terminate them.
> + */
> + nr = next_pidmap(pid_ns, 1);
> + while (nr > 0) {
> + kill_proc(nr, SIGKILL, 1);
> + nr = next_pidmap(pid_ns, nr);

This looks like a proc_flush_task():
[cite]
 * NOTE: This routine is just an optimization so it does not guarantee
 *   that no ... (processes) will exist at process exit time it
 *   just makes it very unlikely that any will persist.
[/cite]

If we really want the namespace to be terminated right when its leader (init)
exits we have to do_wait() for each killed task and resend the signals.

> + }
> + return;
> +}
> +
> #else
>
> static struct pid_namespace *alloc_pid_ns(void)
> @@ -321,6 +357,12 @@ static struct pid_namespace *alloc_pid_n

> WARN_ON_ONCE(1);
> return ERR_PTR(-EINVAL);
> }
> +
> +void zap_pid_ns_processes(struct pid_namespace *pid_ns)
> +{
> + /* Nothing to do when we don't have multiple pid namespaces */
> + return;
> +}
> #endif /*CONFIG_PID_NS*/
>
> static inline struct kmem_cache *select_pid_cache(int num_upids)
>

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