Subject: [PATCH 1/2] signals: kill(-1) should only signal processes in the same namespace
Posted by Daniel Hokka Zakrisso on Thu, 17 Jul 2008 14:55:39 GMT
View Forum Message <> Reply to Message

While moving Linux-VServer to using pid namespaces, I noticed that
kill(-1) from inside a pid namespace is currently signalling every
process in the entire system, including processes that are otherwise
unreachable from the current process.

This patch fixes it by making sure that only processes which are in
the same pid namespace as current get signalled.

Signed-off-by: Daniel Hokka Zakrisson <daniel@hozac.com>

```
diff --git a/include/linux/pid_namespace.h b/include/linux/pid_namespace.h
index caff528..4cf41bd 100644
--- a/include/linux/pid_namespace.h
+++ b/include/linux/pid_namespace.h
@@ -40,6 +40,8 @@ static inline struct pid_namespace *get_pid_ns(struct
pid_namespace *ns)
  extern struct pid_namespace *copy_pid_ns(unsigned long flags, struct
pid_namespace *ns);
  extern void free_pid_ns(struct kref *kref);
  extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);
+extern int task_in_pid_ns(struct task_struct *tsk,
+      struct pid_namespace *pid_ns);

  static inline void put_pid_ns(struct pid_namespace *ns)
  {
@@ -72,6 +74,12 @@ static inline void zap_pid_ns_processes(struct
pid_namespace *ns)
  {
   BUG();
  }
+
+static inline int task_in_pid_ns(struct task_struct *tsk,
+       struct pid_namespace *ns)
+{
+ return 1;
+}
  #endif /* CONFIG_PID_NS */

  static inline struct pid_namespace *task_active_pid_ns(struct
task_struct *tsk)
diff --git a/kernel/pid_namespace.c b/kernel/pid_namespace.c
index 98702b4..3e71011 100644
--- a/kernel/pid_namespace.c
```

```
+++ b/kernel/pid_namespace.c
@@ -188,6 +188,26 @@ void zap_pid_ns_processes(struct pid_namespace *pid_ns)
   return;
  }

+/*
+ * Checks whether tsk has a pid in the pid namespace ns.
+ * Must be called with tasklist_lock read-locked or under rcu_read_lock()
+ */
+int task_in_pid_ns(struct task_struct *tsk, struct pid_namespace *ns)
+{
+ struct pid *pid = task_pid(tsk);
+
+ if (!pid)
+  return 0;
+
+ if (pid->level < ns->level)
+  return 0;
+
+ if (pid->numbers[ns->level].ns != ns)
+  return 0;
+
+ return 1;
+}
+
  static __init int pid_namespaces_init(void)
  {
   pid_ns_cache = KMEM_CACHE(pid_namespace, SLAB_PANIC);
diff --git a/kernel/signal.c b/kernel/signal.c
index 6c0958e..93713a5 100644
--- a/kernel/signal.c
+++ b/kernel/signal.c
@@ -1145,7 +1145,8 @@ static int kill_something_info(int sig, struct
siginfo *info, int pid)
    struct task_struct * p;

    for_each_process(p) {
-   if (p->pid > 1 && !same_thread_group(p, current)) {
+   if (p->pid > 1 && !same_thread_group(p, current) &&
+       task_in_pid_ns(p, current->nsproxy->pid_ns)) {
      int err = group_send_sig_info(sig, info, p);
      ++count;
      if (err != -EPERM)
--
1.5.5.1
```

_____

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

Subject: Re: [PATCH 1/2] signals: kill(-1) should only signal processes in the same namespace
Posted by Pavel Emelianov on Thu, 17 Jul 2008 15:01:47 GMT
View Forum Message <> Reply to Message

Daniel Hokka Zakrisson wrote:
> While moving Linux-VServer to using pid namespaces, I noticed that
> kill(-1) from inside a pid namespace is currently signalling every
> process in the entire system, including processes that are otherwise
> unreachable from the current process.

This is not a "news" actually, buy anyway - thanks :)

> This patch fixes it by making sure that only processes which are in
> the same pid namespace as current get signalled.

This is to be done, indeed, but I do not like the proposed implementation,
since you have to walk all the tasks in the system (under tasklist_lock,
by the way) to search for a couple of interesting ones. Better look at how
zap_pid_ns_processes works (by the way - I saw some patch doing so some
time ago).

> Signed-off-by: Daniel Hokka Zakrisson <daniel@hozac.com>
>
> diff --git a/include/linux/pid_namespace.h b/include/linux/pid_namespace.h
> index caff528..4cf41bd 100644
> --- a/include/linux/pid_namespace.h
> +++ b/include/linux/pid_namespace.h
> @@ -40,6 +40,8 @@ static inline struct pid_namespace *get_pid_ns(struct
> pid_namespace *ns)
> extern struct pid_namespace *copy_pid_ns(unsigned long flags, struct
> pid_namespace *ns);
> extern void free_pid_ns(struct kref *kref);
> extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);
> +extern int task_in_pid_ns(struct task_struct *tsk,
> +   struct pid_namespace *pid_ns);
>
> static inline void put_pid_ns(struct pid_namespace *ns)
> {
> @@ -72,6 +74,12 @@ static inline void zap_pid_ns_processes(struct
> pid_namespace *ns)
> {
>   BUG();
> }
> +
>

```
> +static inline int task_in_pid_ns(struct task_struct *tsk,
> +      struct pid_namespace *ns)
> +{
> + return 1;
> +}
>   #endif /* CONFIG_PID_NS */
>
>   static inline struct pid_namespace *task_active_pid_ns(struct
> task_struct *tsk)
> diff --git a/kernel/pid_namespace.c b/kernel/pid_namespace.c
> index 98702b4..3e71011 100644
> --- a/kernel/pid_namespace.c
> +++ b/kernel/pid_namespace.c
> @@ -188,6 +188,26 @@ void zap_pid_ns_processes(struct pid_namespace *pid_ns)
>    return;
>   }
>
> +/*
> + * Checks whether tsk has a pid in the pid namespace ns.
> + * Must be called with tasklist_lock read-locked or under rcu_read_lock()
> + */
> +int task_in_pid_ns(struct task_struct *tsk, struct pid_namespace *ns)
> +{
> + struct pid *pid = task_pid(tsk);
> +
> + if (!pid)
> +  return 0;
> +
> + if (pid->level < ns->level)
> +  return 0;
> +
> + if (pid->numbers[ns->level].ns != ns)
> +  return 0;
> +
> + return 1;
> +}
> +
>   static __init int pid_namespaces_init(void)
>   {
>    pid_ns_cachep = KMEM_CACHE(pid_namespace, SLAB_PANIC);
> diff --git a/kernel/signal.c b/kernel/signal.c
> index 6c0958e..93713a5 100644
> --- a/kernel/signal.c
> +++ b/kernel/signal.c
> @@ -1145,7 +1145,8 @@ static int kill_something_info(int sig, struct
> siginfo *info, int pid)
>     struct task_struct * p;
>
```

```
>     for_each_process(p) {
> -   if (p->pid > 1 && !same_thread_group(p, current)) {
> +   if (p->pid > 1 && !same_thread_group(p, current) &&
> +       task_in_pid_ns(p, current->nsproxy->pid_ns)) {
>       int err = group_send_sig_info(sig, info, p);
>       ++count;
>       if (err != -EPERM)
```

Subject: Re: [PATCH 1/2] signals: kill(-1) should only signal processes in the same namespace
Posted by Daniel Hokka Zakrisso on Thu, 17 Jul 2008 15:24:40 GMT
View Forum Message <> Reply to Message

Pavel Emelyanov wrote:
> Daniel Hokka Zakrisson wrote:
>> While moving Linux-VServer to using pid namespaces, I noticed that
>> kill(-1) from inside a pid namespace is currently signalling every
>> process in the entire system, including processes that are otherwise
>> unreachable from the current process.
>
> This is not a "news" actually, buy anyway - thanks :)

And yet nobody's fixed it... Kind of a critical thing, if you actually want to use them, since most distribution's rc-scripts do a kill(-1, SIGTERM), followed by kill(-1, SIGKILL) when halting (which, needless to say, would be very bad).

>> This patch fixes it by making sure that only processes which are in
>> the same pid namespace as current get signalled.
>
> This is to be done, indeed, but I do not like the proposed implementation,
> since you have to walk all the tasks in the system (under tasklist_lock,
> by the way) to search for a couple of interesting ones. Better look at how
> zap_pid_ns_processes works (by the way - I saw some patch doing so some
> time ago).

The way zap_pid_ns_processes does it is worse, since it signals every thread in the namespace rather than every thread group. So either we walk the global tasklist, or we create a per-namespace one. Is that what we want?

>> Signed-off-by: Daniel Hokka Zakrisson <daniel@hozac.com>

```
>>
>> diff --git a/include/linux/pid_namespace.h
>> b/include/linux/pid_namespace.h
>> index caff528..4cf41bd 100644
>> --- a/include/linux/pid_namespace.h
>> +++ b/include/linux/pid_namespace.h
>> @@ -40,6 +40,8 @@ static inline struct pid_namespace *get_pid_ns(struct
>> pid_namespace *ns)
>>   extern struct pid_namespace *copy_pid_ns(unsigned long flags, struct
>> pid_namespace *ns);
>>   extern void free_pid_ns(struct kref *kref);
>>   extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);
>> +extern int task_in_pid_ns(struct task_struct *tsk,
>> +      struct pid_namespace *pid_ns);
>>
>>   static inline void put_pid_ns(struct pid_namespace *ns)
>>   {
>> @@ -72,6 +74,12 @@ static inline void zap_pid_ns_processes(struct
>> pid_namespace *ns)
>>   {
>>    BUG();
>>   }
>> +
>> +static inline int task_in_pid_ns(struct task_struct *tsk,
>> +       struct pid_namespace *ns)
>> +{
>> + return 1;
>> +}
>>   #endif /* CONFIG_PID_NS */
>>
>>   static inline struct pid_namespace *task_active_pid_ns(struct
>> task_struct *tsk)
>> diff --git a/kernel/pid_namespace.c b/kernel/pid_namespace.c
>> index 98702b4..3e71011 100644
>> --- a/kernel/pid_namespace.c
>> +++ b/kernel/pid_namespace.c
>> @@ -188,6 +188,26 @@ void zap_pid_ns_processes(struct pid_namespace
>> *pid_ns)
>>    return;
>>   }
>>
>> +/*
>> + * Checks whether tsk has a pid in the pid namespace ns.
>> + * Must be called with tasklist_lock read-locked or under
>> rcu_read_lock()
>> + */
>> +int task_in_pid_ns(struct task_struct *tsk, struct pid_namespace *ns)
>> +{
```

```
>> + struct pid *pid = task_pid(tsk);
>> +
>> + if (!pid)
>> +  return 0;
>> +
>> + if (pid->level < ns->level)
>> +  return 0;
>> +
>> + if (pid->numbers[ns->level].ns != ns)
>> +  return 0;
>> +
>> + return 1;
>> +}
>> +
>>   static __init int pid_namespaces_init(void)
>>   {
>>    pid_ns_cachep = KMEM_CACHE(pid_namespace, SLAB_PANIC);
>> diff --git a/kernel/signal.c b/kernel/signal.c
>> index 6c0958e..93713a5 100644
>> --- a/kernel/signal.c
>> +++ b/kernel/signal.c
>> @@ -1145,7 +1145,8 @@ static int kill_something_info(int sig, struct
>> siginfo *info, int pid)
>>     struct task_struct * p;
>>
>>     for_each_process(p) {
>> -   if (p->pid > 1 && !same_thread_group(p, current)) {
>> +   if (p->pid > 1 && !same_thread_group(p, current) &&
>> +       task_in_pid_ns(p, current->nsproxy->pid_ns)) {
>>       int err = group_send_sig_info(sig, info, p);
>>       ++count;
>>       if (err != -EPERM)
```

--
Daniel Hokka Zakrisson

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

---

Subject: Re: [PATCH 1/2] signals: kill(-1) should only signal processes in the same namespace

Posted by Pavel Emelianov on Thu, 17 Jul 2008 15:54:22 GMT

View Forum Message <> Reply to Message

Daniel Hokka Zakrisson wrote:
> Pavel Emelyanov wrote:

>> Daniel Hokka Zakrisson wrote:
>>> While moving Linux-VServer to using pid namespaces, I noticed that
>>> kill(-1) from inside a pid namespace is currently signalling every
>>> process in the entire system, including processes that are otherwise
>>> unreachable from the current process.
>> This is not a "news" actually, buy anyway - thanks :)
>
> And yet nobody's fixed it... Kind of a critical thing, if you actually
> want to use them, since most distribution's rc-scripts do a kill(-1,
> SIGTERM), followed by kill(-1, SIGKILL) when halting (which, needless to
> say, would be very bad).
>
>>> This patch fixes it by making sure that only processes which are in
>>> the same pid namespace as current get signalled.
>> This is to be done, indeed, but I do not like the proposed implementation,
>> since you have to walk all the tasks in the system (under tasklist_lock,
>> by the way) to search for a couple of interesting ones. Better look at how
>> zap_pid_ns_processes works (by the way - I saw some patch doing so some
>> time ago).
>
> The way zap_pid_ns_processes does it is worse, since it signals every
> thread in the namespace rather than every thread group. So either we walk

It's questionable whether there are more "threads in a pid namespace" than
"processes in a system".

E.g. on my notebook there are ~110 processes and ~150 threads. So having
this setup launched in 10 containers you'll have to walk 1100 tasks, while
zap_pid_ns_processes only 150 ;)

Some real-life example with containers: on one of our servers with 10
containers serving as git repo, bulding system and some other stuff there
are ~200 process totally and ~20 threads in each container. See?

I tend to believe that walking threads in a container is cheaper then
walking processes in a system...

> the global tasklist, or we create a per-namespace one. Is that what we
> want?

We want to kill all tasks in current pid namespace. There are variants of
how to do this. You particular implementation of handling this case seems
poor to me for the reasons described above.

>>> Signed-off-by: Daniel Hokka Zakrisson <daniel@hozac.com>
>>>
>>> diff --git a/include/linux/pid_namespace.h
>>> b/include/linux/pid_namespace.h

```
>>> index caff528..4cf41bd 100644
>>> --- a/include/linux/pid_namespace.h
>>> +++ b/include/linux/pid_namespace.h
>>> @@ -40,6 +40,8 @@ static inline struct pid_namespace *get_pid_ns(struct
>>> pid_namespace *ns)
>>>   extern struct pid_namespace *copy_pid_ns(unsigned long flags, struct
>>> pid_namespace *ns);
>>>   extern void free_pid_ns(struct kref *kref);
>>>   extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);
>>> +extern int task_in_pid_ns(struct task_struct *tsk,
>>> +      struct pid_namespace *pid_ns);
>>>
>>>   static inline void put_pid_ns(struct pid_namespace *ns)
>>>   {
>>> @@ -72,6 +74,12 @@ static inline void zap_pid_ns_processes(struct
>>> pid_namespace *ns)
>>>   {
>>>    BUG();
>>>   }
>>> +
>>> +static inline int task_in_pid_ns(struct task_struct *tsk,
>>> +      struct pid_namespace *ns)
>>> +{
>>> + return 1;
>>> +}
>>>   #endif /* CONFIG_PID_NS */
>>>
>>>   static inline struct pid_namespace *task_active_pid_ns(struct
>>> task_struct *tsk)
>>> diff --git a/kernel/pid_namespace.c b/kernel/pid_namespace.c
>>> index 98702b4..3e71011 100644
>>> --- a/kernel/pid_namespace.c
>>> +++ b/kernel/pid_namespace.c
>>> @@ -188,6 +188,26 @@ void zap_pid_ns_processes(struct pid_namespace
>>> *pid_ns)
>>>    return;
>>>   }
>>>
>>> +/*
>>> + * Checks whether tsk has a pid in the pid namespace ns.
>>> + * Must be called with tasklist_lock read-locked or under
>>> rcu_read_lock()
>>> + */
>>> +int task_in_pid_ns(struct task_struct *tsk, struct pid_namespace *ns)
>>> +{
>>> + struct pid *pid = task_pid(tsk);
>>> +
>>> + if (!pid)
```

```
>>> +  return 0;
>>> +
>>> + if (pid->level < ns->level)
>>> +  return 0;
>>> +
>>> + if (pid->numbers[ns->level].ns != ns)
>>> +  return 0;
>>> +
>>> + return 1;
>>> +}
>>> +
>>>   static __init int pid_namespaces_init(void)
>>>   {
>>>    pid_ns_cachep = KMEM_CACHE(pid_namespace, SLAB_PANIC);
>>> diff --git a/kernel/signal.c b/kernel/signal.c
>>> index 6c0958e..93713a5 100644
>>> --- a/kernel/signal.c
>>> +++ b/kernel/signal.c
>>> @@ -1145,7 +1145,8 @@ static int kill_something_info(int sig, struct
>>> siginfo *info, int pid)
>>>     struct task_struct * p;
>>>
>>>     for_each_process(p) {
>>> -   if (p->pid > 1 && !same_thread_group(p, current)) {
>>> +   if (p->pid > 1 && !same_thread_group(p, current) &&
>>> +       task_in_pid_ns(p, current->nsproxy->pid_ns)) {
>>>       int err = group_send_sig_info(sig, info, p);
>>>       ++count;
>>>       if (err != -EPERM)
>
```

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

---

Subject: Re: [PATCH 1/2] signals: kill(-1) should only signal processes in the same namespace
Posted by Oleg Nesterov on Thu, 17 Jul 2008 17:31:10 GMT
View Forum Message <> Reply to Message

On 07/17, Pavel Emelyanov wrote:
>
> Daniel Hokka Zakrisson wrote:
> >
> > The way zap_pid_ns_processes does it is worse, since it signals every
> > thread in the namespace rather than every thread group. So either we walk

>
> It's questionable whether there are more "threads in a pid namespace" than
> "processes in a system".
>
> E.g. on my notebook there are ~110 processes and ~150 threads. So having
> this setup launched in 10 containers you'll have to walk 1100 tasks, while
> zap_pid_ns_processes only 150 ;)
>
> Some real-life example with containers: on one of our servers with 10
> containers serving as git repo, bulding system and some other stuff there
> are ~200 process totally and ~20 threads in each container. See?
>
> I tend to believe that walking threads in a container is cheaper then
> walking processes in a system...

kill_something_info() can't walk threads, think about the realtime signals.

Anyway, I think we should change kill_something_info(-1) to use rcu_read_lock()
instead of tasklist.

Oleg.

Subject: Re: [PATCH 1/2] signals: kill(-1) should only signal processes in the same namespace
Posted by ebiederm on Thu, 17 Jul 2008 17:45:13 GMT
View Forum Message <> Reply to Message

"Daniel Hokka Zakrisson" <daniel@hozac.com> writes:

> Pavel Emelyanov wrote:
>> Daniel Hokka Zakrisson wrote:
>>> While moving Linux-VServer to using pid namespaces, I noticed that
>>> kill(-1) from inside a pid namespace is currently signalling every
>>> process in the entire system, including processes that are otherwise
>>> unreachable from the current process.
>>
>> This is not a "news" actually, buy anyway - thanks :)
>
> And yet nobody's fixed it... Kind of a critical thing, if you actually
> want to use them, since most distribution's rc-scripts do a kill(-1,
> SIGTERM), followed by kill(-1, SIGKILL) when halting (which, needless to
> say, would be very bad).

>
>>> This patch fixes it by making sure that only processes which are in
>>> the same pid namespace as current get signalled.
>>
>> This is to be done, indeed, but I do not like the proposed implementation,
>> since you have to walk all the tasks in the system (under tasklist_lock,
>> by the way) to search for a couple of interesting ones. Better look at how
>> zap_pid_ns_processes works (by the way - I saw some patch doing so some
>> time ago).
>
> The way zap_pid_ns_processes does it is worse, since it signals every
> thread in the namespace rather than every thread group. So either we walk
> the global tasklist, or we create a per-namespace one. Is that what we
> want?

Can you please introduce kill_pidns_info and have both
kill_something_info and zap_pid_ns_processes call this common
function?

We want to walk the set of all pids in a pid namespace. /proc does
this and it is the recommended idiom. If walking all of the pids in a
pid namespace is not fast enough we can accelerate that.

You are correct signalling every thread in a namespace is worse, in
fact it is semantically incorrect. zap_pid_ns_processes gets away
with it because it is sending SIGKILL. Therefore kill_pidns_info
should skip sending a signal to every task that is not the
thread_group_leader.

We need to hold the tasklist_lock to prevent new processes from
joining the list of all processes. Otherwise we could run the code
under the rcu_read_lock.

Eric

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

---

Subject: Re: [PATCH 1/2] signals: kill(-1) should only signal processes in the same namespace
Posted by ebiederm on Thu, 17 Jul 2008 17:50:56 GMT
View Forum Message <> Reply to Message

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

> kill_something_info() can't walk threads, think about the realtime signals.

walking threads is fine delivering signals to non thread group leaders is
a problem.

> Anyway, I think we should change kill_something_info(-1) to use rcu_read_lock()
> instead of tasklist.

Being dense I think the locking implications of a correct implementation are more
then we are ready to deal with to fix this bug. Although I remember discussing
it and seeing something reasonable.

Eric

Subject: Re: [PATCH 1/2] signals: kill(-1) should only signal processes in the same namespace
Posted by Sukadev Bhattiprolu on Thu, 17 Jul 2008 18:13:13 GMT
View Forum Message <> Reply to Message

Daniel Hokka Zakrisson [daniel@hozac.com] wrote:
| While moving Linux-VServer to using pid namespaces, I noticed that
| kill(-1) from inside a pid namespace is currently signalling every
| process in the entire system, including processes that are otherwise
| unreachable from the current process.
|
| This patch fixes it by making sure that only processes which are in
| the same pid namespace as current get signalled.
|
| Signed-off-by: Daniel Hokka Zakrisson <daniel@hozac.com>
|
| diff --git a/include/linux/pid_namespace.h b/include/linux/pid_namespace.h
| index caff528..4cf41bd 100644
| --- a/include/linux/pid_namespace.h
| +++ b/include/linux/pid_namespace.h
| @@ -40,6 +40,8 @@ static inline struct pid_namespace *get_pid_ns(struct
| pid_namespace *ns)
| extern struct pid_namespace *copy_pid_ns(unsigned long flags, struct
| pid_namespace *ns);
| extern void free_pid_ns(struct kref *kref);
| extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);
| +extern int task_in_pid_ns(struct task_struct *tsk,
| +   struct pid_namespace *pid_ns);
|

```
|   static inline void put_pid_ns(struct pid_namespace *ns)
|   {
| @@ -72,6 +74,12 @@ static inline void zap_pid_ns_processes(struct
| pid_namespace *ns)
|   {
|    BUG();
|   }
| +
| +static inline int task_in_pid_ns(struct task_struct *tsk,
| +      struct pid_namespace *ns)
| +{
| + return 1;
| +}
|   #endif /* CONFIG_PID_NS */
|
|   static inline struct pid_namespace *task_active_pid_ns(struct
| task_struct *tsk)
| diff --git a/kernel/pid_namespace.c b/kernel/pid_namespace.c
| index 98702b4..3e71011 100644
| --- a/kernel/pid_namespace.c
| +++ b/kernel/pid_namespace.c
| @@ -188,6 +188,26 @@ void zap_pid_ns_processes(struct pid_namespace *pid_ns)
|    return;
|   }
|
| +/*
| + * Checks whether tsk has a pid in the pid namespace ns.
| + * Must be called with tasklist_lock read-locked or under rcu_read_lock()
| + */
| +int task_in_pid_ns(struct task_struct *tsk, struct pid_namespace *ns)
| +{
| + struct pid *pid = task_pid(tsk);
| +
| + if (!pid)
| +  return 0;
| +
| + if (pid->level < ns->level)
| +  return 0;
```

ns can be NULL if tsk is exiting.

Like Pavel said, we had couple of attempts to fix the larger problem of signal semantics in containers but did not have a consensus on handling blocked/unhandled signals to container-init.

It would still be good to fix this "kill -1" problem.

Eric had a slightly optimized interface, 'pid_in_pid_ns()' in following

patchset. Maybe we could use that ?

https://lists.linux-foundation.org/pipermail/containers/2007-December/009174.html

```
| +
| + if (pid->numbers[ns->level].ns != ns)
| +  return 0;
| +
| + return 1;
| +}
| +
|   static __init int pid_namespaces_init(void)
|   {
|    pid_ns_cachep = KMEM_CACHE(pid_namespace, SLAB_PANIC);
| diff --git a/kernel/signal.c b/kernel/signal.c
| index 6c0958e..93713a5 100644
| --- a/kernel/signal.c
| +++ b/kernel/signal.c
| @@ -1145,7 +1145,8 @@ static int kill_something_info(int sig, struct
| siginfo *info, int pid)
|     struct task_struct * p;
|
|     for_each_process(p) {
| -   if (p->pid > 1 && !same_thread_group(p, current)) {
| +   if (p->pid > 1 && !same_thread_group(p, current) &&
| +       task_in_pid_ns(p, current->nsproxy->pid_ns)) {
|       int err = group_send_sig_info(sig, info, p);
|       ++count;
|       if (err != -EPERM)
| --
| 1.5.5.1
| _____________________________________________________
| 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: [PATCH 1/2] signals: kill(-1) should only signal processes in the same namespace
Posted by Daniel Hokka Zakrisso on Thu, 17 Jul 2008 18:39:14 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:

> "Daniel Hokka Zakrisson" <daniel@hozac.com> writes:
>
>> Pavel Emelyanov wrote:
>>> Daniel Hokka Zakrisson wrote:
>>>> While moving Linux-VServer to using pid namespaces, I noticed that
>>>> kill(-1) from inside a pid namespace is currently signalling every
>>>> process in the entire system, including processes that are otherwise
>>>> unreachable from the current process.
>>>
>>> This is not a "news" actually, buy anyway - thanks :)
>>
>> And yet nobody's fixed it... Kind of a critical thing, if you actually
>> want to use them, since most distribution's rc-scripts do a kill(-1,
>> SIGTERM), followed by kill(-1, SIGKILL) when halting (which, needless to
>> say, would be very bad).
>>
>>>> This patch fixes it by making sure that only processes which are in
>>>> the same pid namespace as current get signalled.
>>>
>>> This is to be done, indeed, but I do not like the proposed
>>> implementation,
>>> since you have to walk all the tasks in the system (under
>>> tasklist_lock,
>>> by the way) to search for a couple of interesting ones. Better look at
>>> how
>>> zap_pid_ns_processes works (by the way - I saw some patch doing so some
>>> time ago).
>>
>> The way zap_pid_ns_processes does it is worse, since it signals every
>> thread in the namespace rather than every thread group. So either we
>> walk
>> the global tasklist, or we create a per-namespace one. Is that what we
>> want?
>
> Can you please introduce kill_pidns_info and have both
> kill_something_info and zap_pid_ns_processes call this common
> function?

Looks like you've already done that. :-) (Referring to Sukadev's email.)
Is there any reason we don't just merge that patch?

> We want to walk the set of all pids in a pid namespace. /proc does
> this and it is the recommended idiom. If walking all of the pids in a
> pid namespace is not fast enough we can accelerate that.
>
> You are correct signalling every thread in a namespace is worse, in
> fact it is semantically incorrect. zap_pid_ns_processes gets away
> with it because it is sending SIGKILL. Therefore kill_pidns_info

> should skip sending a signal to every task that is not the
> thread_group_leader.
>
> We need to hold the tasklist_lock to prevent new processes from
> joining the list of all processes. Otherwise we could run the code
> under the rcu_read_lock.
>
> Eric

--
Daniel Hokka Zakrisson

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

---

Subject: Re: [PATCH 1/2] signals: kill(-1) should only signal processes in the same namespace
Posted by Daniel Hokka Zakrisso on Thu, 17 Jul 2008 18:44:32 GMT
View Forum Message <> Reply to Message

sukadev@us.ibm.com wrote:
> Daniel Hokka Zakrisson [daniel@hozac.com] wrote:
> | While moving Linux-VServer to using pid namespaces, I noticed that
> | kill(-1) from inside a pid namespace is currently signalling every
> | process in the entire system, including processes that are otherwise
> | unreachable from the current process.
> |
> | This patch fixes it by making sure that only processes which are in
> | the same pid namespace as current get signalled.
> |
> | Signed-off-by: Daniel Hokka Zakrisson <daniel@hozac.com>
> |
> | diff --git a/include/linux/pid_namespace.h
> b/include/linux/pid_namespace.h
> | index caff528..4cf41bd 100644
> | --- a/include/linux/pid_namespace.h
> | +++ b/include/linux/pid_namespace.h
> | @@ -40,6 +40,8 @@ static inline struct pid_namespace *get_pid_ns(struct
> | pid_namespace *ns)
> | extern struct pid_namespace *copy_pid_ns(unsigned long flags, struct
> | pid_namespace *ns);
> | extern void free_pid_ns(struct kref *kref);
> | extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);
> | +extern int task_in_pid_ns(struct task_struct *tsk,
> | +   struct pid_namespace *pid_ns);
> |

> | static inline void put_pid_ns(struct pid_namespace *ns)
> | {
> | @@ -72,6 +74,12 @@ static inline void zap_pid_ns_processes(struct
> | pid_namespace *ns)
> | {
> | BUG();
> | }
> | +
> | +static inline int task_in_pid_ns(struct task_struct *tsk,
> | +   struct pid_namespace *ns)
> | +{
> | + return 1;
> | +}
> | #endif /* CONFIG_PID_NS */
> |
> | static inline struct pid_namespace *task_active_pid_ns(struct
> | task_struct *tsk)
> | diff --git a/kernel/pid_namespace.c b/kernel/pid_namespace.c
> | index 98702b4..3e71011 100644
> | --- a/kernel/pid_namespace.c
> | +++ b/kernel/pid_namespace.c
> | @@ -188,6 +188,26 @@ void zap_pid_ns_processes(struct pid_namespace
> *pid_ns)
> | return;
> | }
> |
> | +/*
> | + * Checks whether tsk has a pid in the pid namespace ns.
> | + * Must be called with tasklist_lock read-locked or under
> rcu_read_lock()
> | + */
> | +int task_in_pid_ns(struct task_struct *tsk, struct pid_namespace *ns)
> | +{
> | + struct pid *pid = task_pid(tsk);
> | +
> | + if (!pid)
> | + return 0;
> | +
> | + if (pid->level < ns->level)
> | + return 0;
>
> ns can be NULL if tsk is exiting.

ns is from current, and this is currently only called from
kill_something_info, so it should not be exiting in this path.

> Like Pavel said, we had couple of attempts to fix the larger problem of
> signal semantics in containers but did not have a consensus on handling

> blocked/unhandled signals to container-init.
>
> It would still be good to fix this "kill -1" problem.

It is a separate issue, so, yeah.

> Eric had a slightly optimized interface, 'pid_in_pid_ns()' in following
> patchset. Maybe we could use that ?
>
> https://lists.linux-foundation.org/pipermail/containers/2007-December/009174.html

See my response to Eric. I think that patch looks good... (Well, nr could
be set to 2 initially, to avoid the nr <= 1 check.)

> | +
> | + if (pid->numbers[ns->level].ns != ns)
> | + return 0;
> | +
> | + return 1;
> | +}
> | +
> | static __init int pid_namespaces_init(void)
> | {
> | pid_ns_cachep = KMEM_CACHE(pid_namespace, SLAB_PANIC);
> | diff --git a/kernel/signal.c b/kernel/signal.c
> | index 6c0958e..93713a5 100644
> | --- a/kernel/signal.c
> | +++ b/kernel/signal.c
> | @@ -1145,7 +1145,8 @@ static int kill_something_info(int sig, struct
> | siginfo *info, int pid)
> | struct task_struct * p;
> |
> | for_each_process(p) {
> | -   if (p->pid > 1 && !same_thread_group(p, current)) {
> | +   if (p->pid > 1 && !same_thread_group(p, current) &&
> | +   task_in_pid_ns(p, current->nsproxy->pid_ns)) {
> | int err = group_send_sig_info(sig, info, p);
> | ++count;
> | if (err != -EPERM)
> | --
> | 1.5.5.1

--
Daniel Hokka Zakrisson

Subject: Re: [PATCH 1/2] signals: kill(-1) should only signal processes in the same namespace
Posted by ebiederm on Thu, 17 Jul 2008 18:45:28 GMT
View Forum Message <> Reply to Message

"Daniel Hokka Zakrisson" <daniel@hozac.com> writes:

> Looks like you've already done that. :-) (Referring to Sukadev's email.)
> Is there any reason we don't just merge that patch?

I knew I had done something like that. Sure let's revive the patch and send
it. I don't know why it got lost the first time.

Eric

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

Subject: Re: [PATCH 1/2] signals: kill(-1) should only signal processes in the same namespace
Posted by ebiederm on Thu, 17 Jul 2008 18:46:23 GMT
View Forum Message <> Reply to Message

sukadev@us.ibm.com writes:
>

> Like Pavel said, we had couple of attempts to fix the larger problem of
> signal semantics in containers but did not have a consensus on handling
> blocked/unhandled signals to container-init.

Oh. I thought we were pretty close then I or somebody ran out of steam.

> It would still be good to fix this "kill -1" problem.
>
> Eric had a slightly optimized interface, 'pid_in_pid_ns()' in following
> patchset. Maybe we could use that ?
>
> https://lists.linux-foundation.org/pipermail/containers/2007-December/009174.html

Eric

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

Subject: Re: [PATCH 1/2] signals: kill(-1) should only signal processes in the same namespace
Posted by Oleg Nesterov on Wed, 23 Jul 2008 14:33:35 GMT
View Forum Message <> Reply to Message

```
up);
+ if (!is_timer_slack_allowed(cgroup_to_tslack_cgroup(cgroup), val))
+  return -EPERM;
+
+ /* Change timer slack value for all tasks in the cgroup */
+ cgroup_iter_start(cgroup, &it);
+ while ((task = cgroup_iter_next(cgroup, &it)))
+  task->timer_slack_ns = val;
+ cgroup_iter_end(cgroup, &it);
+
+ return 0;
+}
+
+static u64 tslack_read_range(struct cgroup *cgroup, struct cftype *cft)
+{
+ struct timer_slack_cgroup *tslack_cgroup;
+
+ tslack_cgroup = cgroup_to_tslack_cgroup(cgroup);
+ switch (cft->private) {
+ case TIMER_SLACK_MIN:
+  return tslack_cgroup->min_slack_ns;
+ case TIMER_SLACK_MAX:
+  return tslack_cgroup->max_slack_ns;
+ default:
+  BUG();
+ };
+}
+
+static int tslack_write_range(struct cgroup *cgroup, struct cftype *cft,
+  u64 val)
+{
+ struct timer_slack_cgroup *tslack_cgroup;
+ struct cgroup_iter it;
+ struct task_struct *task;
+
+ if (cgroup->parent) {
+  struct timer_slack_cgroup *parent;
+  parent = cgroup_to_tslack_cgroup(cgroup->parent);
+  if (!is_timer_slack_allowed(parent, val))
+   return -EPERM;
+ }
+
+ tslack_cgroup = cgroup_to_tslack_cgroup(cgroup);
+ switch (cft->private) {
```

```
+ case TIMER_SLACK_MIN:
+  if (val > tslack_cgroup->max_slack_ns)
+   return -EINVAL;
+  tslack_cgroup->min_slack_ns = val;
+  break;
+ case TIMER_SLACK_MAX:
+  if (val < tslack_cgroup->min_slack_ns)
+   return -EINVAL;
+  tslack_cgroup->max_slack_ns = val;
+  break;
+ default:
+  BUG();
+ }
+
+ /*
+  * Adjust timer slack value for all tasks in the cgroup to fit
+  * min-max range.
+  */
+ cgroup_iter_start(cgroup, &it);
+ w
```

Subject: Re: [PATCH 1/2] signals: kill(-1) should only signal processes in the same namespace
Posted by Daniel Hokka Zakrisso on Wed, 23 Jul 2008 16:09:40 GMT
View Forum Message <> Reply to Message

Are there any more granular patches than this very grand one available?

http://download.openvz.org/kernel/022stab044.1/patches/patch -022stab044-combined

Have you given any thought to merging some of the functionality from
your patch back into the mainline kernel?

-- Dave> Are there any more granular patches than this very grand one available?
1. a bit more granular patches are available in SRC RPM. But these are
drivers only :(
2. they are not publicly available since it is private company CVS.
I think we are to setup git repository in near future. This will make
our developement process much more transparent.

if you are interested in some specific patch or piece of code, we can
extract it for you, though I understand it is not very much convinient
way of doing things.

> http://download.openvz.org/kernel/022stab044.1/patches/patch -022stab044-combined
>
> Have you given any thought to merging some of the functionality from

> your patch back into the mainline kernel?
We always send mainstream fixes to Linus et al.
Other functionality will be definetely sent to mainstream step by step,
but we do not wait it to be an easy and quick task.

KirillAll,

Please, add devel@openvz.org to CC on any
LKML/Linus/Morton/... communication.

KirillPatch from Pavel (xemul@):
Missed newline spoils ouptut.

```
--- ./fs/dcache.c.nl 2005-11-29 19:23:41.000000000 +0300
+++ ./fs/dcache.c 2005-11-30 16:51:48.089948864 +0300
@@ -1779,7 +1779,7 @@ static void check_alert(struct vfsmount

  sb = dentry->d_sb;
  printk(KERN_ALERT "%s check alert! file:[%s] from %d/%s, dev%x\n"
-   "Task %d/%d[%s] from VE%d, execenv %d",
+   "Task %d/%d[%s] from VE%d, execenv %d\n",
    str, p, VE_OWNER_FSTYPE(sb->s_type)->veid,
    sb->s_type->nam
```