
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 Emelianov 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, but 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, building 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 than 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. Your 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>
