
Subject: Re: [PATCH] pidns: Limit kill -1 and cap_set_all
Posted by [Kirill Korotaev](#) on Mon, 29 Oct 2007 08:36:29 GMT
[View Forum Message](#) <> [Reply to Message](#)

I dislike this patch:
it's not scalable/efficient to travers all the tasks
while we know the pid namespace we care about.

Kirill

Eric W. Biederman wrote:

```
> 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 efb9cd..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)
```

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