
Subject: Re: [PATCH] pidns: Place under CONFIG_EXPERIMENTAL (take 2)

Posted by [Kirill Korotaev](#) on Mon, 29 Oct 2007 07:49:22 GMT

[View Forum Message](#) <> [Reply to Message](#)

Can you please send namespace related patches to containers@ ML first
before sending them to Linus/Andrew?

Thanks,
Kirill

Eric W. Biederman wrote:

> This is my trivial patch to swat innumerable little bugs
> with a single blow.
>
> After some intensive review (my apologies for not having
> gotten to this sooner) what we have looks like a good
> base to build on with the current pid namespace code but
> it is not complete, and it is still much to simple to find
> issues where the kernel does the wrong thing outside of
> the initial pid namespace.
>
> Until the dust settles and we are certain we have the ABI and
> the implementation is as correct as humanly possible let's keep
> process ID namespaces behind CONFIG_EXPERIMENTAL.
>
> Allowing us the option of fixing any ABI or other bugs
> we find as long as they are minor.
>
> Allowing users of the kernel to avoid those bugs simply
> by ensuring their kernel does not have support for multiple
> pid namespaces.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
> ---
> include/linux/pid_namespace.h | 22 ++++++
> init/Kconfig | 12 +++++++
> kernel/pid.c | 2 ++
> 3 files changed, 36 insertions(+), 0 deletions(-)
>
> diff --git a/include/linux/pid_namespace.h b/include/linux/pid_namespace.h
> index 0135c76..0227e68 100644
> --- a/include/linux/pid_namespace.h
> +++ b/include/linux/pid_namespace.h
> @@ -29,6 +29,7 @@ struct pid_namespace {
>
> extern struct pid_namespace init_pid_ns;
>

```

> +ifdef CONFIG_PID_NS
> static inline struct pid_namespace *get_pid_ns(struct pid_namespace *ns)
> {
>   if (ns != &init_pid_ns)
> @@ -45,6 +46,27 @@ static inline void put_pid_ns(struct pid_namespace *ns)
>     kref_put(&ns->kref, free_pid_ns);
> }
>
> +#else /* !CONFIG_PID_NS */
> +#include <linux/err.h>
> +
> +static inline struct pid_namespace *get_pid_ns(struct pid_namespace *ns)
> +{
> +  return ns;
> +}
> +
> +static inline struct pid_namespace *copy_pid_ns(unsigned long flags, struct pid_namespace *ns)
> +{
> +  if (flags & CLONE_NEWPID)
> +    ns = ERR_PTR(-EINVAL);
> +  return ns;
> +}
> +
> +static inline void put_pid_ns(struct pid_namespace *ns)
> +{
> +}
> +
> +endif /* CONFIG_PID_NS */
> +
> static inline struct pid_namespace *task_active_pid_ns(struct task_struct *tsk)
> {
>   return tsk->nsproxy->pid_ns;
> diff --git a/init/Kconfig b/init/Kconfig
> index 8b88d0b..72e37c0 100644
> --- a/init/Kconfig
> +++ b/init/Kconfig
> @@ -215,6 +215,18 @@ config USER_NS
>   vservers, to use user namespaces to provide different
>   user info for different servers. If unsure, say N.
>
> +config PID_NS
> +  bool "PID Namespaces (EXPERIMENTAL)"
> +  default n
> +  depends on EXPERIMENTAL
> +  help
> +    Support process id namespaces. This allows having multiple
> +    processes with the same pid as long as they are in different

```

```
> + pid namespaces. This is a building block of containers.  
> +  
> + Unless you want to work with an experimental feature  
> + say N here.  
> +  
> config AUDIT  
> bool "Auditing support"  
> depends on NET  
> diff --git a/kernel/pid.c b/kernel/pid.c  
> index d1db36b..f815455 100644  
> --- a/kernel/pid.c  
> +++ b/kernel/pid.c  
> @@ -537,6 +537,7 @@ err_alloc:  
>     return NULL;  
> }  
>  
> +#ifdef CONFIG_PID_NS  
> static struct pid_namespace *create_pid_namespace(int level)  
> {  
>     struct pid_namespace *ns;  
> @@ -621,6 +622,7 @@ void free_pid_ns(struct kref *kref)  
>     if (parent != NULL)  
>         put_pid_ns(parent);  
> }  
> +#endif /* CONFIG_PID_NS */  
>  
> void zap_pid_ns_processes(struct pid_namespace *pid_ns)  
> {
```

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