Subject: Re: Screamm.. commit f400e198b2ed26ce55b22a1412ded0896e7516ac
Posted by serue on Thu, 29 Mar 2007 14:15:26 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> "Serge E. Hallyn" <serue@us.ibm.com> writes:
>
> > Yup. Looks like ambiguous naming once again hid some real (future)
> > bugs. This is of course safe so far in mainline, but needs to be split
> > into
> >
> > static inline int is_global_init(struct task_struct *tsk)
> > {
> > return (tsk == &init_task);
> > }
> >
> > and
> >
> > static inline int
> > is_container_init(struct task_struct *task, struct pid_namespace *ns)
> > {
> > return (__pid_nr(task, ns) == 1);
> > }
>
> Conceptually yes. The implementation of is_global_init is just wrong.
> &init_task is the first processors idle thread.

Uh, yeah. This is "do_what_I_mean" compiler code. I wasn't even
sure offhand whether init_task existed. :)

> is_container_init looks correct but I don't know if the ns parameter
> makes any sense.

I'm not sure yet, but I suspect we will want to treat, for instance,
signal delivery to a task which is pid==1 for a child namespace
differently based on whether the signal comes from inside the pidns
where it is pid==1, or from a parent pidns.

> > Where the latter is needed in, for instance, kernel/capability.c.
>
> Yes.
>
> I think more clear cut examples could be made. It isn't clear to me
> why we skip pid == 1 in kernel/capability.c

Because the capset(2) manpage says:

 For capset(), pid can also be: -1, meaning

perform the change on all threads except the caller and
init(8);

> I believe a good example is that inside a container you should not
> be able to send pid == 1 a signal it doesn't have a handler for.
> While from outside the container we need that capability.

Exactly.

thanks,
-serge