

---

Subject: Re: [PATCH] Make access to task's nsproxy liter

Posted by [serue](#) on Mon, 13 Aug 2007 15:01:54 GMT

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

---

Quoting Eric W. Biederman (ebiederm@xmission.com):

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

>

>> On 08/10, Pavel Emelyanov wrote:

>>>

>>> Oleg Nesterov wrote:

>>>> On 08/10, Serge E. Hallyn wrote:

>>>>> Quoting Pavel Emelyanov (xemul@openvz.org):

>>>>>>+/\*

>>>>>>+ \* the namespaces access rules are:

>>>>>>+ \*

>>>>>>+ \* 1. only current task is allowed to change tsk->nsproxy pointer or  
>>>>>>+ \* any pointer on the nsproxy itself

>>>>>>+ \*

>>>>>>+ \* 2. when accessing (i.e. reading) current task's namespaces - no  
>>>>>>+ \* precautions should be taken - just dereference the pointers

>>>>>>+ \*

>>>>>>+ \* 3. the access to other task namespaces is performed like this

>>>>>>+ \* rcu\_read\_lock();

>>>>>>+ \* nsproxy = task\_nsproxy(tsk);

>>>>>>+ \* if (nsproxy != NULL) {

>>>>>>+ \* / \*

>>>>>>+ \* \* work with the namespaces here

>>>>>>+ \* \* e.g. get the reference on one of them

>>>>>>+ \* \*/

>>>>>>+ \* } / \*

>>>>>>+ \* \* NULL task\_nsproxy() means that this task is

>>>>>>+ \* \* almost dead (zombie)

>>>>>>+ \* \*/

>>>>>>+ \* rcu\_read\_unlock();

>>>>>> And lastly, I guess that the caller to switch\_task\_namespaces() has

>>>>>> to ensure that new\_nsproxy either (1) is the init namespace, (2) is a

>>>>>> brand-new namespace to which noone else has a reference, or (3) the

>>>>>> caller has to hold a reference to the new\_nsproxy across the call to

>>>>>> switch\_task\_namespaces().

>>>>>>

>>>>>> As it happens the current calls fit (1) or (2). Again if we happen to

>>>>>> jump into the game of switching a task into another task's nsproxy,

>>>>>> we'll need to be mindful of (3) so that new\_nsproxy can't be tossed into

>>>>>> the bin between

>>>>>>

>>>>>> if (new)

>>>>>> get\_nsproxy(new);

>>>>>>

> >> >4) Unless `tsk == current`, `get_task_namespaces(tsk)` and `get_nsproxy(tsk)`  
> >> > are racy even if done under `rcu_read_lock()`.  
> >>  
> >> Yup :)  
> >>  
> >> It is already written in comment that only the current is allowed  
> >> to change its nsproxy. I.e. when `switch_task_nsproxy()` is called  
> >> for `tsk` other than `current` it's a BUG  
> >  
> > Yes, but what I meant is that this code  
> >  
> > `rcu_read_lock();`  
> > `nsproxy = task_nsproxy(tsk);`  
> > `if (nsproxy != NULL)`  
> > `get_nsproxy(nsproxy);`  
> > `rcu_read_unlock();`  
> >  
> > `if (nsproxy) {`  
> > `use_it(nsproxy);`  
> > `put_nsproxy(nsproxy);`  
> > `}`  
> >  
> > is not safe despite the fact we are `_not_` changing `tsk->nsproxy`.  
> >  
> > The patch itself is correct because we don't do that, and the comment  
> > is right. Just it is not immediately obvious.  
>  
> Ugh. That is nasty, non obvious and almost a problem. I don't want  
> to do `get_net(nsproxy->net_ns)` from another task so I can migrate  
> network between namespaces.  
>  
> But thinking about it because we don't do the other decrements  
> until later we can still increment the counts on the individual  
> namespaces. We just can't share nsproxy.  
>  
> So if you did want to do an enter thing you could copy the  
> nsproxy object of a task under the `rcu_read_lock()`, and  
> you would be fine.

Yup, that makes sense, good idea.

-serge

---