
Subject: Re: [patch -mm 08/17] nsproxy: add hashtable
Posted by [Cedric Le Goater](#) on Mon, 11 Dec 2006 16:09:33 GMT
[View Forum Message](#) <> [Reply to Message](#)

Herbert Poetzl wrote:

> On Fri, Dec 08, 2006 at 01:57:38PM -0700, Eric W. Biederman wrote:
>> "Serge E. Hallyn" <serue@us.ibm.com> writes:
>>
>>> Quoting Eric W. Biederman (ebiederm@xmission.com):
>>>> clg@fr.ibm.com writes:
>>>>
>>>>> From: Cedric Le Goater <clg@fr.ibm.com>
>>>>>
>>>>> This patch adds a hashtable of nsproxy using the nsproxy as a key.
>>>>> init_nsproxy is hashed at init with key 0. This is considered to be
>>>>> the 'host' nsproxy.
>>>> NAK. Which namespace do these ids live in?
>
> well, I gave a similar answer in another email,
> so I fully agree with the NAK here ...

hmm, I wasn't that clear to me. OK, let's dig :)

>>>> It sounds like you are setting up to make the 'host' nsproxy
>>>> special and have special rules. That also sounds wrong.
>>>>
>>>> Even letting the concept of nsproxy escape to user space sounds
>>>> wrong. nsproxy is an internal space optimization. It's not struct
>>>> container and I don't think we want it to become that.
>>>>
>>>> Eric
>>> So would you advocate referring to containers just by the pid of
>>> a process containing the nsproxy, and letting userspace maintain
>>> a mapping of id's to containers through container create/enter
>>> commands? Or is there some other way you were thinking of doing
>>> this?
>
>> There are two possible ways.
>> 1) Just use a process using the namespace.
>> This is easiest to implement.
>
>> 2) Have a struct pid reference in the namespace itself,
>> and probably an extra pointer in struct pid to find it.
>> This is the most stable, because fork/exit won't affect
>> which pid you need to use.
>
> while I agree that nsproxy is definitely the wrong
> point to tie a 'context' too, as it can contain a

- > mixture of spaces from inside and outside a context,
- > and it would require to forbid doing things like
- > clone() with the space flags, both inside and outside
- > a 'container' to allow to use them for actual vps
- > applications, I think that we have to have some kind
- > of handle to tie specific sets of namespaces too

this is nsproxy ...

- > that 'can' be an nsproxy or something different, but
- > I'm absolutely unhappy with tying it to a process,

hmm, what do you mean ? nsproxy survives the death of any process. It's not tied to any process in particular. One process creates it with an unshare but that's all.

the ->nsproxy in task_struct is a way to find it.

- > as I already mentioned several times, that lightweight
- > 'containers' do not use/have an init process, and no
- > single process might survive the entire life span of
- > that 'container' ...

I think there is a misunderstanding here. a 'container' or 'nsproxy' or what ever is a set of namespaces which are not tied to a process.

you can do that today on 2.6.19 with utsname.

- >> Beyond that yes it seems to make sense to let user space
- >> maintain any mapping of containers to ids.
- >
- > I agree with that, but we need something to move
- > around between the various spaces ...

the bind_ns syscall lets the user specify the mapping. this is not done by the kernel.

I had to introduce some rules, like giving more capabilities to some processes, but that can be changed. For the moment, they have to live in "init_proxy".

- > for example, Linux-VServer ties the namespaces to
- > the context structure (atm) which allows userspace
- > to set and enter specific spaces of a guest context
- > (I assume OpenVZ does similar)

What's the big difference with nsproxy ?

C.

Containers mailing list

Containers@lists.osdl.org

<https://lists.osdl.org/mailman/listinfo/containers>
