Subject: Re: Namespaces exhausted CLONE_XXX bits problem
Posted by Oren Laadan on Mon, 14 Jan 2008 21:36:13 GMT
View Forum Message <> Reply to Message

Serge E. Hallyn wrote:
> Quoting Pavel Emelyanov (xemul@openvz.org):
>> Serge E. Hallyn wrote:
>>> Quoting Cedric Le Goater (clg@fr.ibm.com):
>>>> to be more precise :
>>>>
>>>> long sys_clone_something(struct clone_something_args args)
>>>>
>>>> and
>>>>
>>>> long sys_unshare_something(struct unshare_something_args args)
>>>>
>>>> The arg passing will be slower bc of the copy_from_user() but we will
>>>> still have the sys_clone syscall for the fast path.
>>>>
>>>> C.
>>> I'm fine with the direction you're going, but just as one more option,
>>> we could follow more of the selinux/lsm approach of first requesting
>>> clone/unshare options, then doing the actual clone/unshare. So
>>> something like
>>>
>>> sys_clone_request(extended_64bit_clone_flags)
>> What if we someday hit the 64-bit limit? :)
>>
>>> sys_clone(usual args)

One (security ?) problem with a two stage approach is that the operation
may not be completed in an atomic manner; e.g. if there are two threads
doing the first call before any of them gets to the second call. Or at
least ensure that such races cannot occur by design. (In contrast, with
sys_indirect() everything is atomic).

Also, in a two-step approach, using /proc as opposed to a specialized
system call incurs higher overhead should ultra-fast clone()s are a
goal by itself.

I second the concern of running out of 64 bits of flags. In fact, the
problem with the flags is likely to be valid outside our context, and
general to the linux kernel soon. Should we not discuss it there too ?

>>>
>>> or
>>>
>>> echo pid,mqueue,user,ipc,uts,net > /proc/self/clone_unshare

>>> clone()
>> Well, this is how sys_indirect() was intended to work. Nobody
>> liked it, so I'm afraid this will also not be accepted.
>
> I would have thought sys_indirect would be disliked because
> it looks like an ioctl type multiplexor. Whereas sys_clone_request()
> or /proc/self/clone_unshare simply sets arguments in advance, the
> way /proc/self/attr/current does.

I find the sys_indirect() approach very appealing, in particular
because it is designed and motivated by a non-ioctl multiplexing
and backward compatibility in mind. Like any API it can be abused
and misused, but since it applies to actual system calls and not
obscured ioctl, it is far less likely to become a victim (so to
speak ...).

While I prefer the sys_indirect() (personally I find it elegant,
but it isn't clear that it will be merged), from a technical point
of view any of new system call, sys_indirect() or a 2-step approach
approach - all three seem plausible. The final solution, however
needs to be coordinated with the rest of the kernel developers.

Oren.

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

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