
Subject: Re: Namespaces exhausted CLONE_XXX bits problem
Posted by [Cedric Le Goater](#) on Tue, 15 Jan 2008 09:40:37 GMT
[View Forum Message](#) <> [Reply to Message](#)

Pavel Emelyanov wrote:

> Cedric Le Goater wrote:

>> Pavel Emelyanov wrote:

>>> Dave Hansen wrote:

>>>> On Mon, 2008-01-14 at 16:36 -0500, Oren Laadan wrote:

>>>>> 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 ?

>>>> It would be pretty easy to make a new one expandable:

>>>>

>>>> sys_newclone(int len, unsigned long *flags_array)

>>>>

>>>> Then you could give it a virtually unlimited number of "unsigned long"s
>>>> pointed to by "flags_array".

>>>>

>>>> Plus, the old clone just becomes:

>>>>

>>>> sys_oldclone(unsigned long flags)

>>>> {

>>>> do_newclone(1, &flags);

>>>> }

>>>>

>>>> We could validate the flags array address in sys_newclone(), then call

>>>> do_newclone().

>>> Hmm. I have an idea how to make this w/o a new system call. This might

>>> look wierd, but. Why not stopple the last bit with a CLONE_NEWCLONE and

>>> consider the parent_tidptr/child_tidptr in this case as the pointer to

>>> an array of extra arguments/flags?

>> It's a bit hacky but it looks like a good idea to me !

>>

>> Shall we use parent_tidptr or child_tidptr to pass a extended array of

>> flags only ? if we could pass the pid of the task to be cloned, it would

>> be useful for c/r.

>

> Yup. I think we can declare a

>

> struct new_clone_arg {

> unsigned int size;

> };

>

> and consider the xx_tidptr to be a pointer on it. After this we

> may sen patches that add fields to this structure.

>

```

> E.g. first
>
> struct new_clone_arg {
>   unsigned int size;
>   + unsigned long new_flags;
> };
>
> to add flags for cloning new namespaces. Later
>
> struct new_clone_arg {
>   unsigned int size;
>   unsigned long new_flags;
>   + int desired_pid;
> };
>
> and each code that needs to access the extra argument would need
> to check for new_clone_arg->size to be not less than the offset
> of the field he need an access to. E.g. like this:
>
> #define clone_arg_has(arg, member) ({ \
>   struct new_clone_arg *__carg = arg; \
>   (__carg->size >= offsetof(struct new_clone_arg, member) + \
>   sizeof(__carg->member)) })
>
> ...
>
> if (!clone_arg_has(arg, desired_pid))
>   return -EINVAL;
>
> This would keep the API always compatible.

```

Pavel, this is pretty neat.

I think we need to work on a patch now and send it to andrew and lkml@ to have a larger audience.

I doesn't seem to be a really big patch and I wondering how I could help. We still have to prepare something for security_task_create()

Thanks !

C.

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