Subject: Re: Re: namespace and nsproxy syscalls
Posted by Cedric Le Goater on Tue, 03 Oct 2006 16:51:19 GMT
View Forum Message <> Reply to Message

Serge E. Hallyn wrote:
> Quoting Herbert Poetzl (herbert@13thfloor.at):
>>>> how to avoid having duplicate identifiers when there
>>>> is a chance that the same pid will be used again
>>>> to create a second namespace?
>>> Well at least that's simple, the pid will no longer be a valid handle to
>>> the first namespace ever since that process died :)
>> which simply makes it inaccesible which is not
>> what you actually want, sorry ...
>
> Nonsense. It is still accessible via any other pids for processes in
> that namespace. (i.e. if you're in pidns 1, and (pidns 2, pid 1)
> has started (pidns 2, pid 2) and then exited, then (pidns 2, pid 2)
> will also be known by some (pidns 1, pid X), so you can access the
> namespace via pid X from your pidns 1 process.

hmm, a few comments on the pid namespace :

* the current model we have been talking about does not map all
  processes of a pid namespace in the parent namespace. only the first
  process of a child namespace is required to but not its children.

* but we also said that a pid namespace can not survive the death of its
  pid 1.

> How to actually find a pid that will last long enough for you to find
> it and then access it is an exercise left to the reader :)

well, if pid 1 is always around, it could be used as a handle but it
would be only valid if we are unsharing pid namespaces. what about
the other namespaces ? we could unshare the utsname only and still
want to reference it one way or the other.

> In other words, I was saying that the duplicate identifiers is not a
> bug, but I thought I had left it clearly implied that the approach not
> practical, and we will need namespace ids.

yes, i'm testing such a patch as discussed on the list. I have good
results for a full nsproxy but i'm having trouble with the mnt namespace
(used to be called namespace) which is stored in nsproxy and the
fs_struct which is stored in the task_struct.

C.