Subject: Re: [patch -mm 08/17] nsproxy: add hashtable
Posted by serue on Mon, 11 Dec 2006 15:29:35 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> Herbert Poetzl <herbert@13thfloor.at> writes:
>
> >> 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.
> >
> > that 'can' be an nsproxy or something different, but
> > I'm absolutely unhappy with tying it to a process,
> > 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' ...
>
> Herbert think of a session id. That is a pid that is
> tied to something besides a single process.
>
> It is easy and recursion safe to tie a pid to a namespace
> or anything else that make sense, as I suggested above.

Recursion safe, but limiting in that you can only descend one pid namespace at a time. That limitation aside, providing task notifiers for all unshares, plus a syscall to jump into all namespaces belonging to a process known to you by a particular pid, could be a good approach. Now you can have a userspace daemon keeping namespace id's tied to processes, giving you the ability to say

```
 ns_exec -a -l ns12 my_prog
  (unshare all namespaces and run my_prog in
  a container known as 'ns12')
 ns_enter -l ns12 /bin/ps
  (jump into ns12 and run /bin/ps)
```

The likely requirement to run a namespace tracking daemon in each pid namespace that wants such functionality could become a resource hog, but that may be just a theoretical problem, since you'll only need that if you want to play with namespaces, meaning that for it to be a problem you'd have to have lots of

virtual servers each maintaining namespaces to either do
process migration or spawn more virtual servers (which each
maintain namespaces to...)

> The pid namespace feels like the right place for this kind
> of activity.
>
> >> 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 ...
>
> If you have CAP_SYS_PTRACE or you have a child process
> in a container you can create another with ptrace.
>
> Now I don't mind optimizing that case, with something like
> the proposed bind_ns syscall. But we need to be darn certain
> why it is safe, and does not change the security model that
> we currently have.

Sigh, and that's going to have to be a discussion per namespace.

> I have not seen that discussion yet, and until I do I have
> serious concerns. That discussion needs to be on lkml as
> well. Why did Al Viro think this was a bad idea when it
> was proposed for the mount namespace?
>
> This is where you are on the edge of some very weird interface
> interactions. Without suid programs it would be completely safe
> for anyone to unshare their mount namespace. With suid programs
> allowed an unprivileged unshare mount namespace unshare is next to
> impossible.
>
> > 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)
>
> Yep, and we certainly need to find a way to fulfill this usage
> requirement.

_____