Subject: Re: [ckrm-tech] [PATCH 7/7] containers (V7): Container interface to nsproxy subsystem
Posted by serue on Mon, 26 Mar 2007 21:55:30 GMT
View Forum Message <> Reply to Message

Quoting Srivatsa Vaddagiri (vatsa@in.ibm.com):
> On Mon, Feb 12, 2007 at 12:15:28AM -0800, menage@google.com wrote:
> > +/*
> > + * Rules: you can only create a container if
> > + *   1. you are capable(CAP_SYS_ADMIN)
> > + *   2. the target container is a descendant of your own container
> > + */
> > +static int ns_create(struct container_subsys *ss, struct container *cont)
> > +{
> > + struct nscont *ns;
> > +
> > + if (!capable(CAP_SYS_ADMIN))
> > + return -EPERM;
>
> Does this check break existing namespace semantics in a subtle way?
> It now requires that unshare() of namespaces by any task requires
> CAP_SYS_ADMIN capabilities.
>
> clone(.., CLONE_NEWUTS, ..)->copy_namespaces()->ns_container_clone()->
> ->container_clone()-> .. -> container_create() -> ns_create()
>
> Earlier, one could unshare his uts namespace w/o CAP_SYS_ADMIN
> capabilities.

Not so, CAP_SYS_ADMIN is required. Depending on your tree, check
kernel/utsname.c or kernel/nsproxy.c.

Mind you, whether there is a reason to require CAP_SYS_ADMIN for utsname
I'm not sure. But if there is any way of finding privileged software
which would behave differently based on utsname information, then we
should.

> Now it is required. Is that fine? Don't know.
>
> I feel we can avoid this check totally and let the directory permissions
> take care of these checks.
>
> Serge, what do you think?

There is no way we can think about doing that until we determine that
every kernel will have the ns container subsystem compiled in and
mounted, right?

-serge