Subject: Re: Re: [RFC] [PATCH 0/3] containers: introduction
Posted by Paul Menage on Fri, 12 Jan 2007 01:33:19 GMT
View Forum Message <> Reply to Message

On 1/11/07, Serge E. Hallyn <serue@us.ibm.com> wrote:
>
> That's what's holding me back here - I'm still not sure whether
> to proceed with a separate implementation, proceed with the
> current implementation of Paul's containers, or wait for an
> update from Paul responding to your feedback.

I think that having just a single process-grouping infrastructure in the kernel, rather than separate ones for CPUsets, virtual servers, BeanCounters, ResGroups, etc, is definitely something we should aim for, provided that the requirements for virtual servers aren't too different to those for other users.

I've been thinking about how the container_clone() function would need to work. Essentially, the sequence would be something like the following:

1) do_fork() or sys_unshare() see that some ns_proxy bits have changed, and call container_clone(&nsproxy_subsys)

In container_clone():

2) Check that the ns_proxy container subsystem was bound to some hierarchy; if not, find an unused hierarchy and bind ns_proxy subsystem to it. (Or fail with -EBUSY if there are no free container subsystems). By the end of this step, ns_proxy is bound to hierarchy H.

3) Create a new subcontainer of the current process' container in hierarchy H. This will involve some VFS manipulation, since normally container creation operations are as part of a mkdir operation, but shouldn't be too tricky.

4) Move the current process (or possibly the new child process in the case of do_fork()) into that container. (This doesn't affect the container mappings of the current or child process in any hierarchies other than H).

So in general if you wanted to combine ns_proxy isolation and resource isolation, you'd mount an instance of containerfs that bound both the ns_proxy subsystem and any resource controllers that you wanted, prior to creating any virtual servers. Alternatively you could have a separate hierarchy for the resource controllers and manually move the new virtual server's root process into the appropriate resource control container.

Paul

Subject: Re: Re: [RFC] [PATCH 0/3] containers: introduction
Posted by serue on Fri, 12 Jan 2007 18:29:24 GMT
View Forum Message <> Reply to Message

Quoting Paul Menage (menage@google.com):
> On 1/11/07, Serge E. Hallyn <serue@us.ibm.com> wrote:
> >
> > That's what's holding me back here - I'm still not sure whether
> > to proceed with a separate implementation, proceed with the
> > current implementation of Paul's containers, or wait for an
> > update from Paul responding to your feedback.
>
> I think that having just a single process-grouping infrastructure in
> the kernel, rather than separate ones for CPUsets, virtual servers,
> BeanCounters, ResGroups, etc, is definitely something we should aim
> for, provided that the requirements for virtual servers aren't too
> different to those for other users.

I agree, so long as "provided requirements aren't too different" is
replaced by "provided there is commonality to be merged." Differences in
lifetime rules and fs behavior could make it pounding a round peg into
a square hole...

> I've been thinking about how the container_clone() function would need
> to work. Essentially, the sequence would be something like the
> following:
>
> 1) do_fork() or sys_unshare() see that some ns_proxy bits have
> changed, and call container_clone(&nsproxy_subsys)

There will be another possibility.

We were thinking that each container directory would have a file
representing each namespace in the nsproxy. To enter only a few
namespaces out of an existing namespace container, then, you could
create a directory for a new namespace container, link the namespaces
you want out of other containers, then enter the container (presumably
by doing 'echo (container_path) > /proc/$$/ns_container)'

So in some ways that's actually closer to what you currently have
than the default container creation rules.

> In container_clone():

>
> 2) Check that the ns_proxy container subsystem was bound to some
> hierarchy; if not, find an unused hierarchy and bind ns_proxy
> subsystem to it. (Or fail with -EBUSY if there are no free container
> subsystems). By the end of this step, ns_proxy is bound to hierarchy
> H.
>
> 3) Create a new subcontainer of the current process' container in
> hierarchy H. This will involve some VFS manipulation, since normally
> container creation operations are as part of a mkdir operation, but
> shouldn't be too tricky.
>
> 4) Move the current process (or possibly the new child process in the
> case of do_fork()) into that container. (This doesn't affect the
> container mappings of the current or child process in any hierarchies
> other than H).
>
> So in general if you wanted to combine ns_proxy isolation and resource
> isolation, you'd mount an instance of containerfs that bound both the
> ns_proxy subsystem and any resource controllers that you wanted, prior
> to creating any virtual servers.

That does sound useful.

> Alternatively you could have a
> separate hierarchy for the resource controllers and manually move the
> new virtual server's root process into the appropriate resource
> control container.

Boy, I really would like for you and Eric to work out your differences :)
so this could make its way into -mm (or be rejected, but hopefully not).
That would make basing the namespace-fs on your containers much easier...

thanks,
-serge