Subject: Re: [RFC] ns containers (v2): namespace entering
Posted by Herbert Poetzl on Sat, 10 Mar 2007 01:36:29 GMT
View Forum Message <> Reply to Message

On Mon, Feb 19, 2007 at 04:14:46PM -0600, Serge E. Hallyn wrote:
> (Updated patchset addressing Paul's comments. Still looking more
> for discussion on functionality, but barring much of that I guess
> I'll do something with tsk->fs then send the set to lkml)
>
> The following patchset uses the ns container subsystem to implement
> namespace entering. It applies over the container patchset Paul
> sent out earlier today.
>
> = Usage =
>
> Following is example usage:
>
> mkdir /container_ns
> mount -t container -ons nsproxy /container_ns
> (start screen with two windows)
> (screen one)
> (unshare uts namespace)
> hostname serge
> (screen two)
> hostname
> (shows old hostname)
> echo $$ > /container_ns/node1/tasks (assuming node1 is screen one's new container)
> hostname
> (shows 'serge')
>
> Of course to get a useful result with the mounts namespace, we would
> have to convert the fs_struct to a namespace, and the mounts and
> fs_struct namespaces would be entered together.
>
> If we follow this path, the next step would be to add files under
> the container directory for each namespace, with ops->link defined
> such that you could
>
> mkdir /container_ns/new_container
> ln /container_ns/vserver1/network /container_ns/new_container/network
> echo $$ > /container_ns/new_container/tasks
>
> and be entered into a set of namespaces containing all of your defaults
> except network, which would come from vserver1.
>
> This is RFC not just on implementation, but also on whether to do
> it at all. If so, then for all namespace, or only some? And if not,
> how to facilitate virtual server management.

>
> = Security =
>
> Currently to enter a namespace, you must have CAP_SYS_ADMIN, and must
> be entering a container which is an immediate child of your current
> container. So from the root container you could enter container /vserver1,
> but from container /vserver1 you could not enter /vserver2 or the root
> container.
>
> This may turn out to be sufficient. If not, then LSM hooks should be
> added for namespace management. Four hooks for nsproxy management (create,
> compose, may_enter, and enter), as well as some security_ns_clone hook for
> each separate namespace, so that the nsproxy enter and compose hooks have
> the information they need to properly authorize.
>
> = Quick question =
>
> Is it deemed ok to allow entering an existing namespace?
>
> If so, the next section can be disregarded. Assuming not, the following
> will need to be worked out.
>
> = Management alternatives =
>
> Mounts
>
> Ram has suggested that for mounts, instead of implementing namespace
> entering, the example from the OLS Shared Subtree paper could be
> used, as follows (quoting from Ram):
>
> > The overall idea is each container creates its own fs-namespace which
> > has a mirror mount tree under /container/c1 in the original
> > fs-namespace. So any changes in the container's fs-namespace reflect in
> > the mount tree under /container/c1 in the original fs-namespace, and
> > vice-versa. And all processes in the original fs-namespace can freely
> > roam around in the mirror trees of all the containers, mounted
> > under /container/cx, giving illusion that they have control over the
> > mounts in all the containers. Whereas the processes in a container can
> > just roam around its own fs-namespace and has no visibility to anything
> > outside its own fs-namespace..
> >
> > As an example the way to accomplish this is:
> >
> >   in the original namespace,
> >   1-1) mkdir /container
> >   1-2) mount --bind /container /container
> >   1-3) mount --make-unbindable /container
> >

> >
> > when a new container c1 is created
> >
> > 2-1) mkdir /container/c1
> > 2-2) mount --rbind / /container/c1
> > 2-3) mount --make-rshared /container/c1
> > 2-4) clone fs-namespace
> > in the new namespace,
> > 2-4-1) pivot_root /container/c1 /tmp
> > 2-4-2) umount -l /tmp
> >
> >
> > the mount at /container is made unbindable in steps 1-1 to 1-3, to
> > prune-out the mounts of other containers from being visible in this
> > container. The mount tree at /container/c1 is made shared in step
> > 2-3, to ensure that the cloned mount tree in the new namespace shall
> > be a exact mirror. The pivot_root in step 2-4-1 followed by umount in
> > step 2-4-2 ensures that the container sees only what is needed and
> > nothing else.
>
> Now, currently a root process in container c1 could make his fs unshared,
> but if that's deemed a problem presumably we can require an extra
> priv for that operation which can be dropped for any vservers.
>
> Herbert, and anyone else who wants mounts namespace entering, is the
> above an acceptable alternative?

sorry for the late answer, I almost missed that one ...

yes, that sounds like an acceptable alternative, but
it might give some interesting issues with references
to devices ... for example:

you mount a filesystem inside a namespace, so that
only the guest will see it (in theory) now you somehow
show that in the namespace copy too (on the host system)
and if some task decides to go camping there (cd into
that) it might keep the guest from unmounting that
device without ever knowing why ... or do you have some
smart solution to that?

> net+pid+uts
>
> Not sure about uts, but I'm pretty sure the vserver folks want the
> ability to enter another existing network namespace, and both vserver
> and openvz have asked for the ability to enter pid namespaces.

yes, definitely, pid and network namespaces have to

be accessible somehow, most administrative work is
done this way, when the administrator also maintains
the guests (i.e. doesn't want to bother accessing the
guest via special console/ssh/logon/whatever)

> The pid namespaces could be solved by always generating as many pids for
> a process as it has parent pid_namespaces. So if I'm in /vserver1, with
> one pid_namespace above me, not only my init process has an entry in the
> root pid_namespace (as I think has been suggested), but all my children
> will also continue to have pids in the root pid_namespace.

yep, sounds okay to me ...
note, our lightweight guests do not have an init
process, which is perfectly fine with the above, as
long as the init process is not considered a special
handle to the pid namespace :)

> Or, if it is ok for the pid namespace operations to be as coarse as
> "kill all processes in /vserver1", then that was going to be implemented
> using the namespace container subsystem as:
>
> rm -rf /container_ns/vserver1

that is definitely something you do not want to make
the general signalling solution, because typically
we have the following scenarios:

 - init less (lightweight) guest
   + a bunch of shutdown scripts are executed
   + term/kill is sent to the processes
   + the context is disposed

 - init based guest
   + a signal is sent to init
   + init executes the shutdown and kills off
     the 'other' processes
   + init finally calls reboot/halt
   + init and the context are disposed

> Any other (a) requirements, (b) ideas for alternate pid and network
> ns management without allowing namespace enters?

entering the spaces seems most natural and quite
essential to me, especially for administration and
debugging purposes ...

best,
Herbert

> -serge