Subject: [RFC] ns containers: namespace entering
Posted by serue on Mon, 12 Feb 2007 22:21:43 GMT
View Forum Message <> Reply to Message

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?

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.

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.

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

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

-serge

Subject: [RFC PATCH 3/4] namespace containers: add nsproxy to nscont struct

From: "Serge E. Hallyn" <serue@us.ibm.com>
Subject: [RFC PATCH 3/4] namespace containers: add nsproxy to nscont struct

Each ns container is associated with an nsproxy. Add that
nsproxy to the nscont struct, set it when a container is auto-
created on clone/unshare, and inc/dec the nsproxy to account for
each container referencing it.

Note that once the nscont->nsproxy is set, it will never change
for the duration of the container's lifetime.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>

---

kernel/container.c | 11 ++++++++++
kernel/ns_container.c | 11 ++++++++++
2 files changed, 22 insertions(+), 0 deletions(-)

e4dd14babf4fef1849cb0c5e797f8d176eb9a0a4
diff --git a/kernel/container.c b/kernel/container.c
index fc559ce..4608a12 100644
--- a/kernel/container.c
+++ b/kernel/container.c
@@ -1581,6 +1581,14 @@ static void get_unused_name(char *buf) {
sprintf(buf, "node%d", atomic_inc_return(&namecnt));
}

+/* XXX need to create include/linux/ns_container.h and move this there */
+#ifdef CONFIG_CONTAINER_NS
+void ns_set_nsproxy_from_task(struct container *cont, struct task_struct *tsk);
+#else
+static inline void
+ns_set_nsproxy_from_task(struct container *cont, struct task_struct *tsk) { }
+#endif
+
/**
* container_clone - duplicate the current container and move this
* task into the new child
@@ -1669,6 +1677,9 @@ int container_clone(struct task_struct *
goto again;
}

+ /* mark the auto-created container with the new namespace list */
+ ns_set_nsproxy_from_task(child, tsk);
+

```
  /* All seems fine. Finish by moving the task into the new container */
  ret = attach_task(child, tsk);
  mutex_unlock(&manage_mutex);
diff --git a/kernel/ns_container.c b/kernel/ns_container.c
index d60d4f5..2d5c578 100644
--- a/kernel/ns_container.c
+++ b/kernel/ns_container.c
@@ -11,6 +11,7 @@

 struct nscont {
  struct container_subsys_state css;
+ struct nsproxy *nsproxy; /* never changes once set */
  spinlock_t lock;
 };

@@ -81,10 +82,20 @@ static int ns_create(struct container_su
  return 0;
 }

+/* called from container_clone */
+void ns_set_nsproxy_from_task(struct container *cont, struct task_struct *tsk)
+{
+ struct nscont *ns = container_nscont(cont);
+ ns->nsproxy = tsk->nsproxy;
+ get_nsproxy(ns->nsproxy);
+}
+
 static void ns_destroy(struct container_subsys *ss,
          struct container *cont)
 {
  struct nscont *ns = container_nscont(cont);
+ if (ns->nsproxy)
+  put_nsproxy(ns->nsproxy);
  kfree(ns);
 }

--
1.1.6
```

Subject: Re: [RFC PATCH 3/4] namespace containers: add nsproxy to nscont struct
Posted by Paul Menage on Mon, 12 Feb 2007 23:27:47 GMT
View Forum Message <> Reply to Message

On 2/12/07, Serge E. Hallyn <serue@us.ibm.com> wrote:
> --- a/kernel/container.c
> +++ b/kernel/container.c
> @@ -1581,6 +1581,14 @@ static void get_unused_name(char *buf) {

> sprintf(buf, "node%d", atomic_inc_return(&namecnt));
> }
>
> +/* XXX need to create include/linux/ns_container.h and move this there */
> +#ifdef CONFIG_CONTAINER_NS
> +void ns_set_nsproxy_from_task(struct container *cont, struct task_struct *tsk);
> +#else
> +static inline void
> +ns_set_nsproxy_from_task(struct container *cont, struct task_struct *tsk) { }
> +#endif
> +

Can't this just go in include/linux/nsproxy.h ?

>
> + /* mark the auto-created container with the new namespace list */
> + ns_set_nsproxy_from_task(child, tsk);
> +

This should be in ns_container_clone() or some (possibly new)
subsystem callback, rather than embedded in the generic containers
code.

Paul

Subject: Re: [RFC PATCH 3/4] namespace containers: add nsproxy to nscont struct
Posted by serue on Tue, 13 Feb 2007 16:34:53 GMT
View Forum Message <> Reply to Message

Quoting Paul Menage (menage@google.com):
> On 2/12/07, Serge E. Hallyn <serue@us.ibm.com> wrote:
> >--- a/kernel/container.c
> >+++ b/kernel/container.c
> >@@ -1581,6 +1581,14 @@ static void get_unused_name(char *buf) {
> > sprintf(buf, "node%d", atomic_inc_return(&namecnt));
> > }
> >
> >+/* XXX need to create include/linux/ns_container.h and move this there */
> >+#ifdef CONFIG_CONTAINER_NS
> >+void ns_set_nsproxy_from_task(struct container *cont, struct task_struct
> >*tsk);
> >+#else
> >+static inline void
> >+ns_set_nsproxy_from_task(struct container *cont, struct task_struct *tsk)
> >{ }
> >+#endif
> >+

>
> Can't this just go in include/linux/nsproxy.h ?

I didn't want nsproxy.h knowing about containers if it didn't have to.
I'd rather have ns_container.h knowing about nsproxy and not the other
way around.

> >
> >+ /* mark the auto-created container with the new namespace list */
> >+ ns_set_nsproxy_from_task(child, tsk);
> >+
>
> This should be in ns_container_clone() or some (possibly new)
> subsystem callback, rather than embedded in the generic containers
> code.

Would you complain if I change kernel/container.c:container_clone() to
return the container *, so that ns_container_clone() doesn't need to
deduce that from tsk? So something like the following (not even
compiled, just to show what I'd be doing):

```
diff --git a/kernel/container.c b/kernel/container.c
index 4608a12..8895137 100644
--- a/kernel/container.c
+++ b/kernel/container.c
@@ -1593,7 +1593,7 @@ ns_set_nsproxy_from_task(struct containe
  * container_clone - duplicate the current container and move this
  * task into the new child
  */
-int container_clone(struct task_struct *tsk, struct container_subsys *subsys,
+struct container *container_clone(struct task_struct *tsk, struct container_subsys *subsys,
     struct nsproxy *nsproxy)
 {
  struct dentry *dentry;
diff --git a/kernel/ns_container.c b/kernel/ns_container.c
index 921c600..4291b18 100644
--- a/kernel/ns_container.c
+++ b/kernel/ns_container.c
@@ -23,9 +23,22 @@ static inline struct nscont *container_n
        struct nscont, css);
 }

+static void ns_set_nsproxy_from_task(struct container *cont, struct task_struct *tsk)
+{
+ struct nscont *ns = container_nscont(cont);
+ ns->nsproxy = tsk->nsproxy;
+ get_nsproxy(ns->nsproxy);
```

```
+}
+
 int ns_container_clone(struct task_struct *tsk, struct nsproxy *nsproxy)
 {
- return container_clone(tsk, &ns_subsys, nsproxy);
+ struct container *c;
+
+ c = container_clone(tsk, &ns_subsys, nsproxy);
+ if (IS_ERR(c))
+  return ERR_PTR(c);
+ ns_set_nsproxy_from_task(c, tsk);
+ return 0;
 }

 /*
@@ -95,14 +108,6 @@ static void ns_create_nsproxy(struct nsc
  get_nsproxy(ns->nsproxy);
 }

-/* called from container_clone */
-void ns_set_nsproxy_from_task(struct container *cont, struct task_struct *tsk)
-{
- struct nscont *ns = container_nscont(cont);
- ns->nsproxy = tsk->nsproxy;
- get_nsproxy(ns->nsproxy);
-}
-
 void ns_attach(struct container_subsys *ss, struct container *cont,
       struct container *old_cont, struct task_struct *tsk)
 {
```

Subject: Re: [RFC PATCH 3/4] namespace containers: add nsproxy to nscont struct
Posted by Paul Menage on Tue, 13 Feb 2007 18:36:20 GMT
View Forum Message <> Reply to Message

On 2/13/07, Serge E. Hallyn <serue@us.ibm.com> wrote:
>
> Would you complain if I change kernel/container.c:container_clone() to
> return the container *, so that ns_container_clone() doesn't need to
> deduce that from tsk? So something like the following (not even
> compiled, just to show what I'd be doing):
>

No real objections, as long as there are no issues with locking, etc.
I don't think it's really needed though, since task_container() will
give you this anyway.

Paul