Subject: Re: [PATCH 0/6] containers: Generic Process Containers (V6)
Posted by serue on Wed, 03 Jan 2007 14:43:30 GMT
View Forum Message <> Reply to Message

From: Serge E. Hallyn <serue@us.ibm.com>
Subject: [RFC] [PATCH 1/1] container: define a namespace container subsystem

Here's a stab at a namespace container subsystem based on
Paul Menage's containers patch, just to experiment with
how semantics suit what we want.

A few things we'll want to address:

1. We'll want to be able to hook things like
   rmdir, so that we can rm -rf /containers/vserver1
   to kill all processes in that container and all
   child containers.

2. We need a semantic difference between attaching
   to a container, and being the first to join the
   container you just created.

3. We will want to be able to give the container
   attach function more info, so that we can ask to
   attach to just the network namespace, but none of
   the others, in the container we're attaching to.

I'm sure there'll be more, but that's a start...

Note that this is far more user-controled than my previous namespace
naming patch. In particular, ns actions - clone/unshare - do not
automatically create new containers. That may be for the best, so
I didn't try to move in that direction this time. However it may be desirable
to at least change the creation semantics such that (after a container create
request) an unshare or clone simultaneously creates the container and joins
the new process as the container's first process.

This version does not point to an nsproxy from the ns_container,
but it probably should, as a definitive way to pick an nsproxy to
attach to if a process wants to enter the container.

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

```
---
 init/Kconfig         |    9 ++++++
 kernel/Makefile      |    1 +
 kernel/ns_container.c |   74 ++++++++++++++++++++++++++++++++++++++++++++++++++++
 3 files changed, 84 insertions(+), 0 deletions(-)
```

```
diff --git a/init/Kconfig b/init/Kconfig
index ebaec57..c00b19c 100644
--- a/init/Kconfig
+++ b/init/Kconfig
@@ -297,6 +297,15 @@ config CONTAINER_CPUACCT
    Provides a simple Resource Controller for monitoring the
    total CPU consumed by the tasks in a container

+config CONTAINER_NS
+ bool "Namespace container subsystem"
+ select CONTAINERS
+ help
+   Provides a simple namespace container subsystem to
+   provide hierarchical naming of sets of namespaces,
+   for instance virtual servers and checkpoint/restart
+   jobs.
+
 config RELAY
  bool "Kernel->user space relay support (formerly relayfs)"
  help
diff --git a/kernel/Makefile b/kernel/Makefile
index feba860..6c73a5e 100644
--- a/kernel/Makefile
+++ b/kernel/Makefile
@@ -39,6 +39,7 @@ obj-$(CONFIG_COMPAT) += compat.o
 obj-$(CONFIG_CONTAINERS) += container.o
 obj-$(CONFIG_CPUSETS) += cpuset.o
 obj-$(CONFIG_CONTAINER_CPUACCT) += cpu_acct.o
+obj-$(CONFIG_CONTAINER_NS) += ns_container.o
 obj-$(CONFIG_IKCONFIG) += configs.o
 obj-$(CONFIG_STOP_MACHINE) += stop_machine.o
 obj-$(CONFIG_AUDIT) += audit.o auditfilter.o
diff --git a/kernel/ns_container.c b/kernel/ns_container.c
new file mode 100644
index 0000000..b122bb4
--- /dev/null
+++ b/kernel/ns_container.c
@@ -0,0 +1,74 @@
+/*
+ * ns_container.c - namespace container subsystem
+ *
+ * Copyright IBM, 2006
+ */
+
+#include <linux/module.h>
+#include <linux/container.h>
+#include <linux/fs.h>
+
```

```
+struct nscont {
+ struct container_subsys_state css;
+ spinlock_t lock;
+};
+
+static struct container_subsys ns_subsys;
+
+static inline struct nscont *container_nscont(struct container *cont)
+{
+ return container_of(container_subsys_state(cont, &ns_subsys),
+       struct nscont, css);
+}
+
+int ns_can_attach(struct container_subsys *ss,
+        struct container *cont, struct task_struct *tsk)
+{
+ struct container *c;
+
+ if (atomic_read(&cont->count) != 0)
+  return -EPERM;
+
+ c = task_container(tsk, &ns_subsys);
+ if (c && c != cont->parent)
+  return -EPERM;
+ printk(KERN_NOTICE "%s: task %lu in container %s, attaching to %s, parent %s\n",
+  __FUNCTION__, tsk->pid, c ? c->dentry->d_name.name : "(null)",
+  cont->dentry->d_name.name, cont->parent->dentry->d_name.name);
+
+ return 0;
+}
+static int ns_create(struct container_subsys *ss, struct container *cont)
+{
+ struct nscont *ns = kzalloc(sizeof(*ns), GFP_KERNEL);
+ if (!ns) return -ENOMEM;
+ spin_lock_init(&ns->lock);
+ cont->subsys[ns_subsys.subsys_id] = &ns->css;
+ return 0;
+}
+
+static void ns_destroy(struct container_subsys *ss,
+       struct container *cont)
+{
+ struct nscont *ns = container_nscont(cont);
+ kfree(ns);
+}
+
+static struct container_subsys ns_subsys = {
+ .name = "ns_container",
```

```
+ .create = ns_create,
+ .destroy  = ns_destroy,
+ .can_attach = ns_can_attach,
+ //.attach = ns_attach,
+ //.post_attach = ns_post_attach,
+ //.populate = ns_populate,
+ .subsys_id = -1,
+};
+
+int __init ns_init(void)
+{
+ int id = container_register_subsys(&ns_subsys);
+ return id < 0 ? id : 0;
+}
+
+module_init(ns_init)
--
1.4.1
```