Subject: [PATCH 2/2][RFC] containers interface to nsproxy subsystem
Posted by serue on Fri, 01 Jun 2007 21:49:43 GMT
View Forum Message <> Reply to Message

Again, not compiles and boots, but not sufficiently tested for
inclusion.

thanks,
-serge

>From 6c03518dbda04bf20a9ddeefb291bf1c63e32ec1 Mon Sep 17 00:00:00 2001
From: Serge E. Hallyn <serue@us.ibm.com>
Date: Fri, 1 Jun 2007 17:30:16 -0400
Subject: [PATCH 2/2] Container interface to nsproxy subsystem

When a task enters a new namespace via a clone() or unshare(), a new
container is created and the task moves into it.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
---
 include/linux/container_subsys.h | 6 ++
 include/linux/nsproxy.h | 7 +++
 init/Kconfig | 9 ++++
 kernel/Makefile | 1 +
 kernel/container.c | 15 +++++-
 kernel/ns_container.c | 96 ++++++++++++++++++++++++++++++++++++++
 kernel/nsproxy.c | 16 ++++++
 7 files changed, 148 insertions(+), 2 deletions(-)
 create mode 100644 kernel/ns_container.c

diff --git a/include/linux/container_subsys.h b/include/linux/container_subsys.h
index 8fea7cf..9861751 100644
--- a/include/linux/container_subsys.h
+++ b/include/linux/container_subsys.h
@@ -24,3 +24,9 @@ SUBSYS(debug)
 #endif

 /* */
+
+#ifdef CONFIG_CONTAINER_NS
+SUBSYS(ns)
+#endif
+
+/* */
diff --git a/include/linux/nsproxy.h b/include/linux/nsproxy.h
index 189e0dc..8be975b 100644
--- a/include/linux/nsproxy.h
+++ b/include/linux/nsproxy.h

```
@@ -54,4 +54,11 @@ static inline void exit_task_namespaces(struct task_struct *p)
   put_nsproxy(ns);
  }
 }
+
+#ifdef CONFIG_CONTAINER_NS
+int ns_container_clone(struct task_struct *tsk);
+#else
+static inline int ns_container_clone(struct task_struct *tsk) { return 0; }
+#endif
+
 #endif
diff --git a/init/Kconfig b/init/Kconfig
index 5861ad9..d79c505 100644
--- a/init/Kconfig
+++ b/init/Kconfig
@@ -355,6 +355,15 @@ 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 PROC_PID_CPUSET
  bool "Include legacy /proc/<pid>/cpuset file"
  depends on CPUSETS
diff --git a/kernel/Makefile b/kernel/Makefile
index f73b3d3..34f2345 100644
--- a/kernel/Makefile
+++ b/kernel/Makefile
@@ -40,6 +40,7 @@ obj-$(CONFIG_CONTAINERS) += container.o
 obj-$(CONFIG_CONTAINER_DEBUG) += container_debug.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/container.c b/kernel/container.c
index 2b2f6f2..ac5879f 100644
--- a/kernel/container.c
+++ b/kernel/container.c
```

```
@@ -2523,14 +2523,25 @@ int container_clone(struct task_struct *tsk, struct container_subsys
*subsys)
  return ret;
 }

-/* See if "cont" is a descendant of the current task's container in
- * the appropriate hierarchy */
+/*
+ * See if "cont" is a descendant of the current task's container in
+ * the appropriate hierarchy
+ *
+ * If we are sending in dummytop, then presumably we are creating
+ * the top container in the subsystem.
+ *
+ * Called only by the ns (nsproxy) container.
+ */

 int container_is_descendant(const struct container *cont)
 {
  int ret;
  struct container *target;
  int subsys_id;
+
+ if (cont == dummytop)
+  return 1;
+
  get_first_subsys(cont, NULL, &subsys_id);
  target = task_container(current, subsys_id);
  while (cont != target && cont!= cont->top_container) {
diff --git a/kernel/ns_container.c b/kernel/ns_container.c
new file mode 100644
index 0000000..afa50d9
--- /dev/null
+++ b/kernel/ns_container.c
@@ -0,0 +1,96 @@
+/*
+ * ns_container.c - namespace container subsystem
+ *
+ * Copyright 2006, 2007 IBM Corp
+ */
+
+#include <linux/module.h>
+#include <linux/container.h>
+#include <linux/fs.h>
+
+struct nscont {
+ struct container_subsys_state css;
+ spinlock_t lock;
```

```
+};
+
+struct container_subsys ns_subsys;
+
+static inline struct nscont *container_nscont(struct container *cont)
+{
+ return container_of(container_subsys_state(cont, ns_subsys_id),
+       struct nscont, css);
+}
+
+int ns_container_clone(struct task_struct *tsk)
+{
+ return container_clone(tsk, &ns_subsys);
+}
+
+/*
+ * Rules:
+ *   1. you can only enter a container which is a child of your current
+ *      container
+ *   2. you can only place another process into a container if
+ *      a. you have CAP_SYS_ADMIN
+ *      b. your container is an ancestor of tsk's destination container
+ *        (hence either you are in the same container as tsk, or in an
+ *         ancestor container thereof)
+ */
+int ns_can_attach(struct container_subsys *ss,
+     struct container *cont, struct task_struct *tsk)
+{
+ struct container *c;
+
+ if (current != tsk) {
+  if (!capable(CAP_SYS_ADMIN))
+   return -EPERM;
+
+  if (!container_is_descendant(cont))
+   return -EPERM;
+ }
+
+ if (atomic_read(&cont->count) != 0)
+  return -EPERM;
+
+ c = task_container(tsk, ns_subsys_id);
+ if (c && c != cont->parent)
+  return -EPERM;
+
+ return 0;
+}
+
```

```
+/*
+ * 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;
+ if (!container_is_descendant(cont))
+  return -EPERM;
+
+ 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);
+}
+
+struct container_subsys ns_subsys = {
+ .name = "ns",
+ .create = ns_create,
+ .destroy  = ns_destroy,
+ .can_attach = ns_can_attach,
+ .subsys_id = ns_subsys_id,
+};
diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
index 1bc4b55..afce808 100644
--- a/kernel/nsproxy.c
+++ b/kernel/nsproxy.c
@@ -124,7 +124,14 @@ int copy_namespaces(int flags, struct task_struct *tsk)
   goto out;
  }

+ err = ns_container_clone(tsk);
+ if (err) {
+  put_nsproxy(new_ns);
+  goto out;
+ }
```

```
+
  tsk->nsproxy = new_ns;
+
 out:
  put_nsproxy(old_ns);
  return err;
@@ -177,6 +184,15 @@ int unshare_nsproxy_namespaces(unsigned long unshare_flags,
  if (IS_ERR(*new_nsp)) {
   err = PTR_ERR(*new_nsp);
   put_nsproxy(old_ns);
+  goto out;
+ }
+
+ err = ns_container_clone(current);
+ if (err) {
+  put_nsproxy(*new_nsp);
+  put_nsproxy(old_ns);
  }
+
+out:
  return err;
 }
--
1.5.1.1.GIT
```