
Subject: [PATCH 7/7] containers (V7): Container interface to nsproxy subsystem
Posted by [Paul Menage](#) on Mon, 12 Feb 2007 08:15:28 GMT
[View Forum Message](#) <> [Reply to Message](#)

When a task enters a new namespace via a clone() or unshare(), a new container is created and the task moves into it. Developed by Serge Hallyn <serue@us.ibm.com>, adapted by Paul Menage <menage@google.com>

```
---
include/linux/nsproxy.h | 6 ++
init/Kconfig            | 9 +++
kernel/Makefile         | 1
kernel/fork.c           | 4 +
kernel/ns_container.c   | 110 +++++
kernel/nsproxy.c        | 6 ++
6 files changed, 136 insertions(+)
```

Index: container-2.6.20/include/linux/nsproxy.h

```
=====
--- container-2.6.20.orig/include/linux/nsproxy.h
+++ container-2.6.20/include/linux/nsproxy.h
@@ -53,4 +53,10 @@ static inline void exit_task_namespaces(
    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
```

Index: container-2.6.20/init/Kconfig

```
=====
--- container-2.6.20.orig/init/Kconfig
+++ container-2.6.20/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
Index: container-2.6.20/kernel/Makefile
=====
--- container-2.6.20.orig/kernel/Makefile
+++ container-2.6.20/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
Index: container-2.6.20/kernel/fork.c
=====
--- container-2.6.20.orig/kernel/fork.c
+++ container-2.6.20/kernel/fork.c
@@ -1661,6 +1661,9 @@ asmlinkage long sys_unshare(unsigned lon
    err = -ENOMEM;
    goto bad_unshare_cleanup_ipc;
  }
+ err = ns_container_clone(current);
+ if (err)
+ goto bad_unshare_cleanup_dupns;
  }

  if (new_fs || new_ns || new_mm || new_fd || new_ulist ||
@@ -1715,6 +1718,7 @@ asmlinkage long sys_unshare(unsigned lon
    task_unlock(current);
  }

+ bad_unshare_cleanup_dupns:
+ if (new_nsproxy)
+ put_nsproxy(new_nsproxy);

Index: container-2.6.20/kernel/ns_container.c
=====
--- /dev/null
+++ container-2.6.20/kernel/ns_container.c
@@ -0,0 +1,110 @@
+/*
+ * 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_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);
+ 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);
+}
+
+static struct container_subsys ns_subsys = {
+ .name = "ns",
+ .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 ret;
+
+ ret = container_register_subsys(&ns_subsys);
+
+ return ret < 0 ? ret : 0;
+}
+
+module_init(ns_init)
Index: container-2.6.20/kernel/nsproxy.c
=====
--- container-2.6.20.orig/kernel/nsproxy.c
+++ container-2.6.20/kernel/nsproxy.c
@@ -116,10 +116,16 @@ int copy_namespaces(int flags, struct ta
 if (err)
    goto out_pid;

+ err = ns_container_clone(tsk);
+ if (err)
+ goto out_container;
out:
    put_nsproxy(old_ns);
    return err;

+ out_container:
+ if (new_ns->pid_ns)
+ put_pid_ns(new_ns->pid_ns);
out_pid:
    if (new_ns->ipc_ns)
        put_ipc_ns(new_ns->ipc_ns);

--

```
