Subject: [PATCH 18/33] containers implement namespace tracking subsystem
Posted by Paul Menage on Mon, 17 Sep 2007 21:03:25 GMT
View Forum Message <> Reply to Message

From: "Serge E. Hallyn" <serue@us.ibm.com>
(container->cgroup renaming by Paul Menage <menage@google.com>)

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

This version names cgroups which are automatically created using cgroup_clone() as "node_<pid>" where pid is the pid of the unsharing or cloned process. (Thanks Pavel for the idea) This is safe because if the process unshares again, it will create

/cgroups/(...)/node_<pid>/node_<pid>

The only possibilities (AFAICT) for a -EEXIST on unshare are

1. pid wraparound
2. a process fails an unshare, then tries again.

Case 1 is unlikely enough that I ignore it (at least for now). In case 2, the node_<pid> will be empty and can be rmdir'ed to make the subsequent unshare() succeed.

Changelog:
Name cloned cgroups as "node_<pid>".

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

---

include/linux/cgroup_subsys.h | 6 +
include/linux/nsproxy.h | 7 ++
init/Kconfig | 9 ++
kernel/Makefile | 1
kernel/ns_cgroup.c | 100 ++++++++++++++++++++++++++++++++++++++
kernel/nsproxy.c | 17 ++++
6 files changed, 139 insertions(+), 1 deletion(-)

diff -puN include/linux/cgroup_subsys.h~cgroups-implement-namespace-tracking-subsystem include/linux/cgroup_subsys.h
--- a/include/linux/cgroup_subsys.h~cgroups-implement-namespace-tracking-subsystem
+++ a/include/linux/cgroup_subsys.h
@@ -24,3 +24,9 @@ SUBSYS(debug)
#endif

```
 /* */
+
+#ifdef CONFIG_CGROUP_NS
+SUBSYS(ns)
+#endif
+
+/* */
diff -puN include/linux/nsproxy.h~cgroups-implement-namespace-tracking-subsystem
include/linux/nsproxy.h
--- a/include/linux/nsproxy.h~cgroups-implement-namespace-tracking-subsystem
+++ a/include/linux/nsproxy.h
@@ -55,4 +55,11 @@ static inline void exit_task_namespaces(
   put_nsproxy(ns);
  }
 }
+
+#ifdef CONFIG_CGROUP_NS
+int ns_cgroup_clone(struct task_struct *tsk);
+#else
+static inline int ns_cgroup_clone(struct task_struct *tsk) { return 0; }
+#endif
+
 #endif
diff -puN init/Kconfig~cgroups-implement-namespace-tracking-subsystem init/Kconfig
--- a/init/Kconfig~cgroups-implement-namespace-tracking-subsystem
+++ a/init/Kconfig
@@ -323,6 +323,15 @@ config SYSFS_DEPRECATED
    If you are using a distro that was released in 2006 or later,
    it should be safe to say N here.

+config CGROUP_NS
+       bool "Namespace cgroup subsystem"
+       select CGROUPS
+       help
+         Provides a simple namespace cgroup 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 -puN kernel/Makefile~cgroups-implement-namespace-tracking-subsystem kernel/Makefile
--- a/kernel/Makefile~cgroups-implement-namespace-tracking-subsystem
+++ a/kernel/Makefile
@@ -42,6 +42,7 @@ obj-$(CONFIG_CGROUPS) += cgroup.o
 obj-$(CONFIG_CGROUP_DEBUG) += cgroup_debug.o
```

```
 obj-$(CONFIG_CPUSETS) += cpuset.o
 obj-$(CONFIG_CGROUP_CPUACCT) += cpu_acct.o
+obj-$(CONFIG_CGROUP_NS) += ns_cgroup.o
 obj-$(CONFIG_IKCONFIG) += configs.o
 obj-$(CONFIG_STOP_MACHINE) += stop_machine.o
 obj-$(CONFIG_AUDIT) += audit.o auditfilter.o
diff -puN /dev/null kernel/ns_cgroup.c
--- /dev/null
+++ a/kernel/ns_cgroup.c
@@ -0,0 +1,100 @@
+/*
+ * ns_cgroup.c - namespace cgroup subsystem
+ *
+ * Copyright 2006, 2007 IBM Corp
+ */
+
+#include <linux/module.h>
+#include <linux/cgroup.h>
+#include <linux/fs.h>
+
+struct ns_cgroup {
+ struct cgroup_subsys_state css;
+ spinlock_t lock;
+};
+
+struct cgroup_subsys ns_subsys;
+
+static inline struct ns_cgroup *cgroup_to_ns(
+  struct cgroup *cgroup)
+{
+ return container_of(cgroup_subsys_state(cgroup, ns_subsys_id),
+       struct ns_cgroup, css);
+}
+
+int ns_cgroup_clone(struct task_struct *task)
+{
+ return cgroup_clone(task, &ns_subsys);
+}
+
+/*
+ * Rules:
+ *   1. you can only enter a cgroup which is a child of your current
+ *      cgroup
+ *   2. you can only place another process into a cgroup if
+ *      a. you have CAP_SYS_ADMIN
+ *      b. your cgroup is an ancestor of task's destination cgroup
+ *        (hence either you are in the same cgroup as task, or in an
+ *         ancestor cgroup thereof)
```

```
+ */
+static int ns_can_attach(struct cgroup_subsys *ss,
+  struct cgroup *new_cgroup, struct task_struct *task)
+{
+ struct cgroup *orig;
+
+ if (current != task) {
+  if (!capable(CAP_SYS_ADMIN))
+   return -EPERM;
+
+  if (!cgroup_is_descendant(new_cgroup))
+   return -EPERM;
+ }
+
+ if (atomic_read(&new_cgroup->count) != 0)
+  return -EPERM;
+
+ orig = task_cgroup(task, ns_subsys_id);
+ if (orig && orig != new_cgroup->parent)
+  return -EPERM;
+
+ return 0;
+}
+
+/*
+ * Rules: you can only create a cgroup if
+ *     1. you are capable(CAP_SYS_ADMIN)
+ *     2. the target cgroup is a descendant of your own cgroup
+ */
+static struct cgroup_subsys_state *ns_create(struct cgroup_subsys *ss,
+      struct cgroup *cgroup)
+{
+ struct ns_cgroup *ns_cgroup;
+
+ if (!capable(CAP_SYS_ADMIN))
+  return ERR_PTR(-EPERM);
+ if (!cgroup_is_descendant(cgroup))
+  return ERR_PTR(-EPERM);
+
+ ns_cgroup = kzalloc(sizeof(*ns_cgroup), GFP_KERNEL);
+ if (!ns_cgroup)
+  return ERR_PTR(-ENOMEM);
+ spin_lock_init(&ns_cgroup->lock);
+ return &ns_cgroup->css;
+}
+
+static void ns_destroy(struct cgroup_subsys *ss,
+   struct cgroup *cgroup)
```

```
+{
+ struct ns_cgroup *ns_cgroup;
+
+ ns_cgroup = cgroup_to_ns(cgroup);
+ kfree(ns_cgroup);
+}
+
+struct cgroup_subsys ns_subsys = {
+ .name = "ns",
+ .can_attach = ns_can_attach,
+ .create = ns_create,
+ .destroy  = ns_destroy,
+ .subsys_id = ns_subsys_id,
+};
diff -puN kernel/nsproxy.c~cgroups-implement-namespace-tracking-subsystem kernel/nsproxy.c
--- a/kernel/nsproxy.c~cgroups-implement-namespace-tracking-subsystem
+++ a/kernel/nsproxy.c
@@ -146,7 +146,14 @@ int copy_namespaces(unsigned long flags,
   goto out;
  }

+ err = ns_cgroup_clone(tsk);
+ if (err) {
+  put_nsproxy(new_ns);
+  goto out;
+ }
+
  tsk->nsproxy = new_ns;
+
 out:
  put_nsproxy(old_ns);
  return err;
@@ -185,8 +192,16 @@ int unshare_nsproxy_namespaces(unsigned

  *new_nsp = create_new_namespaces(unshare_flags, current,
     new_fs ? new_fs : current->fs);
- if (IS_ERR(*new_nsp))
+ if (IS_ERR(*new_nsp)) {
   err = PTR_ERR(*new_nsp);
+  goto out;
+ }
+
+ err = ns_cgroup_clone(current);
+ if (err)
+  put_nsproxy(*new_nsp);
+
+out:
  return err;
```

}

—

--

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers