Subject: [RFC] [PATCH 1/3] container: implement 'containers' as a namespace naming device
Posted by serue on Wed, 20 Dec 2006 06:02:02 GMT
View Forum Message <> Reply to Message

From: Serge E. Hallyn <serue@us.ibm.com>
Subject: [RFC] [PATCH 1/3] container: implement 'containers' as a namespace naming device

Implement containers and their reference counting, as the device
for providing simple, hierarchical naming of namespaces.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
---
 include/linux/container.h | 24 ++++++++++
 include/linux/init_task.h | 13 +++++
 include/linux/nsproxy.h | 2 +
 kernel/Makefile | 2 -
 kernel/container.c | 108 ++
 kernel/nsproxy.c | 11 ++++-
 6 files changed, 158 insertions(+), 2 deletions(-)

diff --git a/include/linux/container.h b/include/linux/container.h
new file mode 100644
index 0000000..fcd85f3
--- /dev/null
+++ b/include/linux/container.h
@@ -0,0 +1,24 @@
+#ifndef _LINUX_CONTAINER_H
+#define _LINUX_CONTAINER_H
+
+#include <linux/spinlock.h>
+#include <linux/list.h>
+#include <linux/kref.h>
+
+struct nsproxy;
+
+struct container {
+ struct container *parent;
+ char *name;
+ struct nsproxy *nsproxy;
+ struct list_head children;
+ struct list_head peers;
+ struct kref ref;
+};
+extern struct container init_container;
+
+void put_container(struct container *c);
+struct container *new_container(struct container *parent,

```
+     struct nsproxy *nsproxy);
+
+#endif
diff --git a/include/linux/init_task.h b/include/linux/init_task.h
index a2d95ff..445a556 100644
--- a/include/linux/init_task.h
+++ b/include/linux/init_task.h
@@ -70,6 +70,18 @@ #define INIT_SIGNALS(sig) {      \
  { .__session      = 1},      \
 }

+/* presumably the init container name will come from .config */
+#define INIT_CONTAINER_NAME "init_container"
+extern struct container init_container;
+#define INIT_CONTAINER(container) {     \
+ .parent  = &init_container,    \
+ .name  = INIT_CONTAINER_NAME,    \
+ .nsproxy = &init_nsproxy,    \
+ .children = LIST_HEAD_INIT(container.children),  \
+ .peers  = LIST_HEAD_INIT(container.peers),  \
+ .ref  = {.refcount = ATOMIC_INIT(1)},   \
+}
+
 extern struct nsproxy init_nsproxy;
 #define INIT_NSPROXY(nsproxy) {      \
  .pid_ns  = &init_pid_ns,     \
@@ -77,6 +89,7 @@ #define INIT_NSPROXY(nsproxy) {      \
  .nslock  = __SPIN_LOCK_UNLOCKED(nsproxy.nslock),  \
  .uts_ns  = &init_uts_ns,     \
  .mnt_ns  = NULL,      \
+ .container = &init_container,    \
  INIT_IPC_NS(ipc_ns)      \
 }

diff --git a/include/linux/nsproxy.h b/include/linux/nsproxy.h
index 0b9f0dc..30c9876 100644
--- a/include/linux/nsproxy.h
+++ b/include/linux/nsproxy.h
@@ -8,6 +8,7 @@ struct mnt_namespace;
 struct uts_namespace;
 struct ipc_namespace;
 struct pid_namespace;
+struct container;

 /*
  * A structure to contain pointers to all per-process
@@ -28,6 +29,7 @@ struct nsproxy {
  struct ipc_namespace *ipc_ns;
```

```
  struct mnt_namespace *mnt_ns;
  struct pid_namespace *pid_ns;
+ struct container *container;
 };
 extern struct nsproxy init_nsproxy;

diff --git a/kernel/Makefile b/kernel/Makefile
index 839a58b..e5d82c1 100644
--- a/kernel/Makefile
+++ b/kernel/Makefile
@@ -8,7 +8,7 @@ obj-y     = sched.o fork.o exec_domain.o
       signal.o sys.o kmod.o workqueue.o pid.o \
       rcupdate.o extable.o params.o posix-timers.o \
       kthread.o wait.o kfifo.o sys_ni.o posix-cpu-timers.o mutex.o \
-      hrtimer.o rwsem.o latency.o nsproxy.o srcu.o
+      hrtimer.o rwsem.o latency.o nsproxy.o srcu.o container.o

 obj-$(CONFIG_STACKTRACE) += stacktrace.o
 obj-y += time/
diff --git a/kernel/container.c b/kernel/container.c
new file mode 100644
index 0000000..ed3269f
--- /dev/null
+++ b/kernel/container.c
@@ -0,0 +1,108 @@
+/*
+ *  Copyright (C) 2006 IBM Corporation
+ *
+ *  Author: Serge Hallyn <serue@us.ibm.com>
+ *
+ *  This program is free software; you can redistribute it and/or
+ *  modify it under the terms of the GNU General Public License as
+ *  published by the Free Software Foundation, version 2 of the
+ *  License.
+ *
+ *  Jun 2006 - namespaces support
+ *             OpenVZ, SWsoft Inc.
+ *             Pavel Emelianov <xemul@openvz.org>
+ */
+
+#include <linux/module.h>
+#include <linux/version.h>
+#include <linux/container.h>
+#include <linux/init_task.h>
+
+struct nsproxy;
+
+struct container init_container = INIT_CONTAINER(init_container);
```

```
+
+/*
+ * free_container: called from rcu_call when all references
+ * are gone
+ * all references won't be gone until all children are already
+ * freed.
+ */
+static void free_container(struct kref *ref)
+{
+ struct container *c = container_of(ref, struct container, ref);
+
+ if (c->parent != c)
+  kfree(c->name);
+ if (!list_empty(&c->peers))
+  list_del(&c->peers);
+ kfree(c);
+}
+
+
+/*
+ * get a container reference
+ * we also grab a reference to all it's parents
+ */
+struct container *get_container(struct container *c)
+{
+ struct container *c2 = c;
+
+ if (c) {
+  kref_get(&c2->ref);
+  while (c2->parent != c2) {
+   c2 = c2->parent;
+   kref_get(&c2->ref);
+  }
+ }
+ return c;
+}
+
+/*
+ * put a container
+ * when we put a container, we also put all it's parents.
+ */
+void put_container(struct container *c)
+{
+ struct container *parent;
+ if (!c)
+  return;
+ parent = c->parent;
+ while (parent != c) {
```

```
+  kref_put(&c->ref, free_container);
+  c = parent;
+  parent = parent->parent;
+ }
+ kref_put(&c->ref, free_container);
+}
+
+/* i expect this lock to be moved into the containers... */
+static DEFINE_SPINLOCK(container_lock);
+static int last_unnamed_container = 1;
+
+struct container *new_container(struct container *parent,
+      struct nsproxy *nsproxy)
+{
+ struct container *c;
+
+ c = kmalloc(sizeof(*c), GFP_KERNEL);
+ if (!c)
+  return NULL;
+ c->name = kzalloc(20, GFP_KERNEL);
+ if (!c->name) {
+  kfree(c);
+  return NULL;
+ }
+ spin_lock(&container_lock);
+ sprintf(c->name, "container_%d", last_unnamed_container++);
+ spin_unlock(&container_lock);
+ INIT_LIST_HEAD(&c->peers);
+ c->parent = parent;
+ get_container(parent);
+ list_add_tail(&c->peers, &parent->children);
+ INIT_LIST_HEAD(&c->children);
+ c->nsproxy = nsproxy;
+ kref_init(&c->ref);
+
+ return c;
+}
diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
index f5b9ee6..0cfa4c4 100644
--- a/kernel/nsproxy.c
+++ b/kernel/nsproxy.c
@@ -20,6 +20,7 @@ #include <linux/init_task.h>
 #include <linux/mnt_namespace.h>
 #include <linux/utsname.h>
 #include <linux/pid_namespace.h>
+#include <linux/container.h>

 struct nsproxy init_nsproxy = INIT_NSPROXY(init_nsproxy);
```

```
@@ -46,8 +47,14 @@ static inline struct nsproxy *clone_name
  struct nsproxy *ns;

  ns = kmemdup(orig, sizeof(struct nsproxy), GFP_KERNEL);
- if (ns)
+ if (ns) {
   atomic_set(&ns->count, 1);
+  ns->container = new_container(orig->container, ns);
+  if (!ns->container) {
+   kfree(ns);
+   ns = NULL;
+  }
+ }
  return ns;
 }

@@ -145,5 +152,7 @@ void free_nsproxy(struct nsproxy *ns)
   put_ipc_ns(ns->ipc_ns);
  if (ns->pid_ns)
   put_pid_ns(ns->pid_ns);
+ put_container(ns->container);
+ ns->container->nsproxy = NULL;
  kfree(ns);
 }
--
1.4.1
```

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers