Subject: [RFC][PATCH 1/5] uts namespaces: Implement utsname namespaces
Posted by serue on Fri, 07 Apr 2006 18:36:00 GMT
View Forum Message <> Reply to Message

This patch defines the uts namespace and some manipulators.
Adds the uts namespace to task_struct, and initializes a
system-wide init namespace which will continue to be used when
it makes sense.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
---
include/linux/init_task.h | 2 +
include/linux/sched.h | 2 +
include/linux/utsname.h | 40 ++++++++++++++++++++++++++++++++++++++-
init/Kconfig | 8 +++++
init/version.c | 70 ++-----
kernel/exit.c | 2 +
kernel/fork.c | 9 +++++-
7 files changed, 122 insertions(+), 11 deletions(-)

14c326d603d88d9ed40a1ddafbf23fc3da68a645
diff --git a/include/linux/init_task.h b/include/linux/init_task.h
index 41ecbb8..21b1751 100644
--- a/include/linux/init_task.h
+++ b/include/linux/init_task.h
@@ -3,6 +3,7 @@

#include <linux/file.h>
#include <linux/rcupdate.h>
+#include <linux/utsname.h>

#define INIT_FDTABLE \
{ \
@@ -123,6 +124,7 @@ extern struct group_info init_groups;
.journal_info = NULL, \
.cpu_timers = INIT_CPU_TIMERS(tsk.cpu_timers), \
.fs_excl = ATOMIC_INIT(0), \
+ .uts_ns = &init_uts_ns, \
}

diff --git a/include/linux/sched.h b/include/linux/sched.h
index 541f482..97c7990 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -684,6 +684,7 @@ static inline void prefetch_stack(struct

struct audit_context; /* See audit.c */

```
 struct mempolicy;
+struct uts_namespace;

 enum sleep_type {
  SLEEP_NORMAL,
@@ -807,6 +808,7 @@ struct task_struct {
  struct files_struct *files;
 /* namespace */
  struct namespace *namespace;
+ struct uts_namespace *uts_ns;
 /* signal handlers */
  struct signal_struct *signal;
  struct sighand_struct *sighand;
diff --git a/include/linux/utsname.h b/include/linux/utsname.h
index 13e1da0..cc28ac5 100644
--- a/include/linux/utsname.h
+++ b/include/linux/utsname.h
@@ -1,5 +1,8 @@
 #ifndef _LINUX_UTSNAME_H
 #define _LINUX_UTSNAME_H
+#include <linux/sched.h>
+#include <linux/kref.h>
+#include <asm/atomic.h>

 #define __OLD_UTS_LEN 8

@@ -30,7 +33,42 @@ struct new_utsname {
  char domainname[65];
 };

-extern struct new_utsname system_utsname;
+struct uts_namespace {
+ struct kref kref;
+ struct new_utsname name;
+};
+extern struct uts_namespace init_uts_ns;
+
+#ifdef CONFIG_UTS_NS
+
+extern struct uts_namespace *clone_uts_ns(struct uts_namespace *old_ns);
+extern struct uts_namespace *unshare_uts_ns(void);
+extern void free_uts_ns(struct kref *kref);
+
+static inline void get_uts_ns(struct uts_namespace *ns)
+{
+ kref_get(&ns->kref);
+}
+
```

```
+static inline void put_uts_ns(struct uts_namespace *ns)
+{
+ kref_put(&ns->kref, free_uts_ns);
+}
+
+#else
+static inline void get_uts_ns(struct uts_namespace *ns)
+{
+}
+static inline void put_uts_ns(struct uts_namespace *ns)
+{
+}
+#endif
+
+static inline struct new_utsname *utsname(void)
+{
+ return &current->uts_ns->name;
+}
+

 extern struct rw_semaphore uts_sem;
 #endif
diff --git a/init/Kconfig b/init/Kconfig
index 3b36a1d..8460e5a 100644
--- a/init/Kconfig
+++ b/init/Kconfig
@@ -166,6 +166,14 @@ config SYSCTL
    building a kernel for install/rescue disks or your system is very
    limited in memory.

+config UTS_NS
+ bool "UTS Namespaces"
+ default n
+ help
+   Support uts namespaces.  This allows containers, i.e.
+   vservers, to use uts namespaces to provide different
+   uts info for different servers.  If unsure, say N.
+
 config AUDIT
  bool "Auditing support"
  depends on NET
diff --git a/init/version.c b/init/version.c
index 3ddc3ce..e128d72 100644
--- a/init/version.c
+++ b/init/version.c
@@ -11,22 +11,76 @@
 #include <linux/uts.h>
 #include <linux/utsname.h>
```

```
 #include <linux/version.h>
+#include <linux/sched.h>

 #define version(a) Version_ ## a
 #define version_string(a) version(a)

 int version_string(LINUX_VERSION_CODE);

-struct new_utsname system_utsname = {
- .sysname = UTS_SYSNAME,
- .nodename = UTS_NODENAME,
- .release = UTS_RELEASE,
- .version = UTS_VERSION,
- .machine = UTS_MACHINE,
- .domainname = UTS_DOMAINNAME,
+struct uts_namespace init_uts_ns = {
+ .kref = {
+  .refcount = ATOMIC_INIT(2),
+ },
+ .name = {
+  .sysname = UTS_SYSNAME,
+  .nodename = UTS_NODENAME,
+  .release = UTS_RELEASE,
+  .version = UTS_VERSION,
+  .machine = UTS_MACHINE,
+  .domainname = UTS_DOMAINNAME,
+ },
 };

-EXPORT_SYMBOL(system_utsname);
+#ifdef CONFIG_UTS_NS
+/*
+ * Clone a new ns copying an original utsname, setting refcount to 1
+ * @old_ns: namespace to clone
+ * Return NULL on error (failure to kmalloc), new ns otherwise
+ */
+struct uts_namespace *clone_uts_ns(struct uts_namespace *old_ns)
+{
+ struct uts_namespace *ns;
+
+ ns = kmalloc(sizeof(struct uts_namespace), GFP_KERNEL);
+ if (ns) {
+  memcpy(&ns->name, &old_ns->name, sizeof(ns->name));
+  kref_init(&ns->kref);
+ }
+ return ns;
+}
+
```

```
+/*
+ * unshare the current process' utsname namespace.  Changes
+ * to the utsname of this process won't be seen by parent, and
+ * vice versa
+ *
+ * Return NULL on error (failure to kmalloc), new ns otherwise
+ *
+ * TODO: decide where this should be locked  (depends on how/where
+ * we decide to use this)
+ */
+struct uts_namespace *unshare_uts_ns(void)
+{
+ struct uts_namespace *old_ns = current->uts_ns;
+ struct uts_namespace *new_ns = clone_uts_ns(old_ns);
+ if (new_ns) {
+  current->uts_ns = new_ns;
+  put_uts_ns(old_ns);
+ }
+ return new_ns;
+}
+EXPORT_SYMBOL(unshare_uts_ns);
+
+void free_uts_ns(struct kref *kref)
+{
+ struct uts_namespace *ns;
+
+ ns = container_of(kref, struct uts_namespace, kref);
+ kfree(ns);
+}
+EXPORT_SYMBOL(free_uts_ns);
+#endif

 const char linux_banner[] =
  "Linux version " UTS_RELEASE " (" LINUX_COMPILE_BY "@"
diff --git a/kernel/exit.c b/kernel/exit.c
index 6c2eeb8..97c5405 100644
--- a/kernel/exit.c
+++ b/kernel/exit.c
@@ -34,6 +34,7 @@
 #include <linux/mutex.h>
 #include <linux/futex.h>
 #include <linux/compat.h>
+#include <linux/utsname.h>

 #include <asm/uaccess.h>
 #include <asm/unistd.h>
@@ -173,6 +174,7 @@ repeat:
  spin_unlock(&p->proc_lock);
```

```
  proc_pid_flush(proc_dentry);
  release_thread(p);
+ put_uts_ns(p->uts_ns);
  call_rcu(&p->rcu, delayed_put_task_struct);

  p = leader;
diff --git a/kernel/fork.c b/kernel/fork.c
index 3384eb8..62e4479 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -44,6 +44,7 @@
 #include <linux/rmap.h>
 #include <linux/acct.h>
 #include <linux/cn_proc.h>
+#include <linux/utsname.h>

 #include <asm/pgtable.h>
 #include <asm/pgalloc.h>
@@ -1119,6 +1120,8 @@ static task_t *copy_process(unsigned lon
  /* Perform scheduler related setup. Assign this task to a CPU. */
  sched_fork(p, clone_flags);

+ get_uts_ns(p->uts_ns);
+
  /* Need tasklist lock for parent etc handling! */
  write_lock_irq(&tasklist_lock);

@@ -1158,7 +1161,7 @@ static task_t *copy_process(unsigned lon
   spin_unlock(&current->sighand->siglock);
   write_unlock_irq(&tasklist_lock);
   retval = -ERESTARTNOINTR;
-  goto bad_fork_cleanup_namespace;
+  goto bad_fork_cleanup_utsns;
  }

  if (clone_flags & CLONE_THREAD) {
@@ -1171,7 +1174,7 @@ static task_t *copy_process(unsigned lon
    spin_unlock(&current->sighand->siglock);
    write_unlock_irq(&tasklist_lock);
    retval = -EAGAIN;
-   goto bad_fork_cleanup_namespace;
+   goto bad_fork_cleanup_utsns;
   }

   p->group_leader = current->group_leader;
@@ -1223,6 +1226,8 @@ static task_t *copy_process(unsigned lon
  proc_fork_connector(p);
  return p;
```

```
+bad_fork_cleanup_utsns:
+ put_uts_ns(p->uts_ns);
 bad_fork_cleanup_namespace:
  exit_namespace(p);
 bad_fork_cleanup_keys:
--
1.2.4
```