Subject: [PATCH 6/9] namespaces: utsname: implement utsname namespaces
Posted by serue on Thu, 18 May 2006 15:50:14 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.

It leaves a #define for system_utsname so sysctl will compile.
This define will be removed in a separate patch.

Signed-off-by: Serge Hallyn <serue@us.ibm.com>

```
 include/linux/init_task.h |    2 ++
 include/linux/nsproxy.h   |    2 ++
 include/linux/sched.h     |    1 +
 include/linux/utsname.h   |   52 ++++++++++++++++++++++++++++++++++++++++++++++++---
 init/Kconfig              |    8 +++++++
 init/version.c            |   22 ++++++++++++++--------
 kernel/Makefile           |    1 +
 kernel/nsproxy.c          |   14 ++++++++++++++
 kernel/utsname.c          |   43 +++++++++++++++++++++++++++++++++++++++++++
 9 files changed, 133 insertions(+), 12 deletions(-)
 create mode 100644 kernel/utsname.c

d590372d659cf4bb3676f8ee7a509e173685b6ee
diff --git a/include/linux/init_task.h b/include/linux/init_task.h
index 672dc04..ceb68b7 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 \
 {       \
@@ -70,6 +71,7 @@ extern struct nsproxy init_nsproxy;
 #define INIT_NSPROXY(nsproxy) {      \
   .count  = ATOMIC_INIT(1),    \
   .nslock  = SPIN_LOCK_UNLOCKED,    \
+  .uts_ns  = &init_uts_ns,     \
   .namespace = NULL,      \
 }
```

```
diff --git a/include/linux/nsproxy.h b/include/linux/nsproxy.h
index 7ebe666..9c2e0ad 100644
--- a/include/linux/nsproxy.h
+++ b/include/linux/nsproxy.h
@@ -5,6 +5,7 @@
 #include <linux/sched.h>

 struct namespace;
+struct uts_namespace;

 /*
  * A structure to contain pointers to all per-process
@@ -21,6 +22,7 @@ struct namespace;
 struct nsproxy {
  atomic_t count;
  spinlock_t nslock;
+ struct uts_namespace *uts_ns;
  struct namespace *namespace;
 };
 extern struct nsproxy init_nsproxy;
diff --git a/include/linux/sched.h b/include/linux/sched.h
index f2c945b..3332d5e 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -685,6 +685,7 @@ static inline void prefetch_stack(struct
 struct audit_context;  /* See audit.c */
 struct mempolicy;
 struct pipe_inode_info;
+struct uts_namespace;

 enum sleep_type {
  SLEEP_NORMAL,
diff --git a/include/linux/utsname.h b/include/linux/utsname.h
index 77e97a5..e6120e7 100644
--- a/include/linux/utsname.h
+++ b/include/linux/utsname.h
@@ -1,6 +1,11 @@
 #ifndef _LINUX_UTSNAME_H
 #define _LINUX_UTSNAME_H

+#include <linux/sched.h>
+#include <linux/kref.h>
+#include <linux/nsproxy.h>
+#include <asm/atomic.h>
+
 #define __OLD_UTS_LEN 8

 struct oldold_utsname {
```

```
@@ -30,17 +35,58 @@ 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;
+
+static inline void get_uts_ns(struct uts_namespace *ns)
+{
+ kref_get(&ns->kref);
+}
+
+#ifdef CONFIG_UTS_NS
+extern int copy_utsname(int flags, struct task_struct *tsk);
+extern void free_uts_ns(struct kref *kref);
+
+static inline void put_uts_ns(struct uts_namespace *ns)
+{
+ kref_put(&ns->kref, free_uts_ns);
+}
+
+static inline void exit_utsname(struct task_struct *p)
+{
+ struct uts_namespace *uts_ns = p->nsproxy->uts_ns;
+ if (uts_ns) {
+  put_uts_ns(uts_ns);
+ }
+}
+
+#else
+static inline int copy_utsname(int flags, struct task_struct *tsk)
+{
+ return 0;
+}
+static inline void put_uts_ns(struct uts_namespace *ns)
+{
+}
+static inline void exit_utsname(struct task_struct *p)
+{
+}
+#endif

 static inline struct new_utsname *utsname(void)
 {
```

```
- return &system_utsname;
+ return &current->nsproxy->uts_ns->name;
 }

 static inline struct new_utsname *init_utsname(void)
 {
- return &system_utsname;
+ return &init_uts_ns.name;
 }

+#define system_utsname init_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..78cef48 100644
--- a/init/version.c
+++ b/init/version.c
@@ -11,23 +11,27 @@
 #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);
-
 const char linux_banner[] =
  "Linux version " UTS_RELEASE " (" LINUX_COMPILE_BY "@"
  LINUX_COMPILE_HOST ") (" LINUX_COMPILER ") " UTS_VERSION "\n";
diff --git a/kernel/Makefile b/kernel/Makefile
index 215fb33..ab7426c 100644
--- a/kernel/Makefile
+++ b/kernel/Makefile
@@ -38,6 +38,7 @@ obj-$(CONFIG_GENERIC_HARDIRQS) += irq/
 obj-$(CONFIG_SECCOMP) += seccomp.o
 obj-$(CONFIG_RCU_TORTURE_TEST) += rcutorture.o
 obj-$(CONFIG_RELAY) += relay.o
+obj-$(CONFIG_UTS_NS) += utsname.o

 ifneq ($(CONFIG_SCHED_NO_NO_OMIT_FRAME_POINTER),y)
 # According to Alan Modra <alan@linuxcare.com.au>, the -fno-omit-frame-pointer is
diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
index 4103f58..d2c6e94 100644
--- a/kernel/nsproxy.c
+++ b/kernel/nsproxy.c
@@ -14,6 +14,7 @@
 #include <linux/version.h>
 #include <linux/nsproxy.h>
 #include <linux/namespace.h>
+#include <linux/utsname.h>
```

```
 static inline void get_nsproxy(struct nsproxy *ns)
 {
@@ -57,6 +58,8 @@ struct nsproxy *dup_namespaces(struct ns
  if (ns) {
   if (ns->namespace)
    get_namespace(ns->namespace);
+  if (ns->uts_ns)
+   get_uts_ns(ns->uts_ns);
  }

  return ns;
@@ -95,6 +98,15 @@ int copy_namespaces(int flags, struct ta
   goto out;
  }

+ err = copy_utsname(flags, tsk);
+ if (err) {
+  if (new_ns->namespace)
+   put_namespace(new_ns->namespace);
+  tsk->nsproxy = old_ns;
+  put_nsproxy(new_ns);
+  goto out;
+ }
+
 out:
  put_nsproxy(old_ns);
  return err;
@@ -104,5 +116,7 @@ void free_nsproxy(struct nsproxy *ns)
 {
   if (ns->namespace)
    put_namespace(ns->namespace);
+  if (ns->uts_ns)
+   put_uts_ns(ns->uts_ns);
   kfree(ns);
 }
diff --git a/kernel/utsname.c b/kernel/utsname.c
new file mode 100644
index 0000000..2818c9b
--- /dev/null
+++ b/kernel/utsname.c
@@ -0,0 +1,43 @@
+/*
+ *  Copyright (C) 2004 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.
+ */
+
+#include <linux/compile.h>
+#include <linux/module.h>
+#include <linux/uts.h>
+#include <linux/utsname.h>
+#include <linux/version.h>
+
+/*
+ * Copy task tsk's utsname namespace, or clone it if flags
+ * specifies CLONE_NEWUTS.  In latter case, changes to the
+ * utsname of this process won't be seen by parent, and vice
+ * versa.
+ */
+int copy_utsname(int flags, struct task_struct *tsk)
+{
+ struct uts_namespace *old_ns = tsk->nsproxy->uts_ns;
+ int err = 0;
+
+ if (!old_ns)
+  return 0;
+
+ get_uts_ns(old_ns);
+
+ return err;
+}
+
+void free_uts_ns(struct kref *kref)
+{
+ struct uts_namespace *ns;
+
+ ns = container_of(kref, struct uts_namespace, kref);
+ kfree(ns);
+}
--
1.1.6
```