Subject: [PATCH 1/9] Add a user_namespace as creator/owner of uts_namespace
Posted by serge on Thu, 17 Feb 2011 15:02:57 GMT
View Forum Message <> Reply to Message

copy_process() handles CLONE_NEWUSER before the rest of the
namespaces. So in the case of clone(CLONE_NEWUSER|CLONE_NEWUTS)
the new uts namespace will have the new user namespace as its
owner. That is what we want, since we want root in that new
userns to be able to have privilege over it.

Changelog:
 Feb 15: don't set uts_ns->user_ns if we didn't create
  a new uts_ns.

Signed-off-by: Serge E. Hallyn <serge.hallyn@canonical.com>

```
---
 include/linux/utsname.h |    3 +++
 init/version.c          |    2 ++
 kernel/nsproxy.c        |    5 +++++
 kernel/user.c           |    8 ++++++--
 kernel/utsname.c        |    4 ++++
 5 files changed, 20 insertions(+), 2 deletions(-)

diff --git a/include/linux/utsname.h b/include/linux/utsname.h
index 69f3997..85171be 100644
--- a/include/linux/utsname.h
+++ b/include/linux/utsname.h
@@ -37,9 +37,12 @@ struct new_utsname {
 #include <linux/nsproxy.h>
 #include <linux/err.h>

+struct user_namespace;
+
 struct uts_namespace {
  struct kref kref;
  struct new_utsname name;
+ struct user_namespace *user_ns;
 };
 extern struct uts_namespace init_uts_ns;

diff --git a/init/version.c b/init/version.c
index adff586..97bb86f 100644
--- a/init/version.c
+++ b/init/version.c
@@ -21,6 +21,7 @@ extern int version_string(LINUX_VERSION_CODE);
 int version_string(LINUX_VERSION_CODE);
 #endif
```

```
+extern struct user_namespace init_user_ns;
 struct uts_namespace init_uts_ns = {
  .kref = {
   .refcount = ATOMIC_INIT(2),
@@ -33,6 +34,7 @@ struct uts_namespace init_uts_ns = {
   .machine = UTS_MACHINE,
   .domainname = UTS_DOMAINNAME,
  },
+ .user_ns = &init_user_ns,
 };
 EXPORT_SYMBOL_GPL(init_uts_ns);

diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
index f74e6c0..034dc2e 100644
--- a/kernel/nsproxy.c
+++ b/kernel/nsproxy.c
@@ -74,6 +74,11 @@ static struct nsproxy *create_new_namespaces(unsigned long flags,
   err = PTR_ERR(new_nsp->uts_ns);
   goto out_uts;
  }
+ if (new_nsp->uts_ns != tsk->nsproxy->uts_ns) {
+  put_user_ns(new_nsp->uts_ns->user_ns);
+  new_nsp->uts_ns->user_ns = task_cred_xxx(tsk, user)->user_ns;
+  get_user_ns(new_nsp->uts_ns->user_ns);
+ }

  new_nsp->ipc_ns = copy_ipcs(flags, tsk->nsproxy->ipc_ns);
  if (IS_ERR(new_nsp->ipc_ns)) {
diff --git a/kernel/user.c b/kernel/user.c
index 5c598ca..9e03e9c 100644
--- a/kernel/user.c
+++ b/kernel/user.c
@@ -17,9 +17,13 @@
 #include <linux/module.h>
 #include <linux/user_namespace.h>

+/*
+ * userns count is 1 for root user, 1 for init_uts_ns,
+ * and 1 for... ?
+ */
 struct user_namespace init_user_ns = {
  .kref = {
-  .refcount = ATOMIC_INIT(2),
+  .refcount = ATOMIC_INIT(3),
  },
  .creator = &root_user,
 };
@@ -47,7 +51,7 @@ static struct kmem_cache *uid_cachep;
```

```
  */
 static DEFINE_SPINLOCK(uidhash_lock);

-/* root_user.__count is 2, 1 for init task cred, 1 for init_user_ns->creator */
+/* root_user.__count is 2, 1 for init task cred, 1 for init_user_ns->user_ns */
 struct user_struct root_user = {
  .__count = ATOMIC_INIT(2),
  .processes = ATOMIC_INIT(1),
diff --git a/kernel/utsname.c b/kernel/utsname.c
index 8a82b4b..a7b3a8d 100644
--- a/kernel/utsname.c
+++ b/kernel/utsname.c
@@ -14,6 +14,7 @@
 #include <linux/utsname.h>
 #include <linux/err.h>
 #include <linux/slab.h>
+#include <linux/user_namespace.h>

 static struct uts_namespace *create_uts_ns(void)
 {
@@ -40,6 +41,8 @@ static struct uts_namespace *clone_uts_ns(struct uts_namespace *old_ns)

  down_read(&uts_sem);
  memcpy(&ns->name, &old_ns->name, sizeof(ns->name));
+ ns->user_ns = old_ns->user_ns;
+ get_user_ns(ns->user_ns);
  up_read(&uts_sem);
  return ns;
 }
@@ -71,5 +74,6 @@ void free_uts_ns(struct kref *kref)
  struct uts_namespace *ns;

  ns = container_of(kref, struct uts_namespace, kref);
+ put_user_ns(ns->user_ns);
  kfree(ns);
 }
--
1.7.0.4
```