Subject: [PATCH 15/15] sysfs: user namespaces: fix bug with clone(CLONE_NEWUSER) with fairsched
Posted by ebiederm on Fri, 04 Jul 2008 01:23:35 GMT
View Forum Message <> Reply to Message

Mark the /sys/kernel/uids directory to be tagged so that processes in
different user namespaces can remount /sys and see their own uid
listings.

Without this patch, having CONFIG_FAIR_SCHED=y makes user namespaces
unusable, because when you
  clone(CLONE_NEWUSER)
it will auto-create the root userid and try to create
/sys/kernel/uids/0. Since that already exists from the parent user
namespace, the create fails, and the clone misleadingly ends up
returning -ENOMEM.

This patch fixes the issue by allowing each user namespace to remount
/sys, and having /sys filter the /sys/kernel/uid/ entries by user
namespace.

Changelong:
v2 - Reworked for the updated sysfs api

Signed-off-by: Serge Hallyn <serue@us.ibm.com>
Signed-off-by: Benjamin Thery <benjamin.thery@bull.net>
Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 include/linux/sched.h | 1 +
 include/linux/sysfs.h | 1 +
 kernel/user.c | 22 ++++++++++++++++++++++
 kernel/user_namespace.c | 1 +
 4 files changed, 25 insertions(+), 0 deletions(-)

diff --git a/include/linux/sched.h b/include/linux/sched.h
index c5d3f84..d2be6a5 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -598,6 +598,7 @@ struct user_struct {
  /* Hash table maintenance information */
  struct hlist_node uidhash_node;
  uid_t uid;
+ struct user_namespace *user_ns;

 #ifdef CONFIG_USER_SCHED
  struct task_group *tg;
diff --git a/include/linux/sysfs.h b/include/linux/sysfs.h

```
index 1ed31bb..ecb942c 100644
--- a/include/linux/sysfs.h
+++ b/include/linux/sysfs.h
@@ -81,6 +81,7 @@ struct sysfs_ops {
 enum sysfs_tag_type {
  SYSFS_TAG_TYPE_NONE = 0,
  SYSFS_TAG_TYPE_NETNS,
+ SYSFS_TAG_TYPE_USERNS,
  SYSFS_TAG_TYPES
 };

diff --git a/kernel/user.c b/kernel/user.c
index 865ecf5..ca29fbc 100644
--- a/kernel/user.c
+++ b/kernel/user.c
@@ -53,6 +53,7 @@ struct user_struct root_user = {
  .files  = ATOMIC_INIT(0),
  .sigpending = ATOMIC_INIT(0),
  .locked_shm     = 0,
+ .user_ns = &init_user_ns,
 #ifdef CONFIG_USER_SCHED
  .tg  = &init_task_group,
 #endif
@@ -230,16 +231,33 @@ static struct attribute *uids_attributes[] = {
  NULL
 };

+static const void *uids_mount_tag(void)
+{
+ return current->nsproxy->user_ns;
+}
+
+static struct sysfs_tag_type_operations uids_tag_type_operations = {
+ .mount_tag = uids_mount_tag,
+};
+
 /* the lifetime of user_struct is not managed by the core (now) */
 static void uids_release(struct kobject *kobj)
 {
  return;
 }

+static const void *uids_sysfs_tag(struct kobject *kobj)
+{
+ struct user_struct *up;
+ up = container_of(kobj, struct user_struct, kobj);
+ return up->user_ns;
+}
```

```
+
 static struct kobj_type uids_ktype = {
  .sysfs_ops = &kobj_sysfs_ops,
  .default_attrs = uids_attributes,
  .release = uids_release,
+ .sysfs_tag = uids_sysfs_tag,
 };

 /* create /sys/kernel/uids/<uid>/cpu_share file for this user */
@@ -272,6 +290,9 @@ int __init uids_sysfs_init(void)
  if (!uids_kset)
   return -ENOMEM;

+ sysfs_register_tag_type(SYSFS_TAG_TYPE_USERNS, &uids_tag_type_operations);
+ sysfs_make_tagged_dir(&uids_kset->kobj, SYSFS_TAG_TYPE_USERNS);
+
  return uids_user_create(&root_user);
 }

@@ -405,6 +426,7 @@ struct user_struct *alloc_uid(struct user_namespace *ns, uid_t uid)

   new->uid = uid;
   atomic_set(&new->__count, 1);
+  new->user_ns = ns;

   if (sched_create_user(new) < 0)
    goto out_free_user;
diff --git a/kernel/user_namespace.c b/kernel/user_namespace.c
index a9ab059..f67bbe0 100644
--- a/kernel/user_namespace.c
+++ b/kernel/user_namespace.c
@@ -71,6 +71,7 @@ void free_user_ns(struct kref *kref)
  struct user_namespace *ns;

  ns = container_of(kref, struct user_namespace, kref);
+ sysfs_exit_tag(SYSFS_TAG_TYPE_USERNS, ns);
  release_uids(ns);
  kfree(ns);
 }
--
1.5.3.rc6.17.g1911
```