Subject: [patch -mm 02/17] user namespace: add the framework
Posted by Cedric Le Goater on Tue, 05 Dec 2006 10:27:54 GMT
View Forum Message <> Reply to Message

From: Cedric Le Goater <clg@fr.ibm.com>

This patch adds the user namespace struct and framework

Basically, it will allow a process to unshare its user_struct table,
resetting at the same time its own user_struct and all the associated
accounting.

A new root user (uid == 0) is added to the user namespace upon
creation. Such root users have full privileges and it seems that
theses privileges should be controlled through some means (process
capabilities ?)

The unshare is not included in this patch.

Changes since [try #3]:
 - moved struct user_namespace to files user_namespace.{c,h}

Changes since [try #2]:
 - removed struct user_namespace* argument from find_user()

Changes since [try #1]:
 - removed struct user_namespace* argument from find_user()
 - added a root_user per user namespace

Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>

---
 include/linux/init_task.h | 2 +
 include/linux/nsproxy.h | 2 +
 include/linux/sched.h | 3 +-
 include/linux/user_namespace.h | 49 ++
 init/Kconfig | 8 ++++++
 kernel/Makefile | 2 -
 kernel/fork.c | 2 -
 kernel/nsproxy.c | 11 +++++++++
 kernel/sys.c | 5 ++--
 kernel/user.c | 18 +++++++--------
 kernel/user_namespace.c | 44 ++++++++++++++++++++++++++++++++++++
 11 files changed, 132 insertions(+), 14 deletions(-)

Index: 2.6.19-rc6-mm2/kernel/user.c
===
--- 2.6.19-rc6-mm2.orig/kernel/user.c

```
+++ 2.6.19-rc6-mm2/kernel/user.c
@@ -14,20 +14,19 @@
 #include <linux/bitops.h>
 #include <linux/key.h>
 #include <linux/interrupt.h>
+#include <linux/module.h>
+#include <linux/user_namespace.h>

 /*
  * UID task count cache, to get fast user lookup in "alloc_uid"
  * when changing user ID's (ie setuid() and friends).
  */

-#define UIDHASH_BITS (CONFIG_BASE_SMALL ? 3 : 8)
-#define UIDHASH_SZ  (1 << UIDHASH_BITS)
 #define UIDHASH_MASK  (UIDHASH_SZ - 1)
 #define __uidhashfn(uid) (((uid >> UIDHASH_BITS) + uid) & UIDHASH_MASK)
-#define uidhashentry(uid) (uidhash_table + __uidhashfn((uid)))
+#define uidhashentry(ns, uid) ((ns)->uidhash_table + __uidhashfn((uid)))

 static kmem_cache_t *uid_cachep;
-static struct list_head uidhash_table[UIDHASH_SZ];

 /*
  * The uidhash_lock is mostly taken from process context, but it is
@@ -94,9 +93,10 @@ struct user_struct *find_user(uid_t uid)
 {
  struct user_struct *ret;
  unsigned long flags;
+ struct user_namespace *ns = current->nsproxy->user_ns;

  spin_lock_irqsave(&uidhash_lock, flags);
- ret = uid_hash_find(uid, uidhashentry(uid));
+ ret = uid_hash_find(uid, uidhashentry(ns, uid));
  spin_unlock_irqrestore(&uidhash_lock, flags);
  return ret;
 }
@@ -120,9 +120,9 @@ void free_uid(struct user_struct *up)
  }
 }

-struct user_struct * alloc_uid(uid_t uid)
+struct user_struct * alloc_uid(struct user_namespace *ns, uid_t uid)
 {
- struct list_head *hashent = uidhashentry(uid);
+ struct list_head *hashent = uidhashentry(ns, uid);
  struct user_struct *up;
```

```
  spin_lock_irq(&uidhash_lock);
@@ -211,11 +211,11 @@ static int __init uid_cache_init(void)
    0, SLAB_HWCACHE_ALIGN|SLAB_PANIC, NULL, NULL);

  for(n = 0; n < UIDHASH_SZ; ++n)
-  INIT_LIST_HEAD(uidhash_table + n);
+  INIT_LIST_HEAD(init_user_ns.uidhash_table + n);

  /* Insert the root user immediately (init already runs as root) */
  spin_lock_irq(&uidhash_lock);
- uid_hash_insert(&root_user, uidhashentry(0));
+ uid_hash_insert(&root_user, uidhashentry(&init_user_ns, 0));
  spin_unlock_irq(&uidhash_lock);

  return 0;
Index: 2.6.19-rc6-mm2/include/linux/nsproxy.h
===================================================================
--- 2.6.19-rc6-mm2.orig/include/linux/nsproxy.h
+++ 2.6.19-rc6-mm2/include/linux/nsproxy.h
@@ -9,6 +9,7 @@ struct uts_namespace;
 struct ipc_namespace;
 struct pid_namespace;
 struct net_namespace;
+struct user_namespace;

 /*
  * A structure to contain pointers to all per-process
@@ -31,6 +32,7 @@ struct nsproxy {
  struct mnt_namespace *mnt_ns;
  struct pid_namespace *pid_ns;
  struct net_namespace *net_ns;
+ struct user_namespace *user_ns;
 };
 extern struct nsproxy init_nsproxy;

Index: 2.6.19-rc6-mm2/include/linux/sched.h
===================================================================
--- 2.6.19-rc6-mm2.orig/include/linux/sched.h
+++ 2.6.19-rc6-mm2/include/linux/sched.h
@@ -252,6 +252,7 @@ extern signed long schedule_timeout_unin
 asmlinkage void schedule(void);

 struct nsproxy;
+struct user_namespace;

 /* Maximum number of active map areas.. This is a random (large) number */
 #define DEFAULT_MAX_MAP_COUNT 65536
@@ -1286,7 +1287,7 @@ extern struct task_struct *find_task_by_
```

```
 extern void __set_special_pids(pid_t session, pid_t pgrp);

 /* per-UID process charging. */
-extern struct user_struct * alloc_uid(uid_t);
+extern struct user_struct * alloc_uid(struct user_namespace *, uid_t);
 static inline struct user_struct *get_uid(struct user_struct *u)
 {
  atomic_inc(&u->__count);
Index: 2.6.19-rc6-mm2/include/linux/init_task.h
===================================================================
--- 2.6.19-rc6-mm2.orig/include/linux/init_task.h
+++ 2.6.19-rc6-mm2/include/linux/init_task.h
@@ -9,6 +9,7 @@
 #include <linux/ipc.h>
 #include <linux/pid_namespace.h>
 #include <linux/net_namespace.h>
+#include <linux/user_namespace.h>

 #define INIT_FDTABLE \
 {       \
@@ -81,6 +82,7 @@ extern struct nsproxy init_nsproxy;
  .mnt_ns  = NULL,      \
  INIT_NET_NS(net_ns)      \
  INIT_IPC_NS(ipc_ns)      \
+ .user_ns = &init_user_ns,    \
 }

 #define INIT_SIGHAND(sighand) {      \
Index: 2.6.19-rc6-mm2/kernel/sys.c
===================================================================
--- 2.6.19-rc6-mm2.orig/kernel/sys.c
+++ 2.6.19-rc6-mm2/kernel/sys.c
@@ -33,6 +33,7 @@
 #include <linux/compat.h>
 #include <linux/syscalls.h>
 #include <linux/kprobes.h>
+#include <linux/user_namespace.h>

 #include <asm/uaccess.h>
 #include <asm/io.h>
@@ -1062,13 +1063,13 @@ static int set_user(uid_t new_ruid, int
 {
  struct user_struct *new_user;

- new_user = alloc_uid(new_ruid);
+ new_user = alloc_uid(current->nsproxy->user_ns, new_ruid);
  if (!new_user)
   return -EAGAIN;
```

```
  if (atomic_read(&new_user->processes) >=
      current->signal->rlim[RLIMIT_NPROC].rlim_cur &&
-    new_user != &root_user) {
+    new_user != current->nsproxy->user_ns->root_user) {
    free_uid(new_user);
    return -EAGAIN;
  }
Index: 2.6.19-rc6-mm2/kernel/fork.c
===================================================================
--- 2.6.19-rc6-mm2.orig/kernel/fork.c
+++ 2.6.19-rc6-mm2/kernel/fork.c
@@ -996,7 +996,7 @@ static struct task_struct *copy_process(
  if (atomic_read(&p->user->processes) >=
      p->signal->rlim[RLIMIT_NPROC].rlim_cur) {
    if (!capable(CAP_SYS_ADMIN) && !capable(CAP_SYS_RESOURCE) &&
-      p->user != &root_user)
+       p->user != current->nsproxy->user_ns->root_user)
      goto bad_fork_free;
  }

Index: 2.6.19-rc6-mm2/kernel/nsproxy.c
===================================================================
--- 2.6.19-rc6-mm2.orig/kernel/nsproxy.c
+++ 2.6.19-rc6-mm2/kernel/nsproxy.c
@@ -74,6 +74,8 @@ struct nsproxy *dup_namespaces(struct ns
     get_pid_ns(ns->pid_ns);
    if (ns->net_ns)
     get_net_ns(ns->net_ns);
+   if (ns->user_ns)
+    get_user_ns(ns->user_ns);
  }

  return ns;
@@ -125,10 +127,17 @@ int copy_namespaces(int flags, struct ta
  if (err)
    goto out_net;

+ err = copy_user_ns(flags, tsk);
+ if (err)
+   goto out_user;
+
 out:
  put_nsproxy(old_ns);
  return err;

+out_user:
+ if (new_ns->net_ns)
```

```
+  put_net_ns(new_ns->net_ns);
 out_net:
  if (new_ns->pid_ns)
   put_pid_ns(new_ns->pid_ns);
@@ -159,5 +168,7 @@ void free_nsproxy(struct nsproxy *ns)
   put_pid_ns(ns->pid_ns);
  if (ns->net_ns)
   put_net_ns(ns->net_ns);
+ if (ns->user_ns)
+  put_user_ns(ns->user_ns);
  kfree(ns);
 }
Index: 2.6.19-rc6-mm2/init/Kconfig
===================================================================
--- 2.6.19-rc6-mm2.orig/init/Kconfig
+++ 2.6.19-rc6-mm2/init/Kconfig
@@ -222,6 +222,14 @@ config UTS_NS
    vservers, to use uts namespaces to provide different
    uts info for different servers.  If unsure, say N.

+config USER_NS
+ bool "User Namespaces"
+ default n
+ help
+   Support user namespaces.  This allows containers, i.e.
+   vservers, to use user namespaces to provide different
+   user info for different servers.  If unsure, say N.
+
 config AUDIT
  bool "Auditing support"
  depends on NET
Index: 2.6.19-rc6-mm2/include/linux/user_namespace.h
===================================================================
--- /dev/null
+++ 2.6.19-rc6-mm2/include/linux/user_namespace.h
@@ -0,0 +1,49 @@
+#ifndef _LINUX_USER_NAMESPACE_H
+#define _LINUX_USER_NAMESPACE_H
+
+#include <linux/kref.h>
+#include <linux/nsproxy.h>
+
+#define UIDHASH_BITS (CONFIG_BASE_SMALL ? 3 : 8)
+#define UIDHASH_SZ (1 << UIDHASH_BITS)
+
+struct user_namespace {
+ struct kref  kref;
+ struct list_head uidhash_table[UIDHASH_SZ];
```

```
+ struct user_struct *root_user;
+};
+
+extern struct user_namespace init_user_ns;
+
+#ifdef CONFIG_USER_NS
+
+static inline void get_user_ns(struct user_namespace *ns)
+{
+ kref_get(&ns->kref);
+}
+
+extern int copy_user_ns(int flags, struct task_struct *tsk);
+extern void free_user_ns(struct kref *kref);
+
+static inline void put_user_ns(struct user_namespace *ns)
+{
+ kref_put(&ns->kref, free_user_ns);
+}
+
+#else
+
+static inline void get_user_ns(struct user_namespace *ns)
+{
+}
+
+static inline int copy_user_ns(int flags, struct task_struct *tsk)
+{
+ return 0;
+}
+
+static inline void put_user_ns(struct user_namespace *ns)
+{
+}
+#endif
+
+#endif /* _LINUX_USER_H */
Index: 2.6.19-rc6-mm2/kernel/user_namespace.c
===================================================================
--- /dev/null
+++ 2.6.19-rc6-mm2/kernel/user_namespace.c
@@ -0,0 +1,44 @@
+/*
+ *  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/module.h>
+#include <linux/version.h>
+#include <linux/nsproxy.h>
+#include <linux/user_namespace.h>
+
+struct user_namespace init_user_ns = {
+ .kref = {
+  .refcount = ATOMIC_INIT(2),
+ },
+ .root_user = &root_user,
+};
+
+EXPORT_SYMBOL_GPL(init_user_ns);
+
+#ifdef CONFIG_USER_NS
+
+int copy_user_ns(int flags, struct task_struct *tsk)
+{
+ struct user_namespace *old_ns = tsk->nsproxy->user_ns;
+ int err = 0;
+
+ if (!old_ns)
+  return 0;
+
+ get_user_ns(old_ns);
+ return err;
+}
+
+void free_user_ns(struct kref *kref)
+{
+ struct user_namespace *ns;
+
+ ns = container_of(kref, struct user_namespace, kref);
+ kfree(ns);
+}
+
+#endif /* CONFIG_USER_NS */
Index: 2.6.19-rc6-mm2/kernel/Makefile
===================================================================
--- 2.6.19-rc6-mm2.orig/kernel/Makefile
+++ 2.6.19-rc6-mm2/kernel/Makefile
@@ -4,7 +4,7 @@

 obj-y     = sched.o fork.o exec_domain.o panic.o printk.o profile.o \
       exit.o itimer.o time.o softirq.o resource.o \
-      sysctl.o capability.o ptrace.o timer.o user.o \
+      sysctl.o capability.o ptrace.o timer.o user.o user_namespace.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 \

--

_____

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