Subject: [PATCH -mm 1/2] user namespace : add unshare
Posted by Cedric Le Goater on Fri, 08 Jun 2007 15:14:07 GMT
View Forum Message <> Reply to Message

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 #4]:
- Updated get_user_ns and put_user_ns to accept NULL, and
  get_user_ns to return the namespace.

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>
Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
Signed-off-by: Andrew Morton <akpm@osdl.org>
Acked-by: Pavel Emelianov <xemul@openvz.org>
Cc: Herbert Poetzl <herbert@13thfloor.at>
Cc: Kirill Korotaev <dev@sw.ru>
Cc: Eric W. Biederman <ebiederm@xmission.com>
---

include/linux/init_task.h | 2 +
include/linux/nsproxy.h | 1
include/linux/sched.h | 3 +-
include/linux/user_namespace.h | 57 ++++++++++++++++++++++++++++++++++++++++
init/Kconfig | 9 ++++++
kernel/Makefile | 2 -
kernel/fork.c | 2 -
kernel/nsproxy.c | 9 ++++++
kernel/sys.c | 5 ++-
kernel/user.c | 18 +++++++-----
kernel/user_namespace.c | 43 ++++++++++++++++++++++++++++++
11 files changed, 137 insertions(+), 14 deletions(-)

```
Index: 2.6.22-rc4-mm2/include/linux/init_task.h
===================================================================
--- 2.6.22-rc4-mm2.orig/include/linux/init_task.h
+++ 2.6.22-rc4-mm2/include/linux/init_task.h
@@ -8,6 +8,7 @@
 #include <linux/lockdep.h>
 #include <linux/ipc.h>
 #include <linux/pid_namespace.h>
+#include <linux/user_namespace.h>

 #define INIT_FDTABLE \
 {       \
@@ -78,6 +79,7 @@ extern struct nsproxy init_nsproxy;
  .uts_ns  = &init_uts_ns,     \
  .mnt_ns  = NULL,      \
  INIT_IPC_NS(ipc_ns)      \
+ .user_ns = &init_user_ns,    \
 }

 #define INIT_SIGHAND(sighand) {      \
Index: 2.6.22-rc4-mm2/include/linux/nsproxy.h
===================================================================
--- 2.6.22-rc4-mm2.orig/include/linux/nsproxy.h
+++ 2.6.22-rc4-mm2/include/linux/nsproxy.h
@@ -28,6 +28,7 @@ struct nsproxy {
  struct ipc_namespace *ipc_ns;
  struct mnt_namespace *mnt_ns;
  struct pid_namespace *pid_ns;
+ struct user_namespace *user_ns;
 };
 extern struct nsproxy init_nsproxy;

Index: 2.6.22-rc4-mm2/include/linux/sched.h
===================================================================
--- 2.6.22-rc4-mm2.orig/include/linux/sched.h
+++ 2.6.22-rc4-mm2/include/linux/sched.h
@@ -268,6 +268,7 @@ 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
@@ -1378,7 +1379,7 @@ 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: 2.6.22-rc4-mm2/include/linux/user_namespace.h
===================================================================
--- /dev/null
+++ 2.6.22-rc4-mm2/include/linux/user_namespace.h
@@ -0,0 +1,57 @@
+#ifndef _LINUX_USER_NAMESPACE_H
+#define _LINUX_USER_NAMESPACE_H
+
+#include <linux/kref.h>
+#include <linux/nsproxy.h>
+#include <linux/sched.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 struct user_namespace *get_user_ns(struct user_namespace *ns)
+{
+ if (ns)
+  kref_get(&ns->kref);
+ return ns;
+}
+
+extern struct user_namespace *copy_user_ns(int flags,
+        struct user_namespace *old_ns);
+extern void free_user_ns(struct kref *kref);
+
+static inline void put_user_ns(struct user_namespace *ns)
+{
+ if (ns)
+  kref_put(&ns->kref, free_user_ns);
+}
+
```

```
+#else
+
+static inline struct user_namespace *get_user_ns(struct user_namespace *ns)
+{
+ return &init_user_ns;
+}
+
+static inline struct user_namespace *copy_user_ns(int flags,
+        struct user_namespace *old_ns)
+{
+ return NULL;
+}
+
+static inline void put_user_ns(struct user_namespace *ns)
+{
+}
+
+#endif
+
+#endif /* _LINUX_USER_H */
Index: 2.6.22-rc4-mm2/init/Kconfig
===================================================================
--- 2.6.22-rc4-mm2.orig/init/Kconfig
+++ 2.6.22-rc4-mm2/init/Kconfig
@@ -231,6 +231,15 @@ config TASK_IO_ACCOUNTING

    Say N if unsure.

+config USER_NS
+ bool "User Namespaces (EXPERIMENTAL)"
+ default n
+ depends on EXPERIMENTAL
+ 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: 2.6.22-rc4-mm2/kernel/Makefile
===================================================================
--- 2.6.22-rc4-mm2.orig/kernel/Makefile
+++ 2.6.22-rc4-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 \
Index: 2.6.22-rc4-mm2/kernel/fork.c
===================================================================
--- 2.6.22-rc4-mm2.orig/kernel/fork.c
+++ 2.6.22-rc4-mm2/kernel/fork.c
@@ -1004,7 +1004,7 @@ 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: 2.6.22-rc4-mm2/kernel/nsproxy.c
===================================================================
--- 2.6.22-rc4-mm2.orig/kernel/nsproxy.c
+++ 2.6.22-rc4-mm2/kernel/nsproxy.c
@@ -79,8 +79,15 @@ static struct nsproxy *create_new_namesp
  if (IS_ERR(new_nsp->pid_ns))
   goto out_pid;

+ new_nsp->user_ns = copy_user_ns(flags, tsk->nsproxy->user_ns);
+ if (IS_ERR(new_nsp->user_ns))
+  goto out_user;
+
  return new_nsp;

+out_user:
+ if (new_nsp->pid_ns)
+  put_pid_ns(new_nsp->pid_ns);
 out_pid:
  if (new_nsp->ipc_ns)
   put_ipc_ns(new_nsp->ipc_ns);
@@ -140,6 +147,8 @@ void free_nsproxy(struct nsproxy *ns)
   put_ipc_ns(ns->ipc_ns);
  if (ns->pid_ns)
   put_pid_ns(ns->pid_ns);
+ if (ns->user_ns)
+  put_user_ns(ns->user_ns);
  kfree(ns);
 }

Index: 2.6.22-rc4-mm2/kernel/sys.c
```

```
===================================================================
--- 2.6.22-rc4-mm2.orig/kernel/sys.c
+++ 2.6.22-rc4-mm2/kernel/sys.c
@@ -35,6 +35,7 @@
 #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>
@@ -1078,13 +1079,13 @@ 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: 2.6.22-rc4-mm2/kernel/user.c
===================================================================
--- 2.6.22-rc4-mm2.orig/kernel/user.c
+++ 2.6.22-rc4-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 struct kmem_cache *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: 2.6.22-rc4-mm2/kernel/user_namespace.c
===================================================================
--- /dev/null
```

```
+++ 2.6.22-rc4-mm2/kernel/user_namespace.c
@@ -0,0 +1,43 @@
+/*
+ *  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
+
+struct user_namespace * copy_user_ns(int flags, struct user_namespace *old_ns)
+{
+ struct user_namespace *new_ns;
+
+ BUG_ON(!old_ns);
+ get_user_ns(old_ns);
+
+ new_ns = old_ns;
+ return new_ns;
+}
+
+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 */
```

---

Containers mailing list
Containers@lists.linux-foundation.org

Subject: Re: [PATCH -mm 1/2] user namespace : add unshare
Posted by Cedric Le Goater on Fri, 08 Jun 2007 15:22:38 GMT
View Forum Message <> Reply to Message

oops, wrong subject. it should be :

"user namespace : add the framework"

Sorry about that,

C.

Cedric Le Goater wrote:
> 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 #4]:
> - Updated get_user_ns and put_user_ns to accept NULL, and
> get_user_ns to return the namespace.
>
> 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>
> Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
> Signed-off-by: Andrew Morton <akpm@osdl.org>
> Acked-by: Pavel Emelianov <xemul@openvz.org>
> Cc: Herbert Poetzl <herbert@13thfloor.at>
> Cc: Kirill Korotaev <dev@sw.ru>
> Cc: Eric W. Biederman <ebiederm@xmission.com>
> ---
>

>  include/linux/init_task.h      |    2 +
>  include/linux/nsproxy.h        |    1
>  include/linux/sched.h          |    3 +-
>  include/linux/user_namespace.h |   57 ++++++++++++++++++++++++++++++++++++++++
>  init/Kconfig                   |    9 ++++++
>  kernel/Makefile                |    2 -
>  kernel/fork.c                  |    2 -
>  kernel/nsproxy.c               |    9 ++++++
>  kernel/sys.c                   |    5 ++-
>  kernel/user.c                  |   18 ++++++------
>  kernel/user_namespace.c        |   43 ++++++++++++++++++++++++++++++
>  11 files changed, 137 insertions(+), 14 deletions(-)
>
> Index: 2.6.22-rc4-mm2/include/linux/init_task.h
> ===================================================================
> --- 2.6.22-rc4-mm2.orig/include/linux/init_task.h
> +++ 2.6.22-rc4-mm2/include/linux/init_task.h
> @@ -8,6 +8,7 @@
>  #include <linux/lockdep.h>
>  #include <linux/ipc.h>
>  #include <linux/pid_namespace.h>
> +#include <linux/user_namespace.h>
>
>  #define INIT_FDTABLE \
>  {       \
> @@ -78,6 +79,7 @@ extern struct nsproxy init_nsproxy;
>    .uts_ns  = &init_uts_ns,     \
>    .mnt_ns  = NULL,      \
>    INIT_IPC_NS(ipc_ns)      \
> + .user_ns = &init_user_ns,    \
>  }
>
>  #define INIT_SIGHAND(sighand) {      \
> Index: 2.6.22-rc4-mm2/include/linux/nsproxy.h
> ===================================================================
> --- 2.6.22-rc4-mm2.orig/include/linux/nsproxy.h
> +++ 2.6.22-rc4-mm2/include/linux/nsproxy.h
> @@ -28,6 +28,7 @@ struct nsproxy {
>    struct ipc_namespace *ipc_ns;
>    struct mnt_namespace *mnt_ns;
>    struct pid_namespace *pid_ns;
> + struct user_namespace *user_ns;
>  };
>  extern struct nsproxy init_nsproxy;
>
> Index: 2.6.22-rc4-mm2/include/linux/sched.h
> ===================================================================
> --- 2.6.22-rc4-mm2.orig/include/linux/sched.h

> +++ 2.6.22-rc4-mm2/include/linux/sched.h
> @@ -268,6 +268,7 @@ 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
> @@ -1378,7 +1379,7 @@ 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: 2.6.22-rc4-mm2/include/linux/user_namespace.h
> ===================================================================
> --- /dev/null
> +++ 2.6.22-rc4-mm2/include/linux/user_namespace.h
> @@ -0,0 +1,57 @@
> +#ifndef _LINUX_USER_NAMESPACE_H
> +#define _LINUX_USER_NAMESPACE_H
> +
> +#include <linux/kref.h>
> +#include <linux/nsproxy.h>
> +#include <linux/sched.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 struct user_namespace *get_user_ns(struct user_namespace *ns)
> +{
> + if (ns)
> +  kref_get(&ns->kref);
> + return ns;

> +}
> +
> +extern struct user_namespace *copy_user_ns(int flags,
> +        struct user_namespace *old_ns);
> +extern void free_user_ns(struct kref *kref);
> +
> +static inline void put_user_ns(struct user_namespace *ns)
> +{
> + if (ns)
> +  kref_put(&ns->kref, free_user_ns);
> +}
> +
> +#else
> +
> +static inline struct user_namespace *get_user_ns(struct user_namespace *ns)
> +{
> + return &init_user_ns;
> +}
> +
> +static inline struct user_namespace *copy_user_ns(int flags,
> +        struct user_namespace *old_ns)
> +{
> + return NULL;
> +}
> +
> +static inline void put_user_ns(struct user_namespace *ns)
> +{
> +}
> +
> +#endif
> +
> +#endif /* _LINUX_USER_H */
> Index: 2.6.22-rc4-mm2/init/Kconfig
> ===================================================================
> --- 2.6.22-rc4-mm2.orig/init/Kconfig
> +++ 2.6.22-rc4-mm2/init/Kconfig
> @@ -231,6 +231,15 @@ config TASK_IO_ACCOUNTING
>
>     Say N if unsure.
>
> +config USER_NS
> + bool "User Namespaces (EXPERIMENTAL)"
> + default n
> + depends on EXPERIMENTAL
> + 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: 2.6.22-rc4-mm2/kernel/Makefile
> ===================================================================
> --- 2.6.22-rc4-mm2.orig/kernel/Makefile
> +++ 2.6.22-rc4-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 \
> Index: 2.6.22-rc4-mm2/kernel/fork.c
> ===================================================================
> --- 2.6.22-rc4-mm2.orig/kernel/fork.c
> +++ 2.6.22-rc4-mm2/kernel/fork.c
> @@ -1004,7 +1004,7 @@ 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: 2.6.22-rc4-mm2/kernel/nsproxy.c
> ===================================================================
> --- 2.6.22-rc4-mm2.orig/kernel/nsproxy.c
> +++ 2.6.22-rc4-mm2/kernel/nsproxy.c
> @@ -79,8 +79,15 @@ static struct nsproxy *create_new_namesp
>   if (IS_ERR(new_nsp->pid_ns))
>    goto out_pid;
>
> + new_nsp->user_ns = copy_user_ns(flags, tsk->nsproxy->user_ns);
> + if (IS_ERR(new_nsp->user_ns))
> +  goto out_user;
> +
>   return new_nsp;
>
> +out_user:
> + if (new_nsp->pid_ns)
> +  put_pid_ns(new_nsp->pid_ns);
>  out_pid:
```

```
>   if (new_nsp->ipc_ns)
>    put_ipc_ns(new_nsp->ipc_ns);
> @@ -140,6 +147,8 @@ void free_nsproxy(struct nsproxy *ns)
>    put_ipc_ns(ns->ipc_ns);
>   if (ns->pid_ns)
>    put_pid_ns(ns->pid_ns);
> + if (ns->user_ns)
> +  put_user_ns(ns->user_ns);
>   kfree(ns);
>  }
>
> Index: 2.6.22-rc4-mm2/kernel/sys.c
> ===================================================================
> --- 2.6.22-rc4-mm2.orig/kernel/sys.c
> +++ 2.6.22-rc4-mm2/kernel/sys.c
> @@ -35,6 +35,7 @@
>  #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>
> @@ -1078,13 +1079,13 @@ 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: 2.6.22-rc4-mm2/kernel/user.c
> ===================================================================
> --- 2.6.22-rc4-mm2.orig/kernel/user.c
> +++ 2.6.22-rc4-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 struct kmem_cache *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: 2.6.22-rc4-mm2/kernel/user_namespace.c
> ===================================================================
> --- /dev/null
> +++ 2.6.22-rc4-mm2/kernel/user_namespace.c
> @@ -0,0 +1,43 @@
> +/*
> + *  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
> +
> +struct user_namespace * copy_user_ns(int flags, struct user_namespace *old_ns)
> +{
> + struct user_namespace *new_ns;
> +
> + BUG_ON(!old_ns);
> + get_user_ns(old_ns);
> +
> + new_ns = old_ns;
> + return new_ns;
> +}
> +
```

> +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 */
> -
> To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
> the body of a message to majordomo@vger.kernel.org
> More majordomo info at  http://vger.kernel.org/majordomo-info.html
> Please read the FAQ at  http://www.tux.org/lkml/
>

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers