
Subject: Re: [PATCH -mm 2/2] user namespace : add unshare
Posted by [Cedric Le Goater](#) on Mon, 11 Jun 2007 21:44:38 GMT
[View Forum Message](#) <> [Reply to Message](#)

Cedric Le Goater wrote:

> Andrew Morton wrote:

>> On Fri, 08 Jun 2007 17:14:12 +0200

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

>>

>>> Changelog: Fix !CONFIG_USER_NS clone with CLONE_NEWUSER so it returns -EINVAL

>>> rather than 0, so that userspace knows they didn't get a new user

>>> namespace.

>> This changelog doesn't seem to match this patch.

>

> Indeed. the changelog entry is now deprecated since I removed

> CONFIG_USER_NS.

>

> that was a fix for the issue you reported about that patch breaking

> the mouse under FC5 when the kernel was compiled without CONFIG_USER_NS.

So here's a more appropriate changelog for the patch.

C.

This patch enables the unshare of user namespaces.

It adds a new clone flag CLONE_NEWUSER and implements copy_user_ns()
which resets the current user_struct and adds a new root user (uid ==
0)

For now, unsharing the user namespace allows a process to reset its
user_struct accounting and uid 0 in the new user namespace should be
contained using appropriate means, for instance selinux

The plan, when the full support is complete (all uid checks covered),
is to keep the original user's rights in the original namespace, and
let a process become uid 0 in the new namespace, with full
capabilities to the new namespace.

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

Signed-off-by: Cedric Le Goater <clg@fr.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/sched.h      | 1
include/linux/user_namespace.h | 4 +++
kernel/fork.c              | 2 -
kernel/nsproxy.c           | 5 ++--
kernel/user_namespace.c     | 46 ++++++-----
5 files changed, 54 insertions(+), 4 deletions(-)

```

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
@@ -25,6 +25,7 @@
#define CLONE_STOPPED 0x02000000 /* Start in stopped state */
#define CLONE_NEWUTS 0x04000000 /* New utsname group? */
#define CLONE_NEWIPC 0x08000000 /* New ipcs */
+#define CLONE_NEWUSER 0x10000000 /* New user namespace */

```

/*

* Scheduling policies

Index: 2.6.22-rc4-mm2/include/linux/user_namespace.h

=====

```

--- 2.6.22-rc4-mm2.orig/include/linux/user_namespace.h
+++ 2.6.22-rc4-mm2/include/linux/user_namespace.h
@@ -4,6 +4,7 @@
#include <linux/kref.h>
#include <linux/nsproxy.h>
#include <linux/sched.h>
+#include <linux/err.h>

#define UIDHASH_BITS (CONFIG_BASE_SMALL ? 3 : 8)
#define UIDHASH_SZ (1 << UIDHASH_BITS)
@@ -45,6 +46,9 @@ static inline struct user_namespace *get
static inline struct user_namespace *copy_user_ns(int flags,
    struct user_namespace *old_ns)
{
+ if (flags & CLONE_NEWUSER)
+ return ERR_PTR(-EINVAL);
+
return NULL;
}

```

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
@@ -1616,7 +1616,7 @@ asmlinkage long sys_unshare(unsigned lon
err = -EINVAL;

```

```

if (unshare_flags & ~(CLONE_THREAD|CLONE_FS|CLONE_NEWNS|CLONE_SIGHAND|
    CLONE_VM|CLONE_FILES|CLONE_SYSVSEM|
-   CLONE_NEWUTS|CLONE_NEWIPC))
+   CLONE_NEWUTS|CLONE_NEWIPC|CLONE_NEWUSER))
    goto bad_unshare_out;

if ((err = unshare_thread(unshare_flags)))
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
@@ -117,7 +117,7 @@ int copy_namespaces(int flags, struct ta

    get_nsproxy(old_ns);

- if (!(flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC)))
+ if (!(flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC | CLONE_NEWUSER)))
    return 0;

    if (!capable(CAP_SYS_ADMIN)) {
@@ -163,7 +163,8 @@ int unshare_nsproxy_namespaces(unsigned
    struct nsproxy *old_ns = current->nsproxy;
    int err = 0;

- if (!(unshare_flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC)))
+ if (!(unshare_flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC |
+   CLONE_NEWUSER)))
    return 0;

    if (!capable(CAP_SYS_ADMIN))
Index: 2.6.22-rc4-mm2/kernel/user_namespace.c
=====
--- 2.6.22-rc4-mm2.orig/kernel/user_namespace.c
+++ 2.6.22-rc4-mm2/kernel/user_namespace.c
@@ -21,6 +21,45 @@ EXPORT_SYMBOL_GPL(init_user_ns);

#ifdef CONFIG_USER_NS

+/*
+ * Clone a new ns copying an original user ns, setting refcount to 1
+ * @old_ns: namespace to clone
+ * Return NULL on error (failure to kmalloc), new ns otherwise
+ */
+static struct user_namespace *clone_user_ns(struct user_namespace *old_ns)
+{
+ struct user_namespace *ns;
+ struct user_struct *new_user;
+ int n;

```

```

+
+ ns = kmalloc(sizeof(struct user_namespace), GFP_KERNEL);
+ if (!ns)
+ return NULL;
+
+ kref_init(&ns->kref);
+
+ for (n = 0; n < UIDHASH_SZ; ++n)
+ INIT_LIST_HEAD(ns->uidhash_table + n);
+
+ /* Insert new root user. */
+ ns->root_user = alloc_uid(ns, 0);
+ if (!ns->root_user) {
+ kfree(ns);
+ return NULL;
+ }
+
+ /* Reset current->user with a new one */
+ new_user = alloc_uid(ns, current->uid);
+ if (!new_user) {
+ free_uid(ns->root_user);
+ kfree(ns);
+ return NULL;
+ }
+
+ switch_uid(new_user);
+ return ns;
+}
+
struct user_namespace * copy_user_ns(int flags, struct user_namespace *old_ns)
{
    struct user_namespace *new_ns;
@@ -28,7 +67,12 @@ struct user_namespace * copy_user_ns(int
    BUG_ON(!old_ns);
    get_user_ns(old_ns);

- new_ns = old_ns;
+ if (!(flags & CLONE_NEWUSER))
+ return old_ns;
+
+ new_ns = clone_user_ns(old_ns);
+
+ put_user_ns(old_ns);
    return new_ns;
}

```

Containers mailing list

