

---

Subject: [patch 03/10] unprivileged mounts: propagate error values from clone\_mnt  
Posted by [Miklos Szeredi](#) on Wed, 16 Jan 2008 12:31:50 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

From: Miklos Szeredi <mszeredi@suse.cz>

Allow clone\_mnt() to return errors other than ENOMEM. This will be used for returning a different error value when the number of user mounts goes over the limit.

Fix copy\_tree() to return EPERM for unbindable mounts.

Signed-off-by: Miklos Szeredi <mszeredi@suse.cz>

Acked-by: Serge Hallyn <serue@us.ibm.com>

---

Index: linux/fs/namespace.c

=====

--- linux.orig/fs/namespace.c 2008-01-16 13:25:06.000000000 +0100

+++ linux/fs/namespace.c 2008-01-16 13:25:07.000000000 +0100

@@ -490,41 +490,42 @@ static struct vfsmount \*clone\_mnt(struct  
struct super\_block \*sb = old->mnt\_sb;  
struct vfsmount \*mnt = alloc\_vfsmnt(old->mnt\_devname);

```
- if (mnt) {  
- mnt->mnt_flags = old->mnt_flags;  
- atomic_inc(&sb->s_active);  
- mnt->mnt_sb = sb;  
- mnt->mnt_root = dget(root);  
- mnt->mnt_mountpoint = mnt->mnt_root;  
- mnt->mnt_parent = mnt;  
-  
- /* don't copy the MNT_USER flag */  
- mnt->mnt_flags &= ~MNT_USER;  
- if (flag & CL_SETUSER)  
- set_mnt_user(mnt);  
-  
- if (flag & CL_SLAVE) {  
- list_add(&mnt->mnt_slave, &old->mnt_slave_list);  
- mnt->mnt_master = old;  
- CLEAR_MNT_SHARED(mnt);  
- } else if (!(flag & CL_PRIVATE)) {  
- if ((flag & CL_PROPAGATION) || IS_MNT_SHARED(old))  
- list_add(&mnt->mnt_share, &old->mnt_share);  
- if (IS_MNT_SLAVE(old))  
- list_add(&mnt->mnt_slave, &old->mnt_slave);  
- mnt->mnt_master = old->mnt_master;  
- }
```

```

- if (flag & CL_MAKE_SHARED)
- set_mnt_shared(mnt);
+ if (!mnt)
+ return ERR_PTR(-ENOMEM);

- /* stick the duplicate mount on the same expiry list
- * as the original if that was on one */
- if (flag & CL_EXPIRE) {
- spin_lock(&vfsmount_lock);
- if (!list_empty(&old->mnt_expire))
- list_add(&mnt->mnt_expire, &old->mnt_expire);
- spin_unlock(&vfsmount_lock);
- }
+ mnt->mnt_flags = old->mnt_flags;
+ atomic_inc(&sb->s_active);
+ mnt->mnt_sb = sb;
+ mnt->mnt_root = dget(root);
+ mnt->mnt_mountpoint = mnt->mnt_root;
+ mnt->mnt_parent = mnt;
+
+ /* don't copy the MNT_USER flag */
+ mnt->mnt_flags &= ~MNT_USER;
+ if (flag & CL_SETUSER)
+ set_mnt_user(mnt);
+
+ if (flag & CL_SLAVE) {
+ list_add(&mnt->mnt_slave, &old->mnt_slave_list);
+ mnt->mnt_master = old;
+ CLEAR_MNT_SHARED(mnt);
+ } else if (!(flag & CL_PRIVATE)) {
+ if ((flag & CL_PROPAGATION) || IS_MNT_SHARED(old))
+ list_add(&mnt->mnt_share, &old->mnt_share);
+ if (IS_MNT_SLAVE(old))
+ list_add(&mnt->mnt_slave, &old->mnt_slave);
+ mnt->mnt_master = old->mnt_master;
+ }
+ if (flag & CL_MAKE_SHARED)
+ set_mnt_shared(mnt);
+
+ /* stick the duplicate mount on the same expiry list
+ * as the original if that was on one */
+ if (flag & CL_EXPIRE) {
+ spin_lock(&vfsmount_lock);
+ if (!list_empty(&old->mnt_expire))
+ list_add(&mnt->mnt_expire, &old->mnt_expire);
+ spin_unlock(&vfsmount_lock);
+ }
return mnt;

```

```

}
@@ -998,11 +999,11 @@ struct vfsmount *copy_tree(struct vfsmou
    struct nameidata nd;

    if (!(flag & CL_COPY_ALL) && IS_MNT_UNBINDABLE(mnt))
-   return NULL;
+   return ERR_PTR(-EPERM);

    res = q = clone_mnt(mnt, dentry, flag);
-   if (!q)
-       goto Enomem;
+   if (IS_ERR(q))
+       goto error;
    q->mnt_mountpoint = mnt->mnt_mountpoint;

    p = mnt;
@@ -1023,8 +1024,8 @@ struct vfsmount *copy_tree(struct vfsmou
    nd.path.mnt = q;
    nd.path.dentry = p->mnt_mountpoint;
    q = clone_mnt(p, p->mnt_root, flag);
-   if (!q)
-       goto Enomem;
+   if (IS_ERR(q))
+       goto error;
    spin_lock(&vfsmount_lock);
    list_add_tail(&q->mnt_list, &res->mnt_list);
    attach_mnt(q, &nd);
@@ -1032,7 +1033,7 @@ struct vfsmount *copy_tree(struct vfsmou
    }
    }
    return res;
-Enomem:
+ error:
    if (res) {
        LIST_HEAD(umount_list);
        spin_lock(&vfsmount_lock);
@@ -1040,7 +1041,7 @@ Enomem:
        spin_unlock(&vfsmount_lock);
        release_mounts(&umount_list);
    }
-   return NULL;
+   return q;
}

struct vfsmount *collect_mounts(struct vfsmount *mnt, struct dentry *dentry)
@@ -1239,13 +1240,13 @@ static int do_loopback(struct nameidata
    goto out;

```

```

clone_fl = (flags & MS_SETUSER) ? CL_SETUSER : 0;
- err = -ENOMEM;
if (flags & MS_REC)
    mnt = copy_tree(old_nd.path.mnt, old_nd.path.dentry, clone_fl);
else
    mnt = clone_mnt(old_nd.path.mnt, old_nd.path.dentry, clone_fl);

- if (!mnt)
+ err = PTR_ERR(mnt);
+ if (IS_ERR(mnt))
    goto out;

err = graft_tree(mnt, nd);
@@ -1816,10 +1817,10 @@ static struct mnt_namespace *dup_mnt_ns(
/* First pass: copy the tree topology */
new_ns->root = copy_tree(mnt_ns->root, mnt_ns->root->mnt_root,
    CL_COPY_ALL | CL_EXPIRE);
- if (!new_ns->root) {
+ if (IS_ERR(new_ns->root)) {
    up_write(&namespace_sem);
    kfree(new_ns);
- return ERR_PTR(-ENOMEM);;
+ return ERR_CAST(new_ns->root);
}
spin_lock(&vfsmount_lock);
list_add_tail(&new_ns->list, &new_ns->root->mnt_list);

```

Index: linux/fs/pnode.c

```

=====
--- linux.orig/fs/pnode.c 2008-01-16 13:24:53.000000000 +0100
+++ linux/fs/pnode.c 2008-01-16 13:25:07.000000000 +0100
@@ -189,8 +189,9 @@ int propagate_mnt(struct vfsmount *dest_

```

```

    source = get_source(m, prev_dest_mnt, prev_src_mnt, &type);

- if (!(child = copy_tree(source, source->mnt_root, type))) {
- ret = -ENOMEM;
+ child = copy_tree(source, source->mnt_root, type);
+ if (IS_ERR(child)) {
+ ret = PTR_ERR(child);
    list_splice(tree_list, tmp_list.prev);
    goto out;
}

--

```

---

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

---