

---

Subject: [patch 5/9] unprivileged mounts: allow unprivileged bind mounts

Posted by [Miklos Szeredi](#) on Tue, 08 Jan 2008 11:35:07 GMT

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

---

From: Miklos Szeredi <[mszeredi@suse.cz](mailto:mszeredi@suse.cz)>

Allow bind mounts to unprivileged users if the following conditions are met:

- mountpoint is not a symlink
- parent mount is owned by the user
- the number of user mounts is below the maximum

Unprivileged mounts imply MS\_SETUSER, and will also have the "nosuid" and "nodev" mount flags set.

In particular, if mounting process doesn't have CAP\_SETUID capability, then the "nosuid" flag will be added, and if it doesn't have CAP\_MKNOD capability, then the "nodev" flag will be added.

Signed-off-by: Miklos Szeredi <[mszeredi@suse.cz](mailto:mszeredi@suse.cz)>

---

Index: linux/fs/namespace.c

=====

--- linux.orig/fs/namespace.c 2008-01-04 13:47:49.000000000 +0100

+++ linux/fs/namespace.c 2008-01-04 13:48:01.000000000 +0100

@@ -487,11 +487,34 @@ static void dec\_nr\_user\_mounts(void)

spin\_unlock(&vfsmount\_lock);

}

-static void set\_mnt\_user(struct vfsmount \*mnt)

+static int reserve\_user\_mount(void)

+{

+ int err = 0;

+

+ spin\_lock(&vfsmount\_lock);

+ if (nr\_user\_mounts >= max\_user\_mounts && !capable(CAP\_SYS\_ADMIN))

+ err = -EPERM;

+ else

+ nr\_user\_mounts++;

+ spin\_unlock(&vfsmount\_lock);

+ return err;

+}

+

+static void \_\_set\_mnt\_user(struct vfsmount \*mnt)

{

BUG\_ON(mnt->mnt\_flags & MNT\_USER);

mnt->mnt\_uid = current->fsuid;

```

mnt->mnt_flags |= MNT_USER;
+
+ if (!capable(CAP_SETUID))
+ mnt->mnt_flags |= MNT_NOSUID;
+ if (!capable(CAP_MKNOD))
+ mnt->mnt_flags |= MNT_NODEV;
+}
+
+static void set_mnt_user(struct vfsmount *mnt)
+{
+ __set_mnt_user(mnt);
+ spin_lock(&vfsmount_lock);
+ nr_user_mounts++;
+ spin_unlock(&vfsmount_lock);
@@ -510,10 +533,16 @@ static struct vfsmount *clone_mnt(struct
    int flag)
    {
        struct super_block *sb = old->mnt_sb;
- struct vfsmount *mnt = alloc_vfsmnt(old->mnt_devname);
+ struct vfsmount *mnt;

+ if (flag & CL_SETUSER) {
+ int err = reserve_user_mount();
+ if (err)
+ return ERR_PTR(err);
+ }
+ mnt = alloc_vfsmnt(old->mnt_devname);
+ if (!mnt)
- return ERR_PTR(-ENOMEM);
+ goto alloc_failed;

    mnt->mnt_flags = old->mnt_flags;
    atomic_inc(&sb->s_active);
@@ -525,7 +554,7 @@ static struct vfsmount *clone_mnt(struct
    /* don't copy the MNT_USER flag */
    mnt->mnt_flags &= ~MNT_USER;
    if (flag & CL_SETUSER)
- set_mnt_user(mnt);
+ __set_mnt_user(mnt);

    if (flag & CL_SLAVE) {
        list_add(&mnt->mnt_slave, &old->mnt_slave_list);
@@ -550,6 +579,11 @@ static struct vfsmount *clone_mnt(struct
        spin_unlock(&vfsmount_lock);
    }
    return mnt;
+
+ alloc_failed:

```

```

+ if (flag & CL_SETUSER)
+ dec_nr_user_mounts();
+ return ERR_PTR(-ENOMEM);
}

static inline void __mntput(struct vfsmount *mnt)
@@ -986,22 +1020,26 @@ asmlinkage long sys_oldumount(char __use

#endif

-static int mount_is_safe(struct nameidata *nd)
+/*
+ * Conditions for unprivileged mounts are:
+ * - mountpoint is not a symlink
+ * - mountpoint is in a mount owned by the user
+ */
+static bool permit_mount(struct nameidata *nd, int *flags)
{
+ struct inode *inode = nd->path.dentry->d_inode;
+
+ if (capable(CAP_SYS_ADMIN))
- return 0;
- return -EPERM;
-#ifndef notyet
- if (S_ISLNK(nd->path.dentry->d_inode->i_mode))
- return -EPERM;
- if (nd->path.dentry->d_inode->i_mode & S_ISVTX) {
- if (current->uid != nd->path.dentry->d_inode->i_uid)
- return -EPERM;
- }
- if (vfs_permission(nd, MAY_WRITE))
- return -EPERM;
- return 0;
-#endif
+ return true;
+
+ if (S_ISLNK(inode->i_mode))
+ return false;
+
+ if (!is_mount_owner(nd->path.mnt, current->fsuid))
+ return false;
+
+ *flags |= MS_SETUSER;
+ return true;
}

static int lives_below_in_same_fs(struct dentry *d, struct dentry *dentry)
@@ -1245,9 +1283,10 @@ static int do_loopback(struct nameidata

```

```
int clone_fl;  
struct nameidata old_nd;  
struct vfsmount *mnt = NULL;  
- int err = mount_is_safe(nd);  
- if (err)  
- return err;  
+ int err;  
+  
+ if (!permit_mount(nd, &flags))  
+ return -EPERM;  
if (!old_name || !*old_name)  
return -EINVAL;  
err = path_lookup(old_name, LOOKUP_FOLLOW, &old_nd);  
  
--
```

---

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

---