Subject: [patch 5/8] allow unprivileged bind mounts
Posted by Miklos Szeredi on Fri, 20 Apr 2007 10:25:37 GMT
View Forum Message <> Reply to Message

From: Miklos Szeredi <mszeredi@suse.cz>

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

- mountpoint is not a symlink or special file
- 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.

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

Index: linux/fs/namespace.c
===
--- linux.orig/fs/namespace.c 2007-04-20 11:55:09.000000000 +0200
+++ linux/fs/namespace.c 2007-04-20 11:55:10.000000000 +0200
@@ -237,11 +237,30 @@ 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->uid;
 mnt->mnt_flags |= MNT_USER;
+ if (!capable(CAP_SYS_ADMIN))
+ mnt->mnt_flags |= MNT_NOSUID | 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);
@@ -260,9 +279,16 @@ 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);
@@ -274,7 +300,7 @@ 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);
@@ -299,6 +325,11 @@ 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)
@@ -745,22 +776,29 @@ 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 or special file
+ * - mountpoint is in a mount owned by the user
+ */
+static bool permit_mount(struct nameidata *nd, int *flags)
 {
+ struct inode *inode = nd->dentry->d_inode;
+
  if (capable(CAP_SYS_ADMIN))
-  return 0;
- return -EPERM;
-#ifdef notyet
- if (S_ISLNK(nd->dentry->d_inode->i_mode))
-  return -EPERM;
- if (nd->dentry->d_inode->i_mode & S_ISVTX) {
-  if (current->uid != nd->dentry->d_inode->i_uid)
-   return -EPERM;
- }
- if (vfs_permission(nd, MAY_WRITE))
-  return -EPERM;
- return 0;
-#endif
+  return true;
+
+ if (!S_ISDIR(inode->i_mode) && !S_ISREG(inode->i_mode))
+  return false;
+
+ if (!(nd->mnt->mnt_flags & MNT_USER))
+  return false;
+
+ if (nd->mnt->mnt_uid != current->uid)
+  return false;
+
+ *flags |= MS_SETUSER;
+ return true;
 }

 static int lives_below_in_same_fs(struct dentry *d, struct dentry *dentry)
@@ -981,9 +1019,10 @@ static int do_loopback(struct nameidata
  int clone_flags;
  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