Subject: [patch 09/10] allow unprivileged mounts
Posted by Miklos Szeredi on Thu, 12 Apr 2007 16:45:50 GMT
View Forum Message <> Reply to Message

From: Miklos Szeredi <mszeredi@suse.cz>

Define a new fs flag FS_SAFE, which denotes, that unprivileged
mounting of this filesystem may not constitute a security problem.

Since most filesystems haven't been designed with unprivileged
mounting in mind, a thorough audit is needed before setting this flag.

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

Index: linux/fs/namespace.c
===
--- linux.orig/fs/namespace.c 2007-04-12 14:04:27.000000000 +0200
+++ linux/fs/namespace.c 2007-04-12 14:04:32.000000000 +0200
@@ -784,7 +784,8 @@ asmlinkage long sys_oldumount(char __use
  * o if it's a sticky directory, it must be owned by the user
  * o it must not be an append-only file/directory
  */
-static int mount_is_safe(struct nameidata *nd, int *flags)
+static int mount_is_safe(struct nameidata *nd, struct file_system_type *type,
+   int *flags)
 {
  struct inode *inode = nd->dentry->d_inode;

@@ -794,6 +795,9 @@ static int mount_is_safe(struct nameidat
  if (!(current->nsproxy->mnt_ns->flags & MNT_NS_PERMIT_USERMOUNTS))
   return -EPERM;

+ if (type && !(type->fs_flags & FS_SAFE))
+ return -EPERM;
+
  if (!S_ISDIR(inode->i_mode) && !S_ISREG(inode->i_mode))
   return -EPERM;

@@ -1025,7 +1029,7 @@ static int do_loopback(struct nameidata
  int clone_flags;
  struct nameidata old_nd;
  struct vfsmount *mnt = NULL;
- int err = mount_is_safe(nd, &flags);
+ int err = mount_is_safe(nd, NULL, &flags);
  if (err)
   return err;
  if (!old_name || !*old_name)

```
@@ -1187,26 +1191,46 @@ out:
  * create a new mount for userspace and request it to be added into the
  * namespace's tree
  */
-static int do_new_mount(struct nameidata *nd, char *type, int flags,
+static int do_new_mount(struct nameidata *nd, char *fstype, int flags,
    int mnt_flags, char *name, void *data)
 {
+ int err;
  struct vfsmount *mnt;
+ struct file_system_type *type;

- if (!type || !memchr(type, 0, PAGE_SIZE))
+ if (!fstype || !memchr(fstype, 0, PAGE_SIZE))
   return -EINVAL;

- /* we need capabilities... */
- if (!capable(CAP_SYS_ADMIN))
-  return -EPERM;
+ type = get_fs_type(fstype);
+ if (!type)
+  return -ENODEV;

- mnt = do_kern_mount(type, flags & ~MS_SETUSER, name, data);
- if (IS_ERR(mnt))
+ err = mount_is_safe(nd, type, &flags);
+ if (err)
+  goto out_put_filesystem;
+
+ if (flags & MS_SETUSER) {
+  err = reserve_user_mount();
+  if (err)
+   goto out_put_filesystem;
+ }
+
+ mnt = vfs_kern_mount(type, flags & ~MS_SETUSER, name, data);
+ put_filesystem(type);
+ if (IS_ERR(mnt)) {
+  if (flags & MS_SETUSER)
+   dec_nr_user_mounts();
   return PTR_ERR(mnt);
+ }

  if (flags & MS_SETUSER)
-  set_mnt_user(mnt);
+  __set_mnt_user(mnt);

  return do_add_mount(mnt, nd, mnt_flags, NULL);
```

```
+
+ out_put_filesystem:
+ put_filesystem(type);
+ return err;
 }

 /*
@@ -1236,7 +1260,7 @@ int do_add_mount(struct vfsmount *newmnt
  if (S_ISLNK(newmnt->mnt_root->d_inode->i_mode))
   goto unlock;

- /* MNT_USER was set earlier */
+ /* some flags may have been set earlier */
  newmnt->mnt_flags |= mnt_flags;
  if ((err = graft_tree(newmnt, nd)))
   goto unlock;
Index: linux/include/linux/fs.h
===================================================================
--- linux.orig/include/linux/fs.h 2007-04-12 14:04:29.000000000 +0200
+++ linux/include/linux/fs.h 2007-04-12 14:04:32.000000000 +0200
@@ -96,6 +96,7 @@ extern int dir_notify_enable;
 #define FS_REQUIRES_DEV 1
 #define FS_BINARY_MOUNTDATA 2
 #define FS_HAS_SUBTYPE 4
+#define FS_SAFE 8  /* Safe to mount by unprivileged users */
 #define FS_REVAL_DOT 16384 /* Check the paths ".", ".." for staleness */
 #define FS_RENAME_DOES_D_MOVE 32768 /* FS will handle d_move()
       * during rename() internally.

--

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