Subject: [patch 06/10] unprivileged mounts: allow unprivileged mounts
Posted by Miklos Szeredi on Wed, 16 Jan 2008 12:31:53 GMT
View Forum Message <> Reply to Message

From: Miklos Szeredi <mszeredi@suse.cz>

For "safe" filesystems allow unprivileged mounting and forced
unmounting.

A filesystem type is considered "safe", if mounting it by an
unprivileged user may not cause a security problem. This is somewhat
subjective, so setting this property is left to userspace (implemented
in the next patch).

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

Make this a separate integer member in 'struct file_system_type'
instead of a flag, since that is easier to handle by sysctl code.

Move subtype handling from do_kern_mount() into do_new_mount(). All
other callers are kernel-internal and do not need subtype support.

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 13:25:08.000000000 +0100
+++ linux/fs/namespace.c 2008-01-16 13:25:09.000000000 +0100
@@ -966,14 +966,16 @@ static bool is_mount_owner(struct vfsmou
 /*
  * umount is permitted for
  * - sysadmin
- * - mount owner, if not forced umount
+ * - mount owner
+ *   o if not forced umount,
+ *   o if forced umount, and filesystem is "safe"
  */
 static bool permit_umount(struct vfsmount *mnt, int flags)
 {
  if (capable(CAP_SYS_ADMIN))
   return true;

- if (flags & MNT_FORCE)
+ if ((flags & MNT_FORCE) && !(mnt->mnt_sb->s_type->fs_safe))

```
   return false;

  return is_mount_owner(mnt, current->fsuid);
@@ -1031,13 +1033,17 @@ asmlinkage long sys_oldumount(char __use
  * - mountpoint is not a symlink
  * - mountpoint is in a mount owned by the user
  */
-static bool permit_mount(struct nameidata *nd, int *flags)
+static bool permit_mount(struct nameidata *nd, struct file_system_type *type,
+     int *flags)
 {
  struct inode *inode = nd->path.dentry->d_inode;

  if (capable(CAP_SYS_ADMIN))
   return true;

+ if (type && !type->fs_safe)
+  return false;
+
  if (S_ISLNK(inode->i_mode))
   return false;

@@ -1291,7 +1297,7 @@ static int do_loopback(struct nameidata
  struct vfsmount *mnt = NULL;
  int err;

- if (!permit_mount(nd, &flags))
+ if (!permit_mount(nd, NULL, &flags))
   return -EPERM;
  if (!old_name || !*old_name)
   return -EINVAL;
@@ -1472,30 +1478,76 @@ out:
  return err;
 }

+static struct vfsmount *fs_set_subtype(struct vfsmount *mnt, const char *fstype)
+{
+ int err;
+ const char *subtype = strchr(fstype, '.');
+ if (subtype) {
+  subtype++;
+  err = -EINVAL;
+  if (!subtype[0])
+   goto err;
+ } else
+  subtype = "";
+
+ mnt->mnt_sb->s_subtype = kstrdup(subtype, GFP_KERNEL);
```

```
+ err = -ENOMEM;
+ if (!mnt->mnt_sb->s_subtype)
+  goto err;
+ return mnt;
+
+ err:
+ mntput(mnt);
+ return ERR_PTR(err);
+}
+
 /*
  * 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;
-
- mnt = do_kern_mount(type, flags & ~MS_SETUSER, name, data);
- if (IS_ERR(mnt))
+ type = get_fs_type(fstype);
+ if (!type)
+  return -ENODEV;
+
+ err = -EPERM;
+ if (!permit_mount(nd, type, &flags))
+  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);
+ if (!IS_ERR(mnt) && (type->fs_flags & FS_HAS_SUBTYPE) &&
```

```
+     !mnt->mnt_sb->s_subtype)
+  mnt = fs_set_subtype(mnt, fstype);
+ 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;
 }

 /*
@@ -1526,7 +1578,7 @@ 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/fs/super.c
===================================================================
--- linux.orig/fs/super.c 2008-01-16 13:24:52.000000000 +0100
+++ linux/fs/super.c 2008-01-16 13:25:09.000000000 +0100
@@ -906,29 +906,6 @@ out:

 EXPORT_SYMBOL_GPL(vfs_kern_mount);

-static struct vfsmount *fs_set_subtype(struct vfsmount *mnt, const char *fstype)
-{
- int err;
- const char *subtype = strchr(fstype, '.');
- if (subtype) {
-  subtype++;
-  err = -EINVAL;
-  if (!subtype[0])
-   goto err;
- } else
```

```
-  subtype = "";
-
- mnt->mnt_sb->s_subtype = kstrdup(subtype, GFP_KERNEL);
- err = -ENOMEM;
- if (!mnt->mnt_sb->s_subtype)
-  goto err;
- return mnt;
-
- err:
- mntput(mnt);
- return ERR_PTR(err);
-}
-
 struct vfsmount *
 do_kern_mount(const char *fstype, int flags, const char *name, void *data)
 {
@@ -937,9 +914,6 @@ do_kern_mount(const char *fstype, int fl
  if (!type)
   return ERR_PTR(-ENODEV);
  mnt = vfs_kern_mount(type, flags, name, data);
- if (!IS_ERR(mnt) && (type->fs_flags & FS_HAS_SUBTYPE) &&
-     !mnt->mnt_sb->s_subtype)
-  mnt = fs_set_subtype(mnt, fstype);
  put_filesystem(type);
  return mnt;
 }
Index: linux/include/linux/fs.h
===================================================================
--- linux.orig/include/linux/fs.h 2008-01-16 13:25:07.000000000 +0100
+++ linux/include/linux/fs.h 2008-01-16 13:25:09.000000000 +0100
@@ -1430,6 +1430,7 @@ int sync_inode(struct inode *inode, stru
 struct file_system_type {
  const char *name;
  int fs_flags;
+ int fs_safe;
  int (*get_sb) (struct file_system_type *, int,
         const char *, void *, struct vfsmount *);
  void (*kill_sb) (struct super_block *);

--
```