Subject: [PATCH] Wake up mandatory locks waiter on chmod (v2)
Posted by Pavel Emelianov on Mon, 17 Sep 2007 08:13:24 GMT
View Forum Message <> Reply to Message

When the process is blocked on mandatory lock and someone changes
the inode's permissions, so that the lock is no longer mandatory,
nobody wakes up the blocked process, but probably should.

Switched to use mandatory_lock() static inline function.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
Cc: J. Bruce Fields <bfields@fieldses.org>

```
 fs/attr.c         |    9 +++++++--
 fs/locks.c        |   17 +++++++++++++++++-
 include/linux/fs.h |    1 +
 3 files changed, 24 insertions(+), 3 deletions(-)

diff --git a/fs/attr.c b/fs/attr.c
index ae58bd3..da643b0 100644
--- a/fs/attr.c
+++ b/fs/attr.c
@@ -104,7 +104,7 @@ int notify_change(struct dentry * dentry
 {
  struct inode *inode = dentry->d_inode;
  mode_t mode;
- int error;
+ int error, mandatory;
  struct timespec now;
  unsigned int ia_valid = attr->ia_valid;

@@ -151,6 +151,8 @@ int notify_change(struct dentry * dentry
  if (ia_valid & ATTR_SIZE)
   down_write(&dentry->d_inode->i_alloc_sem);

+ mandatory = (inode->i_flock && mandatory_lock(inode));
+
  if (inode->i_op && inode->i_op->setattr) {
   error = security_inode_setattr(dentry, attr);
   if (!error)
@@ -171,8 +173,11 @@ int notify_change(struct dentry * dentry
  if (ia_valid & ATTR_SIZE)
   up_write(&dentry->d_inode->i_alloc_sem);

- if (!error)
+ if (!error) {
```

```
   fsnotify_change(dentry, ia_valid);
+  if (mandatory)
+   locks_wakeup_mandatory(inode);
+ }

  return error;
 }
diff --git a/fs/locks.c b/fs/locks.c
index a71c589..6481fd9 100644
--- a/fs/locks.c
+++ b/fs/locks.c
@@ -1110,7 +1110,8 @@ int locks_mandatory_area(int read_write,
    break;
   if (!(fl.fl_flags & FL_SLEEP))
    break;
-  error = wait_event_interruptible(fl.fl_wait, !fl.fl_next);
+  error = wait_event_interruptible(fl.fl_wait,
+    !fl.fl_next || !__mandatory_lock(inode));
   if (!error) {
    /*
     * If we've been sleeping someone might have
@@ -1129,6 +1130,20 @@ int locks_mandatory_area(int read_write,

 EXPORT_SYMBOL(locks_mandatory_area);

+void locks_wakeup_mandatory(struct inode *inode)
+{
+ struct file_lock *fl, **before;
+
+ lock_kernel();
+ for_each_lock(inode, before) {
+  fl = *before;
+
+  if (IS_POSIX(fl))
+   locks_wake_up_blocks(fl);
+ }
+ unlock_kernel();
+}
+
 /* We already had a lease on this file; just change its type */
 int lease_modify(struct file_lock **before, int arg)
 {
diff --git a/include/linux/fs.h b/include/linux/fs.h
index 9c519e6..215eea3 100644
--- a/include/linux/fs.h
+++ b/include/linux/fs.h
@@ -1483,6 +1483,7 @@ extern struct kset fs_subsys;
```

```
 extern int locks_mandatory_locked(struct inode *);
 extern int locks_mandatory_area(int, struct inode *, struct file *, loff_t, size_t);
+extern void locks_wakeup_mandatory(struct inode *);

 /*
  * Candidates for mandatory locking have the setgid bit set
```