Subject: Re: [PATCH] Wake up mandatory locks waiter on chmod
Posted by bfields on Sun, 16 Sep 2007 19:41:35 GMT
View Forum Message <> Reply to Message

On Thu, Sep 13, 2007 at 06:30:43PM +0400, Pavel Emelyanov wrote:
> 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.

I suppose so. Does anyone actually use mandatory locking?

Would it be worth adding a

```
 if (MANDATORY_LOCK(inode))
  return;
```

to the beginning of locks_wakeup_mandatory() to avoid walking the list
of locks in that case? Perhaps setattr is rare enough that this just
isn't worth caring about.

Is there a small chance that a lock may be applied after this check:

> + mandatory = (inode->i_flock && MANDATORY_LOCK(inode));
> +

but early enough that someone can still block on the lock while the file
is still marked for mandatory locking? (And is the inode->i_flock check
there really necessary?)

Well, there are probably worse races in the mandatory locking code.
(For example, my impression is that a mandatory lock can be applied just
after the locks_mandatory_area() checks but before the io actually
completes.)

--b.

```
>   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 83ba887..c0c2281 100644
> --- a/fs/locks.c
> +++ b/fs/locks.c
> @@ -1106,7 +1106,8 @@ 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
> @@ -1125,6 +1126,20 @@ 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 035ffda..af0637f 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