Subject: [PATCH] Wake up mandatory locks waiter on chmod
Posted by Pavel Emelianov on Thu, 13 Sep 2007 14:30:43 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.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

```
---

diff --git a/fs/attr.c b/fs/attr.c
index ae58bd3..7a8506f 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 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
```

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

Subject: Re: [PATCH] Wake up mandatory locks waiter on chmod
Posted by Pavel Emelianov on Mon, 17 Sep 2007 06:37:56 GMT

View Forum Message <> Reply to Message

J. Bruce Fields wrote:
> 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?

:) Good question.

> 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?)

There is, but as you have noticed:

> Well, there are probably worse races in the mandatory locking code.

...there are. The inode->i_lock is protected with lock_kernel() only
and is not in sync with any other checks for inodes. This is sad :(
but a good locking for locks is to be done...

> (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.

Thanks,
Pavel

Subject: Re: [PATCH] Wake up mandatory locks waiter on chmod
Posted by bfields on Mon, 17 Sep 2007 14:59:34 GMT
View Forum Message <> Reply to Message

On Mon, Sep 17, 2007 at 10:37:56AM +0400, Pavel Emelyanov wrote:
> J. Bruce Fields wrote:
> > 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?)
>
> There is, but as you have noticed:

OK, but why not just remove the inode->i_flock check there? I can't see
how it helps anyway.

> > Well, there are probably worse races in the mandatory locking code.
>
> ...there are. The inode->i_lock is protected with lock_kernel() only
> and is not in sync with any other checks for inodes. This is sad :(
> but a good locking for locks is to be done...

I would also prefer a locking scheme that didn't rely on the BKL. That
said, except for this race:

> > (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.)

... I'm not aware of other races in the existing file-locking code. It
sounds like you might be. Could you give specific examples?

--b.

Subject: Re: [PATCH] Wake up mandatory locks waiter on chmod
Posted by Pavel Emelianov on Tue, 18 Sep 2007 06:36:32 GMT

J. Bruce Fields wrote:
> On Mon, Sep 17, 2007 at 10:37:56AM +0400, Pavel Emelyanov wrote:
>> J. Bruce Fields wrote:
>>> 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?)
>> There is, but as you have noticed:
>
> OK, but why not just remove the inode->i_flock check there? I can't see
> how it helps anyway.
>
>>> Well, there are probably worse races in the mandatory locking code.
>> ...there are. The inode->i_lock is protected with lock_kernel() only
>> and is not in sync with any other checks for inodes. This is sad :(
>> but a good locking for locks is to be done...
>
> I would also prefer a locking scheme that didn't rely on the BKL. That
> said, except for this race:

I would as well :) But I don't know the locking code good enough to
start fixing. Besides, even if I send a patch series that handles this,
I don't think that anyone will accept it, due to "this changes too much
code", "can you prove you fixed all the places" and so on...

>>> (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.)
>
> ... I'm not aware of other races in the existing file-locking code. It
> sounds like you might be. Could you give specific examples?

Well, there's a long standing BUG in leases code - when we made all the
checks in inserting lease, we call the locks_alloc_lock() and may fall
asleep. Bu after the wakeup nobody re-checks for the things to change.

I suspect there are other bad places.

> --b.
>

Subject: Re: [PATCH] Wake up mandatory locks waiter on chmod

Posted by bfields on Wed, 19 Sep 2007 18:07:49 GMT
View Forum Message <> Reply to Message

On Tue, Sep 18, 2007 at 10:36:32AM +0400, Pavel Emelyanov wrote:
> J. Bruce Fields wrote:
> > I would also prefer a locking scheme that didn't rely on the BKL. That
> > said, except for this race:
>
> I would as well :) But I don't know the locking code good enough to
> start fixing. Besides, even if I send a patch series that handles this,
> I don't think that anyone will accept it, due to "this changes too much
> code", "can you prove you fixed all the places" and so on...

Several people have expressed interest in a locking scheme for locks.c
(and probably lockd) that doesn't depend on BKL, so I don't think it
would be ignored. But, yes, it would have to be done very carefully;
there have been at least one or two previous attempts that failed.

> >>> (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.)
> >
> > ... I'm not aware of other races in the existing file-locking code. It
> > sounds like you might be. Could you give specific examples?
>
> Well, there's a long standing BUG in leases code - when we made all the
> checks in inserting lease, we call the locks_alloc_lock() and may fall
> asleep. Bu after the wakeup nobody re-checks for the things to change.

Ouch, yes, you're right.

> I suspect there are other bad places.

OK. Thanks in advance for finding any!

--b.

Subject: Re: [PATCH] Wake up mandatory locks waiter on chmod
Posted by Trond Myklebust on Wed, 19 Sep 2007 18:16:13 GMT
View Forum Message <> Reply to Message

On Wed, 2007-09-19 at 14:07 -0400, J. Bruce Fields wrote:
> On Tue, Sep 18, 2007 at 10:36:32AM +0400, Pavel Emelyanov wrote:
> > J. Bruce Fields wrote:
> > > I would also prefer a locking scheme that didn't rely on the BKL. That
> > > said, except for this race:
> >

> > I would as well :) But I don't know the locking code good enough to
> > start fixing. Besides, even if I send a patch series that handles this,
> > I don't think that anyone will accept it, due to "this changes too much
> > code", "can you prove you fixed all the places" and so on...
>
> Several people have expressed interest in a locking scheme for locks.c
> (and probably lockd) that doesn't depend on BKL, so I don't think it
> would be ignored. But, yes, it would have to be done very carefully;
> there have been at least one or two previous attempts that failed.

Another long-term project might be to convert the current list of locks
into a more scalable structure: something like an rbtree might be more
appropriate for really large numbers of locks.

Cheers
  Trond