Subject: Re: [PATCH] Memory shortage can result in inconsistent flocks state
Posted by bfields on Thu, 13 Sep 2007 19:34:39 GMT
View Forum Message <> Reply to Message

```
On Thu, Sep 13, 2007 at 03:27:08PM -0400, Chuck Ebbert wrote:
> On 09/11/2007 08:38 AM, Pavel Emelyanov wrote:
> > diff --git a/fs/locks.c b/fs/locks.c
> > index 0db1a14..f59d066 100644
> > --- a/fs/locks.c
> > +++ b/fs/locks.c
> > @@ -732,6 +732,14 @@ static int flock_lock_file(struct file *
> >   lock_kernel();
> >   if (request->fl_flags & FL_ACCESS)
> >    goto find_conflict;
> > +
> > + if (request->fl_type != F_UNLCK) {
> > +  error = -ENOMEM;
> > +  new_fl = locks_alloc_lock();
> > +  if (new_fl == NULL)
> > +   goto out;
> > + }
> > +
> >   for_each_lock(inode, before) {
> >    struct file_lock *fl = *before;
> >    if (IS_POSIX(fl))
> > @@ -753,10 +761,6 @@ static int flock_lock_file(struct file *
> >    goto out;
> >   }
> >
> > - error = -ENOMEM;
> > - new_fl = locks_alloc_lock();
> > - if (new_fl == NULL)
> > -  goto out;
> >   /*
> >    * If a higher-priority process was blocked on the old file lock,
> >    * give it the opportunity to lock the file.
>
> Doesn't that create a leak in some cases?
>
> >         for_each_lock(inode, before) {
> >                 struct file_lock *fl = *before;
> >                 if (IS_POSIX(fl))
> >                         break;
> >                 if (IS_LEASE(fl))
> >                         continue;
> >                 if (filp != fl->fl_file)
> >                         continue;
> >                 if (request->fl_type == fl->fl_type)
> >                 goto out;  <<<<<<<<<<<<<<<<< LEAK?
```

You mean, a leak of the memory allocated for new_fl? That's freed at the exit labeled with "out". It's the only exit:

```
 out:
        unlock_kernel();
  if (new_fl)
   locks_free_lock(new_fl);
  return error;
```

And new_fl is initially NULL, assigned only once by the allocation, then assigned to NULL only at the very end when we know we've succeeded.

Am I missing something else?

--b.

```
> >             found = 1;
> >             locks_delete_lock(before);
> >             break;
> >       }
```