Subject: [PATCH] Fix potential OOPS in generic_setlease()
Posted by Pavel Emelianov on Wed, 19 Sep 2007 14:26:05 GMT
View Forum Message <> Reply to Message

This code is run under lock_kernel(), which is dropped during
sleeping operations, so the following race is possible:

```
CPU1:                          CPU2:
  vfs_setlease();                vfs_setlease();
  lock_kernel();
                               lock_kernel(); /* spin */
  generic_setlease():
    ...
    for (before = ...)
    /* here we found some lease after
     * which we will insert the new one
     */
    fl = locks_alloc_lock();
    /* go to sleep in this allocation and
     * drop the BKL
     */
                               generic_setlease():
                                 ...
                                 for (before = ...)
                                 /* here we find the "before" pointing
                                  * at the one we found on CPU1
                                  */
                                ->fl_change(my_before, arg);
                                      lease_modify();
                                            locks_free_lock();
                                            /* and we freed it */
                               ...
                               unlock_kernel();
    locks_insert_lock(before, fl);
    /* OOPS! We have just tried to add the lease
     * at the tail of already removed one
     */
```

The similar races are already handled in other code - all the
allocations are performed before any checks/updates.

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

```
diff --git a/fs/locks.c b/fs/locks.c
index 5fa072a..227926e 100644
--- a/fs/locks.c
```

```
+++ b/fs/locks.c
@@ -1341,7 +1341,7 @@ int fcntl_getlease(struct file *filp)
  */
 int generic_setlease(struct file *filp, long arg, struct file_lock **flp)
 {
- struct file_lock *fl, **before, **my_before = NULL, *lease;
+ struct file_lock *fl = NULL, **before, **my_before = NULL, *lease;
  struct dentry *dentry = filp->f_path.dentry;
  struct inode *inode = dentry->d_inode;
  int error, rdlease_count = 0, wrlease_count = 0;
@@ -1368,6 +1368,11 @@ int generic_setlease(struct file *filp,
   || (atomic_read(&inode->i_count) > 1)))
   goto out;

+ error = -ENOMEM;
+ fl = locks_alloc_lock();
+ if (fl == NULL)
+  goto out;
+
  /*
   * At this point, we know that if there is an exclusive
   * lease on this file, then we hold it on this filp
@@ -1410,18 +1415,16 @@ int generic_setlease(struct file *filp,
  if (!leases_enable)
   goto out;

- error = -ENOMEM;
- fl = locks_alloc_lock();
- if (fl == NULL)
-  goto out;
-
  locks_copy_lock(fl, lease);

  locks_insert_lock(before, fl);

  *flp = fl;
- error = 0;
+ return 0;
+
 out:
+ if (fl != NULL)
+  locks_free_lock(fl);
  return error;
 }
 EXPORT_SYMBOL(generic_setlease);
```

Subject: Re: [PATCH] Fix potential OOPS in generic_setlease()
Posted by bfields on Wed, 19 Sep 2007 19:30:01 GMT
View Forum Message <> Reply to Message

On Wed, Sep 19, 2007 at 06:26:05PM +0400, Pavel Emelyanov wrote:
> This code is run under lock_kernel(), which is dropped during
> sleeping operations, so the following race is possible:
>
> CPU1:   CPU2:
>   vfs_setlease();   vfs_setlease();
>   lock_kernel();
>   lock_kernel(); /* spin */
>   generic_setlease():
>   ...
>   for (before = ...)
>   /* here we found some lease after
>   * which we will insert the new one
>   */
>   fl = locks_alloc_lock();
>   /* go to sleep in this allocation and
>   * drop the BKL
>   */
>   generic_setlease():
>   ...
>   for (before = ...)
>   /* here we find the "before" pointing
>   * at the one we found on CPU1
>   */
>   ->fl_change(my_before, arg);
>   lease_modify();
>   locks_free_lock();
>   /* and we freed it */
>   ...
>   unlock_kernel();
>   locks_insert_lock(before, fl);
>   /* OOPS! We have just tried to add the lease
>   * at the tail of already removed one
>   */

Thanks for spotting this!

But--careful-- it looks like "fl" is also used as a temporary variable
in a loop between the new and old location of that allocation. Isn't
that a bug?

--b.

>
> The similar races are already handled in other code - all the

> allocations are performed before any checks/updates.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
>
> ---
>
> diff --git a/fs/locks.c b/fs/locks.c
> index 5fa072a..227926e 100644
> --- a/fs/locks.c
> +++ b/fs/locks.c
> @@ -1341,7 +1341,7 @@ int fcntl_getlease(struct file *filp)
> */
> int generic_setlease(struct file *filp, long arg, struct file_lock **flp)
> {
> - struct file_lock *fl, **before, **my_before = NULL, *lease;
> + struct file_lock *fl = NULL, **before, **my_before = NULL, *lease;
> struct dentry *dentry = filp->f_path.dentry;
> struct inode *inode = dentry->d_inode;
> int error, rdlease_count = 0, wrlease_count = 0;
> @@ -1368,6 +1368,11 @@ int generic_setlease(struct file *filp,
> || (atomic_read(&inode->i_count) > 1)))
> goto out;
>
> + error = -ENOMEM;
> + fl = locks_alloc_lock();
> + if (fl == NULL)
> + goto out;
> +
> /*
> * At this point, we know that if there is an exclusive
> * lease on this file, then we hold it on this filp
> @@ -1410,18 +1415,16 @@ int generic_setlease(struct file *filp,
> if (!leases_enable)
> goto out;
>
> - error = -ENOMEM;
> - fl = locks_alloc_lock();
> - if (fl == NULL)
> - goto out;
> -
> locks_copy_lock(fl, lease);
>
> locks_insert_lock(before, fl);
>
> *flp = fl;
> - error = 0;
> + return 0;
> +

```
>  out:
> + if (fl != NULL)
> +  locks_free_lock(fl);
>   return error;
>  }
>  EXPORT_SYMBOL(generic_setlease);
```

Subject: Re: [PATCH] Fix potential OOPS in generic_setlease()
Posted by Pavel Emelianov on Thu, 20 Sep 2007 08:38:48 GMT
View Forum Message <> Reply to Message

J. Bruce Fields wrote:
> On Wed, Sep 19, 2007 at 06:26:05PM +0400, Pavel Emelyanov wrote:
>> This code is run under lock_kernel(), which is dropped during
>> sleeping operations, so the following race is possible:
>>
>> CPU1:   CPU2:
>>   vfs_setlease();   vfs_setlease();
>>   lock_kernel();
>>   lock_kernel(); /* spin */
>>   generic_setlease():
>>   ...
>>   for (before = ...)
>>   /* here we found some lease after
>>   * which we will insert the new one
>>   */
>>   fl = locks_alloc_lock();
>>   /* go to sleep in this allocation and
>>   * drop the BKL
>>   */
>>   generic_setlease():
>>   ...
>>   for (before = ...)
>>   /* here we find the "before" pointing
>>   * at the one we found on CPU1
>>   */
>>   ->fl_change(my_before, arg);
>>   lease_modify();
>>   locks_free_lock();
>>   /* and we freed it */
>>   ...
>>   unlock_kernel();
>>   locks_insert_lock(before, fl);
>>   /* OOPS! We have just tried to add the lease
>>   * at the tail of already removed one
>>   */
>
```
```

> Thanks for spotting this!
>
> But--careful-- it looks like "fl" is also used as a temporary variable
> in a loop between the new and old location of that allocation. Isn't
> that a bug?

OOPS! Good catch, thanks. I will resend the patch shortly.
