Subject: [PATCH] Fix potential OOPS in generic_setlease() (v2)
Posted by Pavel Emelianov on Thu, 20 Sep 2007 08:48:32 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.

Fixed the problem, spotted by J. Bruce Fields, about the fl
variable reuse.

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

```
diff --git a/fs/locks.c b/fs/locks.c
index a1c1c01..d5b9653 100644
--- a/fs/locks.c
+++ b/fs/locks.c
@@ -1354,6 +1354,7 @@ 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 *new_fl = NULL;
  struct dentry *dentry = filp->f_path.dentry;
  struct inode *inode = dentry->d_inode;
  int error, rdlease_count = 0, wrlease_count = 0;
@@ -1380,6 +1381,11 @@ int generic_setlease(struct file *filp,
   || (atomic_read(&inode->i_count) > 1)))
   goto out;

+ error = -ENOMEM;
+ new_fl = locks_alloc_lock();
+ if (new_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
@@ -1422,18 +1428,15 @@ 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);
+ locks_copy_lock(new_fl, lease);
+ locks_insert_lock(before, new_fl);

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

EXPORT_SYMBOL(generic_setlease);

Subject: Re: [PATCH] Fix potential OOPS in generic_setlease() (v2)
Posted by bfields on Thu, 20 Sep 2007 20:36:12 GMT
View Forum Message <> Reply to Message

OK, this version I can't see any more problem with. Thanks!

--b.

On Thu, Sep 20, 2007 at 12:48:32PM +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
> */
>
> The similar races are already handled in other code - all the
> allocations are performed before any checks/updates.

```
>
> Fixed the problem, spotted by J. Bruce Fields, about the fl
> variable reuse.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
>
> ---
>
> diff --git a/fs/locks.c b/fs/locks.c
> index a1c1c01..d5b9653 100644
> --- a/fs/locks.c
> +++ b/fs/locks.c
> @@ -1354,6 +1354,7 @@ 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 *new_fl = NULL;
>   struct dentry *dentry = filp->f_path.dentry;
>   struct inode *inode = dentry->d_inode;
>   int error, rdlease_count = 0, wrlease_count = 0;
> @@ -1380,6 +1381,11 @@ int generic_setlease(struct file *filp,
>    || (atomic_read(&inode->i_count) > 1)))
>    goto out;
>
> + error = -ENOMEM;
> + new_fl = locks_alloc_lock();
> + if (new_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
> @@ -1422,18 +1428,15 @@ 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);
> + locks_copy_lock(new_fl, lease);
> + locks_insert_lock(before, new_fl);
>
>   *flp = fl;
```

> - error = 0;
> + return 0;
> +
> out:
> + if (new_fl != NULL)
> + locks_free_lock(new_fl);
>   return error;
> }
> EXPORT_SYMBOL(generic_setlease);
