

---

Subject: Re: [PATCH v5] Fix rmmod/read/write races in /proc entries

Posted by [Alexey Dobriyan](#) on Fri, 16 Mar 2007 09:09:42 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

On Thu, Mar 15, 2007 at 05:53:04PM -0800, Andrew Morton wrote:

> My, what a lot of code you have here. I note that nobody can be assed even  
> reviewing it. Now why is that?

I hope, Al could find some time again.

> On Sun, 11 Mar 2007 20:04:56 +0300 Alexey Dobriyan <adobriyan@sw.ru> wrote:

> > Fix following races:

> > =====

> > 1. Write via ->write\_proc sleeps in copy\_from\_user(). Module disappears  
> > meanwhile. Or, more generically, system call done on /proc file, method  
> > supplied by module is called, module dissapeares meanwhile.

> >

> > pde = create\_proc\_entry()

> > if (!pde)

> > return -ENOMEM;

> > pde->write\_proc = ...

> > open

> > write

> > copy\_from\_user

> > pde = create\_proc\_entry();

> > if (!pde) {

> > remove\_proc\_entry();

> > return -ENOMEM;

> > /\* module unloaded \*/

> > }

>

> We usually fix that race by pinning the module: make whoever registered the

> proc entries also register their THIS\_MODULE, do a try\_module\_get() on it

> before we start to play with data structures which the module owns.

>

> Can we do that here?

We can, but it will be unreliable:

Typical proc entry creation sequence is

```
pde = create_proc_entry(...);
```

```
if (pde)
```

```
    pde->owner = THIS_MODULE;
```

Right after create\_proc\_entry() ->owner is NULL, so try\_module\_get()  
won't do anything, but proc\_delete\_inode() could put module which was  
never gotten.

This should be fixable by always setting `->owner` before proc entry is glued to proc entries tree. Something like this:

```
#define create_proc_entry(...) __create_proc_entry(..., THIS_MODULE)
```

However, I think it's not enough: `delete_module(2)` first waits for refcount becoming zero, only then calls module's exit function which starts removing proc entries. In between, proc entries are accessible and fully-functional, so `try_module_get()` can again get module and `module_put(pde->owner)` can happen AFTER module disappears. What will it put?

And how can you fix that? The only way I know is to REMOVE `->owner` completely, once we agree on this `pde_users/pde_unload_lock` stuff.

> And is the above race fix related to the below one in any fashion?

```
> > =====
```

```
> > 2. bogo-revoke aka proc_kill_inodes()
```

```
> >
```

```
> > remove_proc_entry vfs_read
```

```
> > proc_kill_inodes [check ->f_op validness]
```

```
> > [check ->f_op->read validness]
```

```
> > [verify_area, security permissions checks]
```

```
> > ->f_op = NULL;
```

```
> > if (file->f_op->read)
```

```
> > /* ->f_op dereference, boom */
```

```
>
```

> So you fixed this via sort-of-refcounting on `pde->pde_users`.

```
>
```

> hmm.

Probably, you're right and they are independently fixable. It's all about following 3 lines after all. My turn to hmm...

```
> > - proc_kill_inodes(de);
```

```
> > + if (!S_ISREG(de->mode))
```

```
> > + proc_kill_inodes(de);
```

```
> > + spin_lock(&pde->pde_unload_lock);
```

```
> > + pde->pde_users--;
```

```
> > + if (pde->pde_unload_completion && pde->pde_users == 0)
```

```
> > + complete(pde->pde_unload_completion);
```

```
> > +out_unlock:
```

```
> > + spin_unlock(&pde->pde_unload_lock);
```

```
>
```

> The above six lines happen rather a lot - perhaps it could be placed in a  
> helper function?

OK. Should I send incremental updates or full patch again?

---