
Subject: Re: NFS causing oops when freeing namespace
Posted by [Cedric Le Goater](#) on Wed, 17 Jan 2007 19:20:37 GMT
[View Forum Message](#) <> [Reply to Message](#)

Oleg Nesterov wrote:

> On 01/17, Daniel Hokka Zakrisson wrote:

>>>> Call Trace:

>>>> [`<c03be6f0>`] `_spin_lock_irqsave+0x20/0x90`

>>>> [`<c01f6115>`] `lockd_down+0x125/0x190`

>>>> [`<c01d26bd>`] `nfs_free_server+0x6d/0xd0`

>>>> [`<c01d8e9c>`] `nfs_kill_super+0xc/0x20`

>>>> [`<c0161c5d>`] `deactivate_super+0x7d/0xa0`

>>>> [`<c0175e0e>`] `release_mounts+0x6e/0x80`

>>>> [`<c0175e86>`] `__put_mnt_ns+0x66/0x80`

>>>> [`<c0132b3e>`] `free_nsproxy+0x5e/0x60`

>>>> [`<c011f021>`] `do_exit+0x791/0x810`

>>>> [`<c011f0c6>`] `do_group_exit+0x26/0x70`

>>>> [`<c0103142>`] `sysenter_past_esp+0x5f/0x85`

>>>> [`<c03b0033>`] `rpc_wake_up+0x3/0x70`

>> It was the only semi-plausible explanation I could come up with. I added a

>> `printk` in `do_exit` right before `exit_task_namespaces`, where `sighand` was

>> still set, and one right before the `spin_lock_irq` in `lockd_down`, where it

>> had suddenly been set to `NULL`.

>

> I can't reproduce the problem, but

I did on a 2.6.20-rc4-mm1.

> `do_exit`:

> `exit_notify(tsk);`

> `exit_task_namespaces(tsk);`

>

> the task could be reaped by its parent in between.

indeed. while it goes sleeping in `lockd_down()` just before it does

```
spin_lock_irq(&current->sighand->siglock);
```

`current->sighand` is valid before `interruptible_sleep_on_timeout()` and not after.

> We should not use `->signal/->sighand` after `exit_notify()`.

>

> Can we move `exit_task_namespaces()` up?

yes but I moved it down because it invalidates `->nsproxy` ...

C.

Subject: Re: NFS causing oops when freeing namespace
Posted by [Oleg Nesterov](#) on Wed, 17 Jan 2007 19:46:32 GMT
[View Forum Message](#) <> [Reply to Message](#)

On 01/17, Cedric Le Goater wrote:

>
> Oleg Nesterov wrote:
> > On 01/17, Daniel Hokka Zakrisson wrote:
> >> It was the only semi-plausible explanation I could come up with. I added a
> >> printk in do_exit right before exit_task_namespaces, where sighand was
> >> still set, and one right before the spin_lock_irq in lockd_down, where it
> >> had suddenly been set to NULL.
> >
> > I can't reproduce the problem, but
>
> I did on a 2.6.20-rc4-mm1.
>
> > do_exit:
> > exit_notify(tsk);
> > exit_task_namespaces(tsk);
> >
> > the task could be reaped by its parent in between.
>
> indeed. while it goes sleeping in lockd_down() just before it does
>
> spin_lock_irq(¤t->sighand->siglock);
>
> current->sighand is valid before interruptible_sleep_on_timeout() and
> not after.
>
> > We should not use ->signal/->sighand after exit_notify().
> >
> > Can we move exit_task_namespaces() up?
>
> yes but I moved it down because it invalidates ->nsproxy ...

Well, we can fix the symptom if we change lockd_down() to use
lock_task_sighand(), or something like this,

```
--- NFS/fs/lockd/svc.c~lockd_down 2006-11-27 21:20:11.000000000 +0300
+++ NFS/fs/lockd/svc.c 2007-01-17 22:39:47.000000000 +0300
```

```

@@ -314,6 +314,7 @@ void
lockd_down(void)
{
    static int warned;
+   int sigpending;

    mutex_lock(&nlmsvc_mutex);
    if (nlmsvc_users) {
@@ -334,16 +335,15 @@ lockd_down(void)
    * Wait for the lockd process to exit, but since we're holding
    * the lockd semaphore, we can't wait around forever ...
    */
-   clear_thread_flag(TIF_SIGPENDING);
+   sigpending = test_and_clear_thread_flag(TIF_SIGPENDING);
    interruptible_sleep_on_timeout(&lockd_exit, HZ);
    if (nlmsvc_pid) {
        printk(KERN_WARNING
            "lockd_down: lockd failed to exit, clearing pid\n");
        nlmsvc_pid = 0;
    }
-   spin_lock_irq(&current->sigband->siglock);
-   recalc_sigpending();
-   spin_unlock_irq(&current->sigband->siglock);
+   if (sigpending) /* can be wrong at this point, harmless */
+   set_thread_flag(TIF_SIGPENDING);
    out:
    mutex_unlock(&nlmsvc_mutex);
}

```

but this is not good anyway.

Oleg.

Containers mailing list
Containers@lists.osdl.org
<https://lists.osdl.org/mailman/listinfo/containers>
