Subject: Re: NFS causing oops when freeing namespace
Posted by ebiederm on Wed, 17 Jan 2007 12:44:10 GMT
View Forum Message <> Reply to Message

"Daniel Hokka Zakrisson" <daniel@hozac.com> writes:

> The test-case at the bottom causes the following recursive Oopsing on
> 2.6.20-rc5:

A few more people added to the CC who might have a clue.

>
> BUG: unable to handle kernel NULL pointer dereference at virtual address
> 00000504
> printing eip:
> c02292d4
> *pde = 00000000
> Oops: 0002 [#1]
> PREEMPT SMP
> Modules linked in:
> CPU: 0
> EIP: 0060:[<c02292d4>]   Not tainted VLI
> EFLAGS: 00010046   (2.6.20-rc5 #11)
> EIP is at _raw_spin_trylock+0x4/0x30
> eax: 00000000   ebx: 00000213   ecx: 00000001   edx: 00000504
> esi: 00000504   edi: c1493da0   ebp: d5b5ff3c   esp: d5b5fee0
> ds: 007b   es: 007b   ss: 0068
> Process mount (pid: 2618, ti=d5b5e000 task=c17fdaa0 task.ti=d5b5e000)
> Stack: c03be6f0 00000000 00000000 c01f6115 c04240dc c17fdaa0 c17fdc2c
> 00000000
>   00000000 d555ebe0 c04734a0 c01d26bd c01d8e9c d553b800 c0161c5d
> d707443c
>   d54cedc0 c0175e0e d779c9a0 d779cd60 c1493f20 d7f1caa0 c0175e86
> d54cee40
> 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
> ======================

> Code: ff ff c3 8d 74 26 00 c7 00 00 00 00 01 c7 40 08 ed 1e af de c7 40 10
> ff ff ff ff c7 40 0c ff ff ff ff c3 8d 74 26 00 89 c2 31 c0 <86> 02 31 c9
> 84 c0 0f 9f c1 85 c9 74 12 65 a1 04 00 00 00 89 42
> EIP: [<c02292d4>] _raw_spin_trylock+0x4/0x30 SS:ESP 0068:d5b5fee0
> <1>BUG: unable to handle kernel NULL pointer dereference at virtual
> address 00000024
> printing eip:
> c011e8dd
> *pde = 00000000
> Oops: 0000 [#2]
> PREEMPT SMP
> Modules linked in:
> CPU: 0
> EIP: 0060:[<c011e8dd>]   Not tainted VLI
> EFLAGS: 00010006   (2.6.20-rc5 #11)
> EIP is at do_exit+0x4d/0x810
> eax: 00000000   ebx: d5b5fea8   ecx: c17fdaa0   edx: 00000001
> esi: c17fdaa0   edi: 00000a3a   ebp: 0000000b   esp: d5b5fde4
> ds: 007b   es: 007b   ss: 0068
> Process mount (pid: 2618, ti=d5b5e000 task=c17fdaa0 task.ti=d5b5e000)
> Stack: 00000a3a d5b5e000 c17fdaa0 d5b5e000 00000003 c046a307 0000000f
> d5b5fee0
>   00000068 00000013 c011c3cb c042b321 d5b5fe24 d5b5fea8 d5b5fee0
> 00000068
>   00000000 c0104816 c041347f 00000068 d5b5fee0 00000001 d5b5fea8
> c0414df5
> Call Trace:
> [<c011c3cb>] printk+0x1b/0x20
> [<c0104816>] die+0x246/0x260
> [<c03c09b0>] do_page_fault+0x2e0/0x630
> [<c011c2cf>] vprintk+0x28f/0x370
> [<c03c06d0>] do_page_fault+0x0/0x630
> [<c03beea4>] error_code+0x7c/0x84
> [<c03b007b>] rpc_wake_up+0x4b/0x70
> [<c02292d4>] _raw_spin_trylock+0x4/0x30
> [<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
> ======================

> Code: 03 00 00 89 e0 25 00 e0 ff ff f7 40 14 00 ff ff 0f 0f 85 69 03 00 00
> 8b be a4 00 00 00 85 ff 0f 84 43 03 00 00 8b 86 48 04 00 00 <8b> 50 24 3b
> 72 10 0f 84 1c 03 00 00 65 a1 08 00 00 00 f6 40 11
> EIP: [<c011e8dd>] do_exit+0x4d/0x810 SS:ESP 0068:d5b5fde4
>
> It seems to go on forever. addr2line output follows:
> $ addr2line -e vmlinux c02292d4 c03be6f0 c01f6115 c01d26bd c01d8e9c
> c0161c5d c0175e0e c0175e86 c0132b3e c011f021 c011e8dd
> include/asm/spinlock.h:90
> kernel/spinlock.c:280
> fs/lockd/svc.c:345
> fs/nfs/client.c:782
> fs/nfs/super.c:679
> fs/super.c:184
> include/linux/list.h:299
> fs/namespace.c:1879
> include/linux/mnt_namespace.h:26
> include/linux/nsproxy.h:42
> include/linux/pid_namespace.h:42
>
> The problem appears to be that fs/lockd/svc.c:lockd_down tries to lock
> current->sighand->siglock, but during the time it's sleeping the process
> is reaped and released, setting sighand to NULL.
>
> This also affects 2.6.19.2, but since there are no pid spaces there, it's
> "just" a single oops, and doesn't make the machine totally unusable.

Ugh. This is why I have been deliberately trying to finish the kernel_thread
to kthread conversion before I finished the pid namespace. It looks
like whatever part of the pid namespace made it into nsproxy for
2.6.20-rcX may need to come out if it is causing problems, but
since it is always shared that doesn't seem right.

I find it weird that you are describing current as having exited,
while it is sleeping.

sighand in the task_struct is non-null is good until release_task is
called.

I don't think we are reapable at the point so sighand going NULL
shouldn't cause problems.

Thanks for the test-case. If it is as easy to reproduce as this looks
we should be able to track this down fairly quickly.

> Test-case, execute as ./a.out mount -t nfs ...:
> #include <stdio.h>
> #include <sched.h>

```
> #include <unistd.h>
>
> int main(int argc, char *argv[])
> {
>  if (unshare(CLONE_NEWNS) == -1) {
>   perror("unshare");
>   exit(1);
>  }
>  execv("/bin/mount", argv+1);
>  perror("execv(mount)");
>  return 1;
> }
>
```

Now back to bed with me.

Eric

Subject: Re: NFS causing oops when freeing namespace
Posted by Daniel Hokka Zakrisso on Wed, 17 Jan 2007 13:13:53 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> "Daniel Hokka Zakrisson" <daniel@hozac.com> writes:
>
>> The test-case at the bottom causes the following recursive Oopsing on
>> 2.6.20-rc5:
>
> A few more people added to the CC who might have a clue.
>
>>
>> BUG: unable to handle kernel NULL pointer dereference at virtual address
>> 00000504
>> printing eip:
>> c02292d4
>> *pde = 00000000
>> Oops: 0002 [#1]
>> PREEMPT SMP
>> Modules linked in:
>> CPU: 0
>> EIP: 0060:[<c02292d4>]   Not tainted VLI
>> EFLAGS: 00010046   (2.6.20-rc5 #11)
>> EIP is at _raw_spin_trylock+0x4/0x30

>> eax: 00000000 ebx: 00000213 ecx: 00000001 edx: 00000504
>> esi: 00000504 edi: c1493da0 ebp: d5b5ff3c esp: d5b5fee0
>> ds: 007b es: 007b ss: 0068
>> Process mount (pid: 2618, ti=d5b5e000 task=c17fdaa0 task.ti=d5b5e000)
>> Stack: c03be6f0 00000000 00000000 c01f6115 c04240dc c17fdaa0 c17fdc2c
>> 00000000
>> 00000000 d555ebe0 c04734a0 c01d26bd c01d8e9c d553b800 c0161c5d
>> d707443c
>> d54cedc0 c0175e0e d779c9a0 d779cd60 c1493f20 d7f1caa0 c0175e86
>> d54cee40
>> 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
>> ========================
>> Code: ff ff c3 8d 74 26 00 c7 00 00 00 00 01 c7 40 08 ed 1e af de c7 40
>> 10
>> ff ff ff ff c7 40 0c ff ff ff ff c3 8d 74 26 00 89 c2 31 c0 <86> 02 31
>> c9
>> 84 c0 0f 9f c1 85 c9 74 12 65 a1 04 00 00 00 89 42
>> EIP: [<c02292d4>] _raw_spin_trylock+0x4/0x30 SS:ESP 0068:d5b5fee0
>> <1>BUG: unable to handle kernel NULL pointer dereference at virtual
>> address 00000024
>> printing eip:
>> c011e8dd
>> *pde = 00000000
>> Oops: 0000 [#2]
>> PREEMPT SMP
>> Modules linked in:
>> CPU: 0
>> EIP: 0060:[<c011e8dd>] Not tainted VLI
>> EFLAGS: 00010006 (2.6.20-rc5 #11)
>> EIP is at do_exit+0x4d/0x810
>> eax: 00000000 ebx: d5b5fea8 ecx: c17fdaa0 edx: 00000001
>> esi: c17fdaa0 edi: 00000a3a ebp: 0000000b esp: d5b5fde4
>> ds: 007b es: 007b ss: 0068
>> Process mount (pid: 2618, ti=d5b5e000 task=c17fdaa0 task.ti=d5b5e000)
>> Stack: 00000a3a d5b5e000 c17fdaa0 d5b5e000 00000003 c046a307 0000000f
>> d5b5fee0

>> 00000068 00000013 c011c3cb c042b321 d5b5fe24 d5b5fea8 d5b5fee0
>> 00000068
>> 00000000 c0104816 c041347f 00000068 d5b5fee0 00000001 d5b5fea8
>> c0414df5
>> Call Trace:
>> [<c011c3cb>] printk+0x1b/0x20
>> [<c0104816>] die+0x246/0x260
>> [<c03c09b0>] do_page_fault+0x2e0/0x630
>> [<c011c2cf>] vprintk+0x28f/0x370
>> [<c03c06d0>] do_page_fault+0x0/0x630
>> [<c03beea4>] error_code+0x7c/0x84
>> [<c03b007b>] rpc_wake_up+0x4b/0x70
>> [<c02292d4>] _raw_spin_trylock+0x4/0x30
>> [<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
>> =======================
>> Code: 03 00 00 89 e0 25 00 e0 ff ff f7 40 14 00 ff ff 0f 0f 85 69 03 00
>> 00
>> 8b be a4 00 00 00 85 ff 0f 84 43 03 00 00 8b 86 48 04 00 00 <8b> 50 24
>> 3b
>> 72 10 0f 84 1c 03 00 00 65 a1 08 00 00 00 f6 40 11
>> EIP: [<c011e8dd>] do_exit+0x4d/0x810 SS:ESP 0068:d5b5fde4
>>
>> It seems to go on forever. addr2line output follows:
>> $ addr2line -e vmlinux c02292d4 c03be6f0 c01f6115 c01d26bd c01d8e9c
>> c0161c5d c0175e0e c0175e86 c0132b3e c011f021 c011e8dd
>> include/asm/spinlock.h:90
>> kernel/spinlock.c:280
>> fs/lockd/svc.c:345
>> fs/nfs/client.c:782
>> fs/nfs/super.c:679
>> fs/super.c:184
>> include/linux/list.h:299
>> fs/namespace.c:1879
>> include/linux/mnt_namespace.h:26
>> include/linux/nsproxy.h:42
>> include/linux/pid_namespace.h:42
>>

>> The problem appears to be that fs/lockd/svc.c:lockd_down tries to lock
>> current->sighand->siglock, but during the time it's sleeping the process
>> is reaped and released, setting sighand to NULL.
>>
>> This also affects 2.6.19.2, but since there are no pid spaces there,
>> it's
>> "just" a single oops, and doesn't make the machine totally unusable.
>
> Ugh. This is why I have been deliberately trying to finish the
> kernel_thread
> to kthread conversion before I finished the pid namespace. It looks
> like whatever part of the pid namespace made it into nsproxy for
> 2.6.20-rcX may need to come out if it is causing problems, but
> since it is always shared that doesn't seem right.
>
> I find it weird that you are describing current as having exited,
> while it is sleeping.

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.

> sighand in the task_struct is non-null is good until release_task is
> called.
>
> I don't think we are reapable at the point so sighand going NULL
> shouldn't cause problems.
>
> Thanks for the test-case. If it is as easy to reproduce as this looks
> we should be able to track this down fairly quickly.
>
>> Test-case, execute as ./a.out mount -t nfs ...:
>> #include <stdio.h>
>> #include <sched.h>
>> #include <unistd.h>
>>
>> int main(int argc, char *argv[])
>> {
>> if (unshare(CLONE_NEWNS) == -1) {
>>   perror("unshare");
>>   exit(1);
>> }
>> execv("/bin/mount", argv+1);
>> perror("execv(mount)");
>> return 1;
>> }
>>

>
> Now back to bed with me.
>
> Eric
>

--
Daniel Hokka Zakrisson

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

Subject: Re: NFS causing oops when freeing namespace
Posted by Oleg Nesterov on Wed, 17 Jan 2007 18:58:23 GMT
View Forum Message <> Reply to Message

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

```
 do_exit:
  exit_notify(tsk);
  exit_task_namespaces(tsk);
```

the task could be reaped by its parent in between.

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

Can we move exit_task_namespaces() up?

Oleg.

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