Subject: [PATCH 2.6.16] Fix NULL bio crash in loop worker thread
Posted by Alexey Dobriyan on Fri, 01 Jun 2007 07:14:01 GMT
View Forum Message <> Reply to Message

After LOOP_SET_FD/LOOP_CLR_FD combo loop device's queue gets request
handler which is persistent.

After, say

```
mount -t iso9660 /dev/loop0 /mnt        # sic
```

this request handler is called directly with
a) ->lo_state being Lo_unbound
b) ->lo_pending being zero

Error path in loop_make_request() completes ->lo_bh_done completion
which is persistent as well.

Now, let's start worker thread as usual. It'll set ->lo_pending to 1,
don't wait for completion because it was already completed (brokenly),
and will not get out of infinite loop because of ->lo_pending. Loop
device doesn't have bios at this point and triggers BUG_ON.

So, don't complete ->lo_bh_done when loop device isn't setup fully.

Steps to reproduce:

```
#!/bin/sh -x
ISO=1.iso
mount -o loop $ISO /mnt
umount /mnt
mount -t iso9660 /dev/loop0 /mnt        # sic
mount -o loop $ISO /mnt
```

Signed-off-by: Alexey Dobriyan <adobriyan@sw.ru>

```
drivers/block/loop.c |    3 ++-
1 file changed, 2 insertions(+), 1 deletion(-)

--- a/drivers/block/loop.c
+++ b/drivers/block/loop.c
@@ -521,7 +521,7 @@ static int loop_make_request(request_queue_t *q, struct bio *old_bio)

  spin_lock_irq(&lo->lo_lock);
  if (lo->lo_state != Lo_bound)
-  goto out;
+  goto out_not_bound;
```

```
  if (unlikely(rw == WRITE && (lo->lo_flags & LO_FLAGS_READ_ONLY)))
   goto out;
  lo->lo_pending++;
@@ -533,6 +533,7 @@ static int loop_make_request(request_queue_t *q, struct bio *old_bio)
 out:
  if (lo->lo_pending == 0)
   complete(&lo->lo_bh_done);
+out_not_bound:
  spin_unlock_irq(&lo->lo_lock);
  bio_io_error(old_bio, old_bio->bi_size);
  return 0;
```