author	Pavel Begunkov <asml.silence@gmail.com>	2022-04-14 08:50:50 +0100
committer	Greg Kroah-Hartman <gregkh@linuxfoundation.org>	2022-04-15 14:18:41 +0200
commit	1a623d361ffe5cecd4244a02f449528416360038 (patch)	
tree	b99709904f01723c8475b87507637b4427b9c94c /fs/io_uring.c	
parent	33fcb359a64213bea0510770abc761de9780ca89 (diff)	
download	linux-1a623d361ffe5cecd4244a02f449528416360038.tar.gz	

io_uring: fix fs->users overflow

```
There is a bunch of cases where we can grab req->fs but not put it, this
can be used to cause a controllable overflow with further implications.
Release req->fs in the request free path and make sure we zero the field
to be sure we don't do it twice.

Fixes: cac68d12c531 ("io_uring: grab ->fs as part of async offload")
Reported-by: Bing-Jhong Billy Jheng <billy@starlabs.sg>
Signed-off-by: Pavel Begunkov <asml.silence@gmail.com>
Signed-off-by: Greg Kroah-Hartman <gregkh@linuxfoundation.org>
```

Diffstat (limited to 'fs/io_uring.c')

-rw-r--r-- fs/io_uring.c 28

1 files changed, 18 insertions, 10 deletions

```
diff --git a/fs/io_uring.c b/fs/io_uring.c
index 478df7e10767a..e73969fa96bcb 100644
--- a/fs/io_uring.c
+++ b/fs/io_uring.c
@@ -438,6 +438,22 @@ static struct io_ring_ctx *io_ring_ctx_alloc(struct io_uring_params *p)
        return ctx;
 }

+static void io_req_put_fs(struct io_kiocb *req)
+{
+       struct fs_struct *fs = req->fs;
+
+       if (!fs)
+               return;
+
+       spin_lock(&req->fs->lock);
+       if (--fs->users)
+               fs = NULL;
+       spin_unlock(&req->fs->lock);
+       if (fs)
+               free_fs_struct(fs);
+       req->fs = NULL;
+}
+
 static inline bool __io_sequence_defer(struct io_ring_ctx *ctx,
                                       struct io_kiocb *req)
 {
@@ -695,6 +711,7 @@ static void io_free_req_many(struct io_ring_ctx *ctx, void **reqs, int *nr)

 static void __io_free_req(struct io_kiocb *req)
 {
+       io_req_put_fs(req);
        if (req->file && !(req->flags & REQ_F_FIXED_FILE))
                fput(req->file);
        percpu_ref_put(&req->ctx->refs);
@@ -1701,16 +1718,7 @@ static int io_send_recvmsg(struct io_kiocb *req, const struct io_uring_sqe *sqe,
```

```
                        ret = -EINTR;
        }

-       if (req->fs) {
-               struct fs_struct *fs = req->fs;
-
-               spin_lock(&req->fs->lock);
-               if (--fs->users)
-                       fs = NULL;
-               spin_unlock(&req->fs->lock);
-               if (fs)
-                       free_fs_struct(fs);
-       }
+       io_req_put_fs(req);
        io_cqring_add_event(req->ctx, sqe->user_data, ret);
        io_put_req(req);
        return 0;
```