| | | |
|---|---|---|
| author | Baokun Li <libaokun1@huawei.com> | 2023-01-10 21:34:36 +0800 |
| committer | Greg Kroah-Hartman <gregkh@linuxfoundation.org> | 2023-03-22 13:37:56 +0100 |
| commit | 73f7987fe1b82596f1a380e85cd0097ebaae7e01 (patch) | |
| tree | c058e8fbc0c73e5cebb03029ff9969e27e0ce3e8 | |
| parent | ee0c5277d4fab920bd31345c49e193ecede9ecef (diff) | |
| download | linux-73f7987fe1b82596f1a380e85cd0097ebaae7e01.tar.gz | |



## ext4: fix task hung in ext4_xattr_delete_inode

```
[ Upstream commit 0f7bfd6f8164be32dbbdf36aa1e5d00485c53cd7 ]

Syzbot reported a hung task problem:
=====================================================================
INFO: task syz-executor232:5073 blocked for more than 143 seconds.
      Not tainted 6.2.0-rc2-syzkaller-00024-g512dee0c00ad #0
"echo 0 > /proc/sys/kernel/hung_task_timeout_secs" disables this message.
task:syz-exec232 state:D stack:21024 pid:5073 ppid:5072 flags:0x00004004
Call Trace:
 <TASK>
 context_switch kernel/sched/core.c:5244 [inline]
 __schedule+0x995/0xe20 kernel/sched/core.c:6555
 schedule+0xcb/0x190 kernel/sched/core.c:6631
 __wait_on_freeing_inode fs/inode.c:2196 [inline]
 find_inode_fast+0x35a/0x4c0 fs/inode.c:950
 iget_locked+0xb1/0x830 fs/inode.c:1273
 __ext4_iget+0x22e/0x3ed0 fs/ext4/inode.c:4861
 ext4_xattr_inode_iget+0x68/0x4e0 fs/ext4/xattr.c:389
 ext4_xattr_inode_dec_ref_all+0x1a7/0xe50 fs/ext4/xattr.c:1148
 ext4_xattr_delete_inode+0xb04/0xcd0 fs/ext4/xattr.c:2880
 ext4_evict_inode+0xd7c/0x10b0 fs/ext4/inode.c:296
 evict+0x2a4/0x620 fs/inode.c:664
 ext4_orphan_cleanup+0xb60/0x1340 fs/ext4/orphan.c:474
 __ext4_fill_super fs/ext4/super.c:5516 [inline]
 ext4_fill_super+0x81cd/0x8700 fs/ext4/super.c:5644
 get_tree_bdev+0x400/0x620 fs/super.c:1282
 vfs_get_tree+0x88/0x270 fs/super.c:1489
 do_new_mount+0x289/0xad0 fs/namespace.c:3145
 do_mount fs/namespace.c:3488 [inline]
 __do_sys_mount fs/namespace.c:3697 [inline]
 __se_sys_mount+0x2d3/0x3c0 fs/namespace.c:3674
 do_syscall_x64 arch/x86/entry/common.c:50 [inline]
 do_syscall_64+0x3d/0xb0 arch/x86/entry/common.c:80
 entry_SYSCALL_64_after_hwframe+0x63/0xcd
RIP: 0033:0x7fa5406fd5ea
RSP: 002b:00007ffc7232f968 EFLAGS: 00000202 ORIG_RAX: 00000000000000a5
RAX: ffffffffffffffda RBX: 0000000000000003 RCX: 00007fa5406fd5ea
RDX: 0000000020000440 RSI: 0000000020000000 RDI: 00007ffc7232f970
RBP: 00007ffc7232f970 R08: 00007ffc7232f9b0 R09: 0000000000000432
R10: 0000000000804a03 R11: 0000000000000202 R12: 0000000000000004
R13: 0000555556a7a2c0 R14: 00007ffc7232f9b0 R15: 0000000000000000
 </TASK>
```

```
================================================================

The problem is that the inode contains an xattr entry with ea_inum of 15
when cleaning up an orphan inode <15>. When evict inode <15>, the reference
counting of the corresponding EA inode is decreased. When EA inode <15> is
found by find_inode_fast() in __ext4_iget(), it is found that the EA inode
holds the I_FREEING flag and waits for the EA inode to complete deletion.
As a result, when inode <15> is being deleted, we wait for inode <15> to
complete the deletion, resulting in an infinite loop and triggering Hung
Task. To solve this problem, we only need to check whether the ino of EA
inode and parent is the same before getting EA inode.

Link: https://syzkaller.appspot.com/bug?extid=77d6fcc37bbb92f26048
Reported-by: syzbot+77d6fcc37bbb92f26048@syzkaller.appspotmail.com
Signed-off-by: Baokun Li <libaokun1@huawei.com>
Reviewed-by: Jan Kara <jack@suse.cz>
Link: https://lore.kernel.org/r/20230110133436.996350-1-libaokun1@huawei.com
Signed-off-by: Theodore Ts'o <tytso@mit.edu>
Signed-off-by: Sasha Levin <sashal@kernel.org>
```

**Diffstat**

-rw-r--r-- fs/ext4/xattr.c 11

1 files changed, 11 insertions, 0 deletions

```
diff --git a/fs/ext4/xattr.c b/fs/ext4/xattr.c
index 494994d9a332b0..f66c3fae90584b 100644
--- a/fs/ext4/xattr.c
+++ b/fs/ext4/xattr.c
@@ -388,6 +388,17 @@ static int ext4_xattr_inode_iget(struct inode *parent, unsigned long ea_ino,
         struct inode *inode;
         int err;

+        /*
+         * We have to check for this corruption early as otherwise
+         * iget_locked() could wait indefinitely for the state of our
+         * parent inode.
+         */
+        if (parent->i_ino == ea_ino) {
+                ext4_error(parent->i_sb,
+                           "Parent and EA inode have the same ino %lu", ea_ino);
+                return -EFSCORRUPTED;
+        }
+
         inode = ext4_iget(parent->i_sb, ea_ino, EXT4_IGET_NORMAL);
         if (IS_ERR(inode)) {
                 err = PTR_ERR(inode);
```