author	Ye Bin <yebin10@huawei.com>	2023-03-07 09:52:53 +0800
committer	Greg Kroah-Hartman <gregkh@linuxfoundation.org>	2023-03-17 08:57:48 +0100
commit	92eee6a82a9a6f9f83559e17a2b6b935e1a5cd25 (patch)	
tree	cb6d5f9a5a6eabdc7c65807264bea70c30a9009b	
parent	8aa3cb00909868c26e72b5877aa7375fc4ca6210 (diff)	
download	linux-92eee6a82a9a6f9f83559e17a2b6b935e1a5cd25.tar.gz	

ext4: fix WARNING in ext4_update_inline_data

```
commit 2b96b4a5d9443ca4cad58b0040be455803c05a42 upstream.

Syzbot found the following issue:
EXT4-fs (loop0): mounted filesystem 00000000-0000-0000-0000-000000000000 without journal. Quota mode: none.
fscrypt: AES-256-CTS-CBC using implementation "cts-cbc-aes-aesni"
fscrypt: AES-256-XTS using implementation "xts-aes-aesni"
------------[ cut here ]------------
WARNING: CPU: 0 PID: 5071 at mm/page_alloc.c:5525 __alloc_pages+0x30a/0x560 mm/page_alloc.c:5525
Modules linked in:
CPU: 1 PID: 5071 Comm: syz-executor263 Not tainted 6.2.0-rc1-syzkaller #0
Hardware name: Google Google Compute Engine/Google Compute Engine, BIOS Google 10/26/2022
RIP: 0010:__alloc_pages+0x30a/0x560 mm/page_alloc.c:5525
RSP: 0018:ffffc90003c2f1c0 EFLAGS: 00010246
RAX: ffffc90003c2f220 RBX: 0000000000000014 RCX: 0000000000000000
RDX: 0000000000000028 RSI: 0000000000000000 RDI: ffffc90003c2f248
RBP: ffffc90003c2f2d8 R08: dfffffc0000000000 R09: ffffc90003c2f220
R10: fffff52000785e49 R11: 1ffff92000785e44 R12: 0000000000040d40
R13: 1ffff92000785e40 R14: dfffffc0000000000 R15: 1ffff92000785e3c
FS:  0000555556c0d300(0000) GS:ffff8880b9800000(0000) knlGS:0000000000000000
CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
CR2: 00007f95d5e04138 CR3: 00000000793aa000 CR4: 00000000003506f0
DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
DR3: 0000000000000000 DR6: 00000000fffe0ff0 DR7: 0000000000000400
Call Trace:
 <TASK>
 __alloc_pages_node include/linux/gfp.h:237 [inline]
 alloc_pages_node include/linux/gfp.h:260 [inline]
 __kmalloc_large_node+0x95/0x1e0 mm/slab_common.c:1113
 __do_kmalloc_node mm/slab_common.c:956 [inline]
 __kmalloc+0xfe/0x190 mm/slab_common.c:981
 kmalloc include/linux/slab.h:584 [inline]
 kzalloc include/linux/slab.h:720 [inline]
 ext4_update_inline_data+0x236/0x6b0 fs/ext4/inline.c:346
 ext4_update_inline_dir fs/ext4/inline.c:1115 [inline]
 ext4_try_add_inline_entry+0x328/0x990 fs/ext4/inline.c:1307
 ext4_add_entry+0x5a4/0xeb0 fs/ext4/namei.c:2385
 ext4_add_nondir+0x96/0x260 fs/ext4/namei.c:2772
 ext4_create+0x36c/0x560 fs/ext4/namei.c:2817
 lookup_open fs/namei.c:3413 [inline]
 open_last_lookups fs/namei.c:3481 [inline]
 path_openat+0x12ac/0x2dd0 fs/namei.c:3711
 do_filp_open+0x264/0x4f0 fs/namei.c:3741
 do_sys_openat2+0x124/0x4e0 fs/open.c:1310
 do_sys_open fs/open.c:1326 [inline]
 __do_sys_openat fs/open.c:1342 [inline]
 __se_sys_openat fs/open.c:1337 [inline]
 __x64_sys_openat+0x243/0x290 fs/open.c:1337
```

```
 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

Above issue happens as follows:
ext4_iget
   ext4_find_inline_data_nolock ->i_inline_off=164 i_inline_size=60
ext4_try_add_inline_entry
   __ext4_mark_inode_dirty
      ext4_expand_extra_isize_ea ->i_extra_isize=32 s_want_extra_isize=44
         ext4_xattr_shift_entries
         ->after shift i_inline_off is incorrect, actually is change to 176
ext4_try_add_inline_entry
  ext4_update_inline_dir
    get_max_inline_xattr_value_size
      if (EXT4_I(inode)->i_inline_off)
        entry = (struct ext4_xattr_entry *)((void *)raw_inode +
                        EXT4_I(inode)->i_inline_off);
        free += EXT4_XATTR_SIZE(le32_to_cpu(entry->e_value_size));
        ->As entry is incorrect, then 'free' may be negative
   ext4_update_inline_data
      value = kzalloc(len, GFP_NOFS);
      -> len is unsigned int, maybe very large, then trigger warning when
         'kzalloc()'

To resolve the above issue we need to update 'i_inline_off' after
'ext4_xattr_shift_entries()'.  We do not need to set
EXT4_STATE_MAY_INLINE_DATA flag here, since ext4_mark_inode_dirty()
already sets this flag if needed.  Setting EXT4_STATE_MAY_INLINE_DATA
when it is needed may trigger a BUG_ON in ext4_writepages().

Reported-by: syzbot+d30838395804afc2fa6f@syzkaller.appspotmail.com
Cc: stable@kernel.org
Signed-off-by: Ye Bin <yebin10@huawei.com>
Reviewed-by: Jan Kara <jack@suse.cz>
Link: https://lore.kernel.org/r/20230307015253.2232062-3-yebin@huaweicloud.com
Signed-off-by: Theodore Ts'o <tytso@mit.edu>
Signed-off-by: Greg Kroah-Hartman <gregkh@linuxfoundation.org>
```

Diffstat

-rw-r--r-- fs/ext4/xattr.c 3

1 files changed, 3 insertions, 0 deletions

```
diff --git a/fs/ext4/xattr.c b/fs/ext4/xattr.c
index 0c6b011a91b3f8..494994d9a332b0 100644
--- a/fs/ext4/xattr.c
+++ b/fs/ext4/xattr.c
@@ -2807,6 +2807,9 @@ shift:
                        (void *)header, total_ino);
        EXT4_I(inode)->i_extra_isize = new_extra_isize;

+       if (ext4_has_inline_data(inode))
+               error = ext4_find_inline_data_nolock(inode);
+
 cleanup:
        if (error && (mnt_count != le16_to_cpu(sbi->s_es->s_mnt_count))) {
                ext4_warning(inode->i_sb, "Unable to expand inode %lu. Delete some EAs or run e2fsck.",
```