



author Darrick J. Wong <djwong@kernel.org> 2023-02-16 10:55:48 -0800
committer Greg Kroah-Hartman <gregkh@linuxfoundation.org> 2023-03-17 08:48:48 +0100
commit [c5d7c31e17224d847a330180ec1b03bf390632b2](#) (patch)
tree [62ffe3aaa98cf4a5d88547353302bb4387257ebb](#)
parent [df621af95e89d7dec8f28a6f354dd234344922df](#) (diff)
download [linux-c5d7c31e17224d847a330180ec1b03bf390632b2.tar.gz](#)

diff options

context:

3 ▼

space:

include ▼

mode:

unified ▼

ext4: fix another off-by-one fsmap error on 1k block filesystems

commit c993799baf9c5861f8df91beb80e1611b12efcbd upstream.

Apparently syzbot figured out that issuing this FSMAP call:

```
struct fsmap_head cmd = {
    .fmh_count      = ...;
    .fmh_keys       = {
        { .fmr_device = /* ext4 dev */, .fmr_physical = 0, },
        { .fmr_device = /* ext4 dev */, .fmr_physical = 0, },
    },
    ...
};
ret = ioctl(fd, FS_IOC_GETFSMAP, &cmd);
```

Produces this crash if the underlying filesystem is a 1k-block ext4 filesystem:

kernel BUG at fs/ext4/ext4.h:3331!

invalid opcode: 0000 [#1] PREEMPT SMP

CPU: 3 PID: 3227965 Comm: xfs_io Tainted: G W O 6.2.0-rc8-achx

Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.15.0-1 04/01/2014

RIP: 0010:ext4_mb_load_buddy_gfp+0x47c/0x570 [ext4]

RSP: 0018:ffffc90007c03998 EFLAGS: 00010246

RAX: ffff888004978000 RBX: ffff888007c03a20 RCX: ffff888041618000

RDX: 0000000000000000 RSI: 000000000000005a RDI: ffffffff80c99b11

RBP: ffff888012330000 R08: ffffffff80c2b7d0 R09: 0000000000000040

R10: ffff888007c03950 R11: 0000000000000000 R12: 0000000000000001

R13: 0000000000000000 R14: 00000000000000c4 R15: ffff88802678c398

FS: 00007fd02020c880(0000) GS: ffff88807e100000(0000) knlGS: 0000000000000000

CS: 0010 DS: 0000 ES: 0000 CR0: 0000000080050033

CR2: 00007ffd318a5fe8 CR3: 0000000007f80f001 CR4: 00000000001706e0

Call Trace:

<TASK>

ext4_mballoquery_range+0x4b/0x210 [ext4 dfa189daddffe8fec3cdfd00564e0f265a8ab80]

ext4_getfsmap_datadev+0x713/0x890 [ext4 dfa189daddffe8fec3cdfd00564e0f265a8ab80]

ext4_getfsmap+0x2b7/0x330 [ext4 dfa189daddffe8fec3cdfd00564e0f265a8ab80]

ext4_ioc_getfsmap+0x153/0x2b0 [ext4 dfa189daddffe8fec3cdfd00564e0f265a8ab80]

__ext4_ioctl+0x2a7/0x17e0 [ext4 dfa189daddffe8fec3cdfd00564e0f265a8ab80]

__x64_sys_ioctl+0x82/0xa0

do_syscall_64+0x2b/0x80

entry_SYSCALL_64_after_hwframe+0x46/0xb0

```
RIP: 0033:0x7fdf20558aff
RSP: 002b:00007ffd318a9e30 EFLAGS: 00000246 ORIG_RAX: 0000000000000010
RAX: ffffffffda RBX: 00000000000200c0 RCX: 00007fdf20558aff
RDX: 00007fdf1feb2010 RSI: 00000000c0c0583b RDI: 0000000000000003
RBP: 00005625c0634be0 R08: 00005625c0634c40 R09: 0000000000000001
R10: 0000000000000000 R11: 0000000000000246 R12: 00007fdf1feb2010
R13: 00005625be70d994 R14: 0000000000000800 R15: 0000000000000000
```

For GETFSMAP calls, the caller selects a physical block device by writing its block number into `fsmap_head.fmh_keys[0].fmr_device`. To query mappings for a subrange of the device, the starting byte of the range is written to `fsmap_head.fmh_keys[0].fmr_physical` and the last byte of the range goes in `fsmap_head.fmh_keys[1].fmr_physical`.

IOWs, to query what mappings overlap with bytes 3-14 of `/dev/sda`, you'd set the inputs as follows:

```
fmh_keys[0] = { .fmr_device = major(8, 0), .fmr_physical = 3},
fmh_keys[1] = { .fmr_device = major(8, 0), .fmr_physical = 14},
```

Which would return you whatever is mapped in the 12 bytes starting at physical offset 3.

The crash is due to insufficient range validation of `keys[1]` in `ext4_getfsmap_datadev`. On 1k-block filesystems, block 0 is not part of the filesystem, which means that `s_first_data_block` is nonzero. `ext4_get_group_no_and_offset` subtracts this quantity from the `blocknr` argument before cracking it into a group number and a block number within a group. IOWs, block group 0 spans blocks 1-8192 (1-based) instead of 0-8191 (0-based) like what happens with larger block sizes.

The net result of this encoding is that `blocknr < s_first_data_block` is not a valid input to this function. The `end_fsb` variable is set from the keys that are copied from userspace, which means that in the above example, its value is zero. That leads to an underflow here:

```
blocknr = blocknr - le32_to_cpu(es->s_first_data_block);
```

The division then operates on -1:

```
offset = do_div(blocknr, EXT4_BLOCKS_PER_GROUP(sb)) >>
        EXT4_SB(sb)->s_cluster_bits;
```

Leaving an impossibly large group number ($2^{32}-1$) in `blocknr`. `ext4_getfsmap_check_keys` checked that `keys[0].fmr_physical` and `keys[1].fmr_physical` are in increasing order, but `ext4_getfsmap_datadev` adjusts `keys[0].fmr_physical` to be at least `s_first_data_block`. This implies that we have to check it again after the adjustment, which is the piece that I forgot.

Reported-by: syzbot+6be2b977c89f79b6b153@syzkaller.appspotmail.com
Fixes: 4a4956249dac ("ext4: fix off-by-one fsmap error on 1k block filesystems")
Link: <https://syzkaller.appspot.com/bug?id=79d5768e9bfe362911ac1a5057a36fc6b5c30002>
Cc: stable@vger.kernel.org
Signed-off-by: Darrick J. Wong <djwong@kernel.org>
Link: <https://lore.kernel.org/r/Y+58NPTH7VNGgzdd@magnolia>
Signed-off-by: Theodore Ts'o <tytso@mit.edu>
Signed-off-by: Greg Kroah-Hartman <gregkh@linuxfoundation.org>

Diffstat

```
-rw-r--r-- fs/ext4/fsmap.c 2
```

1 files changed, 2 insertions, 0 deletions

diff --git a/fs/ext4/fsmap.c b/fs/ext4/fsmap.c

index 4493ef0c715e9d..cdf9bfe10137ff 100644

--- a/fs/ext4/fsmap.c

+++ b/fs/ext4/fsmap.c

```
@@ -486,6 +486,8 @@ static int ext4_getfsmap_datadev(struct super_block *sb,
        keys[0].fmr_physical = bofs;
        if (keys[1].fmr_physical >= eofs)
            keys[1].fmr_physical = eofs - 1;
+       if (keys[1].fmr_physical < keys[0].fmr_physical)
+           return 0;
        start_fsb = keys[0].fmr_physical;
        end_fsb = keys[1].fmr_physical;
```