author	David Howells <dhowells@redhat.com>	2022-11-03 16:08:14 +0000
committer	Greg Kroah-Hartman <gregkh@linuxfoundation.org>	2022-11-26 09:27:38 +0100
commit	b2cc07a76f1eb12de3b22caf5fdbf856a7bef16d (patch)	
tree	68125344a957c1824a42e89c517725caa17c1c69	
parent	2fac1a7218ef74def406d39d6d572a0a320240ee (diff)	
download	linux-b2cc07a76f1eb12de3b22caf5fdbf856a7bef16d.tar.gz	

netfs: Fix missing xas_retry() calls in xarray iteration

```
[ Upstream commit 7e043a80b5dae5c2d2cf84031501de7827fd6c00 ]

netfslib has a number of places in which it performs iteration of an xarray
whilst being under the RCU read lock.  It *should* call xas_retry() as the
first thing inside of the loop and do "continue" if it returns true in case
the xarray walker passed out a special value indicating that the walk needs
to be redone from the root[*].

Fix this by adding the missing retry checks.

[*] I wonder if this should be done inside xas_find(), xas_next_node() and
    suchlike, but I'm told that's not an simple change to effect.

This can cause an oops like that below.  Note the faulting address - this
is an internal value (|0x2) returned from xarray.

BUG: kernel NULL pointer dereference, address: 0000000000000402
...
RIP: 0010:netfs_rreq_unlock+0xef/0x380 [netfs]
...
Call Trace:
 netfs_rreq_assess+0xa6/0x240 [netfs]
 netfs_readpage+0x173/0x3b0 [netfs]
 ? init_wait_var_entry+0x50/0x50
 filemap_read_page+0x33/0xf0
 filemap_get_pages+0x2f2/0x3f0
 filemap_read+0xaa/0x320
 ? do_filp_open+0xb2/0x150
 ? rmqueue+0x3be/0xe10
 ceph_read_iter+0x1fe/0x680 [ceph]
 ? new_sync_read+0x115/0x1a0
 new_sync_read+0x115/0x1a0
 vfs_read+0xf3/0x180
 ksys_read+0x5f/0xe0
 do_syscall_64+0x38/0x90
 entry_SYSCALL_64_after_hwframe+0x44/0xae

Changes:
========
ver #2)
 - Changed an unsigned int to a size_t to reduce the likelihood of an
   overflow as per Willy's suggestion.
 - Added an additional patch to fix the maths.

Fixes: 3d3c95046742 ("netfs: Provide readahead and readpage netfs helpers")
Reported-by: George Law <glaw@redhat.com>
Signed-off-by: David Howells <dhowells@redhat.com>
```

Reviewed-by: Jeff Layton <jlayton@kernel.org>
Reviewed-by: Jingbo Xu <jefflexu@linux.alibaba.com>
cc: Matthew Wilcox <willy@infradead.org>
cc: linux-cachefs@redhat.com
cc: linux-fsdevel@vger.kernel.org
Link: https://lore.kernel.org/r/166749229733.107206.17482609105741691452.stgit@warthog.procyon.org.uk/ # v1
Link: https://lore.kernel.org/r/166757987929.950645.12595273010425381286.stgit@warthog.procyon.org.uk/ # v2
Signed-off-by: Sasha Levin <sashal@kernel.org>

Diffstat

-rw-r--r- fs/netfs/buffered_read.c	9
-rw-r--r- fs/netfs/io.c	3

2 files changed, 10 insertions, 2 deletions

```
diff --git a/fs/netfs/buffered_read.c b/fs/netfs/buffered_read.c
index 0ce53585215106..baf668fb431541 100644
--- a/fs/netfs/buffered_read.c
+++ b/fs/netfs/buffered_read.c
@@ -46,10 +46,15 @@ void netfs_rreq_unlock_folios(struct netfs_io_request *rreq)

        rcu_read_lock();
        xas_for_each(&xas, folio, last_page) {
-               unsigned int pgpos = (folio_index(folio) - start_page) * PAGE_SIZE;
-               unsigned int pgend = pgpos + folio_size(folio);
+               unsigned int pgpos, pgend;
                bool pg_failed = false;

+               if (xas_retry(&xas, folio))
+                       continue;
+
+               pgpos = (folio_index(folio) - start_page) * PAGE_SIZE;
+               pgend = pgpos + folio_size(folio);
+
                for (;;) {
                        if (!subreq) {
                                pg_failed = true;

diff --git a/fs/netfs/io.c b/fs/netfs/io.c
index 4289258992826b..e374767d1b6832 100644
--- a/fs/netfs/io.c
+++ b/fs/netfs/io.c
@@ -121,6 +121,9 @@ static void netfs_rreq_unmark_after_write(struct netfs_io_request *rreq,
                XA_STATE(xas, &rreq->mapping->i_pages, subreq->start / PAGE_SIZE);

                xas_for_each(&xas, folio, (subreq->start + subreq->len - 1) / PAGE_SIZE) {
+                       if (xas_retry(&xas, folio))
+                               continue;
+
                        /* We might have multiple writes from the same huge
                         * folio, but we mustn't unlock a folio more than once.
                         */
```