



author Filipe Manana <fdmanana@suse.com> 2022-11-01 16:15:38 +0000
committer Greg Kroah-Hartman <gregkh@linuxfoundation.org> 2022-11-10 18:15:29 +0100
commit 222a3d533027b9492d5b7f5ecdc01a90f57bb5a9 (patch)
tree 95a8cf27fb80fb5d9b7b4ce7076d96cadf36e695
parent 6ba3479f9e96b9ad460c7e77abc26dd16e5dec4f (diff)
download [linux-222a3d533027b9492d5b7f5ecdc01a90f57bb5a9.tar.gz](#)

diff options

context: 3 ▼
space: include ▼
mode: unified ▼

btrfs: fix inode list leak during backref walking at find_parent_nodes()

[Upstream commit 92876eec382a0f19f33d09d2c939e9ca49038ae5]

During backref walking, at find_parent_nodes(), if we are dealing with a data extent and we get an error while resolving the indirect backrefs, at resolve_indirect_refs(), or in the while loop that iterates over the refs in the direct refs rbtree, we end up leaking the inode lists attached to the direct refs we have in the direct refs rbtree that were not yet added to the refs ulist passed as argument to find_parent_nodes(). Since they were not yet added to the refs ulist and prelim_release() does not free the lists, on error the caller can only free the lists attached to the refs that were added to the refs ulist, all the remaining refs get their inode lists never freed, therefore leaking their memory.

Fix this by having prelim_release() always free any attached inode list to each ref found in the rbtree, and have find_parent_nodes() set the ref's inode list to NULL once it transfers ownership of the inode list to a ref added to the refs ulist passed to find_parent_nodes().

Fixes: 86d5f9944252 ("btrfs: convert preliminary reference tracking to use rbtrees")

Signed-off-by: Filipe Manana <fdmanana@suse.com>

Signed-off-by: David Sterba <dsterba@suse.com>

Signed-off-by: Sasha Levin <sashal@kernel.org>

Diffstat

```
-rw-r--r-- fs/btrfs/backref.c 18
```

1 files changed, 17 insertions, 1 deletions

diff --git a/fs/btrfs/backref.c b/fs/btrfs/backref.c
index 26e6867ff023e3..60bec5a108fa3f 100644

--- a/fs/btrfs/backref.c

+++ b/fs/btrfs/backref.c

```
@@ -289,8 +289,10 @@ static void prelim_release(struct preftree *preftree)
     struct prelim_ref *ref, *next_ref;
```

```
     rbtree_postorder_for_each_entry_safe(ref, next_ref,
```

```
     -                                     &preftree->root.rb_root, rbnode)
```

```
     +                                     &preftree->root.rb_root, rbnode) {
```

```
     +         free_inode_elem_list(ref->inode_list);
```

```
         free_pref(ref);
```

```
     + }
```

```

    preftree->root = RB_ROOT_CACHED;
    preftree->count = 0;
@@ -1387,6 +1389,12 @@ again:
        if (ret < 0)
            goto out;
        ref->inode_list = eie;
+
+        /*
+         * We transferred the list ownership to the ref,
+         * so set to NULL to avoid a double free in case
+         * an error happens after this.
+         */
+        eie = NULL;
    }
    ret = ulist_add_merge_ptr(refs, ref->parent,
                             ref->inode_list,
@@ -1412,6 +1420,14 @@ again:
        eie->next = ref->inode_list;
    }
    eie = NULL;
+
+    /*
+     * We have transferred the inode list ownership from
+     * this ref to the ref we added to the 'refs' ulist.
+     * So set this ref's inode list to NULL to avoid
+     * use-after-free when our caller uses it or double
+     * frees in case an error happens before we return.
+     */
+    ref->inode_list = NULL;
}
cond_resched();
}

```