author	Filipe Manana <fdmanana@suse.com>	2022-11-01 16:15:37 +0000
committer	Greg Kroah-Hartman <gregkh@linuxfoundation.org>	2022-11-10 18:15:29 +0100
commit	6ba3479f9e96b9ad460c7e77abc26dd16e5dec4f (patch)	
tree	514c94cb63a5e4add524486ab548950fc29b20f9	
parent	a80634f392af8c42a5a9d2b3822acc6e7795c8be (diff)	
download	linux-6ba3479f9e96b9ad460c7e77abc26dd16e5dec4f.tar.gz	

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

```
[ Upstream commit 5614dc3a47e3310fbc77ea3b67eaadd1c6417bf1 ]

During backref walking, at resolve_indirect_refs(), if we get an error
we jump to the 'out' label and call ulist_free() on the 'parents' ulist,
which frees all the elements in the ulist - however that does not free
any inode lists that may be attached to elements, through the 'aux' field
of a ulist node, so we end up leaking lists if we have any attached to
the unodes.

Fix this by calling free_leaf_list() instead of ulist_free() when we exit
from resolve_indirect_refs(). The static function free_leaf_list() is
moved up for this to be possible and it's slightly simplified by removing
unnecessary code.

Fixes: 3301958b7c1d ("Btrfs: add inodes before dropping the extent lock in find_all_leafs")
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 36

1 files changed, 17 insertions, 19 deletions

```
diff --git a/fs/btrfs/backref.c b/fs/btrfs/backref.c
index 2e7c3e48bc9ce9..26e6867ff023e3 100644
--- a/fs/btrfs/backref.c
+++ b/fs/btrfs/backref.c
@@ -648,6 +648,18 @@ unode_aux_to_inode_list(struct ulist_node *node)
        return (struct extent_inode_elem *)(uintptr_t)node->aux;
 }

+static void free_leaf_list(struct ulist *ulist)
+{
+       struct ulist_node *node;
+       struct ulist_iterator uiter;
+
+       ULIST_ITER_INIT(&uiter);
+       while ((node = ulist_next(ulist, &uiter)))
+               free_inode_elem_list(unode_aux_to_inode_list(node));
+
+       ulist_free(ulist);
```

```
+}
+
 /*
  * We maintain three separate rbtrees: one for direct refs, one for
  * indirect refs which have a key, and one for indirect refs which do not
@@ -762,7 +774,11 @@ static int resolve_indirect_refs(struct btrfs_fs_info *fs_info,
                cond_resched();
        }
 out:
-       ulist_free(parents);
+       /*
+        * We may have inode lists attached to refs in the parents ulist, so we
+        * must free them before freeing the ulist and its refs.
+        */
+       free_leaf_list(parents);
        return ret;
 }

@@ -1412,24 +1428,6 @@ out:
        return ret;
 }

-static void free_leaf_list(struct ulist *blocks)
-{
-       struct ulist_node *node = NULL;
-       struct extent_inode_elem *eie;
-       struct ulist_iterator uiter;
-
-       ULIST_ITER_INIT(&uiter);
-       while ((node = ulist_next(blocks, &uiter))) {
-               if (!node->aux)
-                       continue;
-               eie = unode_aux_to_inode_list(node);
-               free_inode_elem_list(eie);
-               node->aux = 0;
-       }
-
-       ulist_free(blocks);
-}
-
 /*
  * Finds all leafs with a reference to the specified combination of bytenr and
  * offset. key_list_head will point to a list of corresponding keys (caller must
```