A survey of bugs in the Btrfs filesystem

Omar Sandoval

1 Introduction

The Btrfs filesystem [16] is being developed as the next major Linux filesystem. It sports several advanced features, including checksumming of all data and metadata, first-class snapshots, and multidevice support. Btrfs has been in development since 2007, being merged into the Linux kernel in 2009 and declared stable in 2013. Despite this, it is still under active development, and as such, bugs are not a rarity.

We examine a case study of several bugs encountered by following the Btrfs mailing list [3] and bug tracker [10]. Most of these bugs were encountered by real users during normal usage. We also discuss the fixes developed for each of these bugs, which were all posted to the Btrfs mailing list for inclusion into the Linux kernel.

2 Btrfs bugs

The bugs discussed below were found through user bug reports. Fixes were developed and posted on the Btrfs mailing list for inclusion in the Linux kernel; some are still under review, some are staged for the next kernel release, and some are already upstream.

2.1 Bugs in Btrfs subvolume deletion

Btrfs has a notion of subvolumes, which are separate, mountable namespaces which are all stored on the same filesystem instance. Subvolumes are used to implement snapshots, a popular feature in Btrfs. A few bugs in the subvolume deletion code were reported by users, some of which were related.

2.1.1 Deletion of mounted subvolumes

A user reported [11] that since Linux 3.18, it is possible to issue the `btrfs` command to delete a subvolume which is currently mounted. This was not possible previously, and the cause turned out to be unintentional and seemingly unrelated.

A Git bisect indicated that this problem was introduced by a patch series [9] which fixed a DoS attack wherein a user in one mount namespace (used for implementing containers) could prevent the deletion of a directory by mounting a filesystem in its namespace which would be invisible to users in other namespaces but would cause `rmdir` to fail with `EBUSY`. An implementation detail meant that this also no longer caused deletion of a mounted subvolume to fail.

Although a fix was provided [4], after discussion it was no longer clear whether this was actually a bug; subvolumes in Btrfs are internally implemented as directories, so subvolume deletion amounts to the recursive deletion of a directory, and the kernel does not prevent a user from issuing an analogous `rm -rf /`. Additionally, fixing it would reintroduce the DoS attack. The patch was put on hold to work out these issues, but there were related issues to come.

2.1.2 Locking mistake in error handling

Btrfs has a feature called send/receive which allows users to back up their filesystem by replaying the operations made on a subvolume since the last backup. This feature needs mutual exclusion with subvolume deletion because it works by walking the on-disk B-trees. When this mutual exclusion was added (itself a bug fix), the error case forgot to unlock the `i_mutex` of the subvolume inode. This was found while looking at the subvolume deletion code for the previous bug report. The fix was merged for Linux 4.1 [8].

2.1.3 Premature unmounting of submounts

Every Btrfs filesystem has a default subvolume which specifies which subvolume is to be mounted when not specifically passed as a mount option. The default subvolume cannot be deleted, as this could create an unmountable filesystem. However, a user reported [2] that a backup daemon attempting to exactly this caused a kernel oops. The oops was harder to reproduce, but it was easy to leave a machine in an unusable state by recreating the conditions of the bug: after attempting to delete the default subvolume mounted as root, the `procfs`, `sysfs`, and all other mounted filesystems were unmounted even though the root filesystem survived.

This issue turned out to be related to the first one, being exposed by the same patch series. Previously, in this case, the subvolume deletion code would have errored out early because the subvolume was mounted. Now, however, it continued to the point where the Linux dentry cache corresponding to the subvolume was invalidated, which unmounts all submounts under it. The check for whether the default subvolume was being deleted only happened after this. The solution was to only invalidate the subvolume in the dentry cache after the deletion succeeds. A patch [1] and test [7] were posted; the patch is scheduled for Linux 4.2.

2.2 Missing Btrfs mount information in procfs

Two files under `/proc/$PID/`, `mounts` and `mountinfo`, provide userspace with the mount options that were passed when a filesystem was mounted (e.g., `rw,data=ordered`).

When a Btrfs subvolume is mounted, it can be specified either by name (e.g., `subvol=/backups/2015-05-18`) or ID (e.g., `subvolid=266`). However, this information is missing from `mounts` and `mountinfo`. This is primarily a usability concern, as it makes it difficult to determine which subvolumes are mounted.

Mounting by subvolume name and subvolume ID use two separate code paths. The simple way to get the subvolume name in `proc` is by getting the path of the subvolume root `dentry`, but because of an implementation detail, this does not work correctly for the subvolume ID case. The solution was to make mounting by subvolume ID translate the ID to a path and then use the same code path. A patch series [6] implementing this is slated for Linux 4.2.

2.3 Replacement of a missing disk on Btrfs RAID 5/6

Btrfs supports using multiple devices at various RAID levels natively. Support for RAID 5 and RAID 6, however, is fairly new and experimental, only being introduced in Linux 3.9. On multi-device setups, Btrfs supports replacing a drive when it is faulty or even missing; doing this on RAID 5/6, however, is even newer, added in Linux 3.19.

A user reported that when they attempted to replace a missing RAID 5/6 device, the kernel panicked. The superficial cause was easy to find: for some reason, an I/O request was being submitted for a `NULL` block device. However, fixing this was much less straightforward. In summary, the device replace code operated under the assumption that there would always be a valid device containing a copy of the data being copied to the new device, which is true for RAID 1, but not for RAID 5 and 6, which use parity for recovery. The fix involved adding a separate code path for this case that only reads from the present devices to reconstruct the data which was on the missing device. This series [5] has been posted but is still out for review.

3 Other Linux kernel bugs

During the investigation of the Btrfs bugs enough, another bug unrelated to Btrfs was found and fixed.

3.1 Regression in the task scheduler after suspend-to-RAM

A bug unrelated to Btrfs was encountered while developing the previous patch series: sometimes, a tool used for searching through the code base, The Silver Searcher [14], was failing to run because `pthread_setaffinity_np` failed with `EINVAL`, indicating that it was attempting to set the affinity to a CPU which did not exist. A test program reproduced the issue: `pthread_setaffinity_np` and `sched_setaffinity` failed on any CPU other than CPU 0 on a four-core machine. Eventually it became clear that this only happened after a suspend/resume cycle, and the issue was bisected to a patch which fixed a bug in the deadline scheduler introduced during the 4.1 merge window. Luckily, the problem turned out to just be a logic error, so it was possible to fix without being familiar with the code. The patch was applied upstream in Linux 4.1-rc3 [13].

4 Discussion

While each of these bugs is interesting in its own right, it is also worthwhile to discuss what we can learn from them as a whole, and in particular, how to prevent similar bugs in the future.

The locking error in Btrfs subvolume deletion could most likely be found with a static analysis tool, including the kernel's own Sparse [15]. Additionally, a test case in the xfstests filesystem test suite for the original bug fix (mutual exclusion between subvolume deletion and send/receive) would also have uncovered the bug, especially if the kernel's runtime locking correctness validator, lockdep [12], were enabled on the tester's machine.

The deletion of mounted subvolumes and premature unmounting of submounts when attempting to delete the default subvolume were both exposed by a patch series which changed the semantics of the VFS API. The changes were necessary to fix longstanding bugs in Linux, but these side-effects were probably impossible to predict. Additionally, it is unclear what the intended behavior for the deletion of a mounted subvolume was, as it was undocumented and untested.

The RAID 5/6 device replace bug has two underlying issues. Firstly, assumptions previously true in the device replace code were not true for RAID 5/6, and secondly, there was insufficient testing of the feature. Attempting to replace a missing device would have quickly exposed the deficiency.

Finally, the scheduler bug is slightly different. It is harder to test for such a bug. It is more fair to consider it as a simple mistake in the code review process, which is bound to happen every once in a while.

In summary, the results are not shocking: more testing is required, preferably available in a widely-used test suite like xfstests, and more code review is required. Regardless, it is difficult to achieve such lofty goals, and it will probably always to necessary to simply push the testing to users and fix problems as they come up.

References

[1] Btrfs: don't invalidate root dentry when subvolume deletion fails. `http://article.gmane.org/gmane.comp.file-systems.btrfs/45448`.

[2] BTRFS issue: deleting default snapshot causes kernel oops. `http://article.gmane.org/gmane.comp.file-systems.btrfs/45446`.

[3] Btrfs mailing list. `http://vger.kernel.org/vger-lists.html#linux-btrfs`.

[4] Btrfs: prevent deletion of mounted subvolumes. `http://article.gmane.org/gmane.linux.kernel/1920175`.

[5] Btrfs: Raid 5/6 missing device replace. `http://article.gmane.org/gmane.comp.file-systems.btrfs/45045`.

[6] Btrfs: show subvolume name and id in /proc/mounts. `http://article.gmane.org/gmane.comp.file-systems.btrfs/45237`.

[7] btrfs: test premature submount unmounting when deleting default subvolume. `http://article.gmane.org/gmane.comp.file-systems.btrfs/45521`.

[8] btrfs: unlock i_mutex after attempting to delete subvolume during send. `https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=909e26dce3f7600f5e293ac0522c28790a0c8c9c`.

[9] Detaching mounts on unlink. `http://article.gmane.org/gmane.linux.kernel/1648838`.

[10] Kernel bug tracker. `https://bugzilla.kernel.org/`.

[11] Mounted subvolumes can be deleted. `https://bugzilla.kernel.org/show_bug.cgi?id=93021`.

[12] Runtime locking correctness validator. `https://www.kernel.org/doc/Documentation/locking/lockdep-design.txt`.

[13] sched/core: Fix regression in cpuset_cpu_inactive() for suspend. `https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=533445c6e53368569e50ab3fb712230c03d523f3`.

[14] The Silver Searcher: A code searching tool similar to ack, with a focus on speed. `https://github.com/ggreer/the_silver_searcher`.

[15] Sparse. `https://sparse.wiki.kernel.org/`.

[16] O. Rodeh, J. Bacik, and C. Mason. BTRFS: The Linux B-tree filesystem. *ACM Transactions on Storage*, 9(3).