Linux Copy on Write for Incident Responders

Taz Wake | Halkyn Consulting Ltd | Originally published 2024-03-18

Introduction

This article will look at Copy on Write within Linux filesystems. It will look at how file creation and file deletion work as these are two of the most important aspects for any Digital Forensics or Incident Response investigation.

What is Copy on Write?

"Copy on Write" (CoW) is a resource management technique used by some filesystems to handle data modification in an efficient (and hopefully low risk) manner.

With CoW, instead of immediately altering the original data, the filesystem creates a copy when a modification is attempted. The system then writes the changes to this copy, leaving the original data untouched until it is certain that no other processes require access to the unmodified version.

This approach minimises data duplication and improves performance by deferring data copy until it is absolutely required. In filesystems like ZFS and Btrfs, CoW enhances data integrity, supports features like snapshots and cloning, and allows for more efficient disk space usage, as only the changes to data are stored, not complete copies of the data.

What filesystems use Copy on Write?

There are two mainstream Linux filesystems that use CoW:

- **Btrfs (B-tree File System)**: Btrfs is known for its focus on fault tolerance, repair, and easy administration. CoW in Btrfs enables features like snapshotting, efficient data duplication, and data integrity. Btrfs is currently the default filesystem on some versions of SUSE Linux and is used extensively on Synology NAS devices.
- **ZFS (previously "*Zettabyte File System*") / OpenZFS**: Originally developed by Sun Microsystems for Solaris, ZFS is available on Linux through third-party modules. It's a combined file system and logical volume manager. ZFS's CoW mechanism supports high storage capacities, data integrity verification, and repair, along with efficient snapshots and clones. ZFS has, however, undergone some major changes since 2010 and support in current Linux distros can be erratic.

File creation in a Copy on Write file system

The nature of CoW filesystems means that file creation is not a "simple" task. Instead, several steps happen, although this is normally completed in a few microseconds.

This is a high-level view of what happens in the background (for most CoW filesystems) is:

- **Allocating the metadata**: The filesystem allocates metadata for the new file. This metadata includes information about the file size, permissions, ownership, timestamps, and pointers to where the data blocks will be stored. In CoW filesystems, this metadata is often stored in a tree structure (e.g., B-tree in Btrfs, Merkle tree in ZFS). Tree structures are designed to provide improved integrity to the process and efficiency in storage and subsequent accesses.
- **Write the data**: When data is written to the file, the filesystem allocates data blocks to store this information. In a CoW filesystem, these data blocks are not overwritten in place for updates or modifications; instead, new data is written to different blocks, and the metadata is updated to point to these new blocks. This approach prevents data corruption during writes and supports atomic updates.
- **Update the metadata**: After the data is written, the filesystem updates the file's metadata with the locations of the new data blocks. This update is also done in a CoW manner, meaning that changes to metadata are written to new locations. The old versions of the metadata are preserved until they are no longer needed, which is crucial for maintaining a consistent and recoverable state. Yes, this is a bit recursive...
- **Snapshots**: If snapshots are present, the creation of a new file (and any subsequent modifications) does not affect the snapshot's state. Snapshots preserve the state of the filesystem *at the time they were taken*. Because CoW filesystems only write changes to new blocks, the snapshots continue to point to the old data and metadata, ensuring that the snapshot remains unchanged. ***Note: The snapshot holds older data***.
- **Space Management**: CoW filesystems employ space management techniques to allocate and manage storage. This includes handling deduplication (if supported), where identical blocks are only stored once, and dynamically allocating space as needed for file data and metadata.
- **Data Integrity and Error Correction**: Most filesystems incorporate features to ensure data integrity, such as checksums for data and metadata. When a file is created (and as it's modified), these checksums are computed and stored. This allows the filesystem to detect and correct errors, ensuring the reliability and correctness of stored data.
- **Transaction Group Commit**: Some filesystems (e.g. ZFS) group changes, such as file creation, into transaction groups. These groups are periodically committed to disk in a consistent state, ensuring that the filesystem can be reliably recovered in case of a crash or power failure.

File deletion

The second interesting way of looking at CoW filesystems, is looking at how files are deleted. Just like file creation, this may seem like a simple task to the user, but a lot happens under the hood of the system.

Although there are similarities, (the process is broadly the same with Btrfs and ZFS), there are some differences worth identifying here.

Deleting a file in Btrfs

Again, keeping this flow simple and just looking at high-level elements, the process of file deletion in Btrfs is:

- **Mark the File for Deletion**: The first step involves marking the file's inode for deletion and removing its entry from the directory listing. This does not immediately free up the disk space used by the file, but it **makes the file inaccessible to users and applications**.
- **Update the file's metadata**: Btrfs updates its metadata trees to reflect the deletion. Since Btrfs uses CoW, these updates are written to new locations on the disk, preserving the old versions of the metadata until they are no longer needed. This is crucial for maintaining filesystem integrity and supports features like snapshotting. ***Note: This means there might be data available if the disk is imaged and analysed in a forensics tool***.
- **Reference Counting and Space Reclamation**: Btrfs employs reference counting for data blocks and metadata. When a file is deleted, Btrfs decrements the reference counts for the blocks that were exclusively used by that file. If a block's reference count drops to zero (meaning no other file or snapshot is using it), the block is marked as free and can be reused by the filesystem. This process is more complex when blocks are shared due to snapshots or deduplication.
- **Delayed Space Reclamation**: The actual reclamation of space on the disk **might not** happen immediately. Btrfs's space management system decides when to reclaim space based on several factors, including the current workload and the overall filesystem health. This approach helps optimise performance and reduce wear on storage devices.
- **Effect of Snapshots**: If the deleted file was part of a snapshot, the space it occupied will not be immediately reclaimed. In Btrfs, snapshots are treated like any other subvolume, with their own set of references to data blocks. A block is only freed when all references to it (from both the active filesystem and any snapshots) have been removed.
- **Garbage Collection and Balancing**: Btrfs periodically runs garbage collection and balancing operations that help reclaim space and optimise the distribution of data across the disk. These operations can also help recover space from deleted files, especially in filesystems with a lot of snapshots or where data blocks are heavily shared.

- **Defragmentation**: While not directly related to file deletion, Btrfs's CoW nature means that filesystem fragmentation can occur over time, especially with frequent file writes and deletions.

File deletion in ZFS

Although similar to Btrfs, there are some minor differences with ZFS.

- **Mark the File for Deletion**: Similar to Btrfs, when a file is deleted, ZFS marks the inode as deleted. However, the actual data blocks (where the file's data is stored) are not immediately wiped or overwritten.
- **Updating Metadata**: ZFS updates the filesystem metadata, removing references to the deleted file. This metadata update is also done in a CoW manner, ensuring that the integrity and consistency of the filesystem metadata are maintained. If the file was part of a snapshot or had multiple hard links, ZFS handles these cases by only removing the relevant links or snapshot references that pertain to the deleted file.
- **Reclaiming Space**: The space occupied by the deleted file's data is not immediately reclaimed. The actual data blocks of the deleted file become eligible for garbage collection and space reclamation during the filesystem's normal operation. If the deleted file's blocks were part of a snapshot, the blocks remain allocated until all references (from all snapshots and the live filesystem) are removed.
- **Delayed Space Reclamation**: ZFS's space reclamation is often asynchronous and can be influenced by several factors, including system load, the presence of snapshots, and the ZFS pool's space usage. The actual reclamation of disk space previously occupied by the deleted file happens over time, as ZFS continues to manage the pool's storage efficiently.
- **Snapshots and Clones**: If a deleted file is referenced by snapshots or clones, the file's data remains on the disk until all such references are removed. ZFS's snapshot and clone features allow for efficient data management, but they also mean that the physical disk space used by a file might not be released immediately upon file deletion, depending on these references.

Incident Responders Notes

Ultimately the main point for anyone doing DFIR on a CoW filesystem is that file recovery is really going to be down to the exact state of the system when you took an image.

If there are snapshots, or if older versions exist, then it might actually be trivial to recover files. If you image before the unallocated space has been reclaimed, then file carving is likely to be successful. However, if the snapshots are removed and space has been reclaimed then recovery is unlikely.

File recovery considerations

1. Copy on write means the original data isn't overwritten immediately. Even using tools like may leave artifacts or remnants of old file data.

2. Snapshots/Clones. If there is a snapshot which contains an older version of the file, file recovery is trivial.
3. Filesystem compression. This is available in both Btrfs and ZFS and makes recovery actually much harder. If the file is compressed at the filesystem layer, then file carving is significantly harder.
4. Encryption. If filesystem encryption is in use (and this is fairly common in ZFS), then any recovery becomes significantly more challenging - file carving is effectively useless.

About Halkyn Consulting

Halkyn Consulting Ltd is a UK-based security and risk management consultancy providing 24/7 incident response, digital forensics, threat hunting, virtual security leadership (vCISO) and security compliance support to organisations worldwide. We deliver clear, practical, vendor-independent advice based on real-world threats — no hype, no upsell.

More free security resources are available at www.halkynconsulting.co.uk/security-resources.

Web: www.halkynconsulting.co.uk | **Email:** info@halkynconsulting.co.uk | **Tel:** +44 1244 940858