Linux Incident Response - Understanding Btrfs

Taz Wake | Halkyn Consulting Ltd | Originally published 2023-11-24

A Short History of Btrfs

Btrfs, often described as "a revolutionary file system," really came about in 2007 following a presentation at the Usenix conference. Conceived by Chris Mason while at Oracle, Btrfs was designed to address the growing demands of modern data storage and management. It was a response to the limitations faced by existing Linux file systems and was inspired by the capabilities of ZFS, though tailored to integrate with the Linux kernel's architecture.

The early versions of Btrfs aimed to bring advanced features to Linux, such as writeable snapshots, data pooling, and integrated checksums for enhanced data integrity. These features were particularly relevant in an era where data volumes were growing exponentially, and the need for efficient data management was becoming increasingly critical.

Over time, Btrfs has seen substantial development and contributions from a range of entities, including tech giants like Facebook. Facebook's involvement was particularly significant, as they tailored Btrfs to meet the demanding needs of their massive data storage systems. This collaboration led to enhancements in Btrfs's scalability and reliability, making it a robust file system capable of handling large-scale and cloud-based environments.

As Btrfs matured, it became more than just a file system; it evolved into a symbol of Linux's adaptability and innovation in data storage. Its development journey reflects a concerted effort to push the boundaries of file system technology, adapting to the ever-changing landscape of digital data storage and management.

Btrfs: Establishing Its Foothold in Modern Linux Distributions

Today, Btrfs's adoption across various Linux distributions is a testament to its reliability and advanced capabilities. It is the default file system in distributions such as OpenSUSE and Fedora, which are known for their forward-looking and innovative nature. This adoption marks a significant shift in the Linux community's approach to data management, with a growing emphasis on flexibility, scalability, and data integrity.

In the realm of network-attached storage, Btrfs has found a strong foothold, exemplified by its adoption in Synology NAS solutions. This underscores Btrfs's suitability for managing dynamic data sets and complex storage environments, catering to both individual users and enterprise-level requirements.

The choice of Btrfs in these systems reflects a growing recognition of its advanced feature set, which includes dynamic inode allocation, snapshots, and integrated RAID support (although this does lead to some further discussion).

Btrfs vs. XFS and EXT4 - Key differences

Distinctive Features of Btrfs

Btrfs differentiates itself from other Linux filesystems like EXT4 and XFS through several innovative features. EXT4, long revered for its stability and widespread usage, offers a dependable and well-understood environment. XFS, on the other hand, is known for its scalability and efficiency in managing large files and volumes. Btrfs, however, breaks new ground with its dynamic approach to data management.

One of the key features that sets Btrfs apart is its native support for advanced RAID configurations. Unlike EXT4 and XFS, which rely on external volume managers for RAID functionality, Btrfs integrates RAID support directly into the filesystem. This integration allows for more streamlined and efficient data redundancy and distribution across multiple storage devices.

However, Btrfs's RAID implementation, particularly RAID5 and RAID6, has faced scrutiny. Concerns primarily revolve around the 'write hole' issue, where during a power failure, data can become corrupted. This issue poses significant challenges in maintaining data integrity, a critical factor for incident responders and forensic investigators. While Btrfs continues to evolve and address these concerns, the current state of its RAID5 and RAID6 support demands careful consideration and often supplementary backup strategies in critical systems.

Copy on Write (CoW)

Copy on Write (CoW) is a foundational concept in Btrfs, fundamentally altering how data is stored and modified. When changes are made to a file, instead of overwriting the existing data, Btrfs writes the new data to a different location on the disk. Once the write operation is complete, the filesystem metadata is updated to point to the new data block, leaving the original data unaltered. This approach has several implications:

1. **Data Integrity**: By not overwriting data directly, CoW provides a layer of protection against data corruption during write operations, especially critical in scenarios of power failure or system crashes.
2. **Efficient Snapshots**: CoW is integral to Btrfs's snapshotting capability. When a snapshot is taken, it's essentially a metadata copy pointing to the same data blocks as the original filesystem. As changes are made to the filesystem, only the altered data blocks are written anew, with the snapshot continuing to point to the original blocks. This process makes snapshots space-efficient and quick to create.
3. **System Calls and Inode Management**: The CoW process involves a series of system calls that manage the allocation of new data blocks and the updating of inode

information. Inodes in Btrfs, akin to file metadata, are updated to reflect the new data locations, ensuring the filesystem remains consistent and up-to-date.

4. **Impact on Forensic Analysis**: For forensic investigators, CoW adds a layer of complexity. Since data blocks are not overwritten, remnants of the original data may persist elsewhere on the disk, offering potential avenues for data recovery and analysis. This aspect of Btrfs can provide valuable insights during incident investigations, allowing for the reconstruction of file histories and system states.
5. **Performance Considerations**: While CoW offers significant benefits in terms of data integrity and snapshotting, it can introduce performance overhead, particularly in write-heavy environments. This is due to the additional write operations and metadata updates required. Understanding this trade-off is crucial for system administrators and forensic investigators alike.

Snapshots in Btrfs - Forensics goldmine?

The Mechanics of Snapshots in Btrfs

Btrfs's snapshot feature is a powerful tool for data recovery and forensic analysis. Unlike traditional backups, snapshots in Btrfs are created instantly and without significantly impacting disk space. This efficiency stems from Btrfs's CoW mechanism. When a snapshot is created, it's essentially a mirror of the filesystem's metadata at that point in time. The actual data blocks are not duplicated but shared between the original filesystem and the snapshot. As changes occur post-snapshot, only the altered blocks are written to new locations, while the snapshot continues to reference the original blocks.

This approach to snapshots offers several advantages:

1. **Instantaneous Creation**: Snapshots are created almost instantaneously, providing a real-time reflection of the filesystem without the delay associated with traditional backup processes.

2. **Space Efficiency**: Since data blocks are shared between snapshots and the original filesystem, there is minimal additional disk space usage, making it an efficient way to maintain multiple historical states of the filesystem.

3. **Data Integrity**: The integrity of the snapshot data is maintained as it remains unaltered, providing a reliable point of reference for data recovery and forensic analysis.

Forensic Implications of Btrfs Snapshots

For forensic investigators, Btrfs snapshots are invaluable. They allow the reconstruction of a filesystem's state at various points in time, facilitating a thorough investigation into the sequence of events leading up to an incident. This capability is particularly crucial in scenarios such as data breach investigations or recovery of accidentally deleted files. The ability to revert to a previous state of the filesystem can also be instrumental in mitigating the impact of ransomware attacks or other forms of data corruption.

Inode Structure in Btrfs: A Comparative Analysis

Understanding Btrfs Inodes

The inode structure in Btrfs plays a pivotal role in data organisation and access. Unlike traditional filesystems like EXT4, which allocate a fixed number of inodes at filesystem creation, Btrfs adopts a more flexible approach by dynamically allocating inodes. This dynamic allocation is particularly beneficial in environments with variable inode usage patterns, as it avoids the limitations of a fixed inode count.

Btrfs inodes are stored in struct btrfs_inode_item at offset zero in the key, and they store the traditional stat data for files and directories. The inode items are always the lowest-valued key for a given object. This structure ensures efficient data management and quick access to file metadata.

Comparison with EXT4 and NTFS

In comparison to EXT4 and NTFS, Btrfs's inode structure offers several advantages. While EXT4 uses a static approach to inode allocation, Btrfs's dynamic method allows for more efficient use of filesystem resources, especially in large-scale and cloud-based environments. Unlike NTFS, which supports alternate data streams, Btrfs does not provide this feature but focuses on a robust and efficient extent-based approach for file data management.

The Zero Inode Count Phenomenon

A unique aspect of Btrfs is its reporting of inode counts, often showing zero. This occurs due to the filesystem's method of dynamically allocating inodes as needed, rather than maintaining a fixed count. For forensic investigators, this characteristic of Btrfs requires a shift in approach when analyzing filesystem metadata, as traditional methods of inode count analysis may not apply.

Timekeeping and Timestamps in Btrfs: Forensic Significance

Precision in Timekeeping

In Btrfs, timekeeping is handled with exceptional precision, employing 64-bit signed integer offsets from the Unix epoch (1970-01-01T00:00:00Z). This high-resolution approach, with nanosecond-level precision, is essential for forensic accuracy, providing a detailed timeline of file activities.

Timestamps Stored in Inodes

Btrfs inodes store multiple timestamps, offering comprehensive temporal metadata for each file. These include:

- **Creation Time**: Marks when the file or directory was first created. This is often called the **btime** or **crtime**.

- **Modification Time**: Indicates the last time the file's content was modified. This is generally referred to as the **mtime**.
- **Metadata Modification Time**: Reflects the last time the file's metadata, such as permissions or ownership, was changed. This is normally called the **ctime**.
- **Access Time**: Records the last time the file was accessed or read. This is called **atime** and its mechanism is driven by how the filesystem is mounted.

These timestamps are stored at specific offsets within the inode structure, ensuring precise and reliable record-keeping. For forensic investigators, this level of detail is crucial for reconstructing the sequence of events around a file or directory.

Forensic Analysis and Timestamps

The granularity of these timestamps in Btrfs allows forensic investigators to construct a detailed chronology of file activities, a key component in digital investigations. Understanding the exact timing of file creation, modification, and access can provide critical insights into user actions and system events, aiding in the unravelling of complex incidents.

File Deletion Process in Btrfs: A Forensic Perspective

The Deletion Mechanism

In Btrfs, the file deletion process is distinct and carries significant implications for forensic investigations. When a file is deleted, Btrfs removes the inode entry and marks the associated data blocks as free. However, this does not immediately remove the actual data from the disk. Due to the Copy on Write (CoW) feature, previous versions of the data may still exist in other locations on the disk, especially if snapshots were taken before the deletion.

Implications for Data Recovery and Forensic Analysis

This deletion mechanism in Btrfs presents unique opportunities for data recovery and forensic analysis:

1. **Potential Data Remnants**: Even after deletion, remnants of the file can remain in the filesystem, especially within snapshots or unallocated blocks, providing a window for forensic recovery.
2. **Analysis of Deletion Events**: Understanding the deletion process in Btrfs allows forensic investigators to reconstruct file deletion events, potentially revealing crucial information about user actions and system changes.
3. **Challenges in Data Reconstruction**: The CoW feature, while beneficial for data integrity, can complicate forensic analysis. Investigators must navigate through various snapshots and unallocated spaces to piece together the deleted file, a process that requires specialised tools and an understanding of Btrfs's architecture.

Summary

Btrfs is a very new filesystem with lots of advanced features. However, this also means it is often less well-understood and has significantly less support within security tools. The good news for incident responders is that Btrfs is not often found within commercial environments today, but this might change. For police or law enforcement investigators it is unfortunately very commonly found on NAS devices.

Overall, this forces us as responders/investigators to put more time and effort into understanding the filesystem and any nuances it creates.

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