The Risks of File Deletion and Data Persistence in Storage Systems

Barbarella Castillo

Abstract— **In an increasingly digital world, data deletion is often viewed as a simple command. We believe that once something is deleted it is gone forever. However, the assumption that deleted files are permanently erased is very misleading, giving the user a false sense of security that is rarely questioned. This paper explores the technical realities and security risks associated with file deletion and data persistence across various storage systems. By analyzing file system behaviors, secure deletion methods, and regulatory implications, this research reveals critical gaps between user expectations and actual data removal, emphasizing the need for informed deletion practices and compliance strategies.**

I. Introduction

Users commonly believe that deleting a file from their computer permanently removes it from existence. Yet, in many cases, deleted data remains recoverable due to how file systems manage storage. This paper investigates how different file systems handle file deletion, the vulnerabilities introduced by data persistence, and the evolving landscape of secure deletion practices. The disconnect between user assumptions and technical reality underscores a pressing need to understand the mechanics and implications of data deletion in modern storage environments. File deletion involves two main components: metadata management and block deallocation. When a file is deleted, most file systems do not immediately erase the file's contents; instead, they mark the associated blocks as available for reuse and update file metadata to remove references to the file.

- NTFS (Windows): Updates the Master File Table (MFT) to reflect file removal but retains the file data on disk until overwritten.
- ext4 (Linux): Uses inode structures to reference data blocks; deleting a file removes the inode link but leaves data intact.
- APFS (macOS): Employs snapshot and clone technologies; deletion typically removes file references while underlying data persists temporarily.

This delayed overwriting creates opportunities for data recovery using forensic tools, which can reconstruct files from unallocated space. Persistent data poses significant security threats:

- Unauthorized Access: Deleted files can be recovered using forensic tools like Autopsy or PhotoRec.
- Data Breaches: Improper disposal of storage media has led to breaches involving personal and corporate data.
- Privacy Concerns: Users and organizations may unintentionally retain sensitive information, violating privacy policies and user trust.

Real-world cases, such as data recovery from discarded SSDs or improperly wiped hard drives, highlight the ongoing risk posed by residual data.

II. Related Works

Numerous studies and technical reports have explored the challenges of secure data deletion and the persistence of deleted files, especially in the context of modern storage systems.

Garfinkel (2003) was among the first to highlight that deleting files through conventional means (e.g., emptying the Recycle Bin) does not remove the actual data from disk. His work, "Design Principles and Patterns for Computer Forensics Tools", demonstrated how forensic tools could recover deleted files and emphasized the need for improved deletion practices [1].

Gutmann's paper, "Secure Deletion of Data from Magnetic and Solid-State Memory", proposed a method of overwriting data 35 times with carefully chosen patterns to counteract magnetic data remanence. While later studies questioned the necessity of so many passes, this work established a foundational framework for secure deletion [2].

Wei et al. (2011) expanded the conversation to include solid state drives (SSDs), revealing that traditional overwriting techniques often fail due to SSDs' internal architecture. Their study, "Reliably Erasing Data from Flash-Based Solid State Drives", showed that even secure deletion tools were unable to remove data completely from many SSDs [4]. This work highlighted the deficiencies of legacy deletion methods for emerging storage technologies.

More recently, Reardon et al. (2013) examined secure deletion in the context of file systems and data structures, proposing alternatives such as self-encrypting drives and deletion aware file systems [5]. These approaches aim to integrate secure deletion into the design of storage systems, rather than just working with post deletion tools.

In the cloud domain, providers like Google Cloud have published detailed documentation on their internal deletion processes [3]. These documents explain the multi layered deletion pipelines and their attempts to follow compliance with legal requirements such as GDPR and CCPA. This shift toward system level deletion processes at scale is an important evolution in secure data life cycle management.

III. Secure Deletion Techniques

Secure deletion methods actively overwrite or cryptographically destroy the data before releasing it back to the storage system. Unlike traditional deletion, where the file's metadata is removed but the content often remains.

A. *Overwriting*

The most common method of secure deletion is data overwriting, which involves writing random or zeroed data over the file's existing location. Classic methods, such as the Gutmann method, use multiple passes of random data to make recovery practically impossible. However, overwriting only works reliably on traditional spinning hard drives (HDDs). On solid-state drives (SSDs), overwriting is not guaranteed due to wear leveling and internal block management.

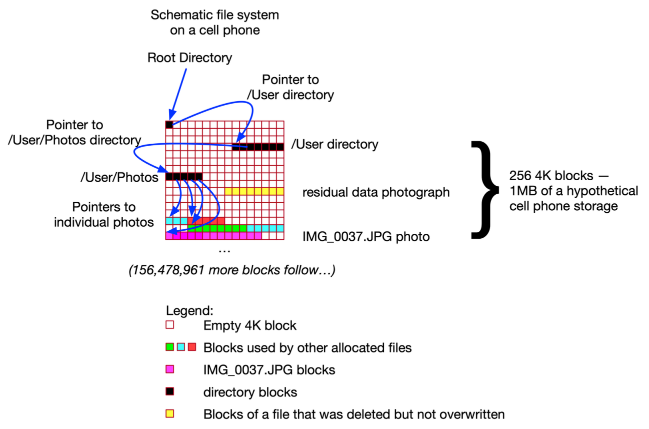
Schematic file system on a cell phone
Root Directory
Pointer to /User directory
Pointer to /User/Photos directory
/User directory
/User/Photos
residual data photograph
256 4K blocks — 1MB of a hypothetical cell phone storage
Pointers to individual photos
IMG_0037.JPG photo
...
(156,478,961 more blocks follow...)
Legend:
Empty 4K block
Blocks used by other allocated files
IMG_0037.JPG blocks
directory blocks
Blocks of a file that was deleted but not overwritten

FIG 1: Modern devices like smartphones and laptops use flash memory divided into blocks, typically 4,096 bytes each. Files, such as images, are stored across these blocks, while file names and metadata are kept in directory blocks that link to the files. These directories are part of a hierarchical structure forming the file system. When a file is deleted, its name is removed from the directory and marked as free space, but its data usually remains intact, making it recoverable with forensic tools.[6]

B. *Cryptographic Erasure*

An increasingly popular and efficient technique is cryptographic erasure, especially used in enterprise environments. Here, data is encrypted by default, and secure deletion is achieved by discarding or overwriting the encryption key. This renders all encrypted data unreadable and inaccessible without requiring physical erasure.

C. *File System and OS Tools*

Many operating systems provide tools to perform secure deletion beyond the traditional "move to trash" functionality. These utilities are designed to reduce the risk of data persistence by overwriting file contents or managing deletion through secure system calls.

- Linux systems offer command-line tools such as shred, wipe, and secure-delete, which can overwrite a file multiple times before deletion.
- Windows provides the Sysinternals utility sdelete, which securely erases file content and cleans free space.
- macOS previously included srm (secure remove), which has since been deprecated, but similar functionality can be achieved with third-party tools.

These tools work best on traditional HDDs. Their effectiveness is limited on SSDs, which abstract physical storage locations through flash translation layers (FTLs), making overwrite-based techniques less reliable.

In addition, major cloud service providers like Google Cloud have developed their own secure data deletion processes see Fig 2. Google Cloud's Data Deletion [3] outlines the life cycle of data deletion across multiple storage layers including user deletion, garbage collection, overwrites, and final erasure from physical media. This process emphasizes how cloud services approach secure deletion in a distributed environment with automated data replication and fail over systems.

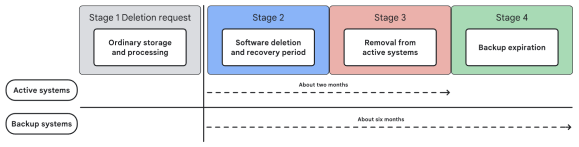

FIG 2: Google Cloud's Data Deletion Process

D. SSD-Specific Challenges and the TRIM Command

Modern SSDs complicate secure deletion due to the use of wear leveling and the TRIM command. When a file is deleted on an SSD, the TRIM command notifies the drive that blocks are no longer in use, allowing the SSD to clean up and reuse them later. However, TRIM does not guarantee the immediate or secure erasure, it only flags the data as disposable. Forensic tools may still recover recently deleted data if the SSD hasn't physically erased the blocks.

E. Demo: Insecure and Simulated Secure Deletion in Python

To illustrate the concepts discussed above, a simple to understand Python project was built to simulate how insecure and "secure" deletion behaviors work. This program creates a sensitive file, then deletes it using regular file system operations, and attempts to recover it. It also simulates an SSD's TRIM behavior by zeroing the file contents before deletion, mimicking the drive's internal garbage collection process.[7]

IV. Evaluation

This research evaluated the effectiveness of file deletion methods and the risks posed by data persistence across storage systems. Traditional deletion techniques, such as removing file references by using os.remove, were shown to be insecure, as the file data can remain recoverable. Secure deletion methods like overwriting and cryptographic erasure provide better protection but will greatly depend in effectiveness based on the storage medium. Especially SSDs, which complicate deletion through wear leveling and the TRIM command.

The Python deletion project reinforced these findings. It demonstrated how deleted files can leave traces and how simulating a TRIM-like operation, zeroing data before deletion can reduce the ease of recovering, although it does not eliminate it. This simple test helped visualize the gap between perceived and actual deletion.[7]

Overall, the evaluation highlights the need for integrated secure deletion practices in both local and cloud environments to ensure data privacy, meet legal obligations, and reduce the risk of data recovery after deletion.

V. Future of Deletion

New storage technologies introduce complexities to safe deletion. Solid-State Drives (SSDs) use TRIM and wear leveling, making secure deletion harder to guarantee [6]. Cloud Storage, makes physical data locations more distant and complex to understand, and may replicate data across servers, complicating deletion verification. File System Snapshots, systems like ZFS and APFS maintain historical states, potentially retaining data users might think is deleted. Researchers continue to explore privacy preserving file systems and deletion aware designs to address these concerns, and as awareness for secure file deletion grows, more innovation will come.

VI. Conclusion

Data persists after deletion, creating security and privacy risks. This paper explores file systems, secure deletion methods, and compliance requirements to emphasize the need for proper data handling strategies. As storage technologies advance, we must adapt our secure deletion methods to balance performance, compliance, and data protection.

References

[1] 1. Garfinkel, S. L. (2003). ”Design Principles and Patterns for Computer Forensics Tools. Communications of the ACM”.

[2] Gutmann, P. (1996). ”Secure Deletion of Data from Magnetic and Solid-State Memory. USENIX”.

[3] Google Cloud. ”Data deletion on Google Cloud”. https://cloud.google.com/docs/security/deletion

[4] Wei, M., Grupp, L. M., Spada, F. M., Swanson, S. (2011). ”Reliably Erasing Data from Flash-Based Solid State Drives. Proceedings of the USENIX Conference on File and Storage Technologies (FAST)”.

[5] Reardon, J., Basin, D., Capkun, S. (2013). ”Secure Data Deletion from Persistent Media. In Proceedings of the USENIX Security Symposium”.

[6] Garfinkel, S. (2024) ’Complete Delete: In Practice, Clicking “Delete” Rarely Deletes. Should it?’.

[7] GitHub Repo Link: https://github.com/barbare11a/CIS5346DeletionResearch/