International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Evaluating and Comparing Deleted File Recovery Techniques in NTFS and FAT: Computer Files Security Implications

Elvis Chukwuani, Collette Eguakun, Crsitian Vasu, Adebunmi Adewusi

*Department of Computer Science Bowling Green State University, Bowling Green, Ohio
{elvisc, ceguaku, cvasu, adewusa}@bgsu.edu

Abstract

In this research, a comparative analysis of deleted file recovery techniques is presented on two of the widely used filesystems i.e. NTFS and FAT which is done according to their efficiency in recovering files with different life spans from an expert computer security perspective as well as evaluated respectively using standards set by The National Institute of Standards and Technology (NIST). Although many works have been dedicated to deleted file recovery, only a few contrast its specialized techniques within these modern environments. In this study, we fill that gap by examining both systems using a series of controlled experiments. Through offering insights valuable to cybersecurity and forensic experts, we intend to provide a systematic view of what recovery methods are effective within NTFS and FAT with the ultimate goal of improving results in digital forensics.

I. Introduction

The integrity of this data before and after deletion has grown in importance, as digital data proliferates its way throughout industries, governments, and even everyday life. In digital forensics, recovering the file is an essential phase for criminal investigation as well as incident response or data preservation. NTFS (New Technology File System) and FAT (File Allocation Table), are examples of some common file systems utilized in Windows environments[1]. File deletion can differ from system to system which also causes various challenges when it comes to recovering lost or deleted files. This paper proposes guidelines to analyze and compare the deleted file recovery methods in NTFS with those on FAT. This research is motivated by the growing sophistication of cybercrime, as it often relies on a deep understanding of file deletion and recovery. For instance, cybercriminals regularly delete data to conceal evidence of a crime and force forensic specialists to use recovery techniques to get that information back[2]. This study aims to bridge this gap by reviewing existing state-of-the-art recovery techniques and quantifying their efficiency, and accuracy capacity.
Research Questions:
Q1: Are NTFS recovery techniques for deleted files as efficient and successful compared to FAT?
RQ2: The strengths and limitations of current recovery methods for each file system
RQ3: How susceptible are these recovery methods to adversarial actions, such as intentional overwriting or encryption?
Expected Contributions:
1. Comparisons of the most commonly used file deletion recovery methods in NTFS and FAT
2. Identified Gaps & Limitations Enhancing Recovery methods
3. The principles of forensic investigation and the considerations with file recovery from a cybersecurity perspective.

II. Related Work

A wealth of work has been done in this direction concerning file recovery techniques, but mostly only for NTFS or FAT and not performing direct comparisons. Developed by Microsoft, NTFS is famous for having sophisticated data structures that offer a high-security level and superior performance features found in FAT (an older system) still widely used because of its simplicity and cross-compatibility with multiple platforms. Many research papers focus on file recovery in NTFS deleted files. For instance, Carrier [3] identified that NTFS' Master File Table (MFT) was very effective at persisting metadata about files even after they had been deleted. Another example is the study that investigated methods for decoding public data from partially erased sectors, common in NTFS[4]. By contrast, this simplicity in FAT file recovery has been discussed at length by previous works, which showed that the structural complexity of NTFS meant it was more difficult to overwrite files but also harder to navigate during a subsequent recovery. Still, there are very few works that take these findings together and provide a comparative analysis of the recovery techniques belonging to both in the case of NTFS and FAT. Moreover, previous research only discusses the success rate of recovery and not how exactly malicious activities (i.e., data corruption or encryption) can interfere with that process in practice. This is the void in literature that this study aims to fill[2].

III. Problem/Approach:

We're trying to tackle a gap in current research: there's not much out there that compares how well we can recover deleted files from NTFS and FAT, especially when it comes to modern cybersecurity issues. Here's what we're planning to do: We'll set up some experiments where we delete data from both NTFS and FAT drives. Then, we'll use different recovery tools to try and get that data back. We'll be looking at a few key things:

1. How often can we recover the files?
2. How long does it take?
3. What happens when part of the data has been overwritten?

IV. Experiment

Our research aims to assess deleted file recovery techniques on NTFS and FAT file systems. The following steps outline the setup and methods that will be used in the experiment, with attention to creating controlled conditions for an accurate comparison of the recovery capabilities on each file system.

A. Setup

1. Setting Up the Virtual Environment.

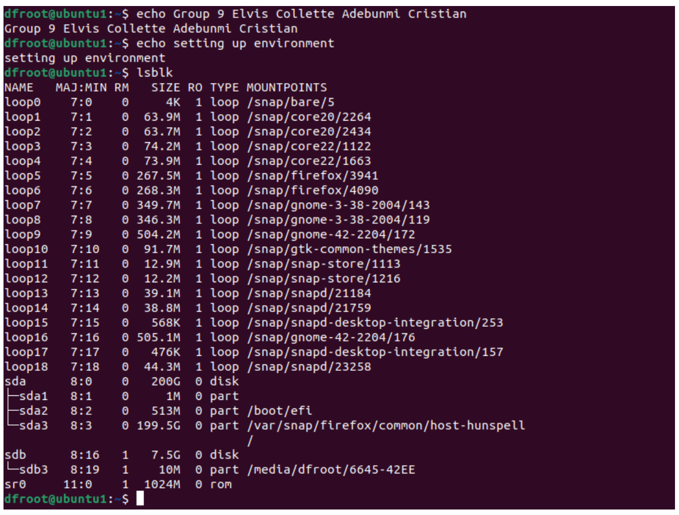
   a. We first downloaded a virtual machine (VM) called Virtual box. This environment is meant to simulate a controlled forensic setting. The following steps detail the setup process.

   b. Then installed Ubuntu Forensics on the VM, serving as the platform for conducting forensic tests. Ubuntu Forensics provides a stable Linux environment equipped with various forensic tools.

   c. Also, on the VM we installed key forensic recovery tools such as Autopsy and the Sleuth Kit (TSK), which includes command-line tools like fls, icat, and others.

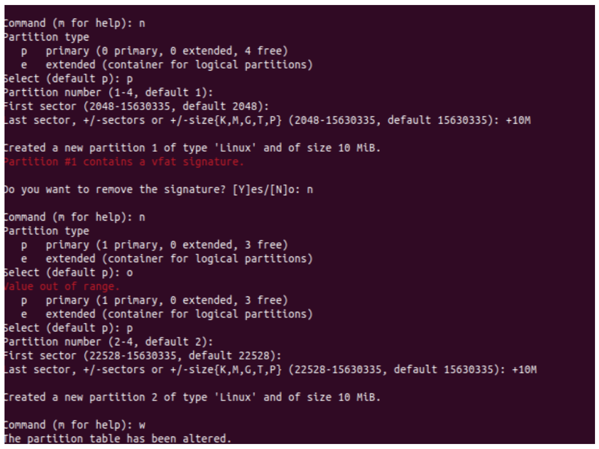
2. Preparing our test storage device

   a. We got a SanDisk USB flash drive for this research then fully formatted it. We erased old partitions and filesystems from the drive. This was done to remove any bias from affecting our results. The image below shows how we used the "lsblk" command on Ubuntu to view our flash drive which is identified as "sdb" in our environment.



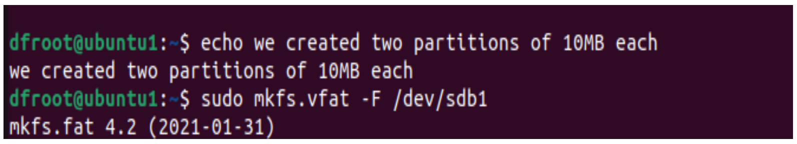
```
dfroot@ubuntu1:~$ echo Group 9 Elvis Collette Adebunmi Cristian
Group 9 Elvis Collette Adebunmi Cristian
dfroot@ubuntu1:~$ echo setting up environment
setting up environment
dfroot@ubuntu1:~$ lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
loop0    7:0    0     4K  1 loop /snap/bare/5
loop1    7:1    0  63.9M  1 loop /snap/core20/2264
loop2    7:2    0  63.7M  1 loop /snap/core20/2434
loop3    7:3    0  74.2M  1 loop /snap/core22/1122
loop4    7:4    0  73.9M  1 loop /snap/core22/1663
loop5    7:5    0 267.5M  1 loop /snap/firefox/3941
loop6    7:6    0 268.3M  1 loop /snap/firefox/4090
loop7    7:7    0 349.7M  1 loop /snap/gnome-3-38-2004/143
loop8    7:8    0 346.3M  1 loop /snap/gnome-3-38-2004/119
loop9    7:9    0 504.2M  1 loop /snap/gnome-42-2204/172
loop10   7:10   0  91.7M  1 loop /snap/gtk-common-themes/1535
loop11   7:11   0  12.9M  1 loop /snap/snap-store/1113
loop12   7:12   0  12.2M  1 loop /snap/snap-store/1216
loop13   7:13   0  39.1M  1 loop /snap/snapd/21184
loop14   7:14   0  38.8M  1 loop /snap/snapd/21759
loop15   7:15   0   568K  1 loop /snap/snapd-desktop-integration/253
loop16   7:16   0 505.1M  1 loop /snap/gnome-42-2204/176
loop17   7:17   0   476K  1 loop /snap/snapd-desktop-integration/157
loop18   7:18   0  44.3M  1 loop /snap/snapd/23258
sda      8:0    0   200G  0 disk
├─sda1   8:1    0     1M  0 part
├─sda2   8:2    0   513M  0 part /boot/efi
└─sda3   8:3    0 199.5G  0 part /var/snap/firefox/common/host-hunspell
                                 /
sdb      8:16   1   7.5G  0 disk
└─sdb3   8:19   1    10M  0 part /media/dfroot/6645-42EE
sr0     11:0    1  1024M  0 rom
dfroot@ubuntu1:~$ 
```

   b. Partition Setup: We then created a DOS partition on the drive. This setup allowed both NTFS and FAT filesystems. The two systems could then be compared for recovery success. The image below shows how we formatted the drive by creating a new DOS partition then created two different primary partitions of size 10mb for our experiment, one is going to be for FAT and the other is going to be NTFS. So basically, you use the "fdisk" command to this, then we use "-o" to create the DOS partition, "-n" for new partition, "-p" for primary partition. Then you can choose your desired size and use the "w" command to save your changes.

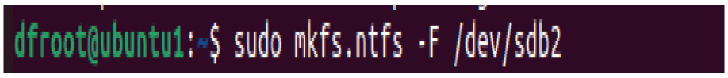


```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-15630335, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-15630335, default 15630335): +10M

Created a new partition 1 of type 'Linux' and of size 10 MiB.
Partition #1 contains a vfat signature.

Do you want to remove the signature? [Y]es/[N]o: n

Command (m for help): n
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): o
Value out of range.
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (2-4, default 2):
First sector (22528-15630335, default 22528):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (22528-15630335, default 15630335): +10M

Created a new partition 2 of type 'Linux' and of size 10 MiB.

Command (m for help): w
The partition table has been altered.
```

c. Filesystem Setup: NTFS and FAT Creation with the DOS partition established, both NTFS and FAT32 filesystems were set up. We used the "mkfs.vfat" and "mkfs.ntfs" commands to create the FAT and NTFS filesystem on our partitions.



```
dfroot@ubuntu1:~$ echo we created two partitions of 10MB each
we created two partitions of 10MB each
dfroot@ubuntu1:~$ sudo mkfs.vfat -F /dev/sdb1
mkfs.fat 4.2 (2021-01-31)
```



```
dfroot@ubuntu1:~$ sudo mkfs.ntfs -F /dev/sdb2
```

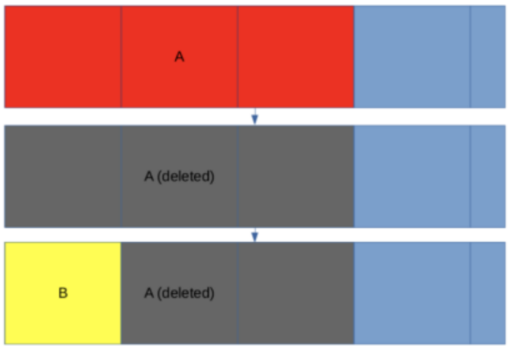
d. Data Preparation and Deletion: For controlled testing, we created two files, file A which was size 3mb and file B which was size 1mb.

i. File Creation: Files are created using the dd command to ensure precise byte sizes.

ii. File Deletion: Files are deleted using standard and permanent deletion methods.

B. Recovery

We will first create the A file of 3mb in each filesystem then delete it. Then create file B of 1mb and try to recover file A in each system and then evaluate our results.



Each tool will attempt to recover the files, and the results will be compared based completeness of the recovered file, and success rates.

The recovery process involved systematic analysis and file recovery using the forensic tools fls and icat from The Sleuth Kit (TSK). These tools were used to recover deleted files from the NTFS and FAT32 file systems. Each step of the process is detailed below, including the specific commands, observations, challenges, and evaluations.

Overview of Forensic Tools

i. fls (File List): The fls tool is used to enumerate files and directories within a file system, including those marked as deleted. It provides detailed metadata such as inode numbers, timestamps, and file paths. In the context of forensic recovery, fls acts as the first step in identifying deleted files and their associated metadata.

ii. icat (Image Cat): icat is a file content extraction tool. It uses inode numbers (or metadata addresses) to directly extract the content of files from a disk image. It is particularly valuable in recovering deleted files identified by fls.

1) Using fls and icat

a. Command Execution (FLS)

The recovery process began with fls, which was used to identify deleted files on both NTFS and FAT32 partitions. The command was executed as follows:

```
-fls -r /mnt/usb > file_list.txt
```

- -r: Enables recursive listing of all files, including those in nested directories.
- /mnt/usb: Specifies the mount point of the test USB drive.
- file_list.txt: Redirects the output to a text file for easier analysis.

Observations on NTFS

The output of fls on the NTFS partition provided detailed metadata, including:

i. File Name: Original names of deleted files.
ii. Inode Number: Unique identifier for each file.
iii. Timestamps: Information about creation, modification, and deletion.
iv. File Type: Distinguishing regular files, directories, and special files.

b. Command Execution (ICAT)

The icat tool was employed to extract the content of deleted files identified by fls. For each file, its inode number was noted from the fls output and used in the following command:

```
icat /mnt/usb 128 > recovered_file.txt
```

- /mnt/usb: Specifies the mount point of the drive.
- 128: Represents the inode number of the target file.
- recovered_file.txt: Redirects the recovered content to a new file.

c. Challenges with icat

a) File Fragmentation: On both NTFS and FAT32, fragmented files posed a challenge. While icat could extract content from a single inode, fragmented files required manual reassembly.
b) Overwriting: Once a file's data blocks were reassigned, recovery became impossible or yielded corrupt results.

C. Results

1) Key Observations from FAT Analysis:

Core Features Evaluation:

1. DFR-CR-01: Successfully identifies deleted files (e.g., aa2M and aa3M).
2. DFR-CR-02: Recovers data when prompted, but recovery is incomplete for overwritten sections.
3. DFR-CR-03: Recovery is limited to sectors identified in residual metadata; overwritten data is not retrieved.
4. DFR-CR-04: Only retrieves sectors from the "Deleted Block Pool."

Fragmentation Handling:

a) FAT relies on directory entries for file size and starting cluster.
b) Fragmentation cannot be fully analyzed post-deletion due to cleared FAT entries.

Overwriting Impact:

- Overwritten sectors are not included in the recovery output, maintaining strict adherence to "Deleted Block Pool."

2) Key Observations from NTFS Analysis:

Core Features Evaluation:

1. DFR-CR-01: Identifies deleted files effectively.
2. DFR-CR-02: Recovers data, including overwritten sections, albeit with merged content.
3. DFR-CR-03: Recovers all unallocated clusters, but also retains clusters overwritten by other files.
4. DFR-CR-04: Recovers overwritten clusters, violating the strict "Deleted Block Pool" criterion.

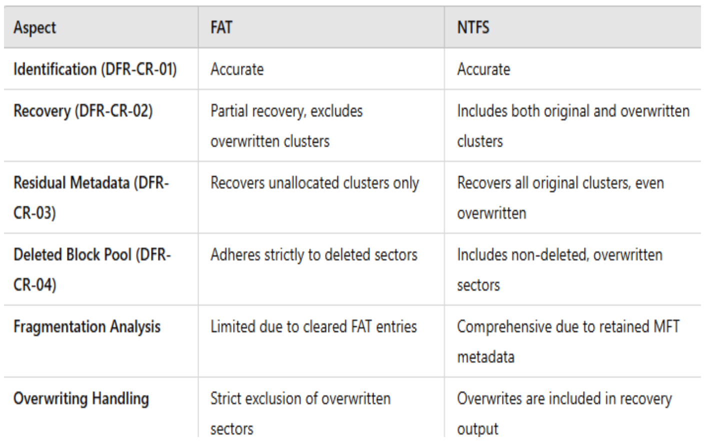
Fragmentation Handling

NTFS retains detailed file cluster mapping in MFT (Master File Table), even after deletion, enabling accurate fragmentation analysis.

Overwriting Impact:

Overwritten clusters are included in recovery, blending data from original and new files.

D. Model Developed



Aspect	FAT	NTFS
Identification (DFR-CR-01)	Accurate	Accurate
Recovery (DFR-CR-02)	Partial recovery, excludes overwritten clusters	Includes both original and overwritten clusters
Residual Metadata (DFR-CR-03)	Recovers unallocated clusters only	Recovers all original clusters, even overwritten
Deleted Block Pool (DFR-CR-04)	Adheres strictly to deleted sectors	Includes non-deleted, overwritten sectors
Fragmentation Analysis	Limited due to cleared FAT entries	Comprehensive due to retained MFT metadata
Overwriting Handling	Strict exclusion of overwritten sectors	Overwrites are included in recovery output

V. Threats to validity

These are some of the threats that may be posed by this research:
i. Hardware variances: a uniform hardware configuration for each experiment will mitigate system-wide performance variance thereby preventing their impact on results[5].
ii. Limitation of tools: Different technologies have different capabilities and can affect the results. Therefore, several instruments will be used to carry out more detailed investigations.
iii. Exogenous factors: We need to allow for events such as system crashes by repeating many times to make them reliable.

VI. Conclusion

This research aims to fill the gap in the comparative analysis of deleted file recovery techniques in NTFS and FAT file systems. By conducting a series of experiments using various recovery tools, we seek to evaluate the strengths, limitations, and vulnerabilities of these methods. The outcome of this research is expected to provide valuable insights into improving file recovery techniques and strengthening cybersecurity practices, particularly in security contexts. We anticipate that this study will also raise awareness of potential threats posed by inadequate file recovery practices and help inform the development of more robust recovery tools.

REFERENCES

[1] D. Kuts et al., “The peculiarities of deleted files recovery in FAT32 file system,” in Proc. USBEREIT, 2023. [Online]. Available: https://bgsu-summon-serialssolutions-com.ezproxy.bgsu.edu/… DOI: 10.1109/USBEREIT58508.2023.10158866.
[2] N. Zhang, Y. Jiang, and J. Wang, “The research of data recovery on Windows file systems,” in Proc. ICITBS, 2020. [Online]. Available: https://go.exlibris.link/78YGWfg3. DOI: 10.1109/ICITBS49701.2020.00141.
[3] B. Carrier, File System Forensic Analysis. 2015. [Online]. Available: https://bgsu-summon-serialssolutions-com.ezproxy.bgsu.edu/…
[4] X. S. Lin and Ohio Library and Information Network, Introductory Computer Forensics: A Hands-on Practical Approach, 1st ed. 2018. [Online]. Available: https://bgsu-summon-serialssolutions-com.ezproxy.bgsu.edu/… DOI: 10.1007/978-3-030-00581-8.
[5] Anonymous, “Disk Recovery Wizard - One of the Wizard Recovery Products uses Sophisticated Data Recovery Algorithms to undelete files from healthy discs,” M2 Presswire, 2016. [Online]. Available: https://bgsu-summon-serialssolutions-com.ezproxy.bgsu.edu/…
[6] E. Casey, Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet, 3rd ed. Academic Press, 2011.
[7] S. Luttgens, M. Pepe, and K. Mandia, Incident Response & Computer Forensics, 3rd ed. McGraw-Hill Education, 2014.
[8] J. Olsson and R. Boldt, “Improving forensic data recovery in FAT32 using cluster remapping,” in Proc. Digital Forensics Research Workshop (DFRWS), 2019.
[9] S. Garfinkel, “Digital forensics research: The next 10 years,” Digital Investigation, vol. 7, pp. S64–S73, 2010.
[10] A. Marziale, G. G. Richard III, and V. Roussev, “Massive threading: Using GPUs to increase the performance of digital forensics tools,” Digital Investigation, vol. 6, pp. S95–S105, 2009.