File Recovery Tool

Members: Bryan Leigh, Jetli Lopez

Date: 12/03/25 | **Course:** IT 360

INTRODUCTION

The File Recovery Tool was made to automatically monitor, detect, and respond to rapid file deletions within a specified directory. By using Python's watchdog library, the tool continuously observes file system activity. It also logs all deletion events in real time. When four or more files are deleted within a minute, the tool initiates a recovery process, restoring deleted files from a snapshot directory to ensure data integrity. This project demonstrates the practical application of automation in digital forensics and system protection, emphasizing proactive file monitoring, event logging, and recovery mechanisms to minimize data loss caused by accidental or malicious actions.

TECHNICAL IMPLEMENTATION

Our project is implemented in Python. We decided to go with Python because it is widely used in digital forensics, and it is easy to maintain and extend. The tool is split into three Python modules:

1. **main.py**

- Handles user input, sets up the folders, and starts the monitoring process of the tool.

2. **monitor.py**
    - Uses the watchdog library to monitor a chosen folder in real-time and detect any deletions.
3. **file_carver.py**
    - Responsible for recovering files from a snapshot folder and verifying them using hashes.
    - We use the watchdog library to listen for deletions.
        i. When a file is deleted, our handler logs the event and stores the timestamp and name of the file into a file called **activity_log.txt.**

If there are four or more deletions within a minute, it will detect rapid deletions and recover the deleted files. For recovery, we maintain a snapshot folder that contains copies of the original files. The file_carver.py module scans this snapshot and copies the following supported file types:

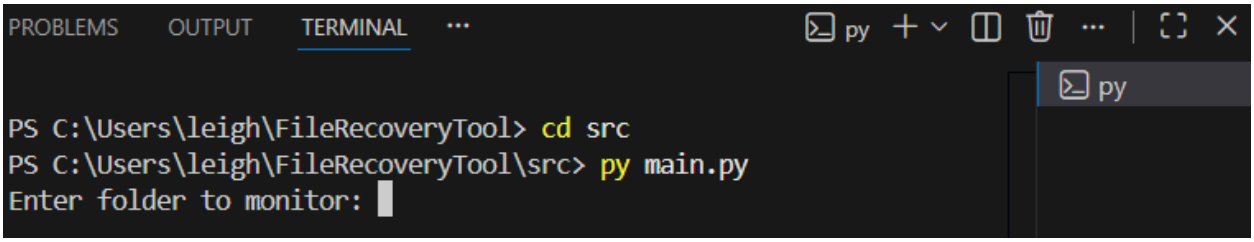
- .jpg
- .png
- .pdf
- .doc
- .ppt

When these files get deleted, a Recovered folder gets created, as well as when an alert gets displayed. We also use the hashlib library to compute a SHA-256 hash for each recovered file, which can be used for integrity checking and forensic documentation.

RESULTS

1. **Confirm Directory**

   First, you want to make sure you are in the right directory, then type py main.py or python main.py, depending on how your Python tool is set up. Your command line should look like the following:



2. **Choose desired folder**

Now you want to type whatever folder you want the tool to monitor. For this, we will use C:\Users\leigh\OneDrive\Desktop\TestFolder for our folder. The tool will start

```
PS C:\Users\leigh\FileRecoveryTool> cd src
PS C:\Users\leigh\FileRecoveryTool\src> py main.py
Enter folder to monitor: C:\Users\leigh\OneDrive\Desktop\TestFolder
[*] Starting File Recovery Tool on C:\Users\leigh\OneDrive\Desktop\
TestFolder...
[+] Monitoring started on C:\Users\leigh\OneDrive\Desktop\TestFolde
r
```

monitoring that folder like the following:

From here, the tool monitors the folder and waits for deletion. If a file were to get deleted, the following result would be displayed:

```
PS C:\Users\leigh\FileRecoveryTool> cd src
PS C:\Users\leigh\FileRecoveryTool\src> py main.py
Enter folder to monitor: C:\Users\leigh\OneDrive\Desktop\TestFolder
[*] Starting File Recovery Tool on C:\Users\leigh\OneDrive\Desktop\
TestFolder...
[+] Monitoring started on C:\Users\leigh\OneDrive\Desktop\TestFolde
r
[-] Deleted: download.jpg
```

As you can see, it detected a file being deleted in the folder which was jpg file. Now if you delete four or more files within a minute it will look like the following:

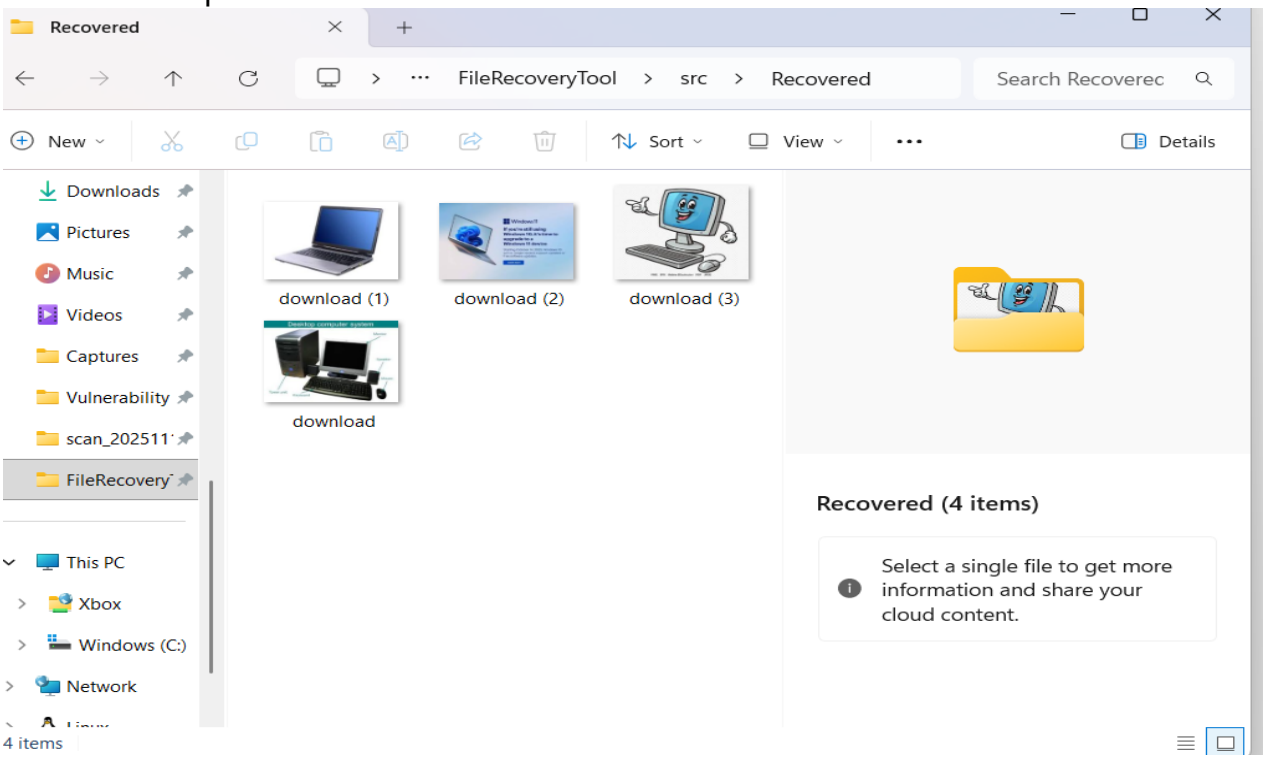
```
PS C:\Users\leigh\FileRecoveryTool> cd src
PS C:\Users\leigh\FileRecoveryTool\src> py main.py
Enter folder to monitor: C:\Users\leigh\OneDrive\Desktop\TestFolder
[*] Starting File Recovery Tool on C:\Users\leigh\OneDrive\Desktop\
TestFolder...
[+] Monitoring started on C:\Users\leigh\OneDrive\Desktop\TestFolde
r
[-] Deleted: download.jpg
[-] Deleted: download (1).jpg
[-] Deleted: download (2).jpg
[-] Deleted: download (3).jpg
[!] Rapid deletions detected!
```

When this happens, it recovers the deleted files and puts them in a recovered folder for you to see.

- All activity is also kept in activity_log.txt file as shown in the following:

```
2025-11-18 12:53:33,181 - INFO - Deleted file: download.jpg
2025-11-18 12:54:26,872 - INFO - Deleted file: download (1).jpg
2025-11-18 12:54:26,878 - INFO - Deleted file: download (2).jpg
2025-11-18 12:54:26,884 - INFO - Deleted file: download (3).jpg
2025-11-18 12:54:26,886 - WARNING - Rapid deletions detected: 4 files in 60 seconds
```

The following image shows the recovered folder with the recovered files that were deleted in the picture above.



LESSONS LEARNED & CONCLUSION

During this project, we learned both what worked well and what was challenging. Splitting the tool into separate parts for monitoring, recovery, and coordination worked well and made the code easier to test and fix. Using Python with libraries like watchdog and logging also helped us quickly add file monitoring and event logging without building everything from scratch.

Some things did not go smoothly at first. We had issues with setting up Python, installing dependencies, and dealing with PATH problems. We also ran into syntax and import errors when separating the code into multiple files. Fine-tuning the rapid deletion

detection so it didn’t trigger too often but still caught real events also took some trial and error.

In the future, we would spend more time early on setting up and testing the environment, design the logging format from the beginning, and possibly add a simple interface to make the tool easier to use. Overall, this project taught us the importance of planning, testing, and improving our design as we go.