AUBURN

UNIVERSITY

Project 2 – Automated File Recovery

Jason Alexander

&

Jennifer Johnson

December 3, 2021

Executive Summary

A disk image was provided to a student digital forensics team to analyze the files contained therein. An automatic file recovery program was developed with Python to automate the search and recovery of the files based on their file signatures. Potential file extensions to locate were given and included the following:

- AVI (Audio Video Interleave)
- BMP (Bitmap)
- DOCX (Microsoft Word document)
- GIF (Graphics Interchange Format)
- JPG (Joint Photographic Experts Group)
- MPG (Moving Picture Experts Group)
- PDF (Portable Document Format)
- PNG (Portable Network Graphics)

Depending upon the file extension, the appropriate file signature was used to analyze the properties of the file such as file size or footer information if available. Knowing these file properties allowed the team to be able to recover the files. Once a file was recovered it was placed in the same directory and a SHA-256 hash was generated from the file. Program output for each file included an appropriate file name and extension, start and end offsets, and the SHA-256 result for each. A total of thirteen files were found and successfully recovered as seen in Table 1.

Table 1: Recovered Files

File Type	File Name	Start Offset	End Offset
AVI	avi_file_1.avi	0x3c000	0x1be0a88
	SHA-256: 91c4520172aaf1f6d5d679db6e0957a7b366c11282e6802497073aa49c0e67b1		
	avi_file_2.avi	0x2427000	0x2d9a36c
	SHA-256: 9966c341d99f5e2fbfc1c607240cb19abc40846f03b799b6debcf2c212c2f1a9		
BMP	bmp_file_1.bmp	0x1e28000	0x1e3b076
	SHA-256: e03847846808d152d5ecbc9e4477eee28d92e4930a5c0db4bffda4d9b7a27dfc		
DOCX	docx_file_1.docx	0x2d9b000	0x2dbc5b7
	SHA-256: 0b6793b6beade3d5cf5ed4dfd2fa8e2ab76bd6a98e02f88fce5ce794cabd0b88		
GIF	gif_file_1.gif	0x214e000	0x23d59e5
	SHA-256: c3c82461c8d7cd3974a82967d5c6cf18449e1b373c8123b01508be277df725e4		
	gif_file_2.gif	0x23d6000	0x24261ee
	SHA-256: 8869dd5fcb077005be3195028db6fe58938c4ec2786a5ff7e818d2f5411ded52		
JPG	jpg_file_1.jpg	0x38000	0x3b055
	SHA-256: 59e0ec78f30c50db44d24a413ca1cccbd7ef5910cad4d3cf0e4753095725ec94		
	jpg_file_2.jpg	0x2148000	0x214d72f
	SHA-256: bde9e54f4e1ec3b6ab8d439aa64eef33216880685f8a4621100533397d114bf9		
	jpg_file_3.jpg	0x1e11000	0x1e27992
	SHA-256: 51481a2994702778ad6cf8b649cb4f33bc69ea27cba226c0fe63eabe2d25003b		
MPG	mpg_file_1.mpg	0x2dbd000	0x2faf8ac
	SHA-256: c9ed8592d0b31b24e5a7286469497cd817e7c32dd6a9347891db8c27c26d0153		
PDF	pdf_file_1.pdf	0x1be1000	0x1dd8491
	SHA-256: b52d27f414edf27872139ce52729c139530a27803b69ba01eec3ea07c55d7366		

	pdf_file_2.pdf	0x1e3c000	0x2147c7c
	SHA-256: 9ef248560dc49384c0ec666db6a7b4320bfec74e62baed1c610a765f056fd3b7		
PNG	png_file_1.png	0x1dd9000	0x1e10a7b
	SHA-256: 3967b4fc85eca8a835cc5c69800362a7c4c5050abe3e36260251edc63eba518f		

Table of Contents

List of Figures

List of Tables

1 Introduction

A disk image entitled Project2.dd was provided to the student digital forensics team to analyze the files contained therein. Our team consisted of two computer science and master's level students, Jason Alexander and Jennifer Johnson. The goal was to create an automatic file recovery program using Python to automate the search and recovery of the files based on their file signatures. We were given that the files on the disk image may possibly have file extensions that matched one or more of the following: AVI, BMP, DOCX, GIF, JPG, MPG, PDF, or PNG.

2 Background Research

To begin this project our team had to first be familiar with the various file signatures associated with each file extension. According to Cameron H. Malin et al. in Malware Forensics Field Guide for Windows Systems, a file signature is a "…unique sequence of identifying bytes written to a file's header…" and that "…different file types have different file signatures." [1] Knowing this information, we compiled a list of potential file signatures to search for from the following sources:

- Gary Kessler's file signature table [2]
- File signatures in database [3]
- Active@ File Recovery for Windows [4]

Additionally, to familiarize ourselves with various related coding techniques such as to read a disk image, search its contents, locate file signatures, and to recover files we used sources such as the following:

- Image carving [5]
- File recovery example using Python [6]

3 Methodology

The program was written using Python 3.8.5. This program is designed to find the file titled Project2.dd in the same directory as the program. For development and testing we used a SIFT-Workstation that contained the disk image and Python program. The workstation's terminal was used to run the program. Recovered files were stored in the same directory as the disk image and program.

Individual modules were constructed to handle each file type with a main method calling each module in turn. Files of type DOCX, GIF, JPG, or PNG were handled in the same manner. The program searches for file signatures that match these types. Headers used for DOCX, GIF, JPG, and PNG were "50 4B 03 04 14 00 06 00", "47 49 46 38 39 61", "FF D8 FF E0", "89 50 4E 47 0D 0A 1A 0A" respectively. File footers for DOCX, GIF, JPG, and PNG were "50 4B 05 06", "00 3B 00 00 00", "FF D9 00 00 00", "49 45 4E 44 AE 42 60 82" respectively. The start and end of the files are determined and documented from this information. Using the start and end of files, the program then recovers or carves out each file found.

The AVI module is slightly different in that the footer information is not available. Header information used for this type of file was "52 49 46 46". A list of the starting of each AVI file is compiled. From this list the size and ending of the file is determined. The program then recovers each file found based on this information.

The PDF module searched for headers matching "25 50 44 46" and footers matching "0A 25 25 46 4F 46". The headers returned were sufficient to deduce the start of file, however, searching for the footers could lead to false footers or rather not truly the end of the file. The first and last headers would be matches with the first and last footers, however, if the number of footers is more than the number of headers found then logic was put in place to match the correct header with the correct footer for ones in between. This logic essentially matches the header with the last footer that is less than the next header. Once these matches are made

then the file is recovered. Please note that we adjusted the names of the files in this document for quick reference and readability, the Python program will return a name such as 'startoffset_endoffset.extension'

The BMP module searches for headers matching "42 4D". There are no footers. Searching for this header may produce multiple false positives. To ensure we are retrieving the beginning of the file, the program checks to ensure the signature is at the beginning of a sector by dividing the offset by 512. If it is divisible by 512 then it is at the beginning of a sector. Additionally, the reserved sections are reviewed to filter out false positives by checking if the reserved section is equal to zero or not. The size of the file is returned from the header information and in combination with the rest of the information obtained the file is recovered.

Finally, the MPG module searches for headers matching "00 00 01 B3" and footers matching "00 00 01 B7". Validations were put in place to ensure the footer did not precede the header. Additionally, if the length of headers did not match the length of the footers then the extra false positives were not included thus giving the start and ending of the file(s) necessary to make a successful recovery.

4 Results and discussion

These are the files that were found and recovered:

1. Avi_file_1: This is an audio/video file of a bear in the woods as seen in Figure 1:

Figure 1: Avi_file_1

2. Avi_file_2: This is a video file of what looks to be an aerial view of a rock formation in the ocean as seen in Figure 2:

Figure 2: Avi_file_2

3. Bmp_file_1: This is an image of purple flowers as seen in Figure 3:

Figure 3: Bmp_file_1

4. Docx_file_1: This is a document which seems to contain a reference to the movie *A Christmas Story* as seen in Figure 4::

3wzf8RYd5zzPh5icWk3YAhnqZ4Rr4y3azeMSGpJc49s4utBMjUyAkeHhTGvzTCZ5gfDmEHLYrXqdVVxuB

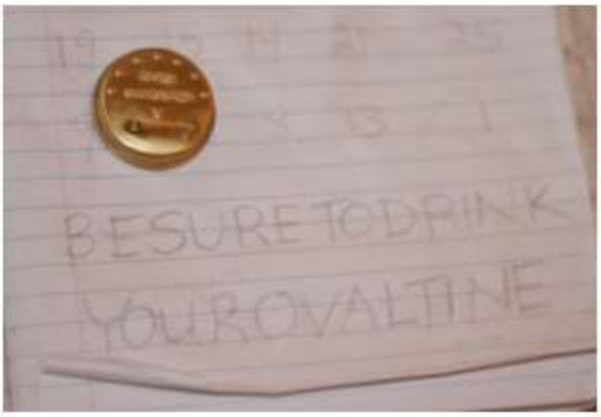

Figure 4: Docx_file_1

5. Gif_file_1: This gif seems to zoom in on the following image (Figure 5):

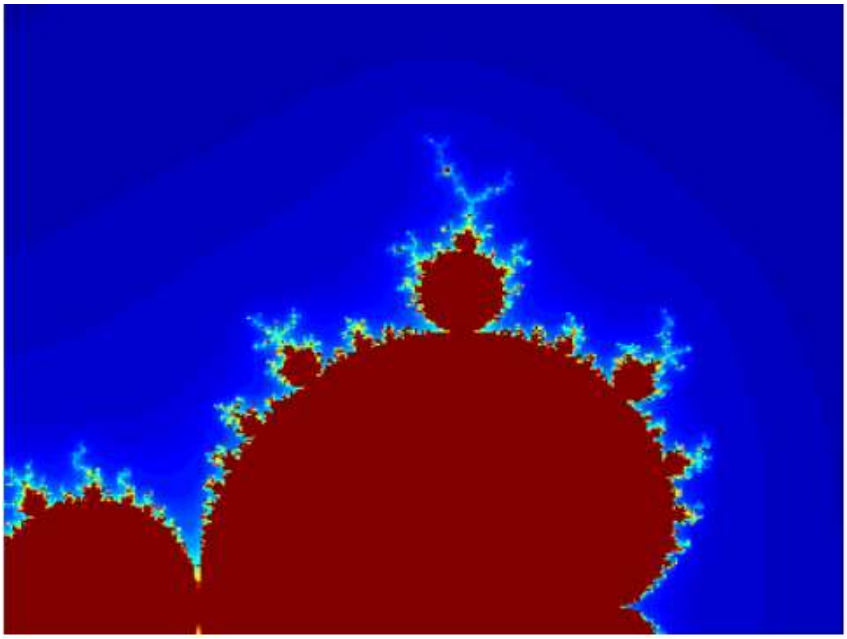

Figure 5: Gif_file_1

6. Gif_file_2: This gif is an image of a Pixar minion with flashing lights (Figure 6):

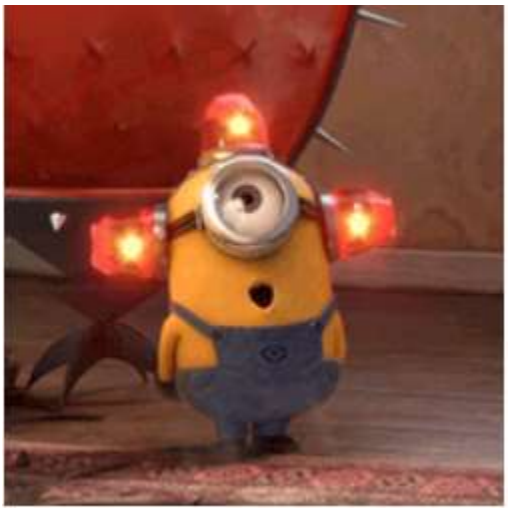

Figure 6: Gif_file_2

7. Jpg_file_1: This is an image of various flags (Figure 7):

Figure 7: Jpg_file_1

8. Jpg_file_2: This is an image of the Auburn University logo (Figure 8):

Figure 8: Jpg_file_2

9. Jpg_file_3: This is an image of the movie poster for the first Iron Man movie (Figure 9):

Figure 9: Jpg_file_3

10. Mpg_file_1: This is an image of the mpg file of the universe (Figure 10):

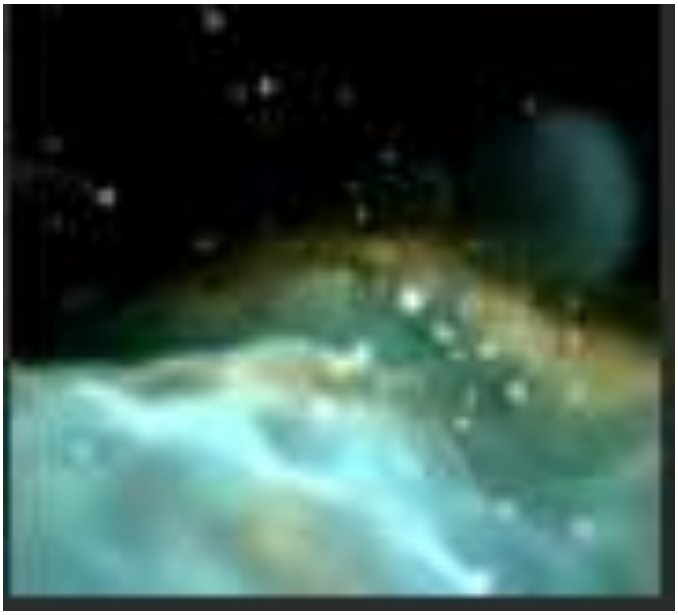

Figure 10: Mpg_file_1

11. Pdf_file_1: This is eBook on A Tale of Two Cities by Charles Dickens (Figure 11):

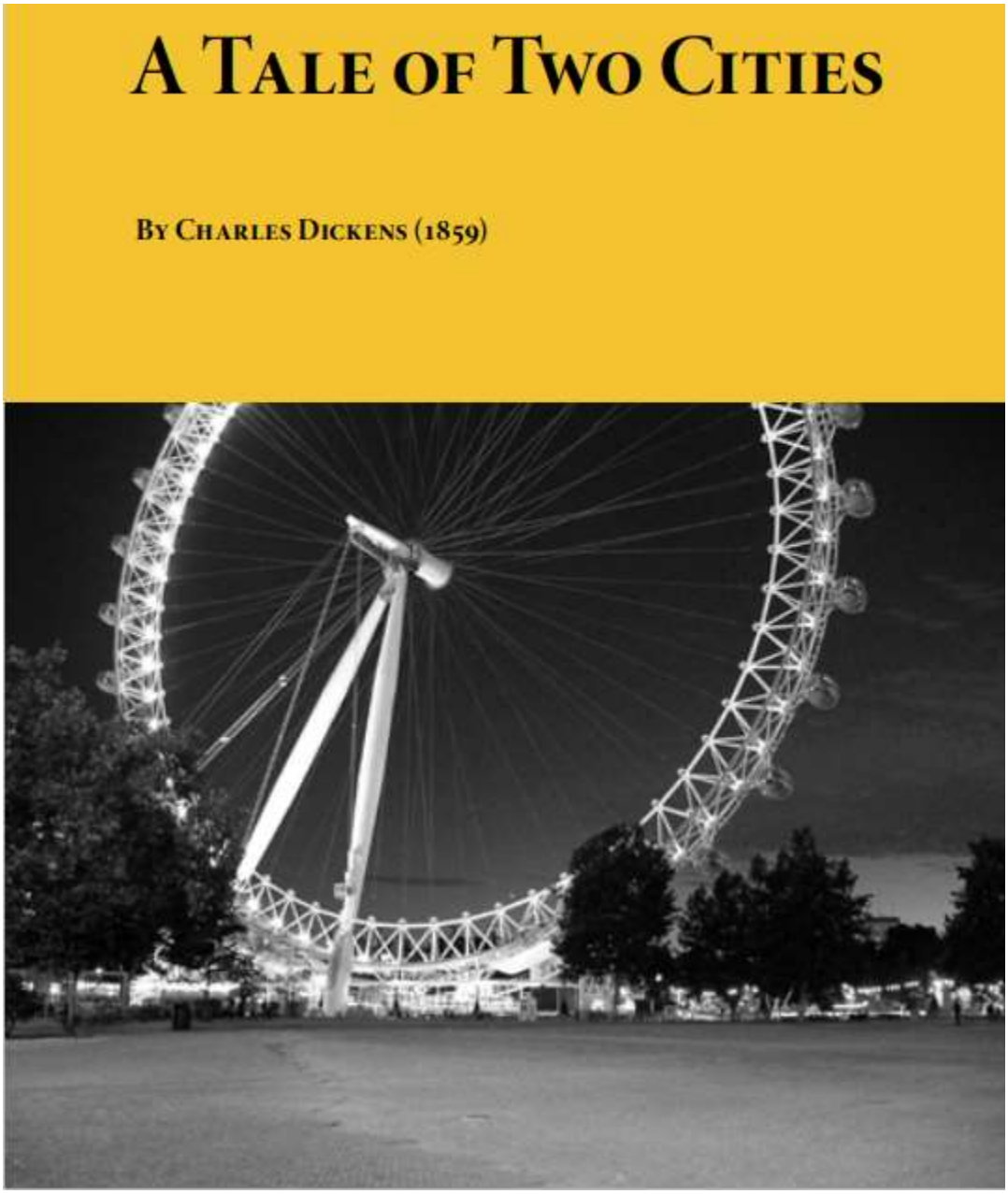

Figure 11: Pdf_file_1

12. Pdf_file_2: This is an eBook for Great Expectations by Charles Dickens (Figure 12):

Figure 12: Pdf_file_2

13. Png_file_1: This is an image of colored dice (Figure 13):

Figure 13: Png_file_1

A sample of the program's output is as follows in Figure 14:

```
sansforensics@siftworkstation: ~/Desktop/Project2
$ python3 main.py
Found PNG starting offset 31297536 End offset 31525491
carving it to 31297536_31525491.png
SHA256: c9ac5048fb37dc0a95f844e190507f7d145d869a0ccbe9d222eb7f5d1365efc2

Found PNG starting offset 47824425 End offset 47948420
carving it to 47824425_47948420.png
SHA256: 36d2109322f9da9cd0f56d2d56987f778692fad9b91a13617a141a65ce32550f

Found JPG starting offset 229376 End offset 241747
carving it to 229376_241747.jpg
SHA256: 59e0ec78f30c50db44d24a413ca1cccbd7ef5910cad4d3cf0e4753095725ec94

Found JPG starting offset 34897920 End offset 34920237
carving it to 34897920_34920237.jpg
SHA256: bde9e54f4e1ec3b6ab8d439aa64eef33216880685f8a4621100533397d114bf9
```

Figure 14: Program Output

Using Active@Disk Editor we viewed the names and previewed files contained on the disk image to verify our results (Figure 15). The number of files, the size of the files, and previewing certain ones verified that the program was returning the correct results.

Name	Size	Date created
Bear.avi	27.6 MB	11/15/20 3:42 AM
Dice.png	223 KB	11/15/20 3:42 AM
Iron.jpg	21.8 KB	11/15/20 3:42 AM
Flags.jpg	90.4 KB	11/15/20 3:42 AM
Great.pdf	3.05 MB	11/15/20 3:42 AM
Ocean.avi	9.45 MB	11/15/20 3:42 AM
Auburn.JPG	12.1 KB	11/15/20 3:42 AM
Cities.pdf	1.97 MB	11/15/20 3:42 AM
Flower.bmp	76.1 KB	11/15/20 3:42 AM
Minion.gif	320 KB	11/15/20 3:42 AM
Universe.mpg	1.95 MB	11/15/20 3:42 AM
Question.docx	133 KB	11/15/20 3:42 AM
Mandelbrot.gif	2.53 MB	11/15/20 3:42 AM

Figure 15: Active@Disk Editor Files

Finally, after completing the program we concluded that using file signatures is a fast and efficient method for file recovery. The program successfully recovered the files on the disk image. In the end we had to make a slight modification to the PNG module as the DOCX file had an embedded PNG file within it. Similar to the BMP module, we filtered out the PNG that did not begin at the start of a sector. Future improvements could be made to make the program more efficient such as reading the image only once rather than multiple times, however, the performance of this program is still significantly faster than manually recovery of each file. We also created individual Python files for each type of file to be recovered in order to increase code reusability and readability.

5 References

[1] Sciencedirect.com. [accessed 2021 Nov 30]. https://www.sciencedirect.com/book/9781597494724/malware-forensics-field-guide-for-windows-systems.
[2] Garykessler.net. [accessed 2021 Nov 30]. https://www.garykessler.net/library/file_sigs.html.
[3] Filesignatures.net. [accessed 2021 Nov 30]. https://filesignatures.net/index.php?search=avi&mode=EXT.
[4] File-recovery.com. [accessed 2021 Nov 30]. https://www.file-recovery.com/recovery.html.
[5] Thehexninja.com. [accessed 2021 Nov 30]. https://www.thehexninja.com/2018/01/practical-exercise-image-carving-ii.html.
[6] Github.com. [accessed 2021 Nov 30]. https://github.com/budrus123/disk-image-file-recovery/blob/main/main.py.