A close up of a sign

Description automatically generated

Digital Forensics Project 2

Uday Kiran and Carson Melton

11/3/23

# Executive Summary

This assignment is centered on the use of file headers and footers to identify, define, and recover a variety of file types from a given disk image. This disk image is passed to a python file as a parameter, and then this disk image is parsed to recover appropriate files and generate the SHA256 hash to ensure integrity. The following file types are accounted for in the assignment: MPG, PDF, BMP, GIF, JPG, DOCX, AVI, PNG, and ZIP.

# Collaboration Summary

This project included a coding portion, this written report, and a video presentation. The coding portion was worked on collaboratively between both members, with most of the debugging and refactoring done by Uday Kiran. The content included in the written report and the video presentation was created by Carson Melton, with Uday Kiran aiding in proofreading and creation of images.

# Table of Contents

Table of Contents

[Abstract 2](#_Toc485290869)

[Table of Contents 3](#_Toc485290870)

[List of Figures 5](#_Toc485290871)

[List of Tables 6](#_Toc485290872)

[1 Introduction 7](#_Toc485290873)

[2 Background 7](#_Toc485290874)

[2.1 Subheading 1 7](#_Toc485290875)

[2.2 Subheading 2 7](#_Toc485290876)

[2.2.1 First Sub-subheading 7](#_Toc485290877)

[2.2.2 Second Sub-subheading 7](#_Toc485290878)

[3 Methodology 8](#_Toc485290879)

[4 Results and discussion 8](#_Toc485290880)

[5 Conclusions and recommendations 8](#_Toc485290881)

[6 Acknowledgements 8](#_Toc485290882)

[7 References 8](#_Toc485290883)

[Appendix A: Place the title of appendix here 9](#_Toc485290884)

# List of Figures

[Figure 1 - File Header and Footer Dictionaries 8](#_Toc149735447)

[Figure 2 - Main Method 9](#_Toc149735448)

[Figure 3 - File Recovery Loop 9](#_Toc149735449)

[Figure 4 - File Size Calculation 11](#_Toc149735450)

[Figure 5 - File Extraction 12](#_Toc149735451)

[Figure 6 - File Size Calculation without File Footer 13](#_Toc149735452)

[Figure 7 - SHA256 Hash Generation 14](#_Toc149735453)

.

# 1 Introduction

This project is meant to reinforce our understanding of the storage of files within a disk image. This project immediately follows a previous project which required file recovery to be done manually without the use of any tools. In this iteration, file headers and footers were allowed for a quicker identification of files. The ‘dd’ command is still being used for actual file recovery.

# 2 Background

The project began with a list of file types that are present on the disk images. These file types are listed previously in the executive summary. The actual value of these headers and footers used within the python code were determined using examples from class lectures and Gary Kessler’s glossary of file types.

## 2.1 File Type Identification

The first requirement for completing this project was to compile dictionaries in python containing all the appropriate headers and footers. As mentioned previously, much of this data was collected from Gary Kessler’s glossary of file headers.

A screenshot of a computer program

Description automatically generated

Figure - File Header and Footer Dictionaries

The headers and footers are represented as hexadecimal values of varying lengths.

The following step was to input the disk image and put it into a usable format. This was done using the ‘read()’ and ‘hex()’ methods existing in python. This step was the first executed when the code is run, and is located within the main method of the file.

A computer code on a black background

Description automatically generated

Figure - Main Method

Once the disk data has been read, the method ‘FileRecovery()’ is called. This contains all the code responsible for locating and recovering all necessary files.

A screen shot of a computer code

Description automatically generated

Figure - File Recovery Loop

The variable ‘total\_found’ is initialized at this stage and is used throughout this code for the naming convention of the recovered files. A for loop is then used to iterate through all relevant file headers, and the ‘find()’ method is called to determine the locations of the current file header in the disk image. Note that GIF has two unique file headers so there is an additional dictionary definition. A while loop then begins to ensure that all locations of a file header are checked before moving to the next header in the dictionary.

## 2.2 File Recovery

Once a valid file header has been located, the code moves into the recovery stage. This portion uses various methods to ensure that the entirety of the file is located before a ‘dd’ command is executed. For most files, the file footer is used to identify the end of the file. This logic applies to file types MPG, PDF, GIF, JPG, DOCX, PNG, and ZIP. Additional zeroes were added to several file footers to eliminate any false matches while traversing.

A screenshot of a computer program

Description automatically generated

Figure - File Size Calculation

An initial check of ‘loc % 512’ is done to check that the location of the file header is at the beginning of a sector. Following this, the file footer is then searched for to locate the end of the file. In the above image, multiple file footers are checked due to PDF’s attribute of having multiple file footers. Because find() returns the location of where the file footer starts, an offset variable is calculated so that the file footer will be included upon extraction. This offset is calculated by the length of the file footer. Following this, the begin and end of the file are calculated for extraction. These values are divided by two as a way of converting from hexadecimal to bytes. The ‘ceil()’ method ensures the final footer byte is included in the extraction. Once these locations have been calculated. They are printed to the console, and then they are passed to ‘File\_Extract()’. If a file header is not at the beginning of a sector, the header is skipped and the search is adjusted to move past the found file header, as shown by ‘search = loc + 8’.

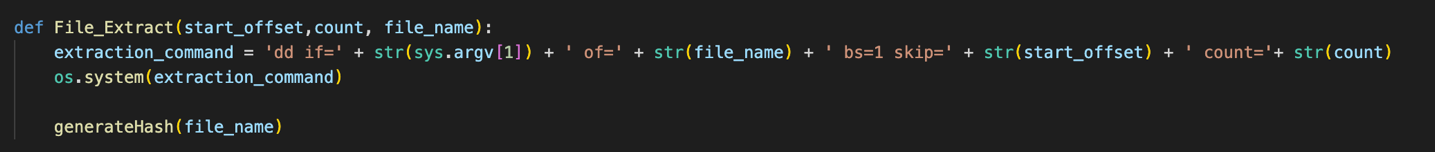


Figure - File Extraction

This method contains the linux ‘dd’ command responsible for the file extraction. This also includes a call to ‘generateHash()’, which is addressed in the following section.

# 2.2.1 Special Cases

File types BMP and AVI have no defined file footers, so the length of the file must be calculated through other means. For these specific file types, the length of the file can be recovered from within the file header.

A screen shot of a computer program

Description automatically generated

Figure - File Size Calculation without File Footer

The logic of these file types remains the same, but the code portion responsible for locating the footer has been replaced with the recovery of the file size. For the BMP example, the 4th hex value to the 16th hex value after the start of the header is the file size. This is recovered in reverse order to account for the little endian standard. The ‘file\_size’ value is then converted to decimal so it can be used in the file recovery process. This same logic applies to the AVI file type.

## 2.3 SHA256 Generation

The Final step in file recovery is the generation of the SHA256 hash to ensure integrity.

A black background with text

Description automatically generated

Figure - SHA256 Hash Generation

This method is called within File\_Extract() and simply executes a sha256sum call on the recovered file.

# 3 Methodology

The methodology used when completing this project was a logical step-by-step approach to the problem. Upon the collection of the disk data, the most important step was the identification of appropriate files within the data using the file headers. Then the appropriate file footers or header information was located to determine the size of the file. Finally the file was extracted and the hash was calculated to confirm integrity.

# 4 Results and discussion

The results from the execution of the python file on the provided disk image resulted in the discovery of 13 files. These files included all the previously mentioned file types aside from ZIP, which was not recovered within the image.

# 5 Conclusions and recommendations

Upon the execution of the python file, the conclusion our team reached is that the use of file headers results in a much more efficient recovery of appropriate files from a disk image. By using file headers, we were able to locate and retrieve files without previously accessing other portions of the disk image such as the root directory or FAT table. This process eliminates several possibilities of human error.

# 6 Acknowledgements

We would like to acknowledge Dr. Farah Kandah for his course material and his lectures, which provided all the necessary information for understanding the purpose and importance of file headers. We would also like to thank Gary Kessler for his incredibly helpful glossary of file headers and their associated footers.

# 7 References

[1] Kessler G. 2019. File Signatures. Garykesslernet. https://www.garykessler.net/library/file\_sigs.html.