OPEN ACCESS
Indo-JC Indonesia Journal of Computing
ISSN 2460-9056
socj.telkomuniversity.ac.id/indojc

Ind. Journal on Computing
Vol. 6, Issue. 1, April 2021. pp. 13-22
doi:10.34818/indojc.2021.6.1.457

Analysis and Implementation of Signature Based Method and Structure File Based Method for File Carving

Anjar Afrizal [#1], Niken Dwi Wahyu Cahyani [*2], Erwid Jadied [#3]

 # Fakultas Informatika, Universitas Telkom
Jl. Telkomunikasi No.1 Terusan Buahbatu Bandung Indonesia

[1] anjarafrizal@student.telkomuniversity.ac.id
[2] nikencahyani@telkomuniversity.ac.id
[3] jadied@telkomuniversity.ac.id

Abstract

File Carving is a data recovery technique based on file structure and content without relying on filesystem information or metadata. The problem in carving files is its high false positive value especially when the file is fragmented (either linear fragmented or non-linear fragmented). The aim of this study is to implement and analyze the performance of two file carving method (Signature Based and File Structure Based) as a solution to the problem of the carving process. By focusing on JPEG, GIF and PNG files, two datasets are used, namely: CFReDS Project (NIST Project) and Basic Data Carving Test (Nick Mikus Project). The analysis is based on the recovery performance (carving recall, supported recall, carving precision), execution time, and memory usage. From the recovery performance parameter, the File Structure Based method gets a higher overall value than the Signature Based method. However, based on the execution time performance parameter, the Signature Based method has better execution time and use fewer resources compared to the File Structure Based method.

Keywords: file carving, signature based method, file structure based method, recovery performance, execution time, memory usage

Abstrak

File carving adalah teknik pemulihan data berdasarkan struktur file dan konten tanpa bergantung pada informasi sistem file atau metadata. Masalah dalam file carving adalah nilai false positive yang tinggi terutama ketika file terfragmentasi (baik terfragmentasi linier maupun terfragmentasi non-linier). Tujuan dari penelitian ini adalah untuk mengimplementasikan dan menganalisis performansi dari metode file carving (Signature Based dan File Structure Based) sebagai solusi dari permasalahan proses carving. Penelitian ini berfokus pada file JPEG, GIF dan PNG. Digunakan dua buah dataset yaitu: CFReDS Project (NIST Project) dan Basic Data Carving Test (Nick Mikus Project). Analisis kinerja didasarkan pada recovery performance (carving recall, supported recall, carving precision), waktu eksekusi, dan penggunaan memori. Berdasarkan parameter recovery performance, metode File Structure Based mendapatkan nilai keseluruhan yang lebih tinggi daripada metode Signature Based. Namun berdasarkan parameter performansi waktu eksekusi, metode Signature Based memiliki waktu eksekusi yang lebih baik dan menggunakan resource yang lebih sedikit dibandingkan dengan metode File Structure Based.

Kata Kunci: file carving, signature based method, file structure based method, recovery performance, execution time, memory usage

Received on January, 2021. Accepted on Mei, 2021

I. Introduction

In the process of digital storage, data can be damaged, lost, and hidden due to human error, errors of digital devices, natural causes, criminal activities, and sabotage. The data recovery process is very much needed to restore damaged or lost data. Previous data recovery techniques used information from the file system to perform recovery, but this technique cannot be used if there is damage to the file system information section. File carving is a digital forensic technique used to recover files from digital storage based on its file structure and content in the absence of filesystem and any metadata information [1], [2].

Three common methods for carving files are Signature Based, File Structure Based and Block Content Based [3]. The Signature Based is a file carving method that identifies the header pattern string as the beginning of the file and the footer pattern string as the end of the file, and considers data between the header and footer string patterns as the contents of the file. The File Structure Based method is the development of the Signature Based method, which in addition to identifying the header and footer of a file, it checks the internal structure of files.

One problem in the carving process is when the file is fragmented (either linear or non-linear fragmentation) i.e. scattered throughout a media instead of one continuous location. Another problem is the high false positive value. To overcome this kind of fragmented file, we create an application and develop 2 workflows namely: Signature Based Workflow and File Structure Based Workflow by combining various techniques in file carving. We then compare its performance.

II. Literature Review

A. Previous Studies

Alshammary [4] describes the currently available carving method for JPEG image files. Also, this paper introduces a hybrid method for handling fragmentation problems. Deris et al. [5] discusses different cases of fragmentation in JPEG files. To make it easier to identify JPEG / JFIF files, a new algorithm is introduced to detect the JPEG / JFIF file header format, which is a dual byte marker. This is because JPEG files can be easily recovered using metadata (headers, footers, and markers). This paper claims that a dual byte marker is better than a single byte marker. JPEG files can be saved in the form of original files, thumbnails, or embed files. Therefore, Abdullah et al. [6] discusses ways to discover thumbnails and JPEG files inserted in another file. The method used in this paper is the Unique Hex Pattern (UHP). Unique Hex Pattern uses unique combinations of markers in JPEG files for recognizes embedded thumbnails and files. For the problem of fragmentation, Mohamad & Deris [7] proposes fragmentation detection point uses markers based on Define Huffman Table Area (DHT) and metadata to recognize all possible fragmentation cases. In this paper, we use byte marker and unique hex pattern.

B. Fragmentation

Fragmentation in a file is normal. When files are added, modified, or deleted by the user / operating system, files can be fragmented. A file is fragmented if it is not stored in the correct order or are not stored sequentially on the disk. Fragmented files can be divided into two categories, namely non-linear fragmentation and linear fragmentation.

Figure 1 (a) is an illustration of 2 non-fragmented files stored on disk. The Blue file consists of 4 blocks and the Green file consists of 2 blocks. Figure 1 (b) is an illustration of non-linear fragmentation in which the files are arranged in a disorderly manner (blocks 3, 4, 2).

There are 6 cases for linear fragmentation [4] (we use JPEG file as example and some cases are illustrated in figure 1 (c)):

- JPEG file intertwined with the non-JPEG
- JPEG file intertwined with JPEG file
- JPEG file fragmented with a gap between fragments
- JPEG file fragmented with containing a missing fragment
- JPEG file fragmented into Multi-fragments
- JPEG file fragmented with a missing header

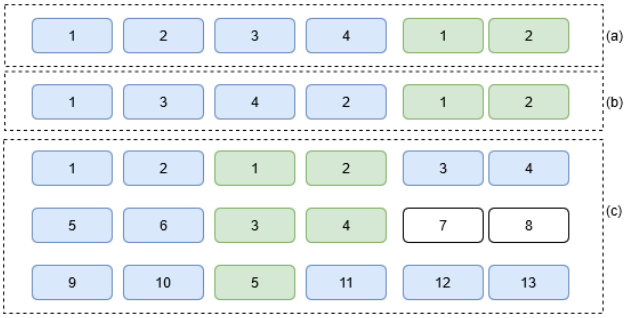

Fig. 1 Illustrations of fragmented files

III. RESEARCH METHOD

A. Signature Based Workflow

The way the Signature Based method works is to look for the header string pattern as the start of the file and the footer string as the end from the file, then all the data blocks between the header and footer are saved. Signature Based Workflow (Fig. 2) is loosely adapted from Scalpel framework [8], [9].

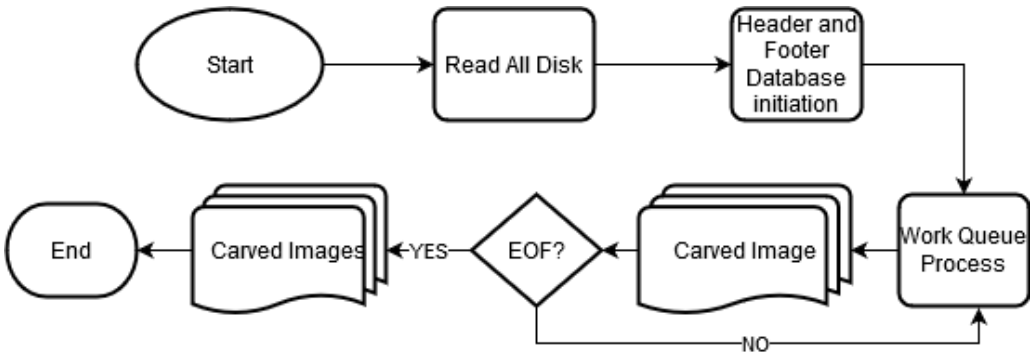

Fig. 2 Signature Based Workflow

There are two important stages in the workflow to carry out the carving process. The first step is to create a database to store a) header and footer file string patterns b) address information from each header and footer found. The second stage is to do the work queue process. The program will start the carving process

automatically based on information from the header and footer database and then continue to scan until the end of the disk.

B. File Structure Based Workflow

File Structure Based works by identifying the header and footer string pattern of the target file same as the Signature Based method plus an analysis of the internal file structure. File Structure Based Workflow (Fig. 3) is developed using myKarve framework [10].

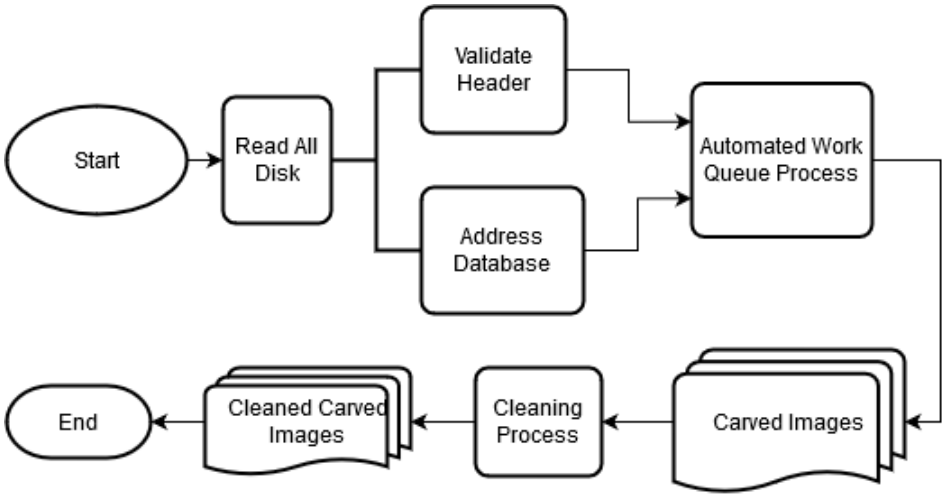

Fig. 3 File Structure Workflow

The stages of the carving process are as follows:

1) Read all the data on the disk to be carved

2) Validated Header saves the file header string pattern that has been validated by referring to the file format based on the standard (This step is used to analyse the internal structure of the file). Address Database is identifying the pattern of the header string, footer, and additional markers and stored on database component (same as Signature Based method).

3) Automated Work Queue Automatically carving process based on data from the database and header validation.

4) During the carving process, various types of files will be found (including junk files due to fragmentation). Then the next step is garbage elimination, which is checking and deleting unusable files.

C. Dataset

Two datasets were used in this study: Database Carving Test # 1 (Nick Mikus Project) [11] and File Carving Dataset (CFReDS Project) [12]. Basic Data Carving Test # 1 can be accessed at http://dftt.sourceforge.net/test11/index.html. This dataset has a total of 15 files with various file types including 4 jpeg files and 1 gif file. There is no fragmentation in this dataset.

The second dataset is CFReDS Project (https://www.cfreds.nist.gov/FileCarving/index.html) with 6 file types: JPEG, PNG, BMP, GIF, TIF, PCX. There are 6 cases in this dataset. The characteristics of each case are as follows:

TABLE I
CHARACTERISTIC OF EACH DATASET IN CFREDS

No	*Dataset*	*Characteristics*	*Total file*
1	*L0 Graphic.dd*	*Non-fragmented graphics files*	*6*
2	*L1 Graphic.dd*	*Sequentially fragmented graphics files*	*6*
3	*L2 Graphic.dd*	*Non-sequentially fragmented graphics files*	*6*
4	*L3 Graphic.dd*	*Graphics files with missing fragments*	*6*
5	*L4 Graphic.dd*	*Graphics files nested within graphics files*	*7*
6	*L5 Graphic.dd*	*Braided graphics files*	*6*

D. Performance Parameters

Three performance parameters are used: recovery performance, execution time and memory usage. For recovery performance we use [13]:

$$carving\ recall\ (CR) = \frac{all - sfn - ufn}{all} \quad (1)$$

$$supported\ recall\ (SR) = \frac{sp - sfn}{sp} \quad (2)$$

$$carving\ precision\ (CP) = \frac{tp}{tp + ufp + \frac{1}{n}kfp} \quad (3)$$

all = refers to all files in the dataset (supported and unsupported file)

sfn = supported false negative

ufn = unsupported false negative

sp = supported file, total file supported by a specific carving tool

tp = true positive, files that correctly carved from the dataset

ufp = unsupported false positive, files not identified as incorrect by a tool

kfp = known true positive, files identified by a tool as incorrect or corrupt

The result of equation (1), (2) and (3) is in the range between 0 (low) and 1 (high). Carving recall is a tool's ability to extract correct file from the dataset (higher is better). Support recall is the same as carving recall but for the supported file type (higher is better). Carving precision is a measure of correctness of the tool (low score means high false positive; a high score means low false positive).

Execution time refers to the time it takes the program to carve the test data provided. Execution Time is measured when the program starts and stops when everything is finished in second. The time value is initialized by calling the time.time() function using the python library. After the program has finished carving, the program will call time.time() function to get the finish time. Execution time is the difference between the initialization time and the finish time. Execution time testing is done 10 times for every case.

Memory Usage refers to the memory used when running the file carving algorithm. In measuring memory usage, we use peak memory [14]. Memory usage testing is done 10 times for every scenario.

E. Hardware and Software Specification

To build and test the system we use Ubuntu 18.0, python 2.7, python library Pillow 3.0 (for decode/display) and Tkinter 3.0. (for UI). All the above software runs on Intel i3 2.4GHz with 8GB of DDR3 RAM and NVidia Geforce 610m 2GB

IV. Results And Discussion

A. Recovery Performance

In this test, we compare the results of the carving process with the actual dataset visually and using md5 hash. Then we use carving recall (CR), supported recall (SR) and carving precision (CP) to quantify it.

First, for the L0 and L1 dataset (non-fragmented and sequentially fragmented each 1 file GIF, 1 file PNG, 1 file JPG), Signature Based able to recover PNG and GIF but not perfectly as shown in Fig. 4. File structure based able to recover all three of them perfectly (see Fig. 5)

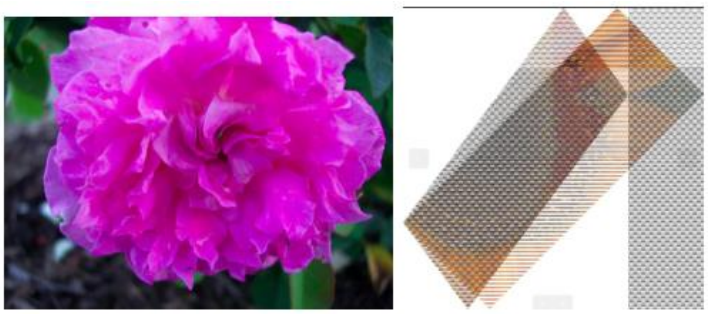

Fig. 4 Recovery of L0 and L1 dataset using Signature Based (PNG is fully recovered-left, GIF is partially recovered-right)

Fig. 5 Recovery of L0 and L1 dataset using File Structure Based (all 3 files are fully recovered)

Second, for the L2 dataset (non-sequentially fragmented with 1 file GIF, 1 file PNG, 1 file JPG), Signature Based only able to recover GIF image partially (Fig. 6 – left). But File Structure Based able to recover JPG partially (Fig. 6 – centre) and recover PNG fully (Fig. 6 – right).

Fig. 6 Recovery from L2 dataset

Third, for the L3 dataset (graphics files with missing fragments), only File Structure Based that able to recover it but only 1 file JPG and partially as shown in Fig. 7.

Fig. 7 Partially recovered file using File Structure Based

Fourth, for the L4 dataset (Graphics files nested within graphics files). Signature Based able to recover it partially and file structure able to recover it all fully. Fifth, for the L5 dataset (Braided graphics files) only File Structure Based able to recover it fully. Using this information, we calculate CR, SR, CP and create table II.

From table II, the Signature Based method is only capable carving files that are non-fragmented as in (row 1.1. and 2.1) or fragmented in sequence (2.2) i.e. not zero scores in CR, SR or CP.

On the other hand, the File Structure Based method is capable of carving files in non-fragmented, fragmented sequentially, or there are image files that are inserted in other image files, as shown in the L4 Graphic.dd dataset. But, for non-sequential fragmentation and missing fragments, it has not been able to do the carving perfectly (there are still residual images remaining from another file).

TABLE II
COMPARISON OF RECOVERY PERFORMANCE BETWEEN TWO METHODS

No	Dataset	Signature Based			File Structure based		
		CR	SR	CP	CR	SR	CP
1	Basic data carving test						
1.1	11-carve-fat.dd	0.133	0.4	0.571	0.333	1	1
2	CFReDS Project						
2.1	L0 Graphic.dd	0.166	0.33	0.5	0.5	1	1
2.2	L1 Graphic.dd	0.166	0.33	0.5	0.5	1	1
2.3	L2 Graphic.dd	0	0	0	0.166	0.333	0.5
2.4	L3 Graphic.dd	0	0	0	0.166	0.333	0.5
2.5	L4 Graphic.dd	0.166	0.333	0.5	0.574	1	1
2.6	L5 Graphic.dd	0	0	0	0.166	0.333	0.5

A. Execution Time Performance

The Signature Based method has a better execution time (almost twice faster) than the File Structure Based method for both datasets. The Signature Based method is faster because the carving process only requires searching for a header and footer string pattern. While the File Structure Based looks for a header and footer string pattern, it also validates the internal structure of files such as additional markers of each type the file.

TABLE III
COMPARISON OF EXECUTION TIME BETWEEN TWO METHODS

No	Dataset	Signature Based Time (s)	File Structure based Time (s)
1	**Basic data carving test**		
1.1	11-carve-fat.dd	0.83	1.64
2	**CFReDS Project**		
2.1	L0 Graphic.dd	0.90	1.71
2.2	L1 Graphic.dd	0.86	1.76
2.3	L2 Graphic.dd	0.74	1.96
2.4	L3 Graphic.dd	0.53	1.19
2.5	L4 Graphic.dd	0.72	1.50
2.6	L5 Graphic.dd	0.75	1.35
	Average	**0.75**	**1.57**

B. Memory Usage Performance

Memory usage performance is to determine the amount of resources used by both methods to carry out the carving process, especially memory usage. Based on [14], memory usage is obtained by calculating the **average peak value** of memory usage during the process carving.

Peak memory is the highest value of memory usage during the carving process. The peak value for memory is obtained by calling the "psutil" python library (memory info function). When the program runs the carving process, the program initializes the initial memory value needed by the program by calling the memory info function based on the program pid number. When the carving process is complete, then the peak memory value will be stored in the variable. Then we average its peak value.

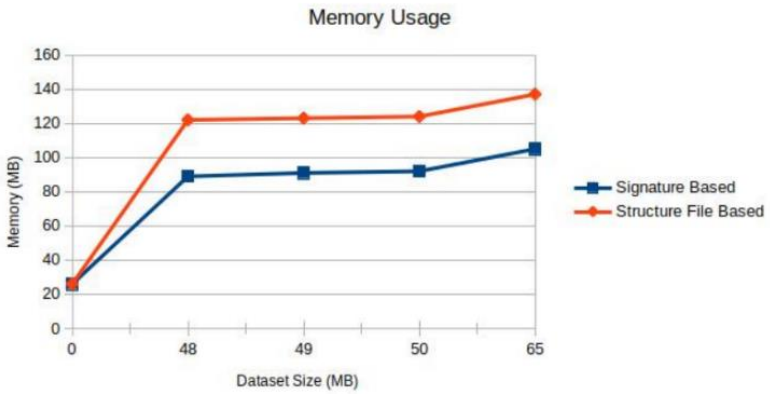

Fig. 8 Memory usage comparison between two methods

Based on Fig. 8, the Signature Based method uses lower memory resources compared to the File Structure Based method. This is due to the stage's differences in the carving process of the two methods, wherein the carving process steps in Signature Based is simpler than the File Structure Based method. In addition, the larger the size the dataset is carved out, the larger the memory size used because this program has not used other advanced methods such as caching or mapping.

V. CONCLUSION

The Signature Based method can only be used for non-fragmented file, while the File Structure Based can be used for non-fragmented file, sequentially fragmented file, and file inserted in another image file. Positive false problems that occur can be resolved by using the file structure method based due to the validation process of the internal file structure and the pattern matching process, so that it can identify and discard the string footer pattern that is not appropriate. Execution time and memory usage test show the Signature Based method has faster time and less memory usage compared to the File Structure Based method. It will carry out the carving process based only on the header and footer string pattern without checking the detailed internal layout file.

As for future research, the content-based method can be added and studied further to increase its accuracy and ability to handle the fragmented file with nonsequential or random fragmentation

References

[1] A. Dewald, M. Luft and J. Suleder, "Incident Analysis and Forensics in Docker Environments.," *ERNW WHITE PAPER 64*, 2018.

[2] R. Ali, K. Mohamad, S. Jamel and S. Khalid, "A review of digital forensics methods for JPEG file carving," *Journal of Theoretical and Applied Information Technology*, vol. 96 , no. 17, p. 5841, 2018.

[3] N. Alherbawi, Z. Shukur and R. Sulaiman, "A survey on data carving in digital forensic," *Asian Journal of Information Technology*, vol. 15, no. (24), pp. 5137-5144, 2016.

[4] E. Alshammary and A. Hadi, "Reviewing and evaluating existing file carving techniques for jpeg files," in *Cybersecurity and Cyberforensics Conference (CCC)*, 2016.

[5] K. Mohamad, T. Herawan and M. Deris, "Dual-byte-marker algorithm for detecting JFIF header," in *International Conference on Information Security and Assurance*, Springer, Berlin, Heidelberg, 2010.

[6] N. Abdullah, R. Ibrahim and K. Mohamad, "Carving thumbnail/s and embedded JPEG files using image pattern matching," *Journal of Software Engineering and Applications* , vol. 6 (3B), p. 62, 2013.

[7] K. Mohamad and M. Deris, "Fragmentation point detection of JPEG images at DHT using validator.," in *International Conference on Future Generation Information Technology*, Springer, Berlin, Heidelberg, 2009.

[8] G. Richard and V. Roussev, "Scalpel: A Frugal, High Performance File Carver," in *The Digital Forensic Research Conference*, New Orleans, LA, 2005.

[9] X. Zha and S. Sahni, "Fast in-place file carving for digital forensics," in *International Conference on Forensics in Telecommunications, Information, and Multimedia* , Springer, Berlin, Heidelberg, 2010.

[10] K. Mohamad, A. Patel, T. Herawan and M. Deris, "myKarve: JPEG image and thumbnail carver," *Journal of Digital Forensic Practice*, vol. 3, no. (2-4), pp. 74-97, 2010.

[11] N. Mikus, "Basic Data Carving Test #1," Source Forge, 14 March 2005. [Online]. Available: http://dftt.sourceforge.net/test11/index.html. [Accessed June 2020].

[12] NIST, "Forensic Images for File Carving," NIST, 19 October 2019. [Online]. Available: https://www.cfreds.nist.gov/FileCarving/index.html. [Accessed June 2020].

[13] T. Laurenson, " Performance analysis of file carving tools," in *IFIP International Information Security Conference* , Springer, Berlin, Heidelberg, 2013.

[14] J. De Bock and P. De Smet, "JPGcarve: an advanced tool for automated recovery of fragmented JPEG files," *IEEE transactions on Information Forensics and Security*, vol. 11, no. (1), pp. 19-34, 2015.