View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by UTHM Institutional Repository

Fragmentation Point Detection of JPEG Images at DHT Using Validator

Kamaruddin Malik Mohamad, Mustafa Mat Deris

Faculty of Information Technology and Multimedia,
Universiti Tun Hussein Onn Malaysia (UTHM),
86400 Parit Raja, Batu Pahat, Johor, Malaysia.
{malik, mmustafa}@uthm.edu.my

Abstract. File carving is an important, practical technique for data recovery in digital forensics investigation and is particularly useful when filesystem metadata is unavailable or damaged. The research on reassembly of JPEG files with RST markers, fragmented within the scan area have been done before. However, fragmentation within Define Huffman Table (DHT) segment is yet to be resolved. This paper analyzes the fragmentation within the DHT area and list out all the fragmentation possibilities. Two main contributions are made in this paper. Firstly, three fragmentation points within DHT area are listed. Secondly, few novel validators are proposed to detect these fragmentations. The result obtained from tests done on manually fragmented JPEG files, showed that all three fragmentation points within DHT are successfully detected using validators.

Keywords: File Carving, Digital Evidence, Digital Forensics, Data Recovery

1 Introduction

Forensics or digital investigations can be lengthy for storage with GB or TB, thus need rapid turnaround for time-sensitive cases involving potential loss of life or property. File carving tools typically produce many false positives and could miss key evidence [1]. Digital Forensics Research Workshop (DFRWS) 2007 carving challenge has boosted the pace for file carving research aimed to improve the state of the art in fully or semi-automated carving techniques [2]. [3] determined that fragmentation on a typical disk is less than 10%, however the fragmentation level of forensically important file types (like images, office files and email) is relatively high. He found that 16% of JPEGs are fragmented. As files are added, modified, and deleted, most file systems get fragmented [4]. However, in digital forensic, reassembling of fragmented documents has received little attention [5].

Most file carvers identify specific types of file headers and/or footers. Unfortunately, not all file types have a standard footer signature, so determining the end can be difficult [1]. In-Place Carving is another approach which allows inspection of recovered files without actually copying the contents [6]. This results in significantly reduction in storage requirements.

Entropy, compressibility and ASCII proportionality metrics can be used for file type identification especially for files that do not have magic numbers or files that are incomplete (e.g. due to deletion and overwritten) [7, 8]. N-gram analysis can also be used for file type identification [9]. File type identification can be used to detect file masquerade as another type of file e.g. by changing the 'jpg' file extension to 'txt'. Nevertheless, these metrics are not suitable for detecting fragmentation point. In the DFRWS 2007 forensics challenge, data in the challenge involved fragmented files where fragments were sequential, out of order, or missing. None of the submissions to the forensic challenge of year 2007 completely solve the problems presented [2].

Baseline JPEG (simply called JPEG from this point onwards) file is used in the experiments because it is widely used in the Internet, and in many applications [9, 10]. Nevertheless, the introduced algorithms can be altered to extend the fragmentation point detection experiment to other JPEG file types (e.g. progressive, lossless and hierarchical JPEG) [11, 12]. It is important to detect fragmentation within the DHT because corrupted DHT would cause image distortion or corruption.

This paper introduces several scenarios (refer to Table 2 and Figure 3) of possible fragmentation points within DHT segment. Validators have been developed to detect these fragmentation points on JPEG files that are manually inserted with dummy data to simulate the three fragmentation scenarios.

The rest of the paper is organized as follows. Section 2 describes related work, section 3 discussed about fragmentation scenarios and validators, section 4 discussed about the experiments done, section 5 discussed about the result and discussion and finally section 6 concludes this paper.

2 Related Works

File carving processes have been defined by [3], [13] and [14]. Bifragment Gap Carving (BGC) [3] is introduced for bifragmented JPEG file recovery. The recovery is done by exhaustively searching all combinations of blocks between an identified header and footer while excluding different number of blocks until a successful decoding/validation is possible [13]. In [13], the author uses sequential hypothesis testing on data block to determine if consecutive clusters should be merged together. He uses forward fragment point detection test, and reverse sequential test only if the forward test is inconclusive. [14] focuses on solving fragmentation of JPEG image file containing RST markers that cuts through scan segment. A scan segment or area is an entropy-coded data segment which starts with start-of-scan (SOS) marker [11]. [3] and [13] are handling a general JPEG fragmentation case, while [14] is specifically focusing on fragmentation in JPEG scan area. On the other hand, this paper focuses on fragmentation in JPEG DHT segment. Thus, all are handling different type of fragmentation cases. It is important to solve fragmentation that occurs within DHT because; DHT corruption would cause image distortion or corruption during encoding process. Nevertheless, this paper will only discussed on the fragmentation detection within DHT using validators in detail but leave the fixing of the problem into our future work.

3 Fragmentation

It is important to know how many DHT tables are there in the baseline JPEG file and how they are stored in these files. Baseline JPEG file can be identified by searching for start-of-frame (SOF) marker (0xFFC0) using any available hex editor (e.g. HexAssistant [15], BinText [16]). According to [11], there are four possible DHTs in baseline JPEG, but does not clearly stated on how are these DHTs are stored in the baseline JPEG file. To test this, 100 JPEG files are downloaded from the Internet (Google images) and renamed (e.g. 01.jpg). A sample of 100 JPEG files is more than enough because all baseline JPEG files have similar structure. For this reason, only a sample of five files is shown in Table 1. From the result obtained, it shows that all these files contain 4 DHTs, namely 2 DHT AC tables and 2 DHT DC tables with DHT table index 0x00, 0x10, 0x01 and 0x11. Each DHT marker is followed by one DHT table. The sequence of these DHT segments is illustrated as in Figure 1.

The offsets of 4 DHT tables and SOS in each JPEG files are tabulated as in Table 1. All the downloaded are baseline JPEG File Interchange Format (JFIF) because this type of file is the de facto standard for Internet [12]. All these files found to be using 4 different DHT tables in each file. Nevertheless, [14] found that, JPEG Exchangeable Image File Format (Exif) images taken from 76 popular digital cameras, 69 (91%) of them are using the same DHT tables.

3.1 Fragmentation Scenarios

In order to come up with DHT fragmentation detection algorithm (validator), first, we need to know all the JPEG file fragmentation possibilities or scenarios within the DHT segment. A standard structure of DHT is illustrated in Figure 2. A list of possible fragmentation scenarios is shown in Table 2. A visual representation of fragmentation scenarios is illustrated in Figure 3. Few novel validators are developed for detecting these fragmentations.

3.2 Validator for scenario 1

After listing the possibility of fragmentation in DHT, so next, we need to know how to validate each of those scenarios. For scenario 1, the valid DHT length value for baseline JPEG must be greater than 19 because the first 2 bytes constitutes the DHT length field, followed by a byte of DC/AC table index and the next 16 bytes represent 16-bit Huffman codes (refer to Figure 2). The variable length data is not included here. If we assume that there is a minimum data of one byte in the variable length data, the size should be 20 bytes. Thus, the DHT length value of less than 20 would be detected by the validator as error (or fragmentation is detected for scenario 1 or called as DHT-FragType-1 (refer to Table 3)). The DHT marker for baseline JPEG normally appears 4 times (refer Figure 2), one for each table index (0x00 [DC table], 0x10 [AC table], 0x01 [DC table], 0x11 [AC table]). The algorithm for fragment point detection for scenario 1 (DHT-FragType-1) is illustrated in below.

Example of algorithm for scenario 1.

```
if DHT marker is found
    get 2-byte DHT length
    if (first byte of DHT length = 0x0)  and
       (second byte of DHT length < 0x14)
                Error : Fragmentation Point Detected
                (DHT-FragType-1)
    endif
endif
```

3.3 Validator for scenario 2

For scenario 2, the table index value is stored in a single byte (8 bits). There are two components in a single byte of table index. The first four bits represent class component (have valid values of 0 for DC table or 1 for AC table). The next four bits represents the "Table id" (with a value 0 or 1). So, there are only four valid values for the one-byte table index i.e. 2 DC tables of values 0x00, 0x01 and another 2 AC tables of values 0x10, 0x11. If the validator found a value other than these four, error or fragmentation point (DHT-FragType-2) is detected at this location in DHT segment. The algorithm of validator for scenario 2 is illustrated below.

Example of algorithm for scenario 2.

```
get DHT structure
get DHT index (from DHT structure)
if (DHT_index<>0x0) and (DHT_index<>0x01) and
   (DHT_index<>10) and (DHT_index<>11))
          Error : Fragmentation Point Detected
                  (DHT-FragType-2)
endif
```

3.4 Validator for scenario 3

For scenario 3, discrepancy can be detected by checking the total values stored in all sixteen bytes of Huffman codes. Nevertheless, when this occurs, it does not exactly show the exact location where the split or fragmentation point occurs. What it tells you is that, there is just a fragmentation point somewhere within the 16 byte Huffman code. To test this scenario, big values need to be used for the dummy data for creating the test file. The validator detects error or fragmentation point only when the total values stored in the 16 byte Huffman codes exceeds 255. This is because only a maximum of 255 ASCII characters can be compressed or represented as a 1-bit to 16-bit codes (or called as 16-bit Huffman codes). The algorithm of validator for scenario 3 is illustrated below.

Example of algorithm for scenario 3.

```
get DHT structure
calculate total of 16-byte DHT Huffman bit code
if (total > 255)
    Error : Fragmentation Point Detected
            (DHT-FragType-3)
endif
```

Table 1. Offset of 4 DHT tables and SOS for 15 JPEG JFIF files downloaded from Internet.

Filename/ Markers Offset	0xFFC4 (DHT table DC index (0x00)	0xFFC4 (DHT table AC index (0x10)	0xFFC4 (DHT table DC index (0x01)	0xFFC4 (DHT table AC index (0x11)	0xFFDA (SOS)
01.jpg	177	206	281	310	366
02.jpg	177	207	276	303	341
03.jpg	177	206	269	296	341
04.jpg	177	207	270	297	347
05.jpg	177	206	268	295	334

DHT marker – DHT length (DHL)– DHT table index (DTI) DC **0x00** table – DHT 16 byte Huffman bit code (DHT16) – Variable Length Data (VLD)
DHT marker – DHL – DTI AC **0x10** table – DHT16 - VLD
DHT marker – DHL – DTI DC **0x11** table – DHT16 – VLD
DHT marker – DHL – DTI AC **0x11** table – DHT16 - VLD

Fig. 1. Sequence of DHT segments in a single baseline JPEG image file.

Table 2. Fragmentation in DHT segment scenarios.

Fragmentation Scenarios	**Description**
Scenario 1	The JPEG file is split between the DHT marker and the "length" field (DHT structure).
Scenario 2	The JPEG file is split between the "length" and "index" field.
Scenario 3	JPEG file split in the middle of DHT structure i.e. between the index field and the "16-byte Huffman bit codes".

DHT Marker (DHT)	Length	Index	16-byte Huffman bit code (HC)	Variable Data Length (VDL)

Fig. 2. DHT without fragmentation.

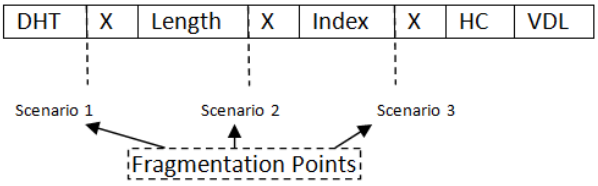

Fig. 3. Fragmentation within DHT area illustrated according to scenarios given in Table 2.

4 Experimentation

There are two experiments done to show that some fragmentations within DHT area can be detected. These are some of the assumptions that we made when doing these experiments:

- We are using baseline JPEG only. Baseline JPEG is widely used and has simple file structure.
- All the file headers and footers are in sequential order and not corrupted.
- Only a single pass is used. The header, footer and the fragmentation point will be indexed. This information can also be displayed to the user, but input output activities will increase the amount the processing time.
- The validator stops once the first fragmentation point is detected.
- Fragmentation codes (refer to Table 3) are introduced to represent the fragmentation points detected (refer to Table 1).

4.1 Experiment 1

Download 100 baseline JPEG images from the Internet. Baseline JPEG can be validated by checking the existence of start-of-frame (SOF) where the SOF0 marker must equal to 0xFFC0. Each file will be copied to three other files and renamed (ft1.jpg to ft3.jpg) as in Table 4. Each file will be fragmented with dummy data to represent each JPEG fragmentation scenarios as illustrated in Table 2 or Figure 3. The file ft1.jpg represents the JPEG file with DHT-FragType-1 or fragmentation in DHT area as in scenario 1. The same goes for other files. These fragmented JPEG files either have their images distorted or corrupted. The developed C program will be run using each of these input files separately. The sample output screen is illustrated in Figure 4 and the results obtained from one of the downloaded JPEG test files are shown in Table 4. Other test files also shown similar results but differs only in the offset address value.

4.2 Experiment 2

The hard disk space is sanitized by creating a simple text file sanitize.txt in the 8MB partition with no content (size of 0kb). Open the file using a hex editor (e.g. HexAssistant [15]) and then use the “insert block” option to insert up to maximum size of 8MB into the file with 0x00. From the window explorer, the property for the 8MB partition shows “used space” is full. Delete the sanitize.txt file and then copy three fragmented JPEG test files (ft1.jpg to ft3.jpg) to the empty 8MB HDD partition. Image of the HDD is taken using Helix Live CD and named as hdd.dd. Similar results to experiment 1 are obtained, as shown in Table 4. The experiment can be repeated with another set of three fragmented files. Similar results are obtained but differ only in the offset address value.

Table 3. List of fragmentation point codes used in the validators.

Fragmentation Code Type	Fragmentation Scenarios
DHT-FragType-1	Scenario 1
DHT-FragType-2	Scenario 2
DHT-FragType-3	Scenario 3

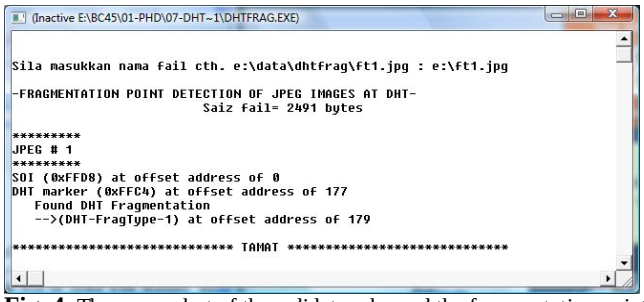

Fig. 4. The screenshot of the validator showed the fragmentation point detected for ft1.jpg.

Table 4. List of detected fragmentation type from the test files.

Filename	Fragmentation detected	Actual fragmentation	Offset Address
ft1.jpg	DHT-FragType-1	DHT-FragType-1	DHT marker at offset address 177, DHT-FragType-1 is detected at offset address 179.
ft2.jpg	DHT-FragType-2	DHT-FragType-2	DHT marker at offset address 177, DHT-FragType-2 is detected at offset address 181.
ft3.jpg	DHT-FragType-3	DHT-FragType-3	DHT marker at offset address 177, DHT-FragType-3 is detected at offset address 182.

5 Result and Discussion

From the experiments done, all these fragmentation points (scenarios 1 to 3) are successfully detected by displaying the fragmentation codes (e.g. DHT-FragType-3) and the fragmentation point addresses.

6 Conclusion and Future Works

File carving is an important, practical technique for data recovery in digital forensics investigation and is particularly useful when filesystem metadata is unavailable or damaged. One of the most popular file carvers is Scalpel. However, these file carvers still fail to merge files that are fragmented. Detection and classification of fragmentation points, made it easier for file recovery. The research on reassembly of

JPEG files with RST markers, fragmented within the scan area have been done before. Fragmentation within Define Huffman Table (DHT) segment should be given attention because it could cause image distortion or corruption during decoding process. However, fragmentation within the DHT segment is yet to be solved. This paper analyzes the fragmentation within the DHT area and list out all the fragmentation possibilities. Two main contributions are made in this paper. Firstly, three fragmentation points within DHT area are listed. Secondly, few novel validators are proposed to detect these fragmentations. The result obtained from tests done on manually fragmented JPEG files, showed that all three fragmentation points within DHT are successfully detected using validators. For future research, experiments can be extended to carve these JPEG files fragmented within DHT.

7 Acknowledgement

This work was supported by Universiti Tun Hussein Onn Malaysia (UTHM).

References

1. http://www.korelogic.com/Resources/Projects/dfrws_challenge_2006/DFRWS_2006_File_Carving_Challenge.pdf
2. Digital Forensics Research Workshop (DFRWS), (2007).
3. Garfinkel, S.: Carving contiguous and fragmented files with fast object validation. In: Proceedings of the 2007 digital forensics research workshop, DFRWS, Pittsburg (2007)
4. Pal, A. , Memon, N. : Evolution of file carving. In: IEEE Signal Processing Magazine, pp. 59-71, (2009)
5. Pal, A., Shanmugasundaram, K., Memon, N.: Automated Reassembly Of Fragmented Images. AFOSR Grant F49620-01-1-0243 (2003)
6. Richard III, G. G., Roussev, V., Marzial, L.: In-Place File Carving. In: National Science Foundation under grant # CNS-0627226 (2007)
7. Hall, G.A., Davis, W.P.: Sliding Window Measurement for File Type Identification. (2006)
8. Shannon, M.: Forensic Relative Strength Scoring: ASCII and Entropy Scoring. In: International Journal of Digital Evidence, Spring 2004, vol. 2, issue 4 (2004)
9. Li, W., Wang, K., Stolfo, S.J., Herzog, B.: Fileprints: Identifying File Types by n-gram Analysis. IEEE (2005)
10. Wallace, G.K.: The JPEG Still Picture Compression Standard. In: IEEE Transactions on Consumer Electronics (1991)
11. ITU T.81, CCITT.: Information Technology – Digital Compression and Coding of Continuous-Tone Still Images –Requirements and Guideline (1992)
12. Hamilton, E.: JPEG file interchange format v1.02. In: Technical report, C-Cube Microsystems (1992).
13. Pal,A., Sencar, H.T., Memon, N.: Detecting File Fragmentation Point Using Sequential Hypothesis Testing. In: Journal of Digital Investigations, pp. s2-s13 (2008)
14. Karresand, M., Shahmehri, N.: Reassembly of Fragmented JPEG Images Containing Restart Markers. In: Proceeding of European Conference on Computer Network Defense, IEEE (2008)
15. http://www.verytools.com
16. http://www.foundstone.com/us/resources/proddesc/bintext.htm