 ISSN : 0976-8491 (Online) | ISSN : 2229-4333 (Print)

Image Carving From Fragmented Clusters

[1]Balan C, [2]Nikhil R
[1]Scientist E, C DAC, Trivandrum, Kerala, India
[2]ER & DCI-IT, Trivandrum, Kerala, India

Abstract
The interaction with technology continues to grow in our day today life, so does the amount of data created through this interaction. The role of a digital forensics analyst is to acquire the data and draw conclusions about the events or actions, either caused by accident or with criminal intent create, delete or manipulate data and discover the facts about who or what is responsible for the event. ''Image carving'' reconstructs images based on their content, rather than using metadata that points to the content. The knowledge of common file structure, information contained in files, and heuristics regarding how file systems fragment data are the factors used in this carving process. This paper aims at carving files from unallocated space, free clusters, lost clusters, slack area, formatted partition, deleted partition or even in cases where file system could be unknown, corrupt or even missing. This paper also aims to carve the files in which the data are contiguously-allocated or fragmented.

Keywords
Fragmentation, Carving, Signature Based Approach, Candidate Weights

I. Introduction

The important problem in digital forensic world is recovering files from digital media for which no file system or metadata information available. There are several reasons for the missing of file system information. First, the destruction of file system because of formatting. Second, the desired file may have been deleted such that the metadata information no longer refers to the file content. Third, the presence of unknown file system in the digital media. Last, the usage of unallocated clusters and slack spaces for hiding certain files. The file content in all the above cases is usually unchanged until the clusters belong to the file are overwritten with other files. File carving is a process of file recovery from digital media by locating file signatures (header, footer) and extracting data between these end points. The start of a file in the digital media can be at cluster, sector, or at any byte (only in case of embedded files). For the optimization in the search for the header signature, it is sufficient to search first few bytes of every cluster or sector.

File carving can be done by using different tools, based on different techniques. The simplest type of carving is the Header-Footer carving technique. This technique works by searching the header and footer in the unallocated space of Image and it will extract the data in between them. This carving process has many disadvantages and they are listed below:

1. **Short Header and Footer:** If the header and/or footers are short, then it can appear multiple times when the search has been done.
2. **Fragmentation:** The fragmented files cannot be carved by using this technique. In fragmentation the data in between header and footer may not match the actual data of the file and the fragmentation process can break the file into certain pieces and store them in different available location. So Header-Footer carving will give us a mixed data not the original file.
3. **No Header and/or Footer:** Some files do not have fixed header and footer. Header- Footer carving is based on header and footer, so it cannot carve files with no header and footer. This is the case for plaintext file formats, like text documents and html files.

Basically, the Header-Footer carving cannot be used in certain cases explained above. So we need to develop a carver that a carve files when the data present in the files are fragmented. This paper deals with the carving of fragmented image files. And it also uses the Header-Footer carving technique to carve the contiguously allocated files.

II. Related Works

Data carving is a very important section in Recovery process. Data carving sometimes referred to as file carving or simply carving, is usually defined as the process of finding and extracting useful data from a data source. Files recovered from hard drive images, memory dumps, or other sources of data can be a source of evidence in an investigation. This has made carving an area of great interest within the digital forensics field and new methods, techniques, algorithms, and tools are being actively developed.

The evolution of data carving tools started with Start of File (SOF)/End of File (EOF) carving [5]. SOF/EOF carvers use a simple method of scanning the data image for common file type headers and possibly common file footers, then extracting data in between. Although the method is simplistic, these carvers were (and sometimes still are) fairly effective and yielded good results. However, fragmentation poses a large problem for SOF/EOF carvers.

One of the initial documents setting the stone rolling has been published by Richard and Roussev [6]. Their carver was signature-based, thus only allowing to recover unfragmented files. Files have been identified by so called header and footer signatures. File signatures are sequences of bytes that are characteristic for a specific file type: e. g. JPEG files start with a sequence of 0xFF 0xD8 and end with a sequence of 0xFF 0xD9 bytes. All data between such signatures of the same file-type was extracted.

Martin Karre and Nahid Shahmehri [2] propose a method, called Oscar, for determining the probable file type of binary data fragments. The Oscar method is based on building models, called centroids, of the mean and standard deviation of the byte frequency distribution of different file types. A weighted quadratic distance metric is then used to measure the distance between the centroid and sample data fragments. If the distance falls below a threshold, the sample is categorized as probably belonging to the modeled file type.

Anandabrata Pal, Husrev T. Sencar, Nasir Memon [3] presents a technique to identifying the fragmentation point of a file by

utilizing Sequential Hypothesis Test (SHT) procedure. The technique begins with a header block identifying the start of a file and then attempts to validate via SHT each subsequent block following the header block. The fragmentation point is identified when SHT identifies a block as not belonging to the file. For a file fragmented into more than two pieces, the techniques for identifying the starting file fragmentation point are no different than the techniques for identifying subsequent file fragmentation points. Other than the rare scenario where a file is fragmented because two or more of its fragments are swapped with each other, the gap between each of a file's fragment ending points and the next correct fragment's starting point contains data that does not belong to the file. A computer file is a collection of bytes, which correspond to eight-bit numbers capable of representing numeric values from 0 to 255 inclusive [4]. By counting the number of occurrences of each byte value in a file, a frequency distribution can be obtained. Many file types have consistent patterns to their frequency distributions, providing information useful for identifying the type of unknown files. The first step in building a byte frequency fingerprint is to count the number of occurrences of each byte value for a single input file. For each byte in the file, the appropriate element of the array is incremented by one. Once the number of occurrences of each byte value is obtained, each element in the array is divided by the number of occurrences of the most frequent byte value. This normalizes the array to frequencies in the range of 0 to 1, inclusive. This normalization step prevents one very large file from skewing the file type fingerprint.

III. Method

First we need to obtain some ideas about the incoming data types present in the input image. For this purpose we use a technique called Rate of Change (ROC)[2]. The rate of change method looks at the absolute values of the difference in byte value between two consecutive bytes. The number of rates of change values of a model and a sample are compared using the weighted sum of squares distance metric.

A. Rate of Change

The rate of change [2] can be defined as the absolute value of the difference between two consecutive byte values in a data fragment. In this way the ordering of the bytes is to some extent taken into consideration, but the algorithm cannot tell what byte values give a specific rate of change, apart from a rate of change of 255, of course. Neither can the rate of change method tell whether the change is in a positive or negative direction, i.e. a positive or negative derivative of the byte stream

The frequency distribution of the rate of change can be used as follows: A centroid is created, containing a mean vector and a standard deviation vector. The centroid is then compared to a sample vector by measuring the quadratic distance between the sample and the centroid, weighted by the standard deviation vector of the centroid, as described in Equation (1).

$$d(s,c) = \sum (s_i - c_i) / (\sigma_i - \alpha) \quad (1)$$

This method is meant to be combined with the original byte frequency distribution method used for the Oscar method [2]. The combination is made as a logical AND operation, because our idea is to let the detection abilities of both algorithms complement each other, thus cancelling out the weaknesses, i.e. reduce the number of false alarms. Depending on the similarity of the true positive sets of the two methods, the improvement will be more or less significant. The detection rate decreases if one of the algorithms has a close to optimal true positive set, while the true positive set of the other algorithm is less optimal. If we prioritize the detection rate we should use a logical OR operation instead.

Next we divide the input image into sectors of 512 bytes. This done under the assumption that a file system does not exist in the input image and so the cluster size is not available.

B. Identifying SOI Sectors

A JPEG image consist of a sequence of segments, each beginning with a marker, each of which begins with a 0xFF byte followed by a byte indicating what kind of marker it is. Some markers consist of just those two bytes; others are followed by two bytes indicating the length of marker specified payload data that follows (The length include the two bytes for the length, but not the two bytes for markers). Some markers are followed by the entropy coded data. Note that consecutive 0xFF bytes are used as fill bytes for padding purposes, although this fill byte padding should only ever take place for markers immediately following entropy coded scan data. We know that every file starts from beginning of a sector. This is same for any image file. We always find the Start of Image ("0xFF 0xD8") of JPEG file at the beginning of sector. So in order to find the starting of JPEG file, search first two bytes of every sector with "0xFF 0xD8". If a match is found, then we can say that a JPEG file starts from that sector.

C. Retrieving Header Part

We know that JPEG uses lossy compression algorithm for creating images. Therefore the JPEG files cannot contains any high values i.e. there is no 0xFF or 255 values in JPEG files. Whenever a high value is found, which means it shows a unique marker. JPEG header contain all the information about the image which include height and width of image in pixels, bit depth, camera information etc. JPEG header starts with Start Of Image (SOI) and end with Start Of Scan (SOS). Finding the header part is important in rebuilding the JPEG files in carving process. For finding and extracting the header, first find out the markers comes after SOI. Then read the two bytes after this marker, which is in big endian order, gives the position or offset of next marker in the same header. If the offset does not contains specific markers or the marker is SOS, then stop the process and extract the sectors from SOI to SOS.

D. Identifying JPEG Data Part

The data part of JPEG contains actual image data. It starts from start of scan (SOS) and ends with JPEG footer or End Of Image (EOI). The markers in the data parts are FF 00 and FF Dn where n=(0,1…). Within the entropy-coded data, after any 0xFF bytes, a 0x00 byte is inserted by the encoder before the next byte, so that there does not appear to be a marker where none is intended, preventing framing errors. Decoder must skip this 0x00 byte. This technique is called byte stuffing and is only applied to the entropy coded data has a few markers of its own; specifically the Reset markers, which are used to isolate independent chunks of entropy coded data to allow parallel decoding, and encoders are free to insert these Reset markers at regular intervals.

We need to search for any of the markers in every sectors of the input. If there is a hit, then store the sector number into the list and move to the next sector and do the same. On completing all

sectors, we got a list contains sector numbers where these markers were found. The next step is to identify the sectors where actual image data is situated from the list. If the difference between consecutive sectors in the list is less than the threshold value, and the minimum number of consecutive sectors in the list must be greater than 15, then we consider those sectors as a data a part of jpeg image.

E. Appending Data Sectors to Header

For appending data sectors to the header part which is already created, first find out the sector number from the list which comes after the sector after Start Of Scan. Then find out the fragmentation point or End of Image (EOI) by using the same condition to identify data parts, that is starting from the sector comes after the SOS of an image to the sector where difference between consecutive sectors will be greater than 15 or another SOI comes. Using this method, we can append the data part to the header and helps to identify the exact fragmentation point.

F. Is it Fragmented or Not?

After appending the first fragment to the header part, next step is to find out whether the created jpeg complete or not. It can be identified by using the SOI and EOI markers. The jpeg image contains one or more thumbnails within it. These thumbnails have same structure as that of the normal image. So those thumbnails start with same SOI and ends with same EOI as that of normal image. The trick to check whether the recreated image is fragmented or not is to count the number of occurrence of SOI and EOI. If the number of occurrence is an odd number, then we can say that the created jpeg is fragmented otherwise it is a complete file.

G. Stitching Fragmented Images

After finding the image which is fragmented, next step is to find the exact continuation fragmentation point of the image. For [1] determining adjacent fragments, examine pixels that form the boundary when the fragments are joined together. Absolute sum of prediction errors is calculated for the pixels along the boundary between the two fragments. Compute prediction errors for last row of pixels in the first fragment and the first row of pixels in the second fragment. In other words, determine the pixel values along the boundary formed between the two fragments because when the two fragments belong to an image, pixel value change will be smooth in most of the cases. So this fact can be used to determine the adjacent fragments of a file. For calculating candidate weights, corresponding pixels in the last row of the first fragment and the first row of second fragment are compared and a value is found out. Three methods can be used for computing this particular value. An overview[1] of finding adjacent fragments are given below.

1. Pixel Matching (PM)

This is a simple technique for determining adjacent fragments. Here the total number of pixels matching along the edges of size w for the two fragments is summed. Suppose the width (w) is 5, PM compare 1 numbered pixel in fragment i with the same numbered pixel in fragment j and the process is repeated for all boundary pixels. If the pixels matched, PM weight is incremented by one. For PM, if the weight is higher, the match will be better.

2. Sum of Differences (SoD)

Calculate the RGB pixel values of two fragments. Extract the RGB pixel values in the last row of first fragment and first row of second fragment. Then calculate the sum of differences of these values. SoD would sum the absolute value of the difference between each numbered pixel in fragment i with the same numbered pixel in fragment j. For SoD, if the weight is lower, the two fragments compared have more chances of being adjacent.

3. Median Edge Detection (MED)

Pixel value is predicted from the value of the pixel above, to the left and left diagonal to it. Take the difference between the predicted value in fragment j and the actual value and repeat the process for all boundary pixels. Then compute the sum of these differences. For MED, the lower the weight, the better the match [1].

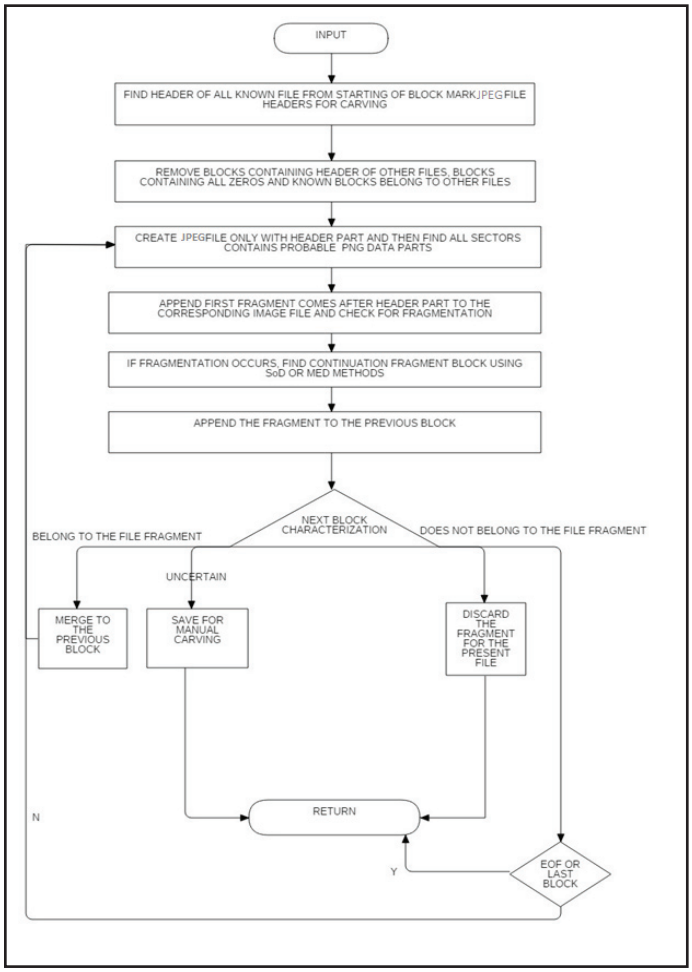

Fig. 1: System Design of Carver that Carve Fragmented Files

Fig. 1 shows the flow chart of the algorithm used in proposed system for carving. The input is loaded into the tool, which may be a disk image or a partition image or any junk data. The database of the algorithm stores all the known file signature. It starts scanning the input to find the signature of known files from the starting of sector. The file system actually stores the data in the clusters or block basics. But if the file system information is not available, then it is impossible to find the cluster size or block size. Therefore we use sectors instead of cluster for analysis. After finding file signature of known files at the starting of sectors, it removes all those sectors except sectors where signatures of JPEG files are found. It also removes sectors containing all zeros and sectors which belongs to other files for sure.

IV. Conclusion and Future Work

Traditional data recovery methods are based on the metadata information. When a file is deleted, the file's metadata structures and clusters will reside in unallocated space until it is reallocated

to other files. When the metadata is corrupt or missing, we cannot use the traditional techniques. There exist a chance for the clusters of deleted file may be reallocated before their metadata entries. In this scenario, by using the technique available to us, we retrieve metadata of deleted file and determine allocated clusters of that file. But now content in that location belong to a new file. So the metadata and content will not match or it will out of sync. So by using the traditional data recovery methods we may obtain certain false positive results. Most of the current carving tools are based on header-footer carving. They are not useful in carving fragmented files. The method implemented here can be used not only to carve fragmented image files but also used as a file type identification method. However, if one or more fragments are not available, then partial image can only be recovered. For determining the best possible matching fragments, we can use SoD or MED techniques. So in future; techniques that make use of both metadata and content to recover deleted files need to be devised. Data carving is based on guess work, so results may include many false hits especially in case of fragmented files. So use of many internal characteristics in addition to headers, footers, identifiers and size information must be entertained. This project is done using MATLAB R2014a and the output only shows the recovered original image. The MATLAB can predict the unrecovered pixels from adjacent pixels that available to us. The Interpolation of pixels can be done using different algorithms like Pixel matching or Median Edge Detection. This technique works based on the concept of Pixels not on the header/footer concept, so it can recover any type of images.

V. Acknowledgment

I express my deep sense of gratitude to my guide Mrs. Ashmi G V, Project Engineer, Dept. of Computer Science and Engineering, ER & DCI Institute of Technology, CDAC-TVM and to my friend Mr. Gokul C Raj, Cyber Forensics Professional, Ascertain Solutions, New Delhi, for all the motivation, inspiration, encouragement and guidance needed throughout the phases of this study.

References

[1] Akshara Ravi, Raj Kumar T, Angelo Renju Mathew,"A Method for Carving Fragmented Document and Image Files", International Conference on Advances in Human Machine Interaction (HMI– 2016).

[2] Martin Karre, Nahid Shahmehri,"Oscar - File Type Identification of Binary Data in Disk Clusters and RAM Pages", SEC, Vol. 201 of IFIP, pp. 413-424, Springer, 2006.

[3] Anandabrata Pal, Husrev T. Sencar, Nasir Memon, "Detecting file fragmentation point using sequential hypothesis testing", Digital Forensic Research workshop (DFRWS), 2008.

[4] M. Hossain Heydari, Mason McDaniel,"Content Based File Type Detection Algorithms", Proceedings of 36th IEEE annual Hawaii Internatioal Conference on System Science (HICSS'03), 2003.

[5] Bhadran V.K; Digambar Povar,"Forensic Data carving", Center for Development of Advanced Computing, Trivandrum, 2014.

[6] A. Pal; N. D. Memon,"The evolution of file carving", IEEE Signal Processing Magazine, pp. 59 – 71, 2009.

Mr. Nikhil R received his B.Tech degree in Electronics and Communications from AWH Engineering College, Calicut in 2014. He is currently pursuing M.Tech degree in Cyber Forensics and information Security at Electronics Research and Development Center of India - Institute of Technology. His fields of interest include Cyber Forensics, Digital Evidence Analysis, Vulnerability testing.

Mr. Balan is the Joint Director at Centre for development of Advanced Computing (CDAC). He has R&D experience of 15+ years in Cyber Forensics and 4 + years in artificial intelligence. He is also a Cyber Forensics faculty for Judicial Officers, Police, Intelligence Officers, Army and Corporate. He has provided technical support to various Law enforcement agencies for cyber-crime investigation. He has Authored number of International and national technical papers on cyber forensics.