DIGITAL INVESTIGATION USING HASH-BASED CARVING

Isabel Maria Sebastian, Noushida A, SafaSaifudeen and Surekha Mariam Varghese

Department of Computer Science and Engineering,
Mar Athanasius College of Engineering, Kothamangalam,Kochi, India

ABSTRACT

File carving is a popular method used for digital investigations for detecting the presence of specific target files on digital media. Hash based sector hashing helps to identify the presence of a target file. The hashes of physical sectors of the media is compared to the database of hashes created by hashing every block of the target files. To enable this, instead of evaluating the hashes of entire files, the hashes of individual data blocks is used for evaluation. Hash-based carving helps to identify fragmented files, files that are incomplete or that have been partially modified. To address the problem of High false identification rate and non-probative blocks, a HASH-SETS algorithm that can help in identification of files and the HASH-RUNS algorithm that helps in reassembling the files is used. This technique is demonstrated using the forensic tool: bulk_extractor along with a hash database: the has hdb and an algorithm implementation written in Python.

KEYWORDS

Forensics investigation, hash-based carving, HASH-SETS, HASH-RUNS algorithms, Sector hashing

1.INTRODUCTION

Digital forensics is a branch of forensic science in corporating the investigation and recovery of evidences found in digital devices, often in relation to computer crime[1].Crime scene investigations involve a scientific examination of the evidences. Digital evidences are mainly used in cases where direct evidences such as eye witnesses are not available or not feasible. File creation and modification dates, internet history cache, emails, chats,windows registery,recyclebin, shadow copies, log files etc. are some of the artifacts that can be extracted and used for forensic investigations[2]. A common method in digital forensics is to search for known files in an accused media. Brute force approach is to compare the known file contents with sectors of disk. Forensic practitioners' often use hash database of file segments to locate the content of sectors on disk with the known content.

Locating files containing child-pornography is a use-case for digital forensics using file carving. This paper presents hash based file carving method for detecting fragments of movies or images in the media.

DOI: 10.5121/ijci.2016.5214

2.RELATED WORK

Hash based file carving was introduced by Garfinkel as part of the solution to the DFRWS 2006 Carving Challenge. Using "the MD5 trick." the extracted text from the carving challenge was used to identify the original target documents on the Internet. He used a manual method to identify the location of the target files i.e. block-hashed the target files and sector-hashed the Carving Challenge and manually matched the two sets[1]'

The term "hash-based carving" was introduced by Collangeet al.in 2009in a pair of publications [3, 4] that explored the use of GPUs to speed the hashing load. But the question of what kind of database would be required to look up hashes produced at such a rate, and how to match files given the fact that the same block hashes appear in many different files was not discussed[1].
Later in 2013,Keydeveloped the File Block Hash Map Analysis (FBHMA) EnScript. This was a tool that createsa hash-map of file blocks from a master file list and selected areas of a target drive was searched for the blocks. But only a few files can be searched by FBHMA at a time.
To serve the purpose of enabling research on file-based digital forensics,Garfinkel et al. (2009) created the GOVDOCS corpus. Also the M57-Patents scenario, a collection of disk images, memory dumps, and network packet captures from a fictional company called M57 was created by Garfinkel et al. (2009).

Shortly, distinct blocks were used to detect the presence of known files on target media. This technique, though discussed by Garfinkel et al. (2010)presented no algorithms for doing so. So in 2012, Foster attempted to determine the reason for the repetition of 50 randomly chosen blocks that were shared between different files in the GOVDOCS corpus.

This idea was expanded by Young et al. (2012) by using a database of distinct block hashes of target files to detect the presence of those files on media being searched. The initial implementation of the hashdb hash database was described in this article [1].

The coverage/time trade-offs for different sample sizes were examined by Taguchi while using random sampling and sector hashing for drive triage. He concluded thatin a wide variety of circumstances, 64 KiB is an optimal read size.

Different file carving taxonomy was proposed by SimsonGarfinkel and Joachim Metz. These methods differed in the way they analyzed their inputs. In block based carving method, the input is analyzed block-by-block to determine if a block is part of a possible output file, on the assumption that each block can only be part of a single file (or embedded file). In statistical carving, the input is analyzed on basis of characteristics and statistics. Other relevant carving methods are Fragment Recovery Carving, File structure based Carving, Header/Footer Carving etc.

3.BACKGROUND

"Hash-based carving" describes the process of recognizing a target file on a piece of searched media by hashing same-sized blocks of data from both the file and the media and looking for hash matches [1]. To search for known contents, the MD5 hash algorithm is used because of its speed.

3.1.Tools Used

Hashdb is the tool that is used for finding previously identified blocks of data in a media such as disk images [9]. The following services are provided by hashdb:

- By importing block hashes and providing lookup services, the hashdb manages block hash databases.
- Hash databases can be created or scanned by other programs using the hashdb library.
- libhashdb of the hashid scanner which is a bulk_extractor plugin,can be used to search for previously identified blocks of data .

Cryptographic hashes (along with their source information) that have been calculated from hash blocks are stored by hashdb. Instead of full file hashing, it relies on block hashing.In block hashing, artifacts are identified at the block scale (usually 4096 bytes). Many of the capabilities of hashdb are best utilized in connection with the bulk_extractor program.

Bulk Extractor (bulk_extractor) is a feature extraction tool written in C++ for extracting features from media images [8].The input which can be a disk image, files or a directory of files, is split into pages and processed by one or more scanners. Bulk Extractor parallel-processes 16MiByte pages of media on multiple cores. 4KiB sectors are recommended for hashing if they are directly supported by the drive. Otherwise combine 8 adjacent 512B sectors into a single 4096-byte super-sector for hashing and for feature extraction.The feature files stores the extracted features so that they can be easily inspected, parsed, or processed with automated tools. These feature files are processed using Python programs.

Python which is a general-purpose, high-level programming language is used for the implementation of "hash-sets" and "hash-runs" algorithm. It is an open-source software.Python is designed to be highly readable. Major advantage of this scripting language is that it has fewer syntactical constructions and best suits a beginner.

Another important tool is md5deep. It is a set of programs used to compute MD5 message digests on an arbitrary number of files. The MD5 hash algorithm (developed by Ronald Rivest) which is an updated version of MD4, is used to generate block hashes because of its speed.It helps to compare and store these small hashes more easily. Alsoin cryptography, verification is done using one-way hashes.

MD5 is just one of many hashing algorithms. There's also SHA-1 (Secure Hash Algorithm), developed by NIST in conjunction with the NSA and produces 160 bit digest. It's more collision resistant than MD5 and is the most commonly used hash for cryptographic purposes these days. Though other hashing algorithms are available, MD5 and SHA-1 are commonly used in cryptography.

3.2.A hash-based carving process

Hash-based carving is a four-step process:

- DATABASE BUILDING: A database of file block hashes is created from the target files.

- SCANNING THE MEDIA: A set of hash values that matched the target files is produced byscanning the searched media. Here we hash 4KiB sectors and searchfor those hashes in the database.
- PROBATIVE BLOCK SELECTION: Most probative (likely) target files on the searched media is determined.
- TARGET ASSEMBLY: Runs of the matching candidate blocks on the searched media are identified and mapped to the corresponding target files.

4.MATCHING SCENARIOS

The basic idea of hash-based carving is to compare the hashes of 4KiB sector clusters from a storage device ("search hashes") with the hashes of every 4KiB block from a file block hash database and identify files based on hashes that the two sets have in common.

- Hash matches can be observed in a variety of scenarios, including: The presence of a copy of an target file, in a complete form on the searched media.
- The presence of a copy of the target file on the searched media at some time in the past which has been deleted at a later point, which may or may not be partially overwritten.
- The presence of a file on the searched media that may have many sectors in common with the required target file.
- The presence of a target file which is embedded in a larger carrying file, detected only if that the file is embedded on an even sector boundary.

5.IMPLEMENTATION

First step in the implementation of hash based carving process is the construction of a database. Here the overlapping 4KiB sectors are hashed using the MD5 tool. Then we scan the disk image of the media using bulk extractor. The output of this stage produces different .txt files. From the obtained output, identify the candidate blocks using probative block selection tests like the ramp test, the white-space test, the 4-Byte histogram test etc. On these selected blocks, apply the HASH-SETS and HASH-RUNS algorithm to determine whether the media contains the required target files. Detailed description of each stage is given below.

5.1.Database Building: creating the target hashdb

A hashdb database that contains the 4KiB block's hash values is created. The output database is renamed with .hdb extension. To build the database, run hashdb from the command line:

- hashdb create sample.hdb

Here sample.hdb is an empty database. ADFXMLfilecontainingsector hash values is required to import data into the database. To populate the hash database with the hashes from the DFXML file called sample.xml:

- hashdb import sample.xml sample.hdb

This command, if executed successfully, will print the number of hash values inserted. For example:

hashdb changes (insert): hashes inserted: 2595

5.2.Media Scanning: Finding Instances Of Known Content On The Searched Media

Here the database is used to search the disk image using the bulk extractor has hdb scanner. Bulk_extractor breaks the disk image into 16MiB "pages" and only processes pages that are not blank. Each page is broken into overlapping 4KiB blocks, each block is hashed with the MD5 algorithm and the resulting hash is used to query the block hash database. Matching sector hashes are reported in bulk_extractor's identified_blocks.txt file. Count value indicates whether each hash matches one or more target files. Matched hashes are used in both the "candidate selection" and "target assembly" phases.

To run bulk_extractor from the command line, type the following command:

- bulk_extractor -o output mydisk.raw

In the above command, output is the directory that will be created to store bulk_extractor results. It cannot already exist. The input mydisk.raw is the disk image to be processed. After the run the resulting output files will be contained in the specified output directory. Open that directory and verify files have been created. There should be 15-25 files. Some will be empty and others will be populated with data. These two steps are combinable using a batch file that incorporates the command line commands for running the hashdb and the bulk extractor.

5.3.Candidate Selection: Identifying Probative Blocks

The hashdb'sexplain_identified_blocks command is used to determine the files to which these block hashes correspond. This is implemented using a data reduction algorithm. For each block inidentified_blocks.txt file, if the block maps to fewer than N files (the default is 20), those files are added to the set of candidate files. To run it from the command line, type the following instructions

- hashdbexplain_identified_blockssample.hdb out/identified_blocks.txt> out/explain_identified_blocks.txt.

The program writes out a new file called explain_identified_blocks.txt with a list of the deduplicated sector hashes, the number of times that the hash appeared in the original file, and all of the source files in which the hash appears.

Next, run the program report_identified_runs.py program which writes theidentified_blocks_explained.txt file.This is implemented using the explain_identified_blocks.txt output file: if the hash is not flagged by any of the ad hoc tests like the ramp test, the white-space test, the 4-Byte histogram test (they identify the non-probative blocks), the hash's sources are added to the set of candidate files.

5.4 .Target Matching:

After selecting the candidate files, the block hashes corresponding to candidates are grouped into source files. They are eventually tallied or reassembled into runs of matching blocks with the HASH-SETS and HASH-RUNS algorithms.

5.4.1.Hash-Sets: Reporting The Fraction Of Target Files

HASH-SETS is a simple, memory efficient algorithm that employs block hashes to generate a report of the fraction of blocks associated with each target file that is found on the searched media [1].

It is implemented as follows:

1. A list of candidate targets are determined by employing the candidate selection algorithm.
2. Repeat for each block hash H in the file identified_blocks_explained.txt:
    (a) Repeat for each target T matching against H:
        i). If T is a Candidate target, add 1 to the score of that particular target.
3. Repeat for each target T
    (a) To compute the fraction of the file present, divide the score by the count of number of blocks in the targetfile.
4. Sort the targets in inverse order of the fraction of the file present so that a list with highest fraction first is obtained.
5. If the fraction recovered exceeds the threshold, report the target file name, number of recovered blocks, and the fraction of the file recovered. The number of blocks recovered is counted by the algorithm and so the score is solely a function of the target file and the searched drive.

Here the number of blocks is counted by the algorithm so that the score would be solely a function of the target file and the searched drive.

5.4.2.Hash-runs: locating target files

The presence of target files is detected by the HASH-SETS algorithm and the HASH-RUNS algorithm is used to report the location.

- Handling of the case when the target file is on the searched media in multiple locations.
- Employs the placement of adjacent logical sectors in a file in adjacent physical sectors of the searched media to its advantage.
- Accounts for different blocks in a file having the same (mod 8) value.
- Detects runs of recognized blocks being separated by null blocks and combines them for efficiency.

In this algorithm, first the data structures are created by the HASH-SETS implementation. It then identifies all of the block runs on the physical disk that correspond to logical runs of the target files [1]. Using the logical block number in the target file, these blocks are sorted by and reported to the analyst.

1. First the identified_blocks _explained.txt file is read by the algorithm (previously produced by the hashdb program). Then, an in-memory database is built that maintains the set of sector hashes associated with each target file.
2. A second database is built by the algorithm for each (target file, sector hash) pair to record the set of block numbers in the target file where the block appears. So if target file A has 6 blocks, and both blocks 1 and 4 match sector hash H1, then the element (A,H1) of the database contains the set {1,4}.
3. Next, the algorithm reads the identified_blocks.txt file. For each sector hash that was found in a target file, it records the disk block at which the hash was found.
4. Finally, there comes the target matching step. For each (target file, mod8) combination:

a) The algorithm builds an array consisting of elements in the form:

[diskblock,{fileblocks},count] Where disk block is the physical disk block, { file blocks} is a set of blocks within the target file where the hash was found, and the number of times in the sector hash database that the sector hash was found.

b) The array is sorted by diskblock.
c) The algorithm runs a sliding window over the rows of the array. It helps to identify

rows that represent sequential disk blocks and file blocks.. A new array of block runs is created, where each element in the array has the values:

- Identified File (from the hash database)
- Score (The number of identified blocks)
- Physical sector start
- Logical block start
- Logical block end

d) If the number of bytes in the sector gap between the two runs matches the number of bytes in the logical blocks, and if all of the sectors on the drive corresponding to the gap contain only NULLs, then the two block run elements are combined.
e) Drop the block runs that are smaller than a predetermined threshold.
f) Finally, for every reported run, use the SleuthKit to determine the allocated or deleted file that corresponds to the first block in the run.

The program's output is obtained in the form of a CSV file that can be readily imported into Microsoft Excel.

6.CONCLUSION

A hash-based carving system implementation is presented here. The bulk_extractor program is used by the carving system to create a block hash database of target files. Also, a sector hash record of searched media, supported by the hashdb hash database is created. Individually recognized block hashes are assembled into carved runs which is performed by a post-processing Python script.

7.IMPROVEMENTS

During the database construction using hashdb, the non-probative blocks can be flagged. These flag status can be stored in a database to help in the probative-block selection process. This improves the overall efficiency. Also, files composed entirely of non-probative blocks can be avoided in the earlier stages itself. File allocation status can be considered along with the (mod 8) value in the target matching. Though this is an expensive filtering step, it contribute a little to the accuracy rate. For an encrypted file system, the media is first mounted on an appropriate decrypting driver and later the unencrypted file is accessed. Also, other hashing techniques like SHA-1 can be used for block hashing.

REFERENCES

[1] SimsonL.Garfinkel& Michael McCarrin (2015), "Hash-based carving: Searching media for complete files and file fragments with sector hashing and hashdb", Digital Investigation 14(2015) 895e8105
[2] https://forensicswiki.org
[3] Collange S, Dandass YS, Daumas M, Defour D. Using graphics processors for parallelizing hashbaseddata carving. CoRR abs/0901.1307. 2009.,http://arxiv.org/abs/0901.1307.
[4] Collange S, Daumas M, Dandass YS, Defour D. Using graphics processors for parallelizing hash-based data carving. In: Proceedings of the 42nd Hawaii International Conference on System Sciences; 2009. Last accessed 03.12.11, http://hal.archives-ouvertes.fr/docs/00/35/09/62/ PDF/ColDanDauDef09.pdf.
[5] Allen B. hashdb. 2014. https://github.com/simsong/hashdb.git.
[6] http://booksite.elsevier.com/samplechapters/9780123742681/Chapter_6.pdf
[7] http://simson.net/ clips/academic/2012.IEEE.SectorHashing.pdf.
[8] https://forensicswiki.org/bulk_extractor
[9] https://forensicswiki.org/hashdb
[10] https://en.wikipedia.org/wiki/Digital_forensics
[11] http://www.tutorialspoint.com/python/
[12] https://github.com/NPS-DEEP/hashdb/wiki
[13] http://sourceforge.net/p/guymager/wiki
[14] http://forensicswiki.org/wiki/Famous_Cases_Involving_Digital_Forensics#

Authors

Isabel Maria Sebastian is currently pursuing B.Tech in Computer Science and Engineering in Mar Athanasius College of Engineering.

Noushida A is currently pursuing B.Tech in Computer Science and Engineering in Mar Athanasius College of Engineering.

SafaSaifudeen is currently pursuing B.Tech in Computer Science and Engineering in Mar Athanasius College of Engineering.

Surekha Mariam Varghese is currently heading the Department of Computer Science and Engineering, M.A. College of Engineering, Kothamangalam, Kerala, India. She received her B-Tech Degree in Computer Science and Engineering in 1990 from College of Engineering, Trivandrum affiliated to Kerala University and M-Tech in Computer and Information Sciences from Cochin University of Science and Technology, Kochi in 1996. She obtained Ph.D in Computer Security from Cochin University of Science and Technology, Kochi in 2009. She has around 25 years of teaching and research experience in various institutions in India. Her research interests include Network Security, Database Management, Data Structures and Algorithms, Operating Systems, Machine Learning and Distributed Computing. She has published 17 papers in international journals and international conference proceedings. She has been in the chair for many international conferences and journals.