Simson L. Garfinkel

Carving contiguous and fragmented files with fast object validation

Presented by Jevin Sweval

Overview

- **Introduction to carving**
- File fragmentation
- Object validation
- Carving methods
- Conclusion

Introduction to carving

- Carving is the recovery of files from a raw dump of a storage device without the file system metadata
  - Data recovery: corrupted drive, physical damage to drive, accidental deletion
  - Forensics recovery: criminal suspect formatted the drive or tried to delete delete evidence

Introduction to carving

- Normally, a file system's metadata contains an index of files' locations on disk sectors
- Without that metadata, these files must be identified heuristically
- This ranges from fairly straightforward (JPEGs) to difficult/impossible (encrypted files) depending on the characteristics of the file type being recovered

Problems with current carvers

- Carvers can only handle contiguous files – files that have been fragmented cannot be reconstructed
- Reconstructed files are not thoroughly verified by carvers leading to high numbers of false-positives

Overview

- Introduction to carving
- **File fragmentation**
- Object validation
- Carving methods
- Conclusion

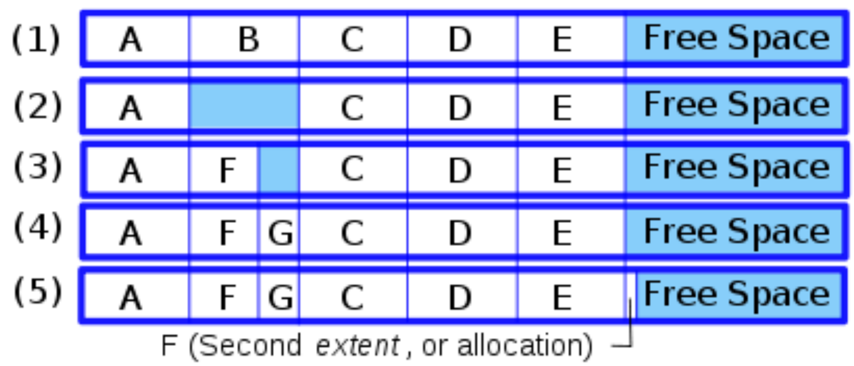
File Fragmentation

- On a disk, sectors may be in use or free for writing
- File systems must decide how to place new data in the free sectors
  - Often try to place the file in contiguous sectors for best performance

File Fragmentation

- At some point, it becomes worthwhile or necessary to split a file across non-contiguous sectors
  - There weren't enough contiguous sectors to hold the file
  - The file was extended but another file was directly after the original

File Fragmentation



Observed File Fragmentation Patterns

- Garfinkel collected over 300 drives from secondary markets from 1998-2006
- 6% of the all files were fragmented
  - ~50% of the drives had no fragmented files

Fraction of files fragmented on drive	Total Drives
f=0%	145
0<f<0.01	42
0.01<f<0.1	107
0.1<f<1	30
Total	324

Distribution of file fragmentation

Observed File Fragmentation Patterns

# Fragments	FAT # Files	NTFS # Files
0	1,286,459	521,663
2	25,154	22,984
3	4932	6474
4	2473	3653
5-10	4340	13,139
11-20	1593	7880
21-100	1246	11,901
101-1000	186	5953
1001-	2	590
Total	1,326,385	594,237

Distribution of number of fragments

Does file fragmentation matter when carving?

- If the most forensically interesting file types are not usually fragmented, fragmentation is less of a concern
- Most interesting file types (.doc, .jpg, .pst) files are fragmented 18-57% of the time while "boring" (.bmp, .ini, .chm) files are commonly fragmented < 10%
  - Yes, fragmentation is a concern!

Highly Fragmented Files

- It was observed that oft-updated system DLL and CAB files became extremely fragmented (> 100 fragments)
- Temporary files were fragmented 66% of the time

Bifragmented Files

- The paper focuses on bifragmented (2 fragment) files
  - Bifragmented files can be reconstructed with relatively simple algorithms
- Most bifragmented files were split by 1, 2, 4, 8 512-byte sectors
  - Believed to be caused by a single-cluster file within the fragmented file
  - The commonly short length of these gaps will make reconstruction quite fast

Overview

- Introduction to carving
- File fragmentation
- **Object validation**
- Carving methods
- Conclusion

Object Validation

- A recovered file should pass some validation steps to ensure that it is valid and meaningful
  - Too little validation leads to many recovered files that were not valid files on the original file system
- Some conflation of terms has occurred
  - Files may have objects embedded within them (JPEG thumbnails, Word documents with pictures, etc)
  - The ability to reconstruct an embedded object without reconstructing the parent file is still useful
  - Object validation is a generalization of file validation

Object Validation

- Naïve object validation
  - Attempt to validate all possible subsequences of a drive
  - 200 GB drive ➔ 2×10^{22} subsequences!
- Files on FAT and NTFS are sector-aligned
  - You can eliminate 511/512 (99.8%) of all subsequences
  - ReiserFS is optimized for many small files and can coalesce multiple small files into one sector (tail packing)

Object Validation

- For objects that can withstand appended, arbitrary data (e.g. JPEG), binary search can be used to greatly reduce the number of validations performed
  - Try validating [obj_start, drive_end]
  - If successful, try [obj_start, (drive_end-obj_start)/2]
  - Repeat the binary search until the you find the smallest object that validates

Object Validation

- By optimizing to sector-alignment and using the binary search method, validations can be reduced to $4x10^8$ plus ~40 validations / identified object

Objects with headers and footers

- If the object contains easily identifiable headers and footers (think magic numbers), it is easy and fast to find potentially valid subsequences
  - JPEG files have a one of two constant 4 byte headers and a constant 2 byte footer
    - A random subsequence will match that pattern just 2 in 2^{48} times

Container Objects

- Most file formats have several internal structures
  - JPEGs have metadata, thumbnails, Huffman tables
  - Archive files have indexes and multiple compressed files
  - Word files have a complicated, almost file system like structure

Container Objects

- The internal structures of these container objects provide more opportunities for validation and rejection of invalid objects
- The structures often have fields that specify the length of a following data structure or pointers to other data structures in the object
  - These fields can be checked to see if they are internally consistent
  - The fields may also give the carver more information about the object like a lower or upper bound on its length

Validation with decompression

- If an object contains a compressed data structure, the carver can attempt to decompress the structure and check for errors
  - This is much slower than simple byte comparisons but may be fast enough for small structures
- The Huffman symbol tables in JPEG files can be quickly checked for validity
- Word documents with non-ASCII or extremely uncommon characters may be discarded

Validation with decompression

- In this paper, if a JPEG Huffman symbol table validated, the carver attempts to decode the entire picture
  - The decoder often successfully decoded multiple invalid sectors before actually detecting an error
  - Luckily, none of the potential JPEGs that had invalid sectors were successfully decoded in their entirety

Semantic validation

- Even with all of the previous validations, a false-positive may still get through
- By using plausibly expected semantics, files can be further validated
- For example, if a validated file contains a lot of engineering jargon with some cooking related terms in the middle, it is likely that a fragmented recipe lies in the middle of a fragmented thesis

Overview

- Introduction to carving
- File fragmentation
- Object validation
- **Carving methods**
- Conclusion

General carving procedure

- The carver presented in the paper worked by taking a sector then gradually expanding it while periodically validating the result
- If an object validates, the sectors in that object are marked as used and are ignored when trying to construct further objects

Contiguous carvers

- Can carve sector-aligned objects to find files (fast) or unaligned objects to find embedded objects (slow)
- Each file type has its own validation function that is passed a string to either declare as valid or to declare as invalid (with some optional additional information that can aid in optimization)

Contiguous carvers

- Header/footer carving
  - Try validating any sequences starting with a valid header and ending with a valid footer
- Header/maximum size carving
  - Uses binary search method to find the largest object that validates
  - The paper did not discuss the advantages of length maximization vs. length minimization (discussed on slide 18)

Contiguous carvers

- Header/embedded length carving
  - The validators are passed longer and longer strings, starting at an identified header, until an internal field is found that indictates the total object length
  - The resulting complete object is passed to the validator for final validation
- File trimming
  - Some validators will try to trim a file down a character at a time until it no longer validates

Bifragmented carver

- For a string that starts with a valid header and ends with a valid footer but does not validate, it is possible that the object was fragmented
- The paper only attempts to validate files that have been split into two fragments

Bifragment carver

- The algorithm inserts a gap (originally one sector long) between the header and footer
- The gap is "slid" around between the header and footer and the defragmented sectors are sent through the validator until the object validates
- If the gap has "slid" through all possible positions without creating a valid object, the gap is grown by a sector and the process is repeated

Bifragment carver

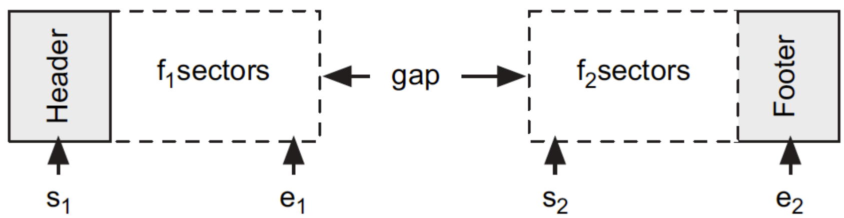


Fig. 2 – In Bifragment Gap Carving the sectors s_1 and e_2 are known; the carver must find e_1 and s_2.

Bifragment carver

- For objects with headers and footers, the algorithm runs in $O(n^2)$ time (where n is the object size)
- To find all such objects on a disk, the runtime is $O(n^4)$ because every possible header/footer combination must be validated

Bifragment carver

- Word Documents
  - Contain a header but no footer
  - The header contains the file offset of an internal structure that has its own header (easy to find on disk) and the total file length

Bifragment carver

- Algorithm to carve a Word file
  - Find the master file header on the drive and the header of the internal structure
  - The internal structure gives the total length L of the file and the master header gives the file offset of the internal structure
  - Use a variation of the gap method that only attempts to validate defragmented objects of length L that contain the internal structure in the second fragment

Bifragment carver

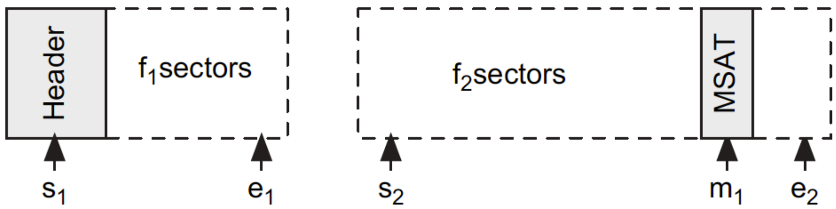


Fig. 3 – In Bifragment Carving with constant size and known offset the sectors s_1 and m_1 and $f_1 + f_2$ are known; the carver must find e_1, s_2 and e_2.

Bifragment carver

- The efficiency of this algorithm is $O(n^3)$ to find a file given the location of the file header and $O(n^4)$ to find all such files on the drive

Conclusions

- Files of forensic interest are often fragmented, evading reconstruction by existing carvers
- Validation of internal structures allows carvers to reduce false-positives and can help carvers attempt to reconstruct fragmented files
- **Encrypt your data if you don't want it to be discovered!**