Mag. iur. Dr. techn. Michael Sonntag

File carving

Institute for Information Processing and
Microprocessor Technology (FIM)
Johannes Kepler University Linz, Austria

E-Mail: sonntag@fim.uni-linz.ac.at
http://www.fim.uni-linz.ac.at/staff/sonntag.htm

Agenda

- What is file carving and why do it?
  - → Deleting files in NTFS and EXT3
  - → Main problems
- Simple file carving
- The file carving process
- File carving software
  - → Scalpel
  - → X-Ways Forensics
  - → CarvFS
  - → Sliding Entropy
- Semantics-based file carving

What is "file carving"?

- Recovering a file from unstructured digital forensic images
  - "Unstructured" → File **meta**data is no longer available
  - I.e., the file **content** is (partially) still on the disk (as sectors), but the sequence of the sectors as well as start, end, length, owner etc. is missing
- Typically last effort: No "undelete" poss., but still suspicions
  - E.g. keyword searches of the whole disk found something
- Reasons for file carving
  - The file system was damaged or deleted
  - Using a modern file system (e.g. ext3)
    - They overwrite important data on deletion
    - But typically low level of file fragmentation (→easier carving)!
  - Hard disks are in use for a long time and are faster
    - Less need for defragmentation; defragmentation more difficult (and therefore rarer) on modern file systems

EXT3 delete

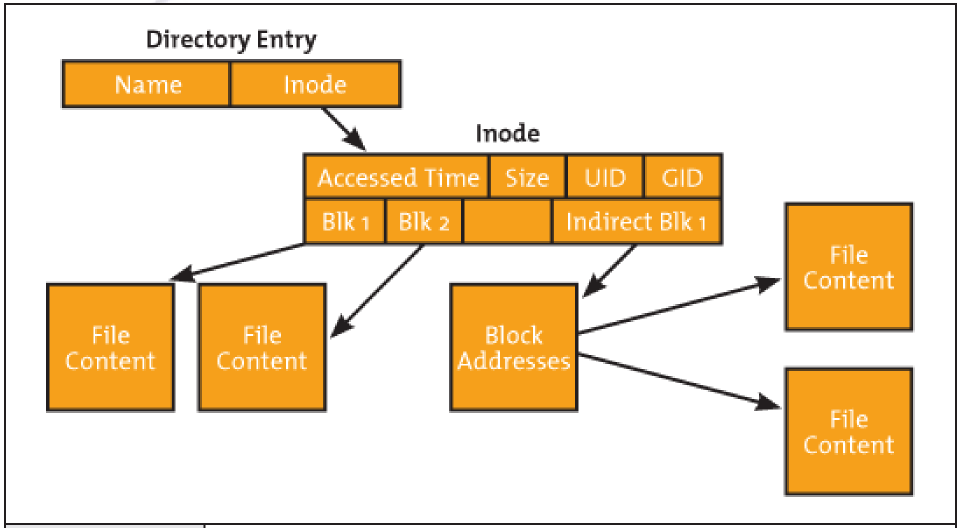


FIGURE 1 RELATIONSHIP BETWEEN THE DIRECTORY ENTRY, AN INODE, AND BLOCKS OF AN ALLOCATED FILE

Before deletion
(file still exists)

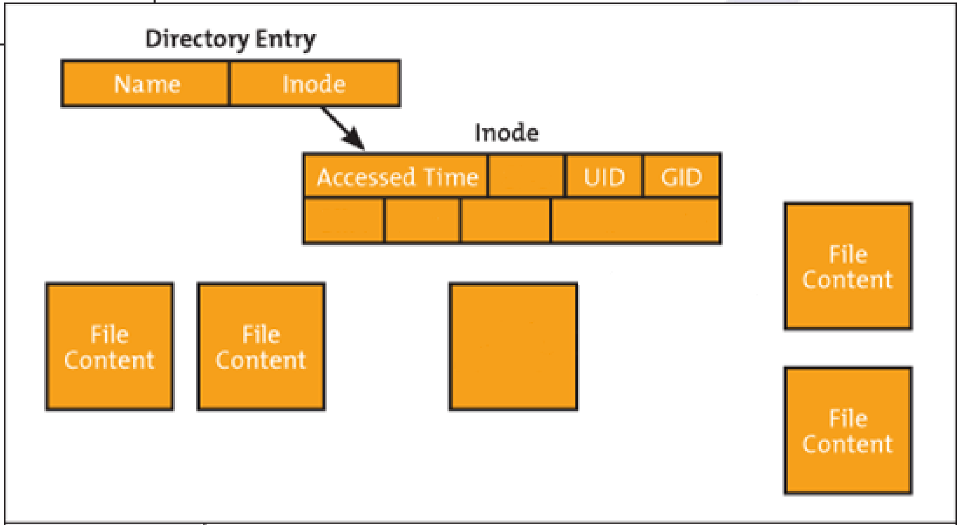


FIGURE 2 RELATIONSHIP BETWEEN THE DIRECTORY ENTRY, AN INODE, AND BLOCKS OF AN UNALLOCATED EXT3 FILE. THE LINKS BETWEEN THE INODE AND BLOCKS HAS BEEN CLEARED.

After deletion
(file removed)

Modified from: Carrier: Why recovering….

Main problems of file carving

- Time complexity of file carving: NP-complete
  - → You must try all possible combinations of fragments/clusters
    - » You don't know in advance how many clusters a file consists of
  - → Optimizations are possible (and necessary!) to reduce this
    - » Depending on the file type in questions
    - » Depending on the file system used
    - » Depending on additional information, e.g. content redundancy
- File systems become ever larger
  - → ≥ 1 TB hard disks are inexpensive and common
  - → Huge numbers of files and huge numbers of fragments!
    - » But individual files usually lightly fragmented
- File start is at sector boundary, but end not (slack space!)
- Files may be incomplete
  - → Start/end/middle sectors may have been reused for new files

1 TB = 4096 * 268.435.456 !!!

Simple file carving

- Pre-knowledge needed:
  - → Where does the file system (=partition) an it’s data area start
  - → How large is a cluster
- Identify start & end of file and extract everything in between
  - → Example: JPEG (Start = FFD8, End= FFD9)
- Will only find files with existing beginning (marker)
  - → First cluster lost → Gone!
- Requires identifying the end of the file
  - → Often very difficult!
- Often produces huge files with lots of irrelevant data
  - → Result contains same data/other carved files several times!
    - » First 20 kB file will be carved for a length of 10 MB and therefore contains also the next ten/twenty/… 20 KB files!

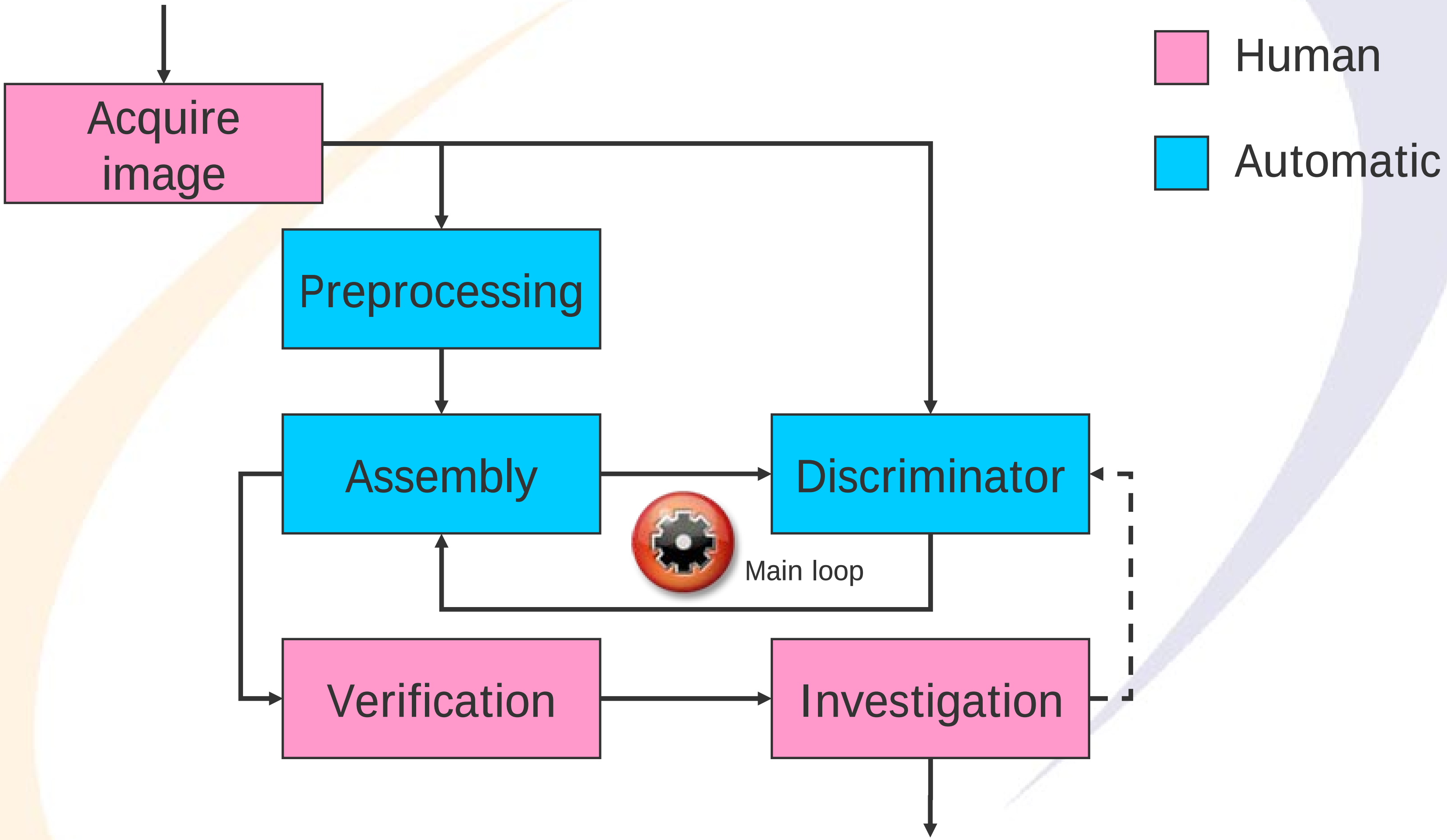
Detecting the end of a file

- If a specific signature exists → Perfect!
  - → Note: Some files have header or footer signatures occurring perhaps several times within the file!
- Length of the file may be found in the header
  - → Requires detailed knowledge of the file format
    - » Especially problematic with proprietary software!
- Header signature of a new file
  - → Embedded files can be troublesome in this respect!
      - – Example: Pictures in text documents, videos in presentations, …
    - » Would mean premature termination → Careful!
  - → But: Would have to be aligned on sector start
- Maximum file length reached
  - → This is a fallback and very inefficient!
  - → File viewers will usually ignore added data after the end
- End of image reached (or partition/disk)

Simple file carving

- Works only for non-fragmented files
  - → Improvement: Exclude all sectors in use by other files
    - » "Real" files (still existing) and those extracted previously
    - » Other approach: Ext2/3 → The 13th block is usually an indirect pointer block (if everything was allocated in sequence)
      - – This might be verified through its internal structure/data
  - → No reordering of sectors, no intertwining allowed
    - » Reordering: Usually because of later appending to a file
      - – Or creating it and very slowly writing to it (size unknown at start)
    - » Intertwining: Space was too small for the file
      - – Can happen also on creation of a “full” file (e.g. copy)
- Requires extensive manual improvement
  - → Removing duplicates and erroneous results
  - → Manual reordering/reassignment of clusters

The file carving process: Overview



After Cohen: Overview of Carving Algorithms (p. 8)

The file carving process: Description of human activities

- Acquire image: Acquiring a forensic duplicate from the original media in a safe way, preserving chain of custody
  - → Use write blockers and store in an appropriate format
- Verification: Making sure the result is actually a result
  - → It not only "looks" like an image/PDF/…, it actually is one!
  - → Check whether it is complete or only partially recovered
  - → Other tasks: Extraction, duplicate removal
- Investigation: Relate the result to the investigation aims
  - → Is it relevant for the case?
    - » If very relevant but incomplete, the main loop might be restarted with additional information from the manual inspection
      - – Or completely manually!
  - → Extraction of the evidential value, correlation with other evidence, documentation, etc.

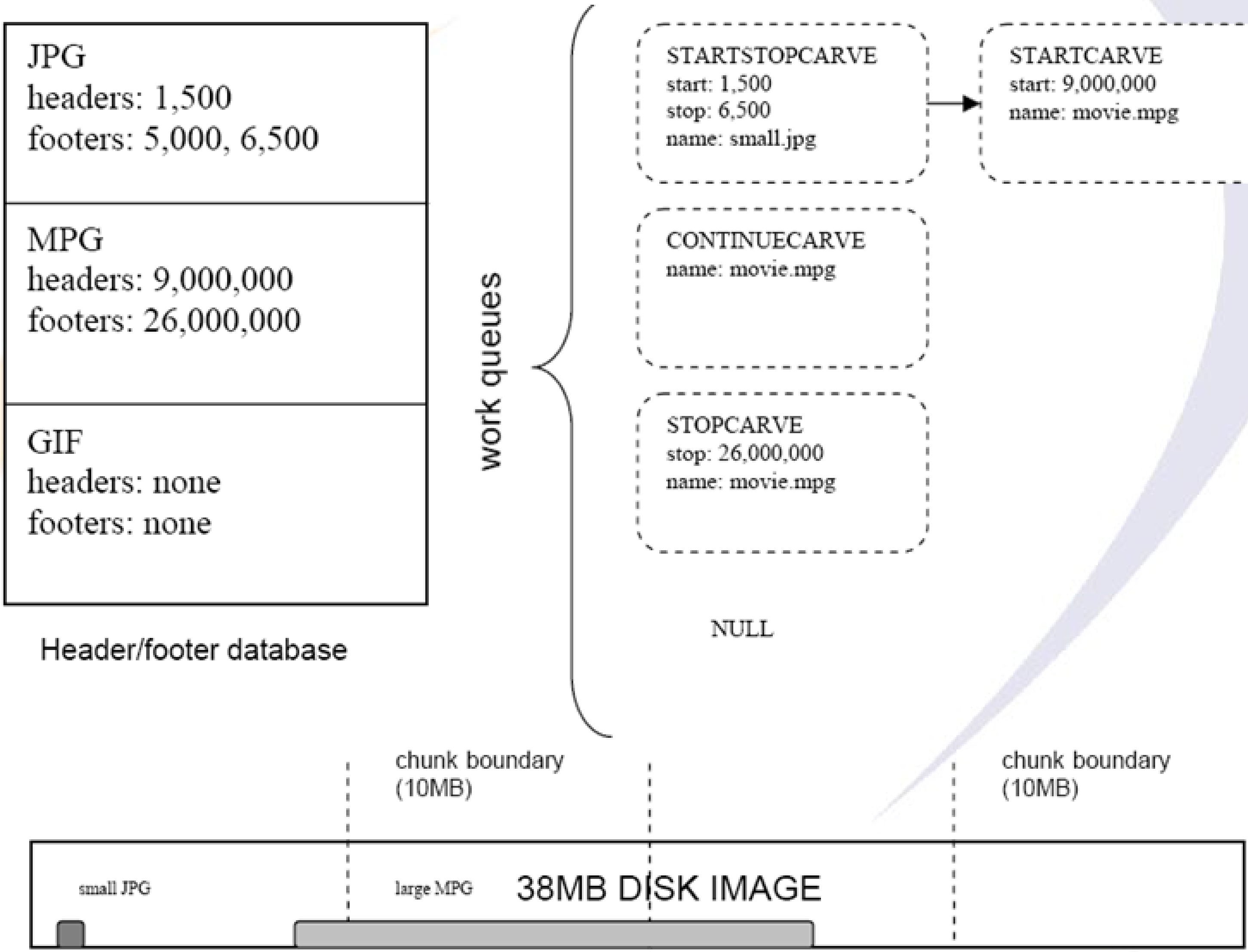
The file carving process: Description of automatic activities

- Preprocessing: Extracting information about the file
  - → Identify file type; identify start and end/length if possible
  - → Select all sectors which potentially could be part of the file
- Assembly: Generate a potential version of the file
  - → Decide which sectors to include
  - → Concatenate these sectors in a "sensible" manner
    - » According to various strategies and based on various data
  - → Note: Try "best" files first to reduce scope of searching!
- Discriminator: Check whether the result could be correct
  - → Can this file be "decompressed" or does it make "sense"?
  - → Where in the file is the erroneous position?
  - → Some parts belonging at an absolute position?
  - → Usually based on viewers/printers
    - » Difficulties: No specific error reporting, internal error recovery
    - » Is additionally problematic if the file was corrupt anyway

File carving software: Scalpel

- Reprogramming of "Foremost" for better performance and less memory requirements
  - → Limited to two sequential passes over the whole image
    - » First: Create DB of file headers and search for possible footers
      - – Only when header found and reasonably near (max. file size)
    - » In between: Matching headers and footers to create files
      - – Creates work queues for each chunk (typ. 10 MB)
    - » Second: Extract all files by working the queues for each chunk
      - – To avoid memory-to-memory copies
- Based on the "simple" approach: File headers and footers
  - → Configuration file needed, which specifies for which information to search (e.g. reducing scope to JPEG images)
  - → Produces therefore a lot of "garbage"!

File carving software: Scalpel



JPG
headers: 1,500
footers: 5,000, 6,500

MPG
headers: 9,000,000
footers: 26,000,000

GIF
headers: none
footers: none

Header/footer database

work queues

STARTSTOPCARVE
start: 1,500
stop: 6,500
name: small.jpg

STARTCARVE
start: 9,000,000
name: movie.mpg

CONTINUECARVE
name: movie.mpg

STOPCARVE
stop: 26,000,000
name: movie.mpg

NULL

chunk boundary
(10MB)

chunk boundary
(10MB)

small JPG

large MPG

38MB DISK IMAGE

Richard/Roussev: Scalpel: A Frugal ...

File carving software: Scalpel example configuration

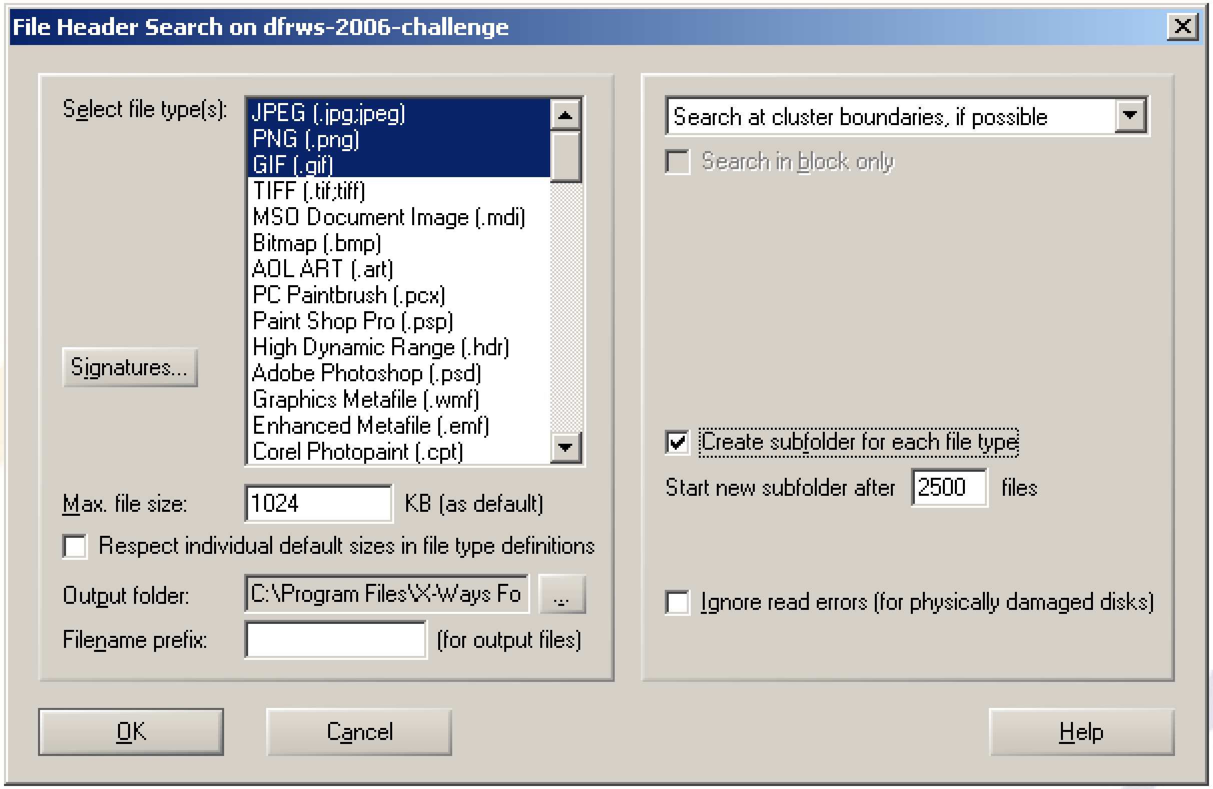
```
1     2         3              4                      5           6
gif   y     5000000     \x47\x49\x46\x38\x37\x61      \x00\x3b
jpg   y     200000000   \xff\xd8\xff\xe0\x00\x10      \xff\xd9
png   y     20000000    \x50\x4e\x47?           \xff\xfc\xfd\xfe
doc   y     10000000    \xd0\xcf\x11\xe0\xa1\xb1\x1a\xe1\x00\x00
                        \xd0\xcf\x11\xe0\xa1\xb1\x1a\xe1\x00\x00 NEXT
doc   y     10000000    \xd0\xcf\x11\xe0\xa1\xb1
pst   y     500000000   \x21\x42\x4e\xa5\x6f\xb5\xa6
htm   n     50000       <html                   </html>
pdf   y     5000000     %PDF                    %EOF\x0d      REVERSE
zip   y     10000000    PK\x03\x04              \x3c\xac
```

- 1: File extension; 2: Case sensitivity of header/footer
- 3: Maximum file size in bytes; 4: Header bytes
- 5: Footer bytes (optional); 6: Footer mode (optional)
  - → NEXT → Header + all data up to and excluding the footer
  - → REVERSE → Header + all data up to last occurrence of footer within maximum file size

File carving software: X-Ways Forensic

- "File recovery by type"
  - → Requires files to be not fragmented at all
    - » Uses no optimizations → Just plain start to end/maximum size!
  - → Alignment of file start can be specified
    - » Cluster: Only possibility for files in a "good" file system
    - » Sector: Find remnants of previous file systems/partitions
    - » Byte: When no alignment is possible
      - – Backup files, embedded objects (image within text documents)
      - – Increases the number of false positives significantly
  - → Signatures are stored in an Excel file
    - » Description, extension, header, offset (of header from file start), footer, default size (override of the manually set size in the UI)
      - – Header/footer are regular expressions (GREP)
      - – Custom extensions to the list are possible
    - » Original size of jpg, gif, png, bmp, tiff, psd, cdr, avi, wav, zip, MS Word/Excel/PowerPoint, rtf, pdf, and html is extracted from file
    - » Footer is only searched up to the maximum file size

File carving software: X-Ways Forensic



File Header Search on dfrws-2006-challenge

Select file type(s):

- JPEG (.jpg;jpeg)
- PNG (.png)
- GIF (.gif)
- TIFF (.tif;tiff)
- MSO Document Image (.mdi)
- Bitmap (.bmp)
- AOL ART (.art)
- PC Paintbrush (.pcx)
- Paint Shop Pro (.psp)
- High Dynamic Range (.hdr)
- Adobe Photoshop (.psd)
- Graphics Metafile (.wmf)
- Enhanced Metafile (.emf)
- Corel Photopaint (.cpt)

Max. file size: 1024 KB (as default)

□ Respect individual default sizes in file type definitions

Output folder: C:\Program Files\X-Ways Fo

Filename prefix: (for output files)

Search at cluster boundaries, if possible

□ Search in block only

☑ Create subfolder for each file type

Start new subfolder after 2500 files

□ Ignore read errors (for physically damaged disks)

→ File types should be recovered separately

» So a different maximum size can be specified!

- Manual recovery possible in addition

→ Identifying sectors + saving and concatenating them

Reducing the space requirements: CarvFS

- With huge hard disks, carving becomes more difficult
  - → Many carved files are very large, as they extend to the maximum size: the footer (no longer/did never) exists!
  - → Copying file content takes a long time
- Solution: CarvFS
  - → Virtual file system on top of FUSE (Linux userland file system)
  - → Mounting an image as a new file system
  - → Files created do not exist separately at all: They only refer to certain positions within the image!
    - » They are really only "symbolic links"
    - » Many and overlapping files → No size on disk required at all!
- Writing is not supported, only reading
- Metadata can be supplied in an additional XML file
  - → Depends on the image used, raw has none, EWF/AFF has!

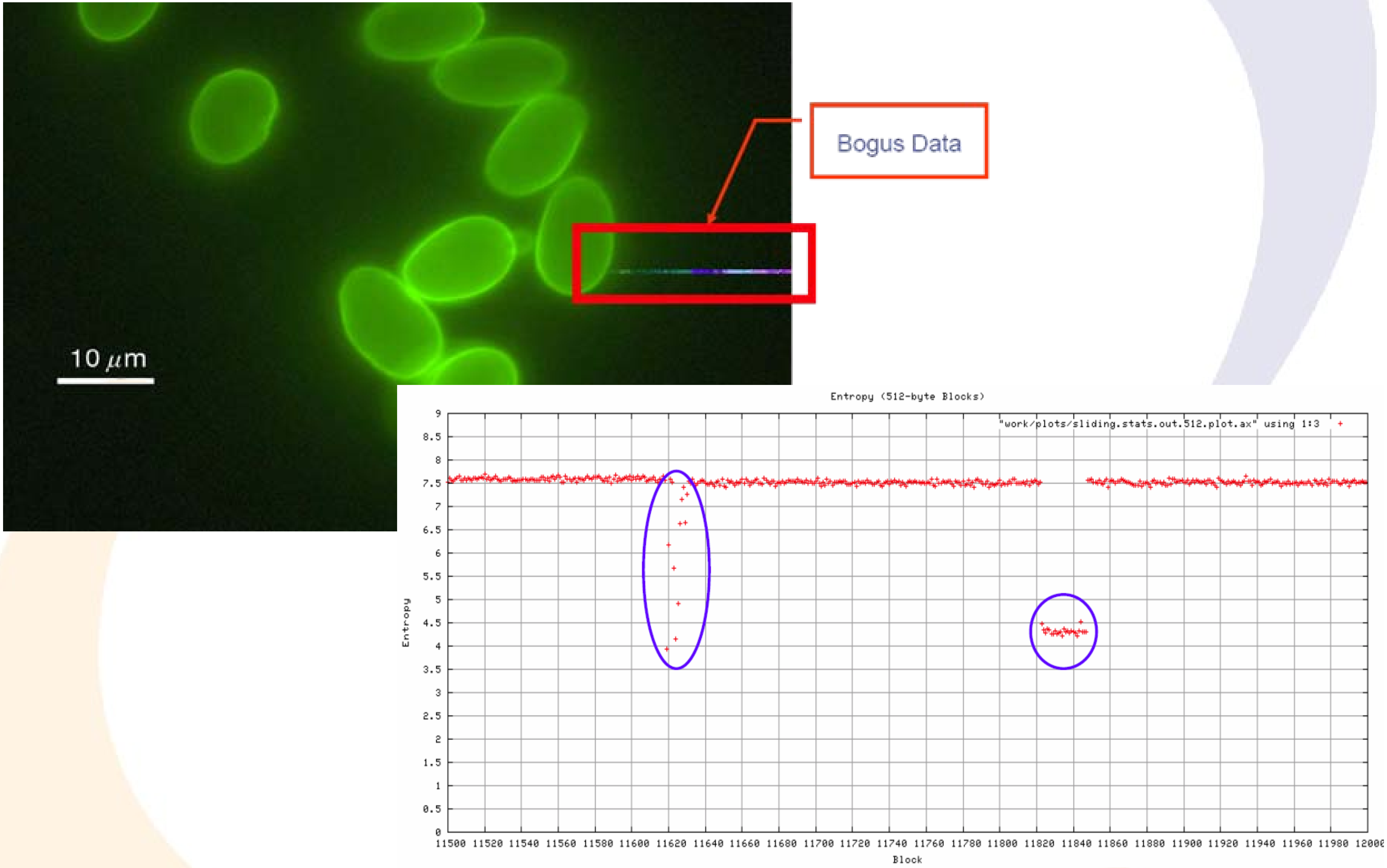
Reducing the space requirements: CarvFS

- The information on the position within the image is encoded into the name of the file
  - → Consists of several fragments
    - » Each fragment is specified by <offset>":"<size>
  - → Fragments are separated by "_"
- Note: You can open ANY file in CarvFS, even if it does not exist, but conforms to the filename specification!
  - → Example: "strings CarvFS/0:512.crv" will search the first 512 image bytes for any text strings contained and print them
- Note: CarvFS is not compatible with other forensic tools!
  - → Tools must be adapted to be able to work with CarvFS, or they will just copy out the data to a "normal" position!
    - » No "automatic" creation of the links when writing to a file!
      - – As writing is not supported at all!
    - » The tool must provide only the "coordinates" where to find a file

Sliding entropy

- Entropy = Measure of randomness
  - → Large changes in entropy will usually indicate that this sector does belong to a different file
    - » Attention: Embedded files; but these are seldom on sector boundaries → Requires a sliding window smaller than a sector!
- Average = Average value of bytes
- Sliding entropy is used to classify different data types
  - → Entropy 0-8 (8=pure random)
    - » 4-6: Text and HTML blocks
    - » 7-8: Zip and JPEG blocks
- Additional measure: ctype
  - → Counts the percentage of certain character classes
    - » Alpha(-numeric), ASCII, lower, printable, punctuation, space, …
- Not easy to fully automate
  - → Changes in entropy are best identified visually

Sliding entropy
Example



Images: Smith/Monroe/Bair: Digital Forensics File Carving Advances

Cross-references within files

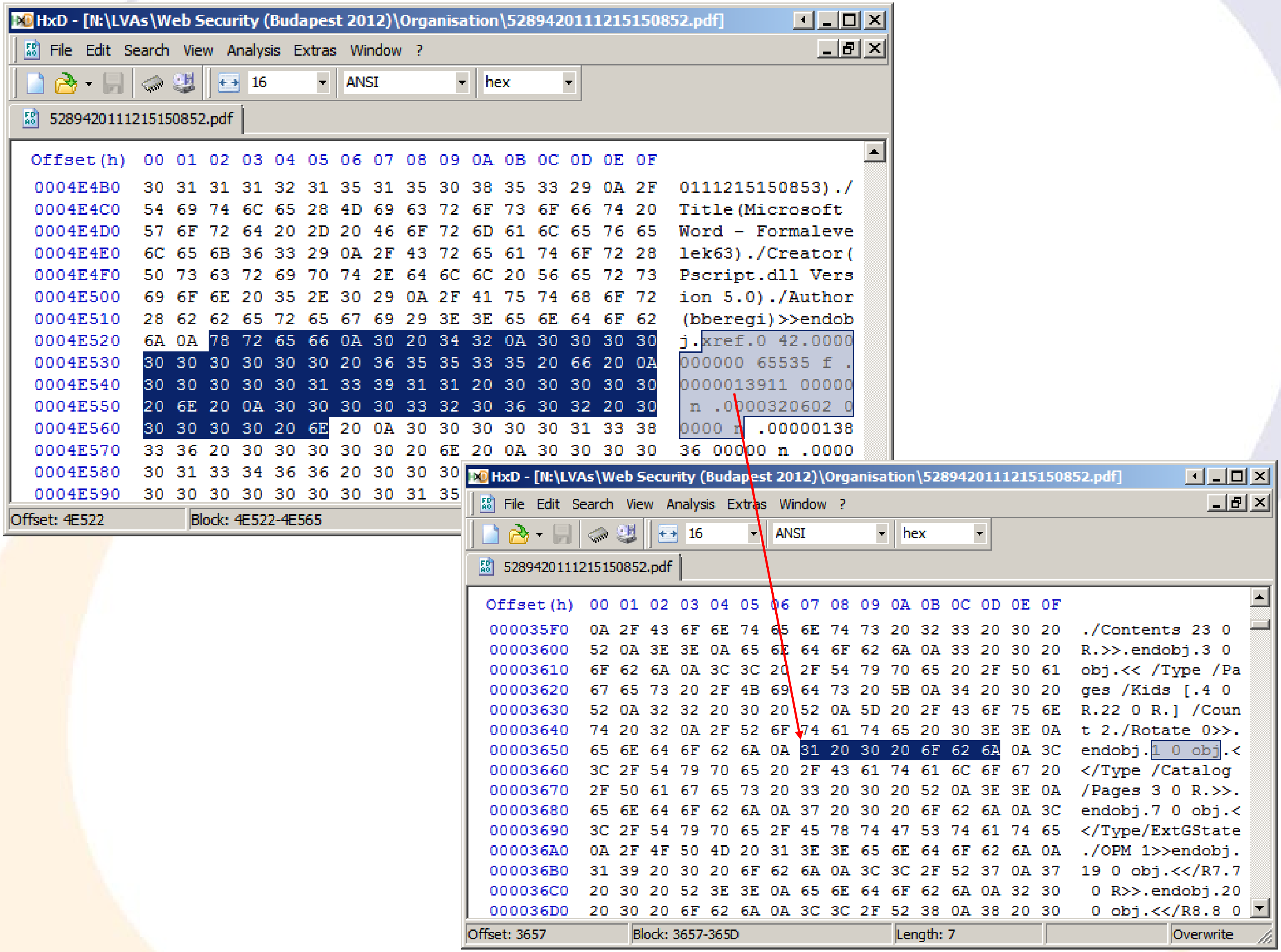
- Cross references to other parts of a file give information on where certain data must be present
  - → Example:
    - » Offset 104: Next “internal record” starts at offset 3570
    - » Offset 3570: Begin marker of internal record expected, or this area must look like such a record
      - – Note: All clusters which do not conform to this can obviously not belong at this position!
  - → Problem: Empty space may remain where no cross-references exist (just continue or leave out)
  - → Requires knowledge of cluster size (normally not a problem!)
- Detailed knowledge of the file format needed
  - → Must contain cross-references
  - → Targets of references must be identifiable as such

Cross-references within files: PDF example

- PDFs consist of objects
  - → <obj. number> <generation> obj
- Cross references do exist
  - → xref
    0 42
    0000000000 65535 f
    0000013911 00000 n
    0000320602 00000 n
- Conclusion: At offset 13911 (=0x3657) must be object number 1: “1 ?????? obj”
- So we search for all clusters where at offset 1623 (13911%4096) this character sequence exists
  - → Which are probably VERY few!

Source: Schuster, Carving – Grundlagen und neue Techniken

Cross-references within files: PDF example


HxD - [N:\LVAs\Web Security (Budapest 2012)\Organisation\5289420111215150852.pdf]
File Edit Search View Analysis Extras Window ?
16
ANSI
hex
5289420111215150852.pdf

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

0004E4B0   30 31 31 31 32 31 35 31 35 30 38 35 33 29 0A 2F  0111215150853)./
0004E4C0   54 69 74 6C 65 28 4D 69 63 72 6F 73 6F 66 74 20  Title(Microsoft 
0004E4D0   57 6F 72 64 20 2D 20 46 6F 72 6D 61 6C 65 76 65  Word - Formaleve
0004E4E0   6C 65 6B 36 33 29 0A 2F 43 72 65 61 74 6F 72 28  lek63)./Creator(
0004E4F0   50 73 63 72 69 70 74 2E 64 6C 6C 20 56 65 72 73  Pscript.dll Vers
0004E500   69 6F 6E 20 35 2E 30 29 0A 2F 41 75 74 68 6F 72  ion 5.0)./Author
0004E510   28 62 62 65 72 65 67 69 29 3E 3E 65 6E 64 6F 62  (bberegi)>>endob
0004E520   6A 0A 78 72 65 66 0A 30 20 34 32 0A 30 30 30 30  j.xref.0 42.0000
0004E530   30 30 30 30 30 30 20 36 35 35 33 35 20 66 20 0A  000000 65535 f .
0004E540   30 30 30 30 30 31 33 39 31 31 20 30 30 30 30 30  0000013911 00000
0004E550   20 6E 20 0A 30 30 30 30 33 32 30 36 30 32 20 30   n .0000320602 0
0004E560   30 30 30 30 20 6E 20 0A 30 30 30 30 30 31 33 38  0000 n .00000138
0004E570   33 36 20 30 30 30 30 30 20 6E 20 0A 30 30 30 30  36 00000 n .0000
0004E580   30 31 33 34 36 36 20 30 30 30
0004E590   30 30 30 30 30 30 30 30 31 35
```

Offset: 4E522 | Block: 4E522-4E565

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

000035F0   0A 2F 43 6F 6E 74 65 6E 74 73 20 32 33 20 30 20  ./Contents 23 0 
00003600   52 0A 3E 3E 0A 65 6E 64 6F 62 6A 0A 33 20 30 20  R.>>.endobj.3 0 
00003610   6F 62 6A 0A 3C 3C 20 2F 54 79 70 65 20 2F 50 61  obj.<< /Type /Pa
00003620   67 65 73 20 2F 4B 69 64 73 20 5B 0A 34 20 30 20  ges /Kids [.4 0 
00003630   52 0A 32 32 20 30 20 52 0A 5D 20 2F 43 6F 75 6E  R.22 0 R.] /Coun
00003640   74 20 32 0A 2F 52 6F 74 61 74 65 20 30 3E 3E 0A  t 2./Rotate 0>>.
00003650   65 6E 64 6F 62 6A 0A 31 20 30 20 6F 62 6A 0A 3C  endobj.1 0 obj.<
00003660   3C 2F 54 79 70 65 20 2F 43 61 74 61 6C 6F 67 20  </Type /Catalog 
00003670   2F 50 61 67 65 73 20 33 20 30 20 52 0A 3E 3E 0A  /Pages 3 0 R.>>.
00003680   65 6E 64 6F 62 6A 0A 37 20 30 20 6F 62 6A 0A 3C  endobj.7 0 obj.<
00003690   3C 2F 54 79 70 65 2F 45 78 74 47 53 74 61 74 65  </Type/ExtGState
000036A0   0A 2F 4F 50 4D 20 31 3E 3E 65 6E 64 6F 62 6A 0A  ./OPM 1>>endobj.
000036B0   31 39 20 30 20 6F 62 6A 0A 3C 3C 2F 52 37 0A 37  19 0 obj.<</R7.7
000036C0   20 30 20 52 3E 3E 0A 65 6E 64 6F 62 6A 0A 32 30   0 R>>.endobj.20
000036D0   20 30 20 6F 62 6A 0A 3C 3C 2F 52 38 0A 38 20 30   0 obj.<</R8.8 0
```

Offset: 3657 | Block: 3657-365D | Length: 7 | Overwrite

Semantics-based file carving

- Research project:
  - → Carving of "text" files based on their semantic content
    - » txt, html, java, c, … Everything for direct human reading
- Basic idea: Searching in several stages
  - → Identify all potential sectors
    - » Recognizing text, programs, etc. is possible with a high certainty
      - – Programming languages: Idioms, reserved words
      - – Natural languages: Check for spaces, letters, non-letters
  - → Detect language of the file
    - » Programming language or natural language
      - – Natural language: Using stop word lists is fast and easy!
      - – Programming language: Reserved words, regular expressions
        - » Example C: include "[a-zA-Z\-_0-9]*.h"\n
  - → Hierarchy check: Nesting for programming languages (indentation) and html files (unopened/unclosed tags)
    - » Allows excluding certain sequences

Semantics-based file carving

  - → Boundary check: Is the first/last word a complete word or only a fragment?
    - » Uses WordNet or custom lists
- Sorting fragments based on Google searches
  - → Build a combination of a small part of the end of a sector and a small part of the start of a sector
  - → Submit it as a fixed-string search to Google
  - → Count the results
  - → Which occurs most often (or is found at all) is the most likely combination of sectors
- Based on the idea, that texts and programs consist of common fragments which can be found in the Internet
  - → Will not work for binary files:
    - » These cannot be found by Google as easily
    - » They are much rarer and often the exact file would be required!

New difficulties and helpers

- Problems:
  - → Compressing file systems
    - » Cluster boundaries don’t match content boundaries any more
    - » Statistics and inspection of individual clusters may not work any more (unless each can be decompressed separately!)
  - → Large file systems: See above!
  - → File formats are complex and often undocumented
- Advantages:
  - → Fewer file formats in widespread use; reuse of existing ones
    - » E.g. SQLite databases for configurations etc.
  - → Huge disks and often used as storage only (e.g. media files: Copied there and read, but not modified in size)
    - » Less fragmentation
  - → More data: Often (!) a lot of evidence exists; we don’t have to find the **single** offending picture/important E-Mail

Conclusions

- File carving is still problematic: It takes a long time and the results are often suboptimal
  - → Large numbers of huge files, which are incomplete
- Fragmentation is not that common anymore, but still a problem even for modern file systems
  - → File carving must cope with out-of-order and missing sectors
  - → Especially problematic are files with a missing start
- Improvements possible and under development towards
  - → Requiring less memory: Verification also "in-place"
  - → Needing less IO: Fewer passes
  - → Specialisation: Working for a single file format very well
    - » Based on the specific structure, content, properties, …

Questions?

Thank you for your attention!

Literature

- Cohen, Michael: Advanced Carving techniques
  http://sandbox.dfrws.org/2007/cohen/Advanced_Carving.pdf
- Kloet, S. J. J: Measuring and Improving the Quality of File Carving Methods
  http://www.uitwisselplatform.nl/frs/download.php/461/thesis.pdf
- Carrier, Brian: Why Recovering a Deleted Ext3 File Is Difficult …
  http://www.linux.sys-con.com/read/117909.htm
- Wood, Carlo: HOWTO recover deleted files on an ext3 file system
  http://www.xs4all.nl/~carlo17/howto/undelete_ext3.html
- Richard, Golden G. III, Roussev, Vassil: Scalpel: A Frugal, High Performance File Carver
  http://dfrws.org/2005/proceedings/richard_scalpel.pdf

Literature

- LibCarvPath and CarvFS http://ocfa.sourceforge.net/libcarvpath/
- Smith, Jay, Monroe, Klayton, Bair, Andy: Digital Forensics File Carving Advances http://www.korelogic.com/Resources/Projects/dfrws_challenge_2006/DFRWS_2006_File_Carving_Challenge.pdf
- Anandabrata Pal, Nasir Memon, The Evolution of File Carving http://digital-assembly.com/technology/research/pubs/ieee-spm-2009.pdf