



Mag. iur. Dr. techn. Michael Sonntag

File carving

Security and Privacy
VSE Prag, 7 - 11.6.2010

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>

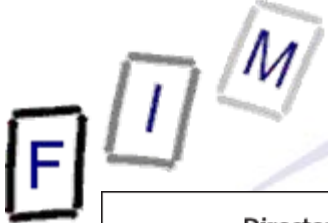


- What is file carving and why do it?
 - Deleting files in NTFS and EXT3
 - Main problems
- Simple file carving
- The file carving process
- File carving software
 - Scalpel
 - X-Ways Forensics
 - CarvFS
 - Sliding Entropy
- Semantics-based file carving

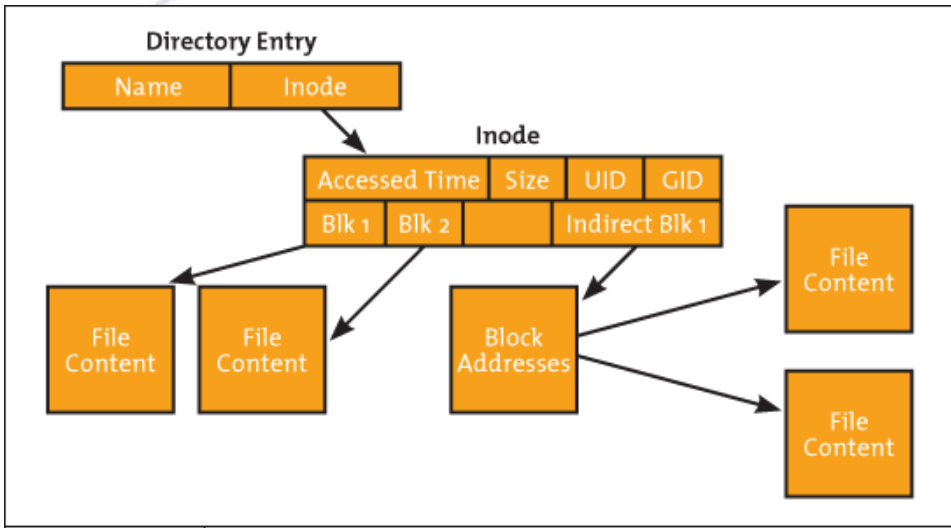


What is "file carving"?

- 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

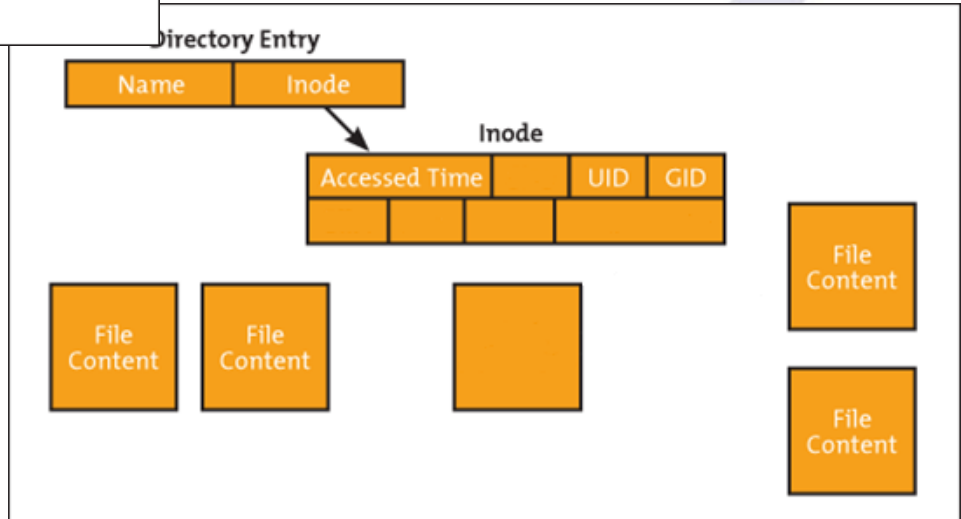


EXT3 delete



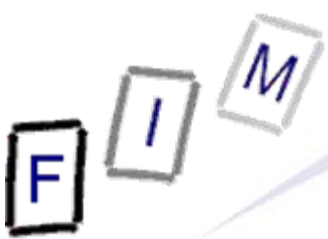
Before deletion
(file still exists)

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



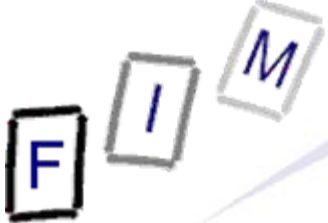
After deletion
(file removed)

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.



Main problems of file carving

- File carving has a time complexity of NP-complete
 - You must try all possible combinations of fragments/sectors
 - Optimizations are possible 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
 - ≥ 500 GB 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



Simple file carving

- Identify start & end of file and extract everything in between
 - Example: JPEG (Start = FFD8, End= FFD9)
- Works only for file types with a specific end marker
 - Will only find files with existing beginning (marker)
 - » First cluster lost → Gone!
 - If no end marker exists: Specify a maximum length
 - » Everything up to then will be extracted
 - Most programs will just ignore everything after "their" end
 - » Other option: Some file types have a length indicator at start
 - Requires parsing the file according to its internal structure



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 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)
- Usually produces large numbers of very large files
 - Very large as often no end marker exists
 - » Carved files contain same data/other carved files several times!
 - First 20 kB file will be carved for a length of 10 MB and therefore contain also the next ten/twenty/... 20 KB files!
 - Manually removing duplicates and erroneous results

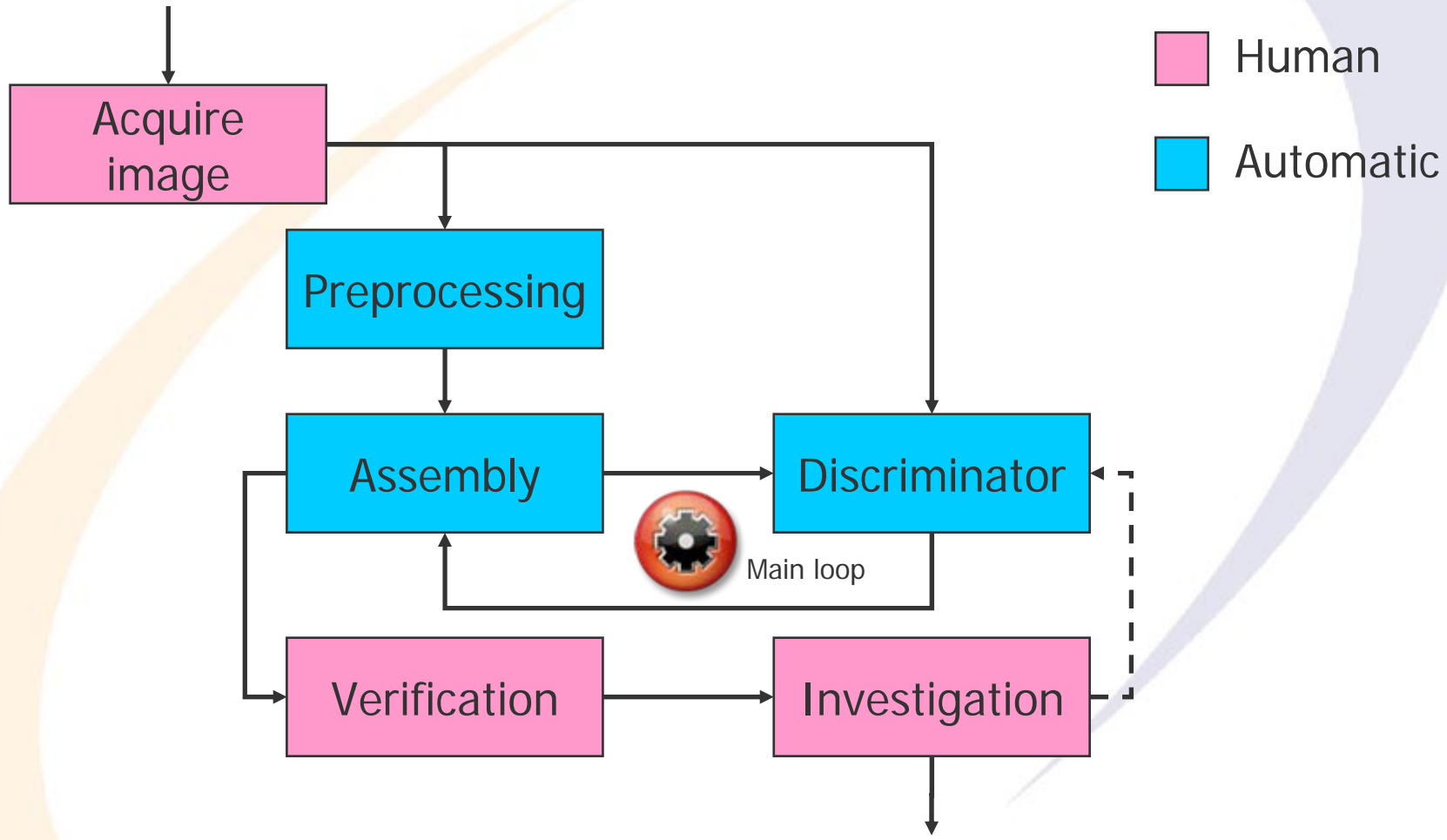


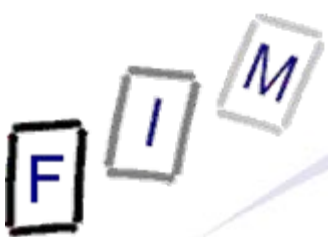
Detecting the end of a file

- 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!
- 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)



The file carving process: Overview





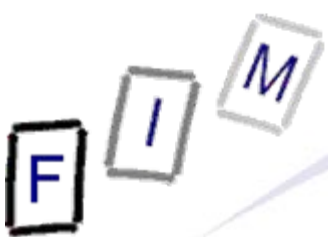
The file carving process: Description of human activities

- 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

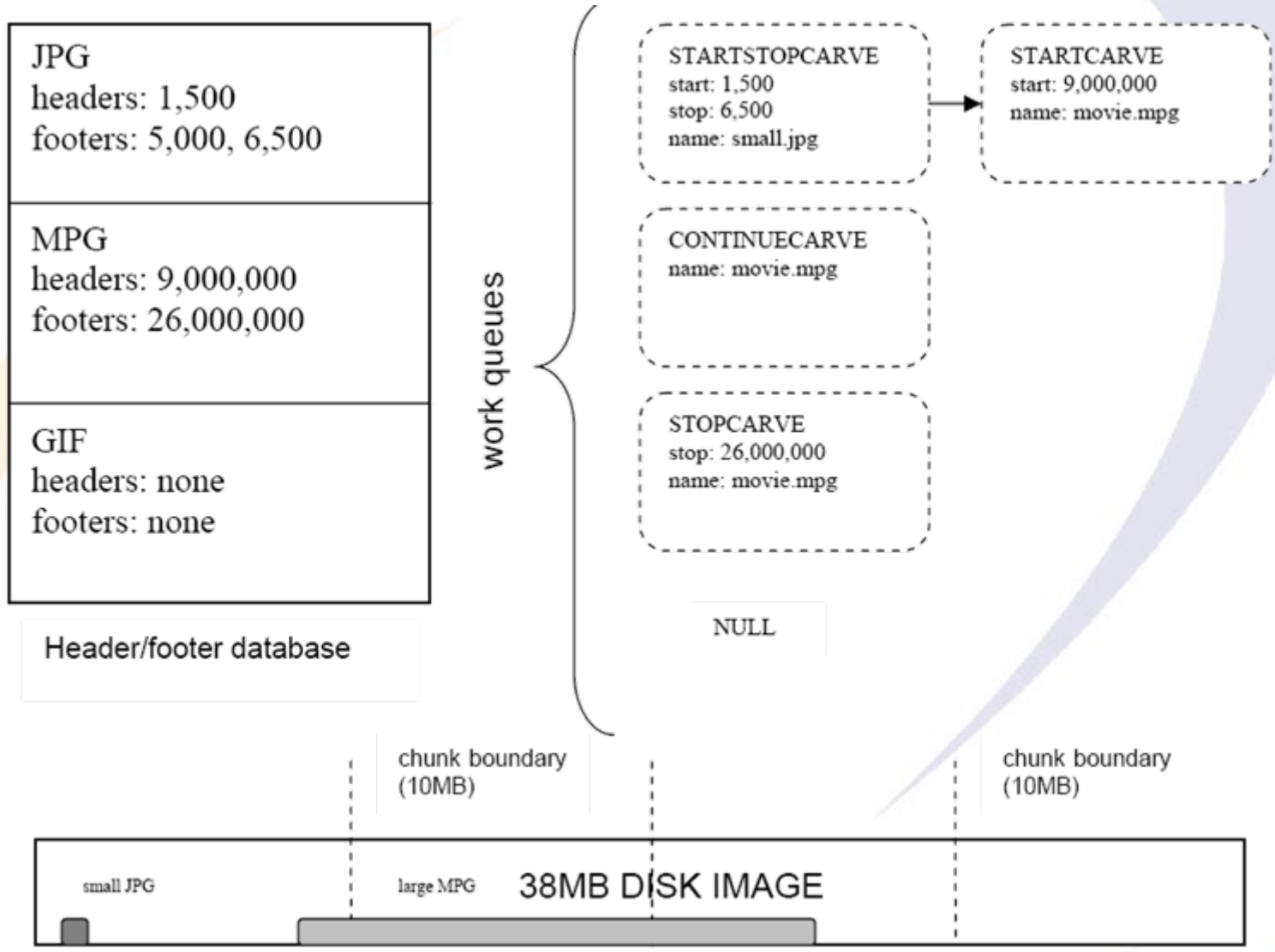
- Preprocessing: Extracting information about the file
 - Identify file type; identify start and end/length if possible
 - Select all sectors which potentially could be part of the file
- Assembly: Generate a potential version of the file
 - Decide which sectors to include
 - Concatenate these sectors 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



- 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

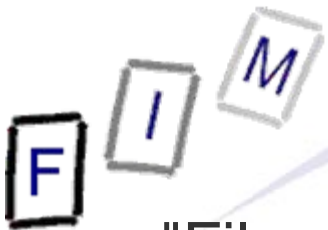




File carving software: Scalpel example configuration

| 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 | |

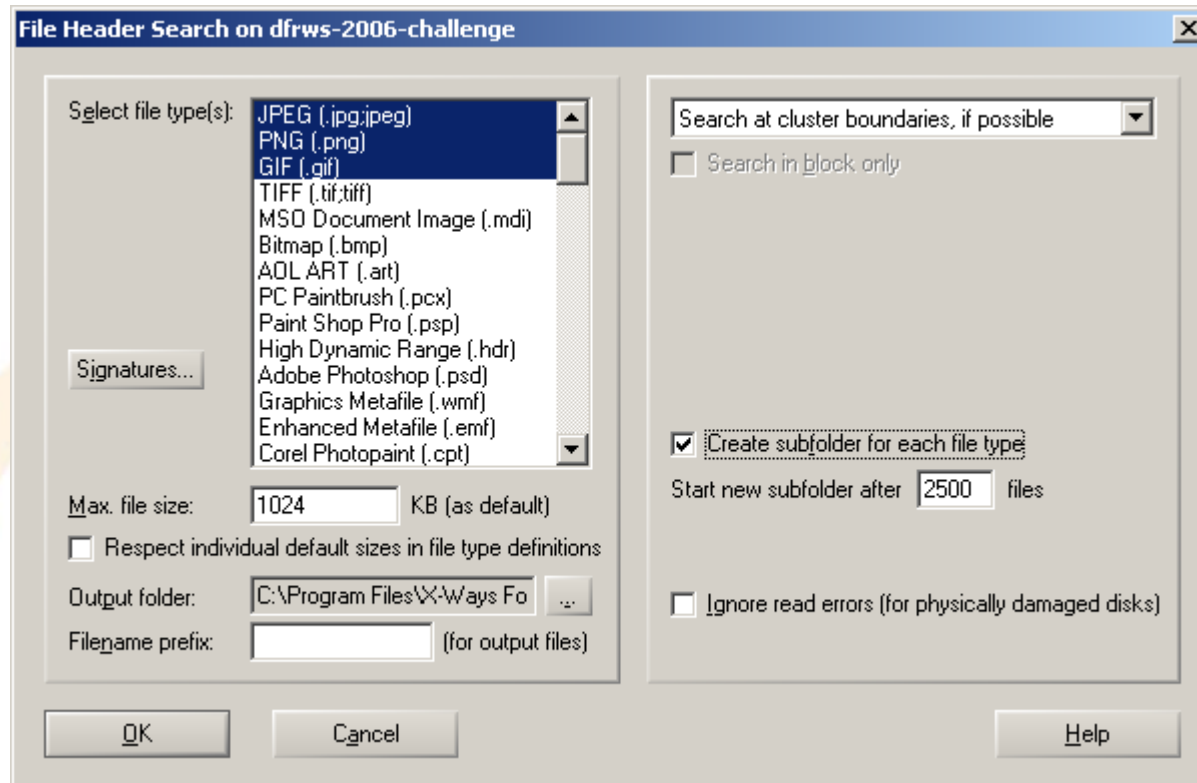
- 1: File extension; 2: Case sensitivity of header/footer
- 3: Maximum file size in bytes; 4: Header bytes
- 5: Footer bytes (optional); 6: Footer mode (optional)
 - NEXT → Header + all data up to and excluding the footer
 - REVERSE → Header + all data up to last occurrence of footer within maximum file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



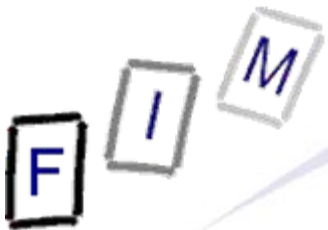
File carving software: X-Ways Forensic

- "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 (GREG)
 - 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

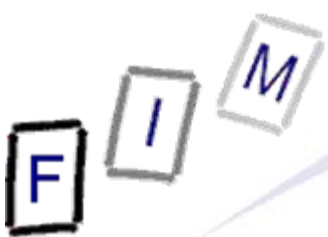


- File types should be recovered separately
 - » So a different maximum size can be specified!
- Manual recovery possible in addition
 - Identifying sectors +saving and concatenating them



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!



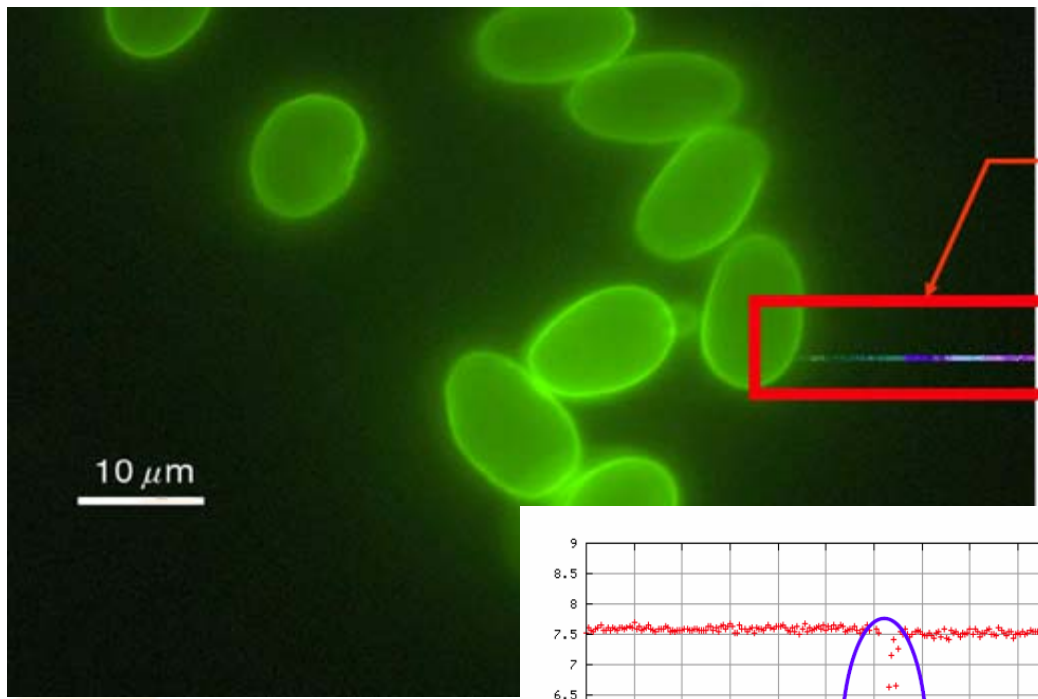
Reducing the space requirements: CarvFS

- 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

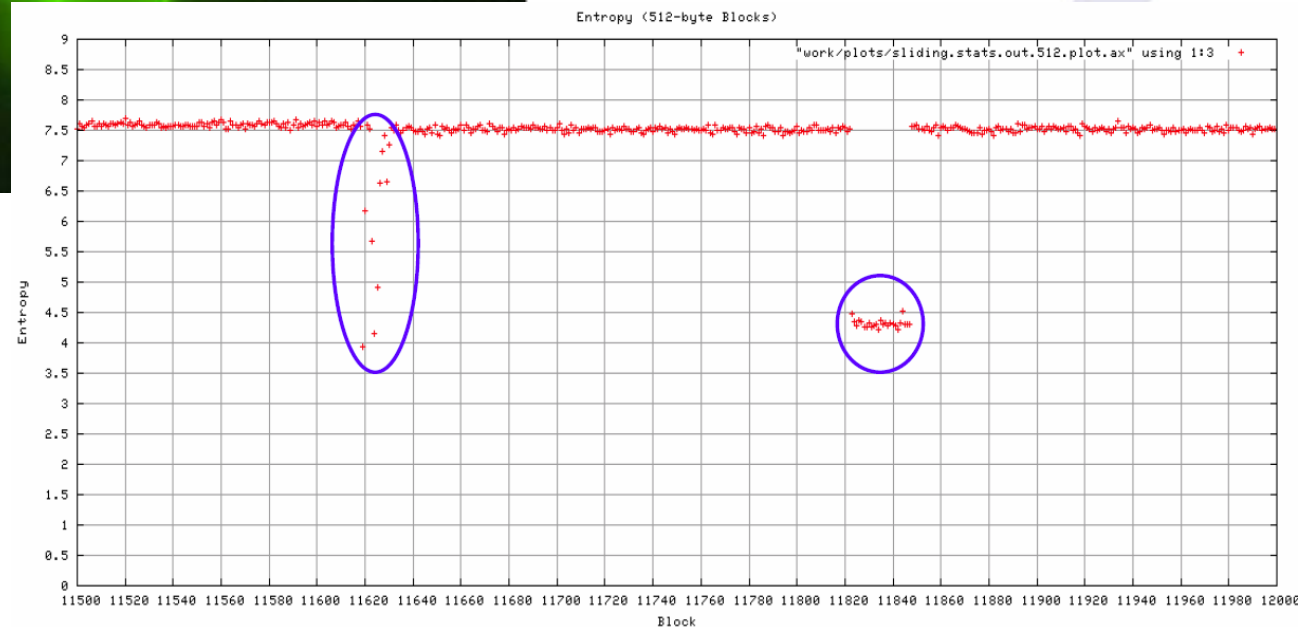


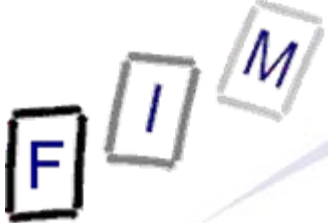
Sliding entropy

- 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



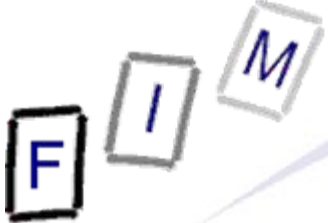
Bogus Data





Semantics-based file carving

- Current 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!



- 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, ...

F I M

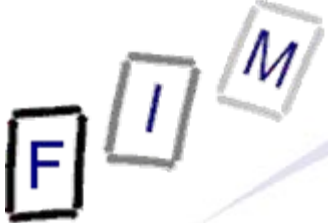
Questions?

Thank you for your attention!



Conclusions

- 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



Conclusions

- 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