



# Forensics II

## File carving

What is file carving and why do it?

Simple file carving

The file carving process

File carving software

Advanced file carving

# File carving fundamentals

- File Carving definition
  - Carving is a general term for extracting structured data (files) out of raw data, based on format specific characteristics present in the structured data
- File carving is a powerful technique because it can
  - Identify and recover files of interest from raw, deleted or damaged file systems, memory, or swap space data
  - Assist in recovering files and data that may not be accounted for by the operating system and file system
    - File metadata is no longer available
  - Assist in simple data recovery
    - [http://www.forensicswiki.org/wiki/Tools:Data\\_Recovery](http://www.forensicswiki.org/wiki/Tools:Data_Recovery)

# File carving details 1

- Identify and recover files based on analysis of file formats
- Header-footer or header-maximum file size carving
  - Many file types have well-known values or magic numbers in the first bytes of the file header and in the last bytes of the file
  - Identify specific types of file headers and/or footers and carve out blocks between these two boundaries
  - Stop carving after a user-specified or set limit has been reached
  - Unfortunately not all file types have a standard footer signature so determining the EOF can be difficult – thus the need for limits
- File structure based carving (metadata in the file as well)
- Content based carving (semantics etc.)

# File Structure and Content-based Based Carving

- File Structure Based Carving
  - This technique uses the internal layout of a file
  - Elements are header, footer, identifier strings and size information etc.
  - Known carvers which use this technique are Scalpel and PhotoRec
- Content-based Carving
  - Content structure
    - Loose structure (MBOX, HTML, XML)
  - Content characteristics
    - Character count
    - Text/Language recognition
    - White and Black listing of data (filter)
    - Statistical attributes
    - Information entropy



# File carving details 2

- Many carving programs have an option to only look at or near sector or cluster boundaries where headers are found
  - File start is always at a sector boundary, but end is not
- However, searching the entire input can find files that have been embedded into other files, such as JPEGs being embedded into OLE documents etc.
  - This may be considered an advantage or a disadvantage, depending on the circumstances
- The majority of file carving programs will only recover files that are contiguous on the media (in other words: files that are not fragmented)
- Files may be incomplete
  - Start, end, middle sectors may have been reused

# FAT delete

FAT File System Structures 1

## Root Directory Entries

File name      Starting block

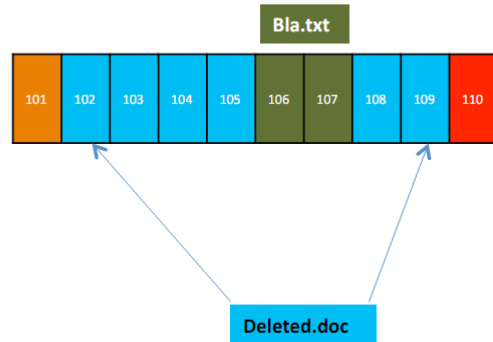
Deleted.doc	102
Bla.txt	106
Archive.pst	110

## FAT

Block      Next  
Index      Block

101	Free
102	103
103	104
104	105
105	108
106	107
107	EOF
108	109
109	EOF
110	111

Media Data Block Area 1



FAT File System Structures 2

## Root Directory Entries

File name      Starting block

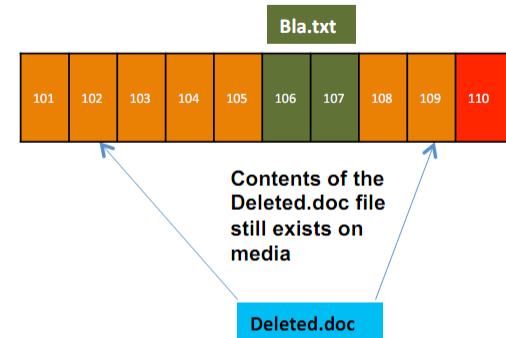
_eleted.doc	102
Bla.txt	106
Archive.pst	110

## FAT

Block      Next  
Index      Block

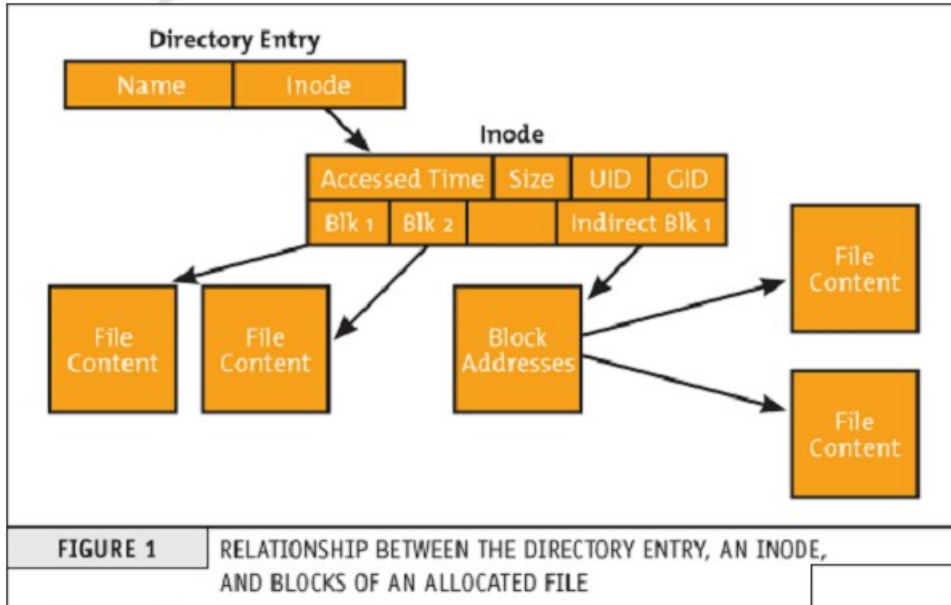
101	Free
102	Free
103	Free
104	Free
105	Free
106	107
107	EOF
108	Free
109	Free
110	111

Media Data Block Area 2



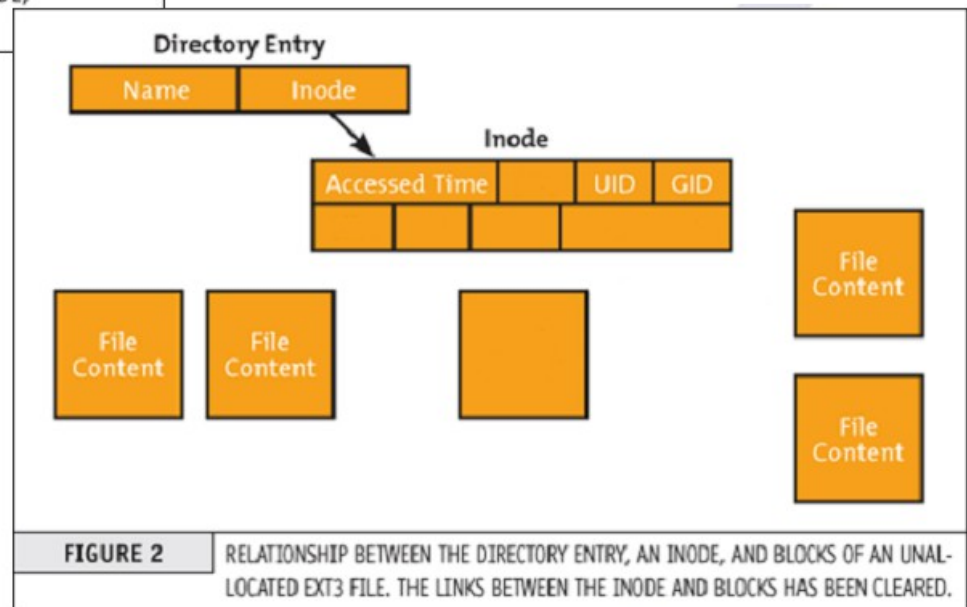
# Ext3 delete

Before deletion



After deletion

Block pointers are zeroed out in the inode



# Simple file carving example

- JPEG files start with 0xFFD8 and end with 0xFFD9
- To recover a JPEG file
  - Find locations of its header and footer
  - Carve out everything between those two endpoints
  - If no end marker exists: Specify a maximum length
    - File size (length) or location of next JPEG file (0xFFD8) etc.
- Works well only for non-fragmented files
  - Improvements: Exclude all sectors in use by other **real (allocated) files** and previously extracted files

Hexdump of sample.jpg

```
ff d8 ff e0 00 10 4a 46 49 46 00 01 01 01 00 50 |.....JFIF....P|
... Data ...
28 a2 80 3f ff d9 |(..?...|
```



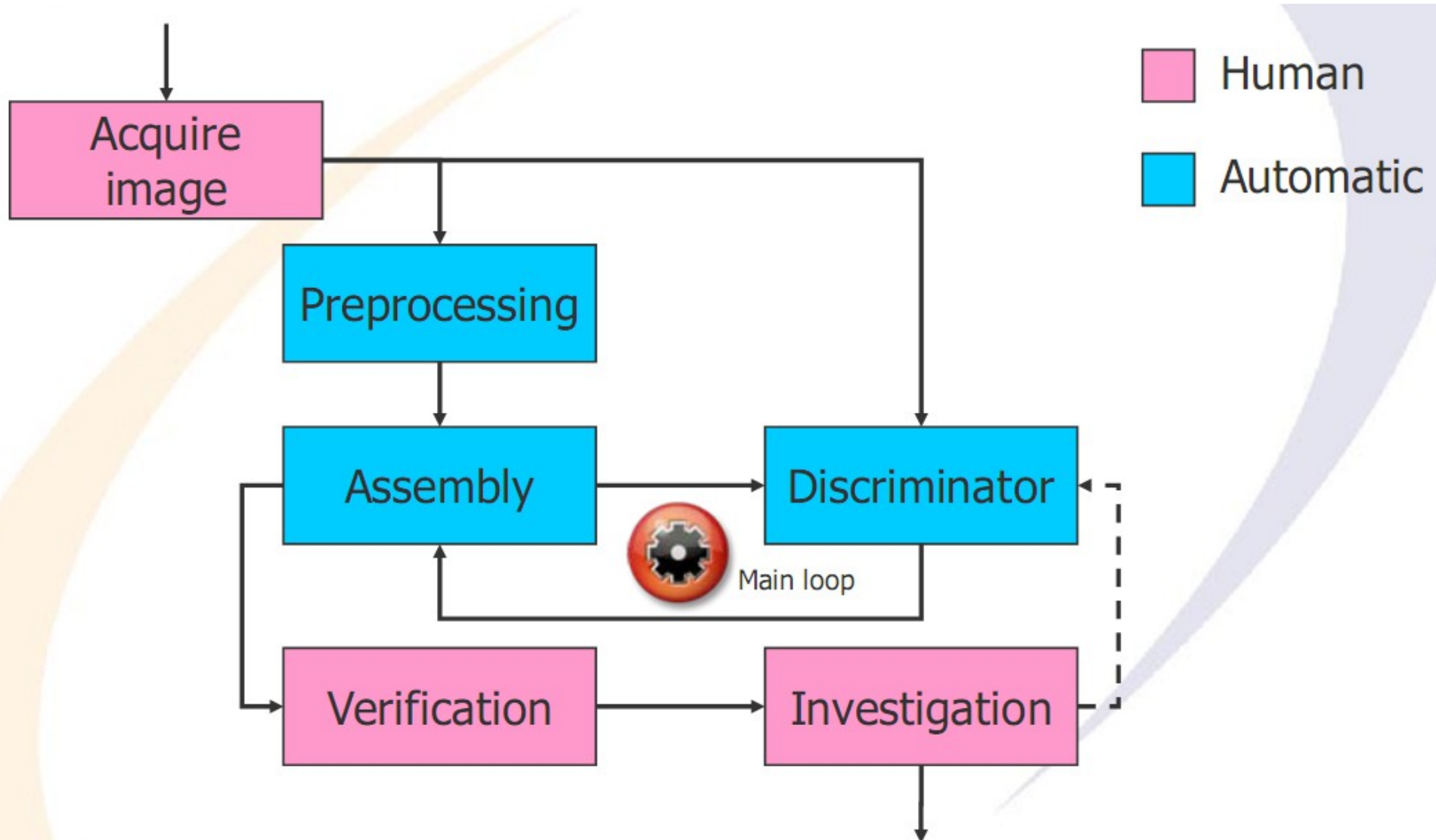
# Main problems of file carving

- File carving has a time complexity of NP-complete
  - Cannot be **solved** and **verified** quickly (time consuming)
  - You must try all possible combinations of fragments/sectors
  - Optimizations are possible to reduce this somewhat
    - Depending on the file system used, file types in question or additional information, e.g. content redundancy etc.
- Many unreadable invalid and partial results
  - May result in more data as output than input
- Quality of the tools are unclear
- File systems become larger (TB disks are inexpensive)
  - Huge numbers of files and huge numbers of fragments!
  - But, individual files usually lightly fragmented and files are usually stored sequentially by the OS on media

# Detecting the end of a file

- If a specific signature exists > Problem solved?
  - Note: Some files may have header signatures or the 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 which is problematic with proprietary software
- Header signature of a new file exist in the carved candidate
  - Embedded files can be troublesome in this respect!
  - Possible premature termination > Be 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 etc.)

# The file carving process 1



# The file carving process 2

- **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" (viewed) or does it make "sense"?
  - Where in the file is the erroneous position?
  - Some parts belonging at an absolute position?
  - Usually based on known file viewers or printers
    - Difficulties: No specific error reporting, internal error recovery
    - Is additionally problematic if the file was corrupt before carving start

# 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 (typical 10 MB)
  - **Second:** Extract all files by working the queues for each chunk
    - To avoid memory-to-memory copies
- Based on: File structure based carving
  - Configuration file needed, which specifies for which information to search (e.g. reducing scope to JPEG images)
  - Produces therefore a lot of "garbage"!

First  
pass

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

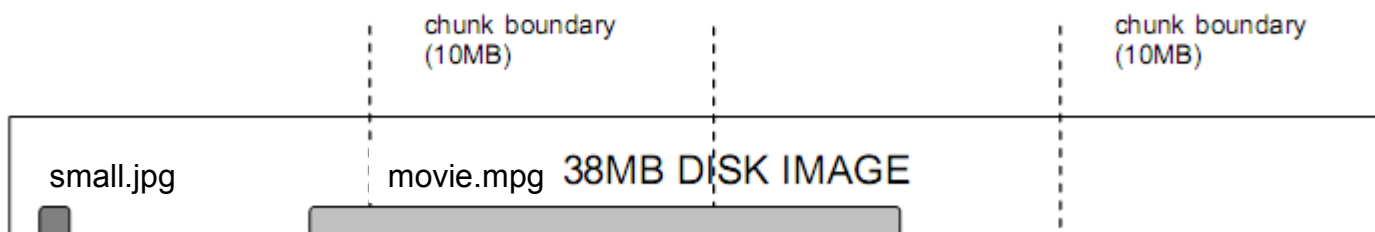
CONTINUECARVE  
name: movie.mpg

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

NULL

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

Second  
pass



**Figure 1.** Work queues in Scalpel. Each 10MB chunk of an image file is assigned a work queue, which contains a sequence of records that define carving operations for that chunk during Scalpel's second sequential pass over the image file.

# 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	200000000	\x50\x4e\x47?	\xff\xfc\xfd\xfe	
doc	y	100000000	\xd0\xcf\x11\xe0\xa1\xb1\x1a\xe1\x00\x00		
			\xd0\xcf\x11\xe0\xa1\xb1\x1a\xe1\x00\x00		NEXT
doc	y	100000000	\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	100000000	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

# Carving software: X-Way Forensics (WinHex)

- File recovery by type
  - Requires files to be not fragmented at all
    - Uses no optimizations, just plain start to end/maximum size!
  - Look for file headers (file system knowledge)
    - Everywhere, free clusters only, allocated space only
  - Search (alignment) of file start can be specified
    - Cluster: Only possible searching for files in a "good" file system
    - Sector: Find remnants of files in previous file systems/partitions
    - Byte: When no alignment is possible (searching from raw data)
      - Backup files, embedded objects (images 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)
      - Custom extensions to the list are possible
    - Footer is only searched up to the maximum file size





# Carving software: X-Way Forensics (WinHex)

- Tools > Disk Tools > File Recovery by Type
  - Mark the file types you want to carve

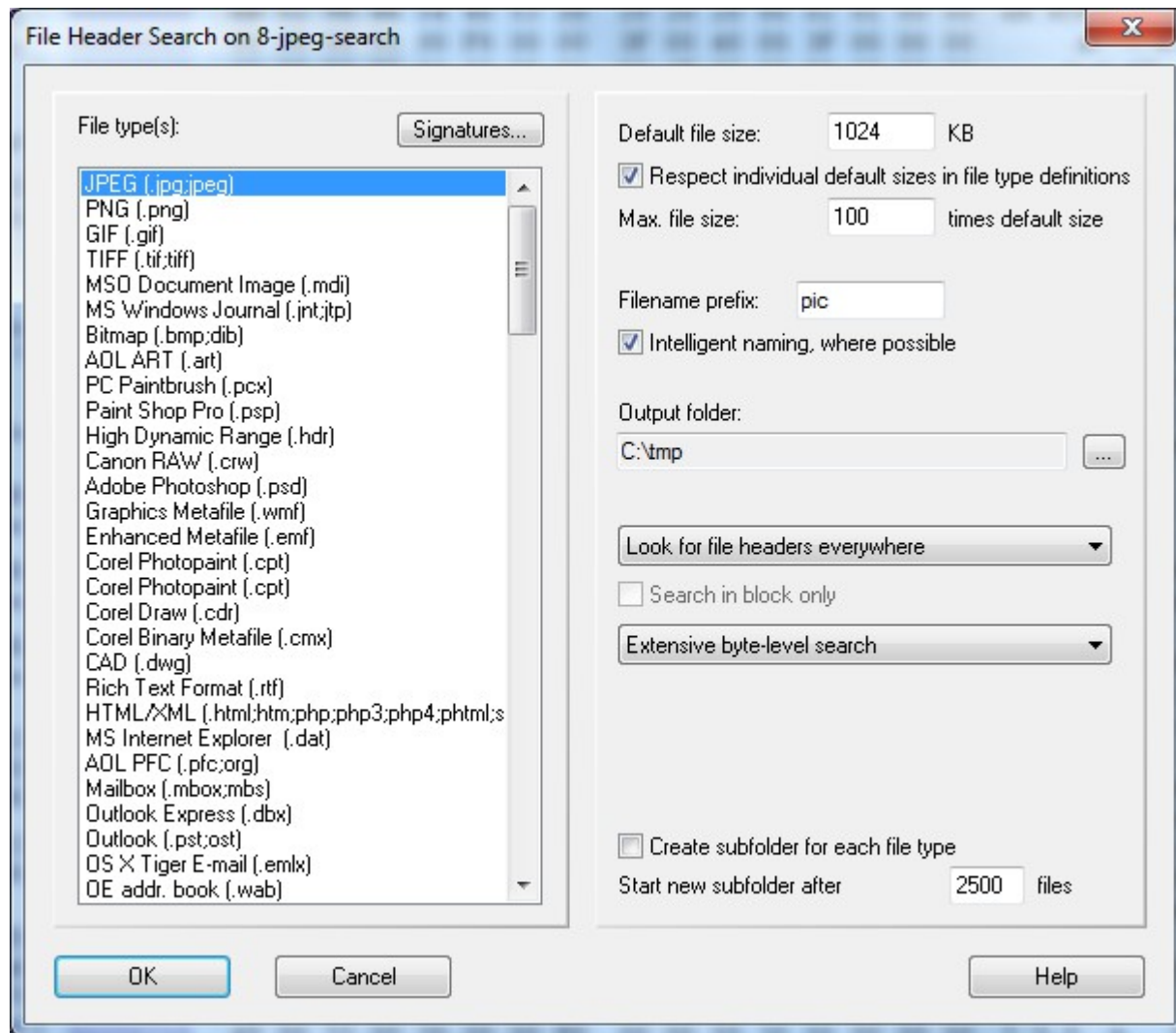
- XWF

- Underrated SW
- Leading supplier of forensic SW in Europe!

- Help

- manual.pdf
- XWFQuickStart.pdf by Brett Shavers
- Quick Guides
- Templates
- YouTube movie

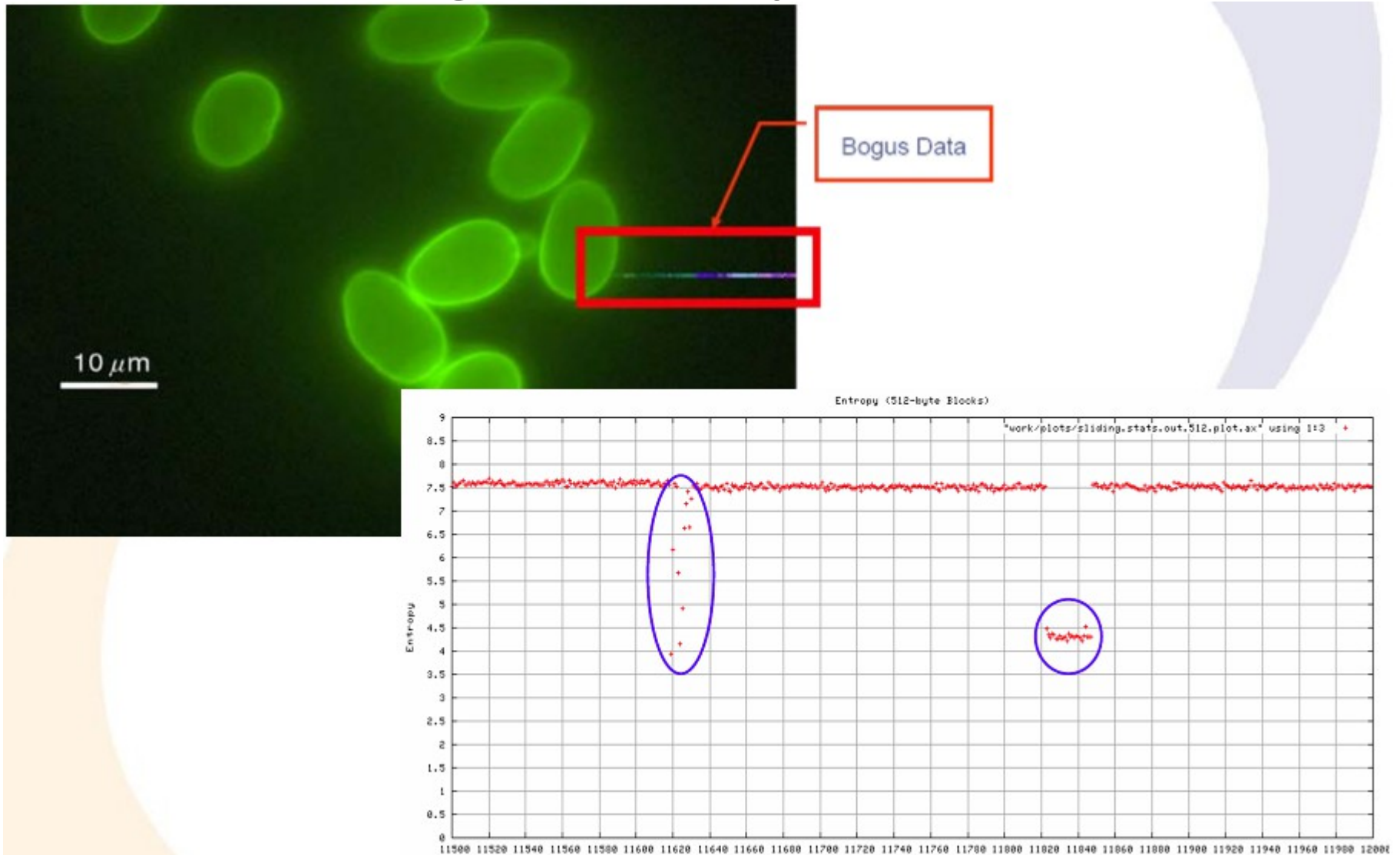
Data recovery [part2]



# 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; since these are seldom on sector boundaries
    - > Requires a sliding window smaller than a sector!
- Average = Measure average value of bytes (in combo with stddev)
- 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: character type
  - 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



# Semantics-based file carving 1

- 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 (for loop etc.), 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 sequence

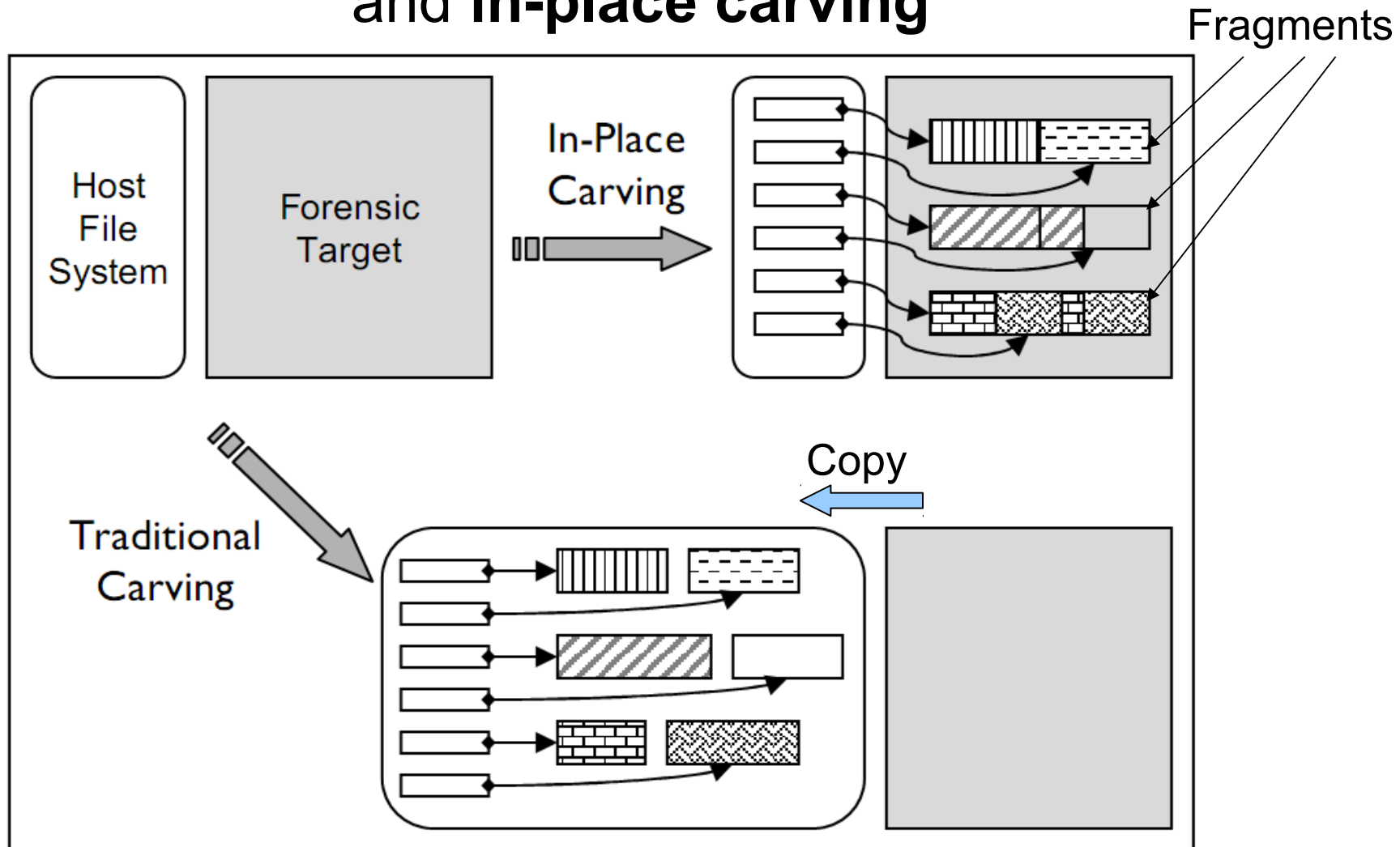
# Semantics-based file carving 2

- Boundary check: Is the first/last word a complete word or only a fragment?
  - Uses WordNet (<http://wordnet.princeton.edu/>) or custom lists
- Sorting fragments based on Google searches
  - Build a combination of a small part of the end of a sector and 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 easily
    - They are much rarer and often the exact file would be required

# 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/at all) exists!
  - Copying file content takes a long time
- Solution: CarvFS
  - Virtual file system on top of FUSE (Linux userland file system)
  - Mounting a forensic image as a new read only file system
  - Files created do not exist separately at all: They only refer to certain positions within the image
    - They are only symbolic links
    - Many and overlapping files > No size on disk required at all!
- Metadata can be supplied in an additional XML file
  - Depends on the image used, raw (dd) has none, EWF/AFF has
- LibCarvPath do the actual fragment mapping and annotation

# Reducing the space requirements: Conceptual differences between traditional and **in-place carving**



# 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 byte 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
    - The tool must provide only the "coordinates" where to find a file
- Work has started to include LibCarvPath support into Photorec



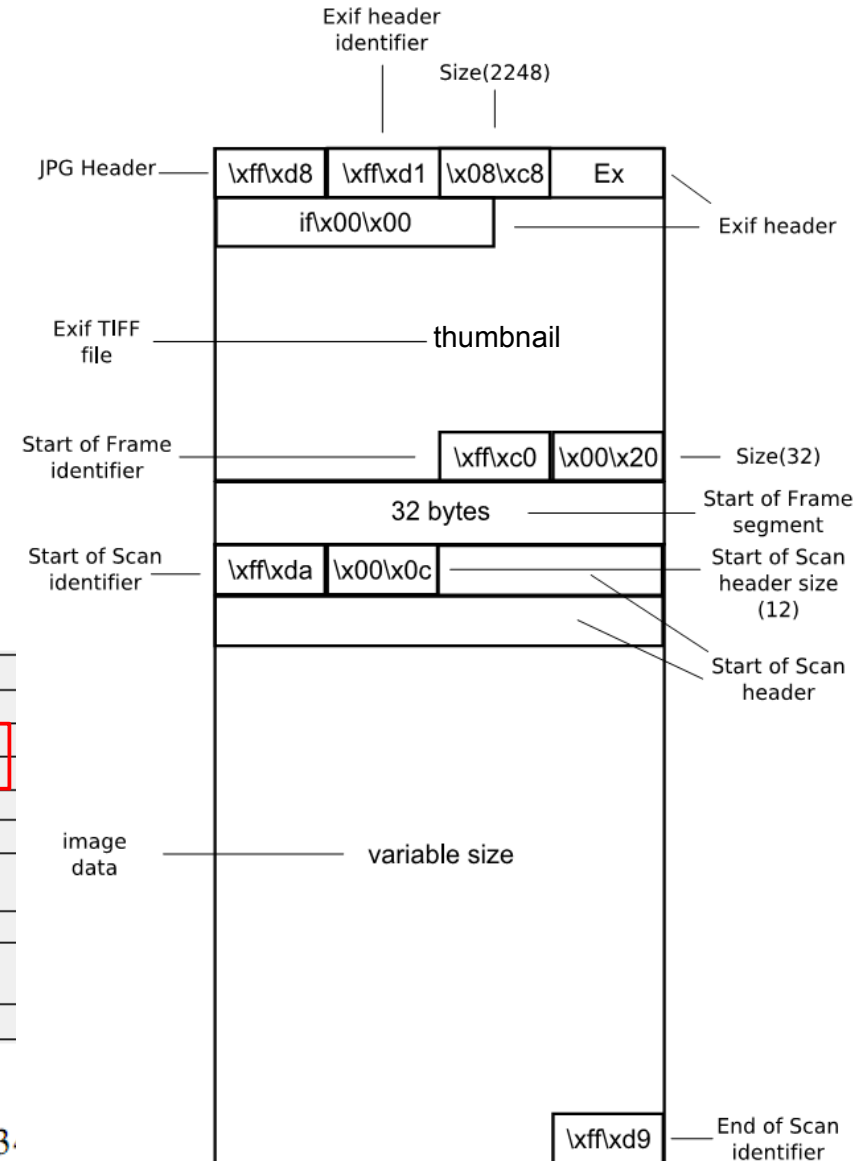
# JPG file structure (Exif)

- APP1 appears at the beginning of an Exif file, contains the thumbnail and cannot be more than 64 Kb in size
  - The Exif JPEG thumbnail may use a JFIF header and footer
  - JFIF standard (non-EXIF) use APP0
- Exif files allow for an optional APP2 which contains FlashPix extensions

Hex	Symbol	Marker Name	Description
FFD8	SOI	Start of Image	Start of compressed data
FFE1	APP1	Application Segment 1	Exif attribute information
FFE2	APP2	Application Segment 2	Exif extended data
FFDB	DQT	Define Quantization Table	Quantization table definition
FFC4	DHT	Define Huffman Table	Huffman table definition
FFDD	DRI	Define Restart Interoperability	Restart Interoperability definition
FFC0	SOF	Start of Frame	Parameters relating to frame
FFDA	SOS	Start of Scan	Parameters relating to components
FFD9	EOI	End of Image	End of compressed data

TABLE I

JPEG MARKER CODE ASSIGNMENTS USED IN EXIF (JEITA CP-3)



# Digital Still Camera Forensics

- SSDDFJ\_V1\_1\_Cohen.pdf
- Some programs do not update the thumbnail  
<http://blogs.23.nu/disLEXia/2004/12/antville-5751/>
- Exif extended data – APP2 (FlashPix)

- This is a jungle and hard to get a grip on!
- Project suggestion?

TABLE V  
METADATA TAGS USED IN EXIF (JEITA CP-3451)

GPS IFD
Tags Relating to GPS
GPSVersionID
GPSLatitudeRef
GPSLatitude
GPSLongitudeRef
GPSLongitude
GPSAltitude
GPSTimeStamp
GPSSatellites
GPSStatus
GPSMeasureMode
GPSDOP
GPSSpeedRef
GPSTrackRef
GPSTrackRef
GPSImgDirectionRef
GPSImgDirectionRef
GPSMapDatum
GPSTestLatitudeRef
GPSTestLatitude
GPSTestLongitudeRef
GPSTestLongitude
GPSTestBearingRef
GPSTestBearing
GPSTestDistanceRef
GPSTestDistanceRef
GPSProcessingMethod
GPSAreaInformation
GPSTestStamp
GPSTestDifferential

TABLE IV  
METADATA TAGS USED IN EXIF (JEITA CP-3451)

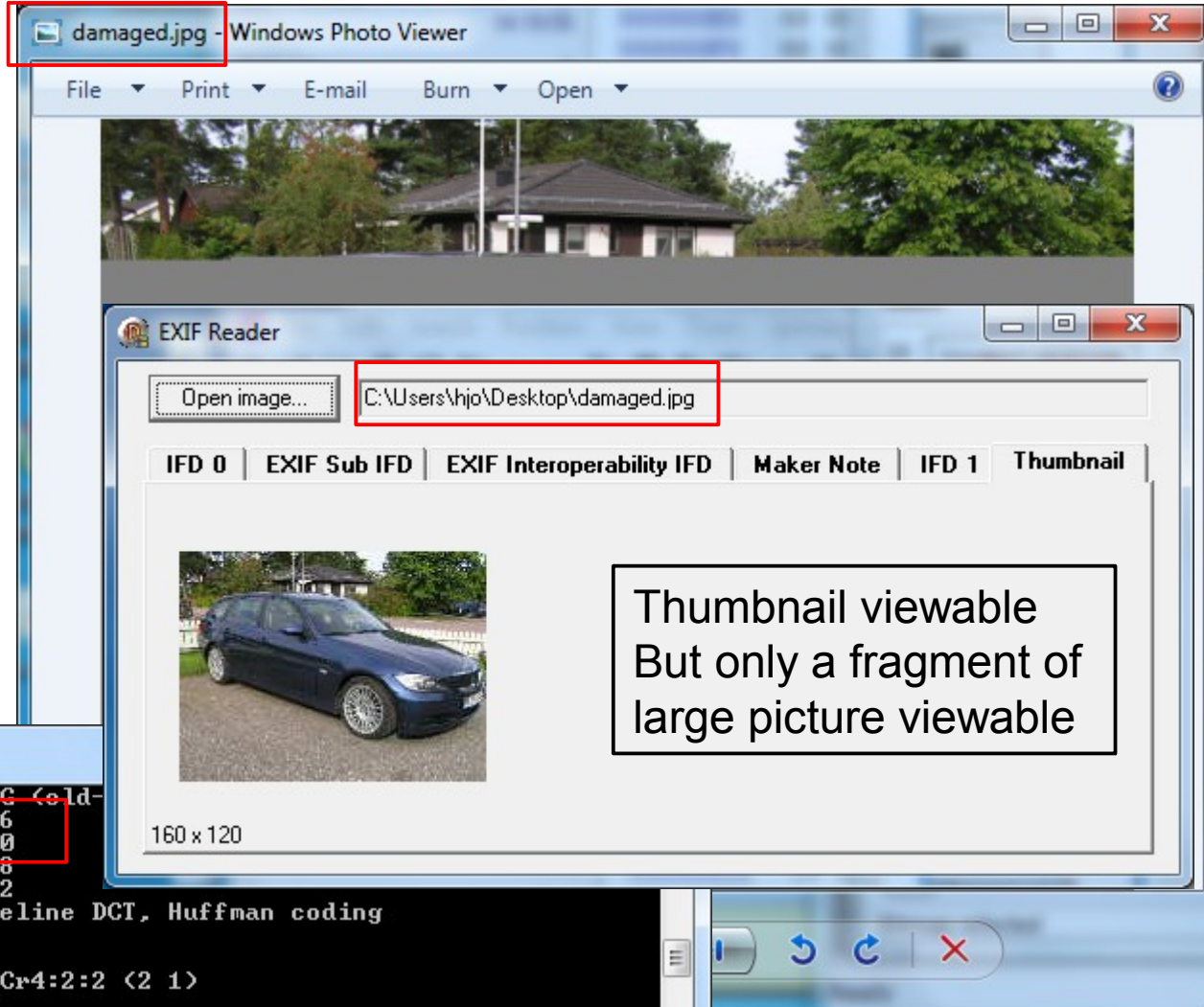
TIFF
Tags relating to image data structure
ImageWidth
ImageLength
BitsPerSample
Compression
PhotometricInterpretation
Orientation
SamplesPerPixel
PlanarConfiguration
YCbCrSubSampling
YCbCrPositioning
XResolution
YResolution
Tags relating to recording offset
StripOffsets
RowsPerStrip
JPEGInterchangeFormat
JPEGInterchangeFormatLength
Tags relating to image data characteristics
TransferFunction
WhitePoint
PrimaryChromaticities
YCbCrCoefficients
ReferenceBlackWhite
Other Tags
DateTime
ImageDescription
Make
Model
Software
Artist
Copyright

TABLE III  
METADATA TAGS USED IN EXIF (JEITA CP-3451)

EXIF IFD
Tags Relating to Version
ExifVersion
FlashpixVersion
Tags Relating to Image Data Characteristics
ColorSpace
Tags Relating to Image Configuration
ComponentsConfiguration
CompressedBitsPerPixel
PixelXDimension
PixelYDimension
Tags Relating to User Information
MakerNote
UserComment
Tags Relating to Related File Information
RelatedSoundFile
Tags Relating to Date and Time
DateTimeOriginal
DateTimeDigitized
SubSecTime
SubSecTimeOriginal
SubSecTimeDigitized
Tags Relating to Picture-Taking Conditions
ExposureTime
FNumber
ExposureProgram
SpectralSensitivity
ISOSpeedRatings
OEFC
ShutterSpeedValue
ApertureValue
BrightnessValue
ExposureBiasValue
MaxApertureValue
SubjectDistance
MeteringMode
LightSource
Flash
FocalLength
SubjectArea
FlashEnergy
SpatialFrequencyResponse
FocalPlaneXResolution
FocalPlaneYResolution
FocalPlaneResolutionUnit
SubjectLocation
ExposureIndex
SensingMethod
FileSource
SceneType
CFAPattern
CustomRendered
ExposureMode
WhiteBalance
DigitalZoomRatio
FocalLengthIn35mmFilm
SceneCaptureType
GainControl
Contrast
Saturation
Sharpness
DeviceSettingDescription
SubjectDistanceRange
Other Tags
ImageUniqueID

# Exif JPEG Thumbnail

## exiftool



```
C:\utils\exiftool\exiftool(-k).exe
Compression : JPEG (old)
Thumbnail Offset : 4096
Thumbnail Length : 5130
Image Width : 2288
Image Height : 1712
Encoding Process : Baseline DCT, Huffman coding
Bits Per Sample : 8
Color Components : 3
Y Cb Cr Sub Sampling : YCbCr4:2:2 (2 1)
Aperture : 4.0
Image Size : 2288x1712
Scale Factor To 35 mm Equivalent : 6.0
Shutter Speed : 1/4000
Thumbnail Image : <Binary data 5130 bytes, use -b option to extract>
Circle Of Confusion : 0.085 mm
Field Of View : 50.6 deg
Focal Length : 6.3 mm (35 mm equivalent: 38.1 mm)
Hyperfocal Distance : 2.00 m
Light Value : 13.3
-- press any key --
```

# Conclusions

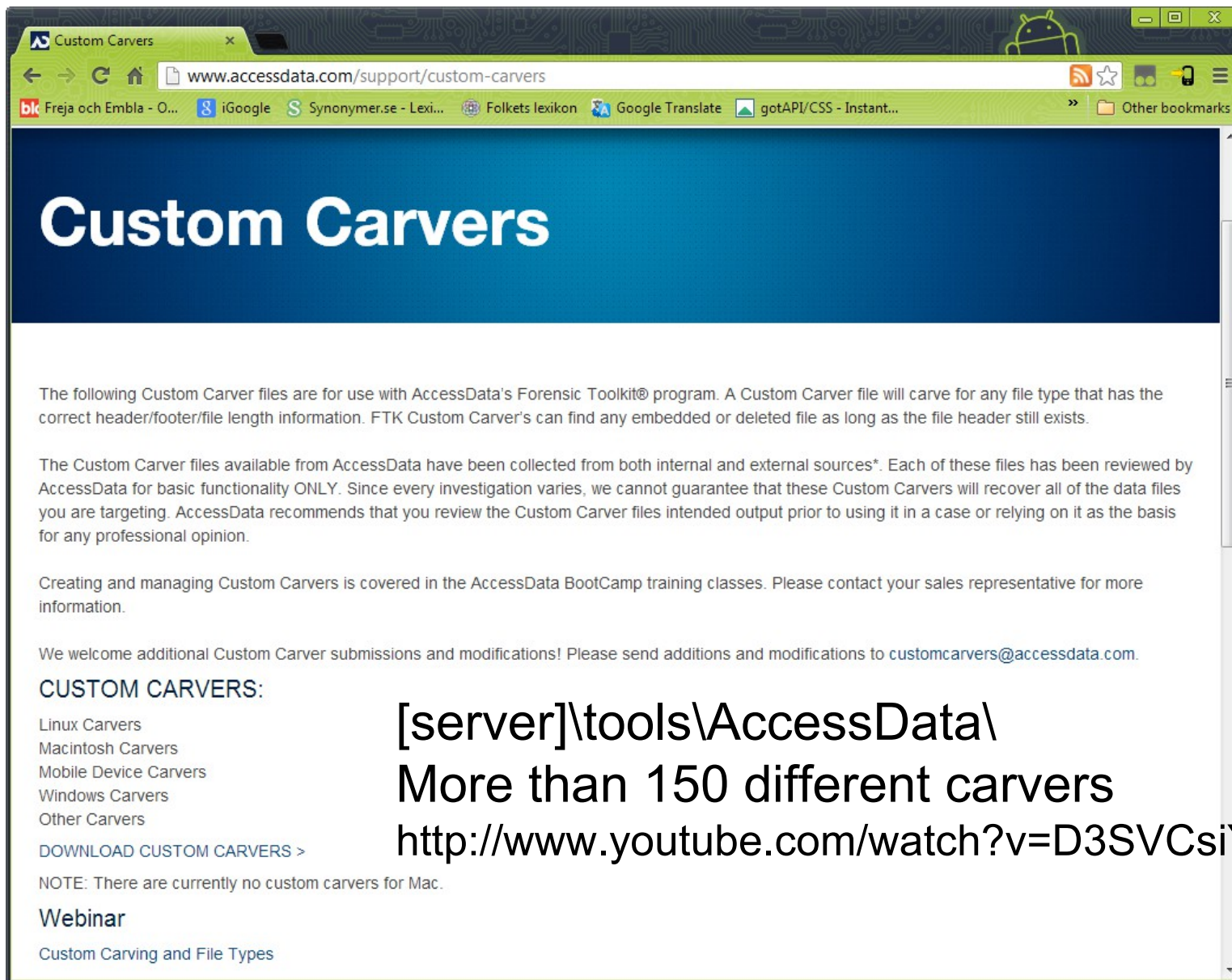
- File carving is problematic: It takes a long time and the results are often suboptimal
  - Large numbers of huge files, which are incomplete
  - View and classificate files takes forever...
- Fragmentation is not that common anymore, but still a problem even for modern file systems
  - Fragmentation may be worse when users upgrade to SSD drives
  - 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, ...

# Ease review of data carving

- Keven Murphy at: <http://www.citadelsystems.net> have in his blog post a solution: <http://blogs.sans.org/computer-forensics/2009/12/09/making-reviewing-files-from-data-carving-easier-documents/comment-page-1/#comment-6856>
- Fully automatic script solution
- Images processor
  - Stego test, exif data and nudity score
- Documents processor
  - Creates thumbnails of documents
- More processor modules on the way...
- Perl scripts
- Plugins support
- Looks very promising!



# FTK >= 3.2 Custom Carvers



The screenshot shows a web browser window with the title "Custom Carvers" and the URL "www.accessdata.com/support/custom-carvers". The page has a blue header with the title "Custom Carvers" in white. Below the header, there is a paragraph of text explaining that Custom Carver files are for use with AccessData's Forensic Toolkit program. Another paragraph states that the Custom Carver files available from AccessData have been collected from both internal and external sources. A third paragraph mentions that creating and managing Custom Carvers is covered in the AccessData BootCamp training classes. A fourth paragraph welcomes additional Custom Carver submissions and modifications. Below this, there is a section titled "CUSTOM CARVERS:" with a list of carvers: Linux Carvers, Macintosh Carvers, Mobile Device Carvers, Windows Carvers, and Other Carvers. There is a link "DOWNLOAD CUSTOM CARVERS >". A note states: "NOTE: There are currently no custom carvers for Mac." Below the note is a section titled "Webinar" with a link "Custom Carving and File Types".

Custom Carvers

The following Custom Carver files are for use with AccessData's Forensic Toolkit® program. A Custom Carver file will carve for any file type that has the correct header/footer/file length information. FTK Custom Carver's can find any embedded or deleted file as long as the file header still exists.

The Custom Carver files available from AccessData have been collected from both internal and external sources\*. Each of these files has been reviewed by AccessData for basic functionality ONLY. Since every investigation varies, we cannot guarantee that these Custom Carvers will recover all of the data files you are targeting. AccessData recommends that you review the Custom Carver files intended output prior to using it in a case or relying on it as the basis for any professional opinion.

Creating and managing Custom Carvers is covered in the AccessData BootCamp training classes. Please contact your sales representative for more information.

We welcome additional Custom Carver submissions and modifications! Please send additions and modifications to [customcarvers@accessdata.com](mailto:customcarvers@accessdata.com).

**CUSTOM CARVERS:**

- Linux Carvers
- Macintosh Carvers
- Mobile Device Carvers
- Windows Carvers
- Other Carvers

[DOWNLOAD CUSTOM CARVERS >](#)

NOTE: There are currently no custom carvers for Mac.

**Webinar**

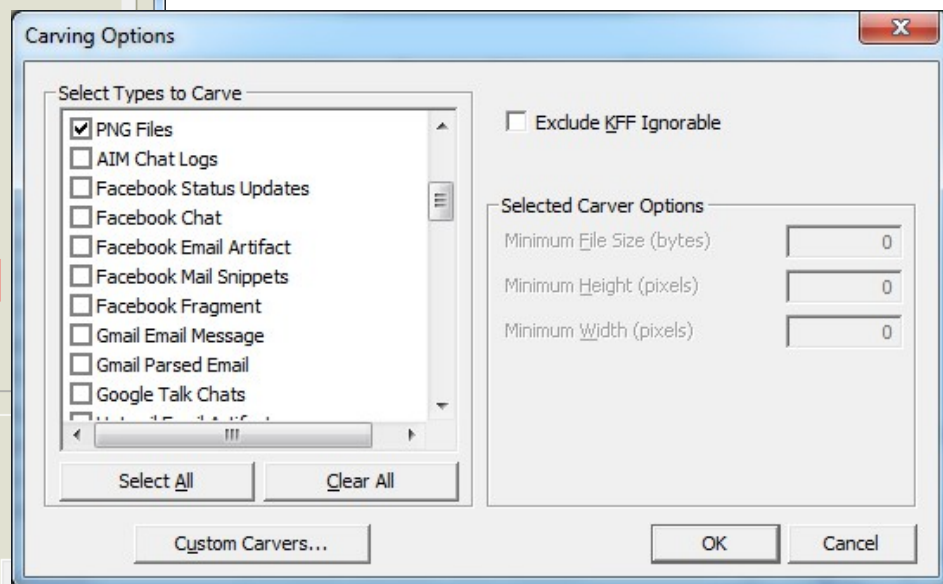
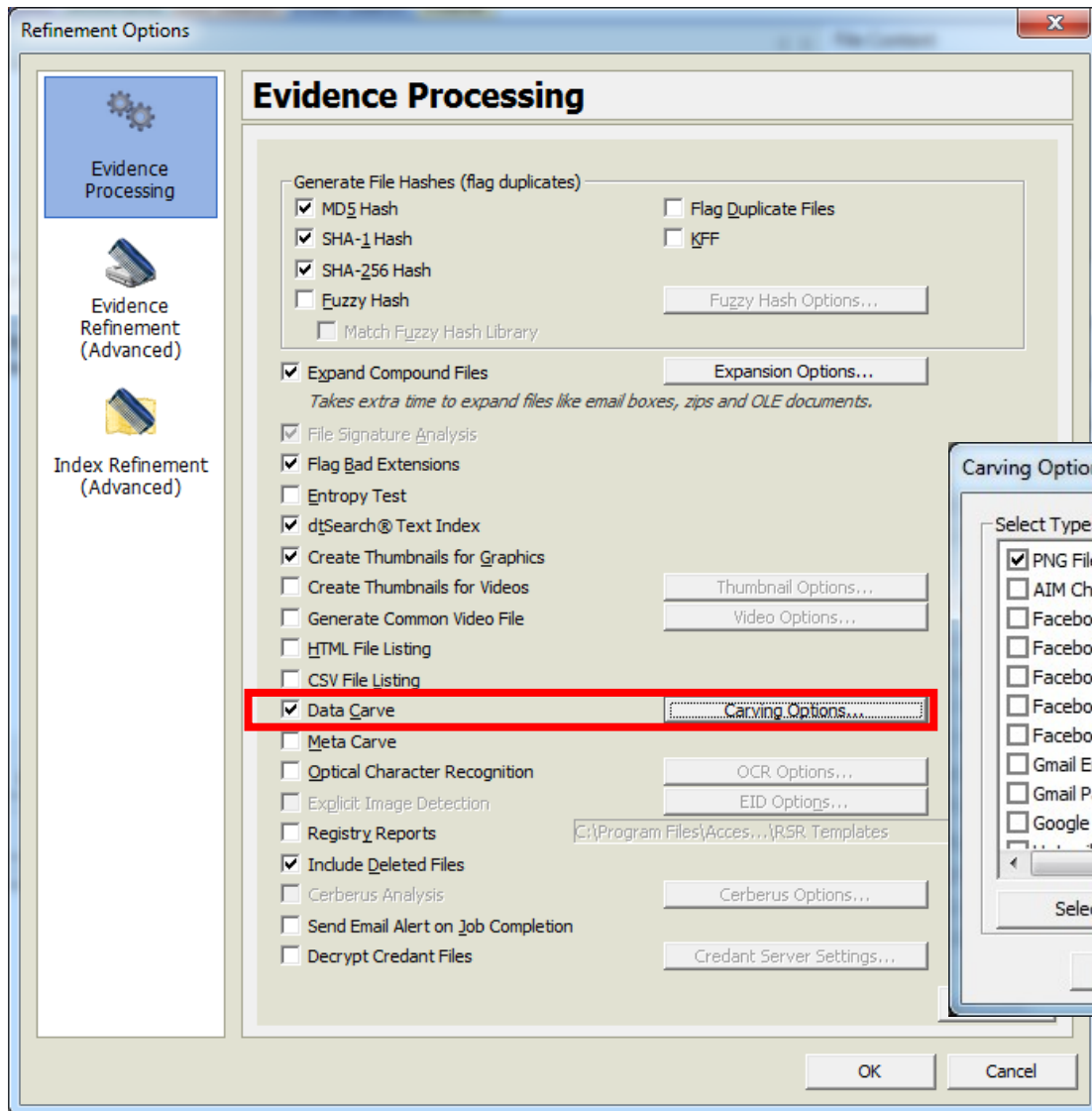
[Custom Carving and File Types](#)

[server]\tools\AccessData\

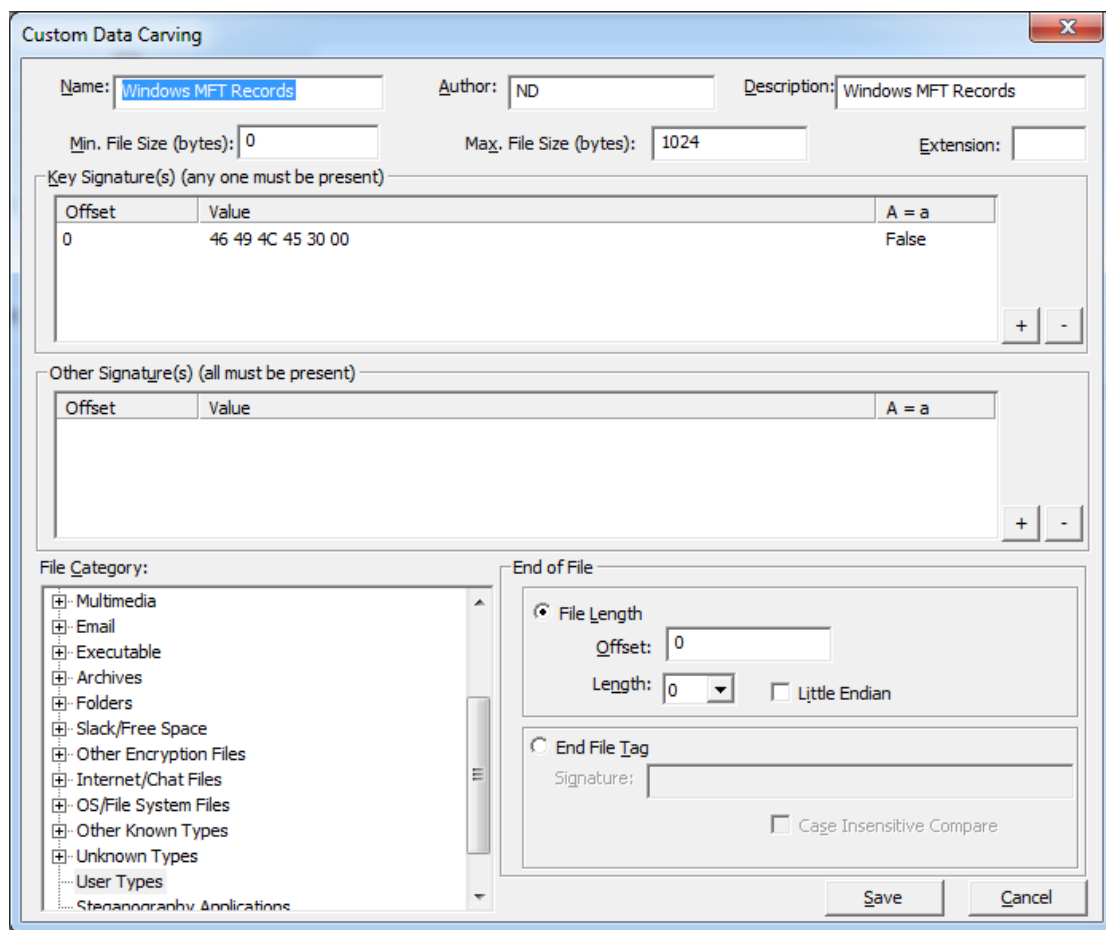
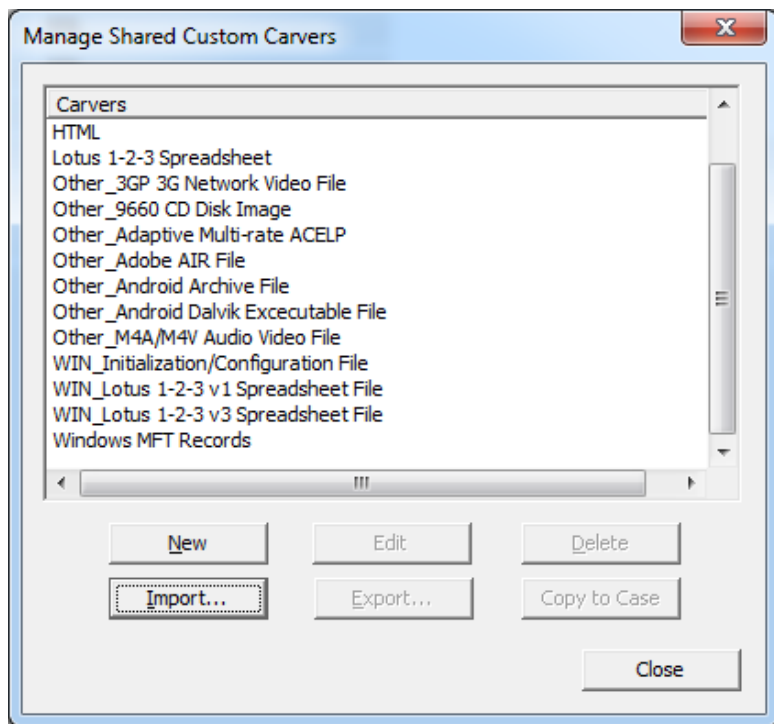
More than 150 different carvers

<http://www.youtube.com/watch?v=D3SVCsiYcug>

# FTK >= 3.2 Custom Carvers

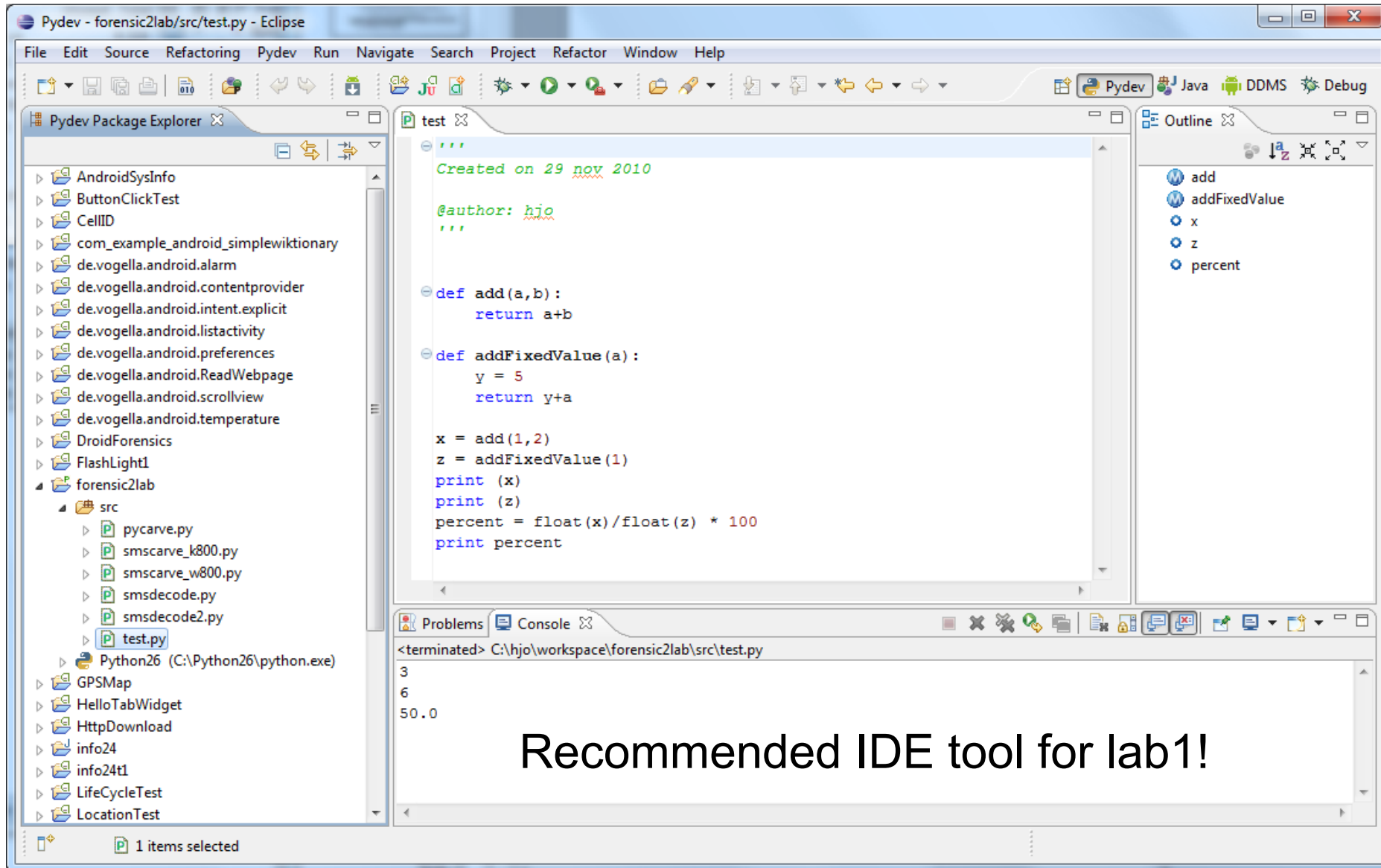


# FTK >= 3.2 Custom Carvers





# Lab 1 - Eclipse with PyDev



Last lecture slide!  
References will follow

# File Carving Taxonomy 1

- Carving (from: <http://www.forensicswiki.org/wiki/Carving>)
  - General term for extracting data (files) out of undifferentiated blocks (raw data), like "carving" a sculpture out of soap stone
- Block Based Carving
  - Any carving method (algorithm) that analyzes the input on block-by-block basis to determine if a block is part of a possible output file. This method assumes that each block can only be part of a single file (or embedded file).
- Characteristic Based Carving
  - Any carving method (algorithm) that analyzes the input on characteristic basis (for example, entropy) to determine if the input is part of a possible output file.
- Header/Footer Carving
  - A method for carving files out of raw data using a distinct header (start of file marker) and footer (end of file marker).

# File Carving Taxonomy 2

- Header/Maximum (file) size Carving
  - A method for carving files out of raw data using a distinct header (start of file marker) and a maximum (file) size. This approach works because many file formats (e.g. JPEG, MP3) do not care if additional junk is appended to the end of a valid file.
- Header/Embedded Length Carving
  - A method for carving files out of raw data using a distinct header and a file length (size) which is embedded in the file format
- File structure based Carving
  - A method for carving files out of raw data using a certain level of knowledge of the internal structure of file types. Garfinkel called this approach "Semantic Carving" in his DFRWS2006 carving challenge submission, while Metz and Mora called the approach "Deep Carving."

# File Carving Taxonomy 3

- Semantic Carving
  - A method for carving files based on a linguistic analysis of the file's content. For example, a semantic carver might conclude that six blocks of french in the middle of a long HTML file written in English is a fragment left from a previous allocated file, and not from the English-language HTML file.
- Carving with Validation
  - A method for carving files out of raw data where the carved files are validated using a file type specific validator.
- Fragment Recovery Carving
  - A carving method in which two or more fragments are reassembled to form the original file or object. Garfinkel previously called this approach "Split Carving."

# Sources and readings 1

- Cohen, Michael: Advanced Carving techniques
  - [http://sandbox.dfrws.org/2007/cohen/Advanced\\_Carving.pdf](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](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](http://dfrws.org/2005/proceedings/richard_scalpel.pdf)

# Sources and readings 2

- LibCarvPathand 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](http://www.korelogic.com/Resources/Projects/dfrws_challenge_2006/DFRWS_2006_File_Carving_Challenge.pdf)