

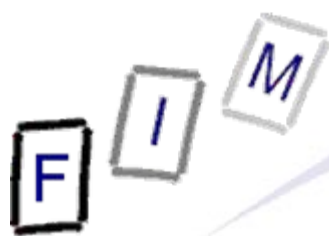


Drive investigation

Computer Forensics, Budapest 2008

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>



- Install the software:
 - Filedisk
 - WinHex, Cygwin
- Search for deleted files and reconstruct them
 - WinHex: Deleted file (FAT)
 - Reconstruct: If possible
- Discovering hidden files: Wrong extension
 - Cygwin: "file" command
- Windows ADS
 - LADS – Find the picture hidden in an ADS
- Timestamps
 - WinHex: Analyze timestamps and convert them
- Running time of your Windows computer
 - Analyze the event log

Scenario

①

②

②

③



- Source of images: <http://dftt.sourceforge.net/>
 - ❶ 6-undel-fat.zip
 - » FAT image
 - ❷ 8-jpeg-search.zip
 - » NTFS image
 - ❸ 5-fat-daylight.zip
 - » FAT image
- Requirements:
 - Operating System: Windows (XP; NT, 2K, Vista: ???)
 - Harddisk space:
 - » Scenarios: 18 MB
 - » Cygwin: 674 MB
 - » Other software: 4 MB



Software installation

- Filedisk: Mounting a disk image as a drive under Windows
 - Requires Administrator access and a reboot
 - Procedure:
 - » Copy driver (filedisk.sys) to %SYSTEMROOT%\system32\drivers
 - » Import "filedisk.reg" into the registry (double-click on it)
 - » Reboot the computer
 - Attention: When mounting an image, you must always give the full path to the file!
 - » E.g. "filedisk /mount 0 C:\temp\image.dd"
- Install "Winhex"
 - Not really needed; can be run directly from CD!
 - » Copy to harddisk for faster start if desired



Software installation

- Install "Cygwin"

- Linux-like environment (and programs) under windows

- Procedure:

- » Execute "setup.exe" and choose to install from local path

- Select the Subdirectory with "ftp..." in it as install source

- No spaces in the path of destination directory

- E.g. **not** C:\Program Files\...

- » Change selection to "install" on the "All" selection

- Add the binary directory to the path

- » Control panel – System – Advanced – Environment Variables →
Add to user variables the complete path, e.g. ";C:\Cygwin\bin"

Search for deleted files: FAT



- Find and recovery the deleted files in image ❶!
- Your task:
 - Find out, which files did at some time exist in the image
 - » Recovery through WinHex!
 - » Manual recovery not possible due to eval. version limitations
 - Recover these files
 - » Check their MD5 values
- Document your actions through a log and screen shots!
- Hints:
 - FAT1 starts at offset 0x1000, FAT2 (=copy) at 0x4000
 - Root directory is at offset 0x7000



Search for deleted files: FAT

- MD5 table of correctly recovered files

→ Filename	File size	MD5 value
→ \SING.DAT	780	59B20779F69FF9F0AC5FCD2C38835A79
→ \MULT1.DAT	3801	FFD27BD782BDCE67750B6B9EE069D2EF
→ \FRAG1.DAT	1584	7A3BC5B763BEF201202108F4BA128149
→ \FRAG2.DAT	3873	0E80AB84EF0087E60DFC67B88A1CF13E
→ \DIR1\ MULT2.DAT	1715	59CF0E9CD107BC1E75AFB7374F6E05BB
→ DIR2\ FRAG3.DAT	2027	21121699487F3FBBDB9A4B3391B6D3E0

Search for deleted files: FAT



- Mount the image ① as readonly (6-fat-undel.dd)
- Look at the image directly (cmd.exe)
 - There is not a single file there!
- Examine it using WinHex
 - Find the FAT and the root directory (see also later)
 - Root dir: Starts at offset 0x7000
 - » Note: The first directory entry is not a file, but the volume label!
- Root dir content:
 - ?RAG1.DAT, ?RAG2.DAT, ?ING.DAT, ?ULT1.DAT, ?YSTEM~1 ("System Volume Information"; long filename)
 - » 0xE5 → This file has been deleted
 - ?IR1: Deleted directory
 - » Difference file vs. directory? Byte 0x0B, Bit 4 (here: 0x10)!



- Manual undelete of ?ING.DAT
 - Overwrite first byte (0xE5) with something else (e.g. 0x53 = S)
 - » Note: You have to change into edit mode with F6!
 - » But writing to the disk does not work in the evaluation version!
 - Default edit mode gives an error message, but in-place mode just **silently ignores** all changes you make!
 - Right-click on the file and then "Recover/Copy..."
 - Copy the files to your hard disk and calculate their MD5 sum
 - Example: "md5sum _ING.DAT"
 - » Note, that the MD5 for ?ING.DAT and ?ULT1.DAT are correct, but those for ?RAG1.DAT and ?RAG2.DAT are not
 - » Why? Examine the FAT table at offset 0x1000 (or 0x4000)!
 - » The FAT has been completely cleared, except for the first sector
 - This is the sector of the root directory!
 - Automatic undelete has therefore the problem, that it cannot know which sectors belong to a file if fragmentation occurs!



- Result: Start at first (=known) sector and copy consecutive number of bytes till the file size has been reached
- Possible chance at detection (but not solution!):
The sector is marked as "in use" by another directory entry
 - » However, this is marked as deleted as well, so which one was the later one cannot really be determined either!
 - MAC dates might help here to some degree
- Deleted directory: Only the directory is marked as deleted
 - The files/directory inside are only **implicitly** deleted!
 - Their first character still exists: MULT2.DAT and subdir DIR2
 - \DIR1\DIR2: FRAG3.DAT
 - MULT2.DAT can be recovered, FRAG3.DAT is again a fragmented file and cannot be recovered
 - » Although extracted without **any** warning or error message!



Discovering hidden files: Wrong extensions

- Find out, which of **all** the files in image ② are jpg pictures!
- Your task:
 - Collect all files, except those in archives
 - » How many are these?
 - Identify their file type
 - » Do this manually (Winhex)
 - Check first in the internet: How to recognize a JPG file
 - » Use the command "file"
 - Inspect the "magic" file and find the description for JPG files
 - » Use command "strings" (file1.jpg, file4.jpg, file12.doc, cmd.exe)
 - Identify the file type of the archives
- Document your actions through a log and screen shots!



Wrong extensions: Exemplary solution – Manual check

- Mount the image ② as readonly (8-jpeg-search.dd)
- Collect all files
 - alloc\file1.jpg, alloc\file2.dat, \invalid\file3.jpg, \invalid\file4.jpg, \invalid\file5.rtf, misc\file11.dat, misc\file12.doc, misc\file13.dll
 - In total 8 files (+3 archives)
- Search the Internet for recognizing JPEG files
 - E.g. search Google for "JPEG magic number"
 - » See [http://en.wikipedia.org/wiki/Magic_number_\(programming\)](http://en.wikipedia.org/wiki/Magic_number_(programming))
 - » JPEG start with 0xFFD8 and end with 0xFFD9
 - Most (JPEG/JFIF type) also contain "JFIF\0"!
- Identify files manually
 - Open alloc\file1.jpg in Winhex
 - » Both start and end match, and "JFIF" is found at offset 0x06
 - This also applies to alloc\file2.dat → Wrong extension!



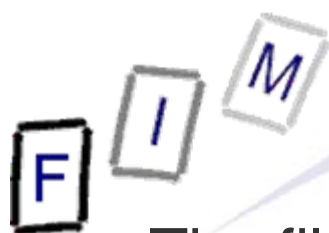
Wrong extensions: Exemplary solution – Manual check

- invalid\file3.jpg has a different start (0x4865) and end (0x300A)
- invalid\file4.jpg claims to be a JPEG, but has only a header
 - » The footer is 0x9F32; additionally there is no "JFIF"
 - Only "JF", and that at index 0x02AB90!
- invalid\file5.rtf has neither header nor footer and no "JFIF"
 - » But signature occurs several times within, e.g. 0x2CF3, 0x4094!
 - » Only "JF" at 0x013062 (no "JFI" or "JFIF"!)!
- misc\file11.dat has a wrong header, but a correct footer
 - » Additionally, "JFIF" occurs at 0x062A
 - » At 0x0624 there is the correct header signature
 - » This could be a JPEG with some other data at the beginning
 - Extract it with WinHex (in 2 parts, e.g. to 0x31FFF; eval. limit)
 - Concatenate with "copy /b part1.bin + part2.bin file11.jpg"
 - » We have recovered a new picture ("I Am Picture #8")!



Wrong extensions: Exemplary solution – Manual check

- misc\file12.doc is similar to file11.dat, but the end of the picture is not the end of the file
 - » Do the same as above
 - » Actually, this is a valid MS Word document with an embedded JPEG; these are not necessarily always stored "plain"!
 - » Extract from 0x1348 until 0x1C26C (inclusive)
 - 0x1348-0xFFFF and 0x010000
 - » We have recovered "I Am Picture #9"!
- misc\file13.dll: Header and footer wrong, no "JFIF"
 - » This is no picture!
 - » It looks more like random data



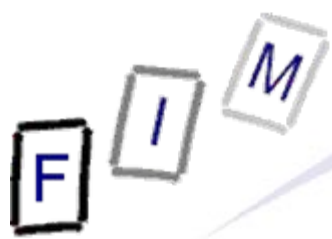
Wrong extensions: Exemplary solution – "file" command

- The file command uses a table of "magic values" to identify file types according to their content
 - These rules can be very simple, but also complex
- Where is the file located: %CYGWIN%\usr\share\file\magic
 - JPEG starts at line 7327
 - » Note: "JFIF" occurs already before, but those are movie files!
 - As can be seen, only the start of the file is checked!
 - » Starts with 0xFFD8
 - » At exactly the position 0x06 the string "JFIF" must occur
 - The ">" is a continuation marker, **not** an index modifier!
- Identify file types:
 - alloc\file1.jpg: JPEG image data, JFIF standard 1.01
 - alloc\file2.dat: JPEG image data, JFIF standard 1.01
 - \invalid\file3.jpg: ASCII English text
 - » Actually, the file only **starts** with ASCII English text. But as only the start is checked, the rest is ignored!



Wrong extensions: Exemplary solution – "file" command

- `\invalid\file4.jpg`: JPEG image data
 - » Note: No "JFIF" and "version ?.?!"
 - Identified, as only the (correct) header is checked, but not the footer
 - The missing "JFIF" should be a warning sign here!
- `\invalid\file5.rtf`: data
 - » Could not be identified; actually it is just random data
- `misc\file11.dat`: data
 - » As only the start is checked, the picture later on is **not found!**
- `misc\file12.doc`: Microsoft Installer
 - » This identification is incorrect!
 - » Try again with "file –k file12.doc": Provides additionally "Microsoft Office Document"
 - Microsoft office files are often hard to identify because of a complex file format (actually a kind of archive with several streams)
- `misc\file13.dll`: data
 - » Just random data, correctly identified



Wrong extensions: Exemplary solution – "strings" command

- Only as an example: The "strings" command
 - More useful for investigating executables
 - » What text do they contain
 - » If a debug version → what methods do they call, messages etc.
- Examine alloc\file1.jpg
 - strings file1.jpg | more
 - » All strings, first one is "JFIF"
 - strings file1.jpg | grep "JFIF"
 - » Select only those lines containing the specific string
- Examine invalid\file4.jpg
 - strings file1.jpg | grep "JFIF"
 - » Returns nothing → no JPEG
 - Actually, just not a JPEG/JFIF!



Wrong extensions: Exemplary solution – "strings" command

- Examine misc\file12.doc
 - Finds "JFIF" very early, but not as the first string
 - The end is more interesting: The document properties!
 - » We can find out that some "Brian Carrier" is somehow involved with this document
 - Detailed investigation: This is the author (of document and image)
 - » We can also see that it was created by Microsoft Word 10.1
- Examine %SYSTEMROOT%\system32\cmd.exe
 - Lots of Windows functions
 - » E.g. MessageBeep, CopyFileExW, RegEnumKeyW, _wcslwr
 - At least one message "CMD Internal Error %s"
 - Some "ASCII art" (probably icon information)



Wrong extensions: Exemplary solution – Archives

- Check the file type of the archives with the "file" command
 - file8.zip: Zip archive data, at least v2.0 to extract
 - » "unzip file8.zip -d C:\temp"
 - file8.jpg + random8.dat (some random data)
 - » file8.jpg is "I Am Picture #5"
 - file9.boo: Zip archive data, at least v2.0 to extract
 - » Unzip as before → file9.jpg + random9.dat
 - » file9.jpg is "I Am picture #6"
 - file10.tar.gz: gzip compressed data, from Unix
 - » "gzip -d file10.tar.gz -c > C:\temp\file10.tar"
 - » "file -z file10.tar.gz" → Identifies a TAR inside a ZIP
 - file10.tar: POSIX tar archive (GNU)
 - A file archive (several files; not compressed!)
 - » "tar -xvf file10.tar"
 - file10.jpg + random10.dat (some random data)
 - » file10.jpg is "I Am Picture #7"



Search for deleted files: Bonus example - NTFS

- Search for deleted files in this image
- Recover them if possible
 - In del1 a deleted JPEG can be recovered (file6.jpg)
 - In del2 another file could theoretically be recovered completely (file7.hmm)
 - » Not actually because of the WinHex evaluation version size limit!
 - » Or recover it manually in two parts and combine them
- Note that this is not a FAT but a NTFS volume!



- Find the hidden picture!
- In the image ② there is an additional picture hidden
 - This is located within an alternate data stream
- Your task:
 - Find the location of the hidden picture
 - Extract the picture into a separate "normal" file
 - Add the picture to another file and to a directory
 - » Not "into" the directory, but to the directory entry itself!
 - » Name the ADS "new*picture"
 - Could you create a normal file with this name?
- Document your actions through a log and screen shots!

Windows ADS: Exemplary solution



- Mount the image ② as **read-write** (8-jpeg-search.dd)
- Run lads for every directory
 - Or use the parameter **"/s"** on the root directory
- Look at the result: The file **"?:\misc\file13.dll"** contains an ADS with the name **"here"**
 - **"?:\misc\file13.dll:here"**
- Extract the ADS through **"more"**
 - **"more < ?:\misc\file13.dll:here >here.jpg"**
 - Examine the file: Is it really a picture?
 - Check the MD5: It should be **9b787e63e3b64562730c5aecaab1e1f8!**
 - » **The result is different! Why?**
 - Open the file in Winhex and compare it to another JPEG file from the same drive
 - » **It seems, that "more" does some textual translation, outputting 0x0D0A instead of 0x00 (translation to plain text!)**



- Extract through "cp" (Cygwin!)
 - "cp ?:\misc\file13.dll:here here.jpg"
 - Check the file with Winhex: Is it a picture? Yes!
 - Calculate the MD5 value and check it
 - » Now it is OK!
- View the picture: Just open it (double-click) or use Paint
 - It shows a green puzzle tile and the text "I Am Picture #10"



- Add the picture to another file
 - Create a small text file named "base.txt"
 - Add the picture to this file under the name "new*picture"
 - » "type here.jpg > base.txt:new*picture"
 - A normal file could not be created with this name, as it contains the illegal character "*" (wildcard)!
 - Confirm (lads+extraction) that the file is there and unchanged
 - » "cp base.txt:new*picture base.txt.jpg"
 - » Check also the MD5 value
- Add the picture to a directory entry
 - Create a directory named "base"
 - Add the picture: "cp here.jpg base:new*picture"
 - Confirm (lads+extraction) that the file is there and unchanged
- Delete the ADSs and files/directories
 - "rm base.txt:new*picture", "rm base:new*picture"



Timestamps

- Find out when the two files in image ③ were actually created
- Your task:
 - Check the date through the Windows command line
 - » Would changing the local time zone influence the output?
 - » Compare this to your Windows drive (hint: FAT ⇔ NTFS!)
 - Find out where the creation time is located on the disk
 - » Don't use the Winhex UI; first think and calculate, then verify!
 - Manually calculate the creation time from the hex values
 - » Search the internet for the exact format
 - Use DCode to decode the creation time
 - When were the files created in UTC?
- Document your actions through a log and screen shots!



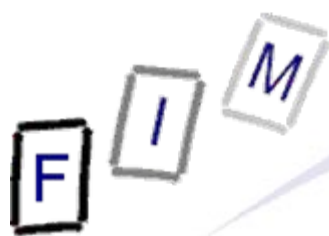
Timestamps: Exemplary solution

- Mount the image ③ as readonly (daylight.dd)
- Check dates through command line
 - Result:
 - » Winter.txt: 1.1.2004 14:00
 - » Summer.txt: 1.6.2004 15:00
 - Changing the local time zone would not change the output!
 - » FAT stores date/time according to the local time of the computer at the moment the action occurs
 - » Therefore it is not "recalculated" according to the local time zone
 - As for example NTFS dates are: These are stored as UTC!
 - File shows 14:26 in TZ Austria (+1), but 13:26 in TZ London (=UTC)!
- Where is the FAT on a FAT-disk?
 - Offset 0x0e in first sector: Number of reserved sectors (=1)
 - FAT therefore starts immediately after the boot sector
 - » This is address 0x0200 (1 sector = 512 bytes)



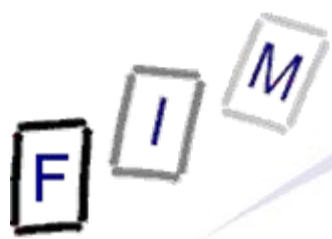
Timestamps: Exemplary solution

- Root directory starts immediately after the FAT-copy
 - » Length of FAT: Offset 0x16 = 9
 - » Boot + FAT1 + FAT2 = 1+9+9 = 19 sectors = 0x2600
- One directory entry = 32 bytes
- Creation time: 0x0D-0x11
- Manual calculation from the hex values
 - » See http://en.wikipedia.org/wiki/File_Allocation_Table for details
 - Value: 0x8600702130
 - » Fine time: 0x86 = 134 * 100ms = 1.34 s
 - » Time: 0x7000 = 14 hours, 0 minutes, 0 * 2 seconds
 - Little endian, therefore to be converted as 0x7000 and not the 0x0070 as found on the disk!
 - » Date: 0x3021 = 24+1980 years, month 1 (=January), day 1
 - » Result: 1.1.2004, 14:00:01.34



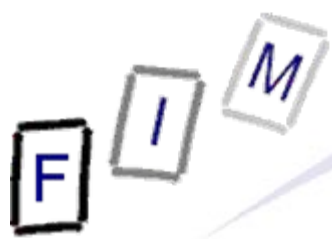
Timestamps: Exemplary solution

- Use DCode to decode the date and time
 - Note DCode only can convert 4-Byte times!
 - Use "MS-DOS: 32 bit Hex Value"
 - To enter: 00702130 (omit first byte; little endian!)
 - Result: 1/Jan/2004 14:0:0 Local
- When were the files created in UTC?
 - This we cannot say, as the date/time is always stored in local time only. Unless we know exactly the time zone where the file was created, we cannot determine it!



Windows Startup/Shutdown time

- Investigate your own computer:
 - When was it turned on and off during the last week?
 - » Investigate in the Internet which events are logged when!
 - Draw a timeline to visualize your results!



Windows Startup/Shutdown time: Exemplary solution

- The Startup/shutdown time is logged in the event log
 - These are part of the "System" log
 - See <http://support.microsoft.com/kb/196452>
 - Date and time of the event are logged as well
 - » Note: Local time!
- Event-IDs:
 - 6009: Startup (OS version, ...)
 - 6005: Event log service started
 - ...
 - 6006: Clean shutdown
 - 6008: Dirty shutdown
 - » Unexpected, e.g. through power failure



Windows Startup/Shutdown time: Exemplary solution

- To ease the gathering, use "View – Filter..." to only show events with specific IDs
- Exemplary results:
 - No 6008 events!

Type	Date	Time	Source	Category	Event
Information	14.01.2008	07:39:35	eventlog	None	6009
Information	11.01.2008	07:42:02	eventlog	None	6009
Information	10.01.2008	07:42:36	eventlog	None	6009
Information	09.01.2008	07:52:19	eventlog	None	6009
Information	08.01.2008	07:52:27	eventlog	None	6009
Information	07.01.2008	07:48:25	eventlog	None	6009
Information	04.01.2008	08:44:11	eventlog	None	6009
Information	03.01.2008	08:46:04	eventlog	None	6009
Information	02.01.2008	08:41:38	eventlog	None	6009
Information	21.12.2007	08:38:17	eventlog	None	6009
Information	20.12.2007	08:42:55	eventlog	None	6009

Type	Date	Time	Source	Category	Event
Information	11.01.2008	18:00:31	eventlog	None	6006
Information	10.01.2008	17:37:32	eventlog	None	6006
Information	09.01.2008	20:11:26	eventlog	None	6006
Information	09.01.2008	07:51:18	eventlog	None	6006
Information	07.01.2008	17:11:09	eventlog	None	6006
Information	04.01.2008	17:13:34	eventlog	None	6006
Information	03.01.2008	17:05:44	eventlog	None	6006
Information	02.01.2008	16:20:33	eventlog	None	6006
Information	19.12.2007	22:54:01	eventlog	None	6006



Windows Startup/Shutdown time: Exemplary solution

- Result:

- Monday 7.1.2008: 7:48-17:11
- Tuesday 8.1.2008: 7:52- 7:51 (next day!)
- Wednesday 9.1.2008: 7:52-20:11
- Thursday 10.1.2008: 7:42-17:37
- Friday 11.1.2008: 7:42-18:00

- Extraordinary period on night of 8.1. to 9.1.

- Further investigation: Some updates occurred at 5 o'clock in the night, so the computer was actually running
- It seems, it was just not turned off
- Whether it was in actual use cannot be decided!
 - » But as no other events occurred during that time, this is unlikely



- Undelete is quite simple on FAT
 - But complex/impossible on NTFS/EXT3!
 - "Plain text" search will still work unless actually overwritten
- Hiding files is quite simple: Wrong extensions and ADS
 - Found only with good knowledge and additional tools
 - » But **VERY** difficult to **REALLY** hide information!
- Even with very simple means a lot of information can be extracted, if it is exactly known where to look for it
 - But also its limitations must be known!
- Timestamps (or timing issues) are an important aspect for every forensic investigation
 - The time zone is very important there
 - » Is the data stored in local or UTC (or ...) time?
 - » What is the difference to UTC now (and what was it then?)

F I M

Questions?

Thank you for your attention!