Mag. iur. Dr. techn. Michael Sonntag

Disk/File system investigation

Institute for Information Processing and 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

© Michael Sonntag 2013

Agenda

- Acquiring a forensic copy
- Preliminary stages
  - → Image hashing
  - → Partition/file system information
- Removing known files
- Identifying file types
  - → Hash databases
- Creating a timeline

Acquiring a forensic copy: Write blockers

- Never work on the original media
  - → Anything goes wrong → Evidence is gone!
    - » Even just a suspicion this occurred may be enough in a process!
- So we need a copy…
  - → But during copying the media is accessed as well!
  - → Additionally, we don't want a copy of the files … we want a copy of the whole medium!
    - » This is not the same: Unallocated clusters are e.g. not copied when transferring (all) files through a share
- Result: Create a binary copy of the source media while applying some kind of write-protection to the original
  - → This may be quite easy: Floppy disks/USB sticks do have (more correct: “had”!) a "write-protect" "switch"
    - » But can we trust it? And what about media without them, e.g. normal hard disks, most modern USB sticks?

Requirements for forensic duplications

- Physical copy:
  - → We need a copy of every single bit, not only of the (currently existing) files on the disk
- Error handling:
  - → Read errors must be detected without fail and be documented
  - → Replaced by pre-determined and known pattern
- Completeness:
  - → Reserved areas of the disk must be detected and deactivated for the copying to obtain a truly complete image
- Unmodified:
  - → Calculate a checksum to be later able to prove the integrity of the image

Acquiring a forensic copy: Write blockers

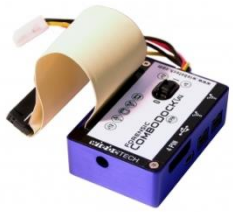
- Therefore we need a separate write blocker
  - → Which is under the control of the person performing the copy!
- Use a trusted hardware write blocker
  - → Exist for all kind of media: IDE, SATA, flash-disks, SCSI, …
    - » Note: More "exotic" or high-performance → Expensive
      - – This is not a mainstream hardware sold in the thousands!
  - → USB is quite universal (USB HD cases!)
- Alternatively use a software write blocker
  - → Problem: Many things can go wrong, e.g. configuring it for the wrong device, bugs etc.
  - → Additionally, it should only be used on a trusted computer
    - » Not: Installing/Running a SW write-blocker on source machine
      - – You don't know what else is installed there and whether this will actually work or not!
  - → Typical example: USB write blocker
  - → Potential problem: Reboot may be required

Hardware write blockers: How they work

- Two kinds exists
  - → Same interface on both sides: IDE – IDE
  - → Different interfaces: SATA – USB/Firewire
    - » The typical computer-side is USB and/or Firewire
      - – Future: Perhaps eSATA or USBv3; but not yet available!
    - » Advantage: USB and Firewire are hot-swappable!
- Basic approach
  - → Intercept commands and drop those writing to the disk
    - » Problem: Custom extensions!
      - – Best approach: Don't allow anything not explicitly known to **not** modify the data and block everything else
      - – Note: This may break compatibility with exotic systems!
    - » Return OK/Failure depending on configuration
  - → Pass all other, i.e. read-only, commands
- See http://www.cftt.nist.gov/hardware_write_block.htm for tested appliances!

Hardware write blockers: Examples

- Examples:
  - → FastBloc
    - » http://www.encase.com/products/ee_hardware.aspx
  - → ICS DriveLock
    - » http://www.icsforensic.com/index.cfm/action/catalog.browse/category/DriveLock/id_category/c14d69f1-dcb6-47ab-8be6-1b13217f5b84
  - → WiebeTech Forensic ComboDock v4
    - » http://wiebetech.com/products/ForensicComboDock.php
    - » http://www.wiebetech.com/products/USB-WriteBlocker.php
  - → Tableau
    - » http://tableau.com/index.php?pageid=products&category=forensic_bridges
  - → MyKey NoWrite FPU (owns a patent on write-blocking)
    - » http://www.mykeytech.com/



Software write blockers: How they work

- Basic principle: Access the media without passing on write requests; only allow read requests
  - → I.e., on Linux do not mount it in read/write mode, or just "refrain from writing" (USB)
    - » "Not writing" may (and often will!) still change access times (→ Windows registry flag)
    - » Attention on journaling file systems!
- Not recommended: Setting the USB-write-protection flag in the Windows registry
  - → This requires a reboot and is not guaranteed to work!
- In general, SW blockers do the same as Hardware ones
- Comparing Hardware and Software blockers:
  - → SW +: Cheaper and flexible (all devices)
  - → SW -: Platform specific, working not immediately apparent
  - → HW +: Hot-swap, interface conversion, easier to verify
  - → HW -: Expensive, only for selected interfaces

Software write blockers: Examples

- Digital Intelligence PDBlock
  - → http://www.digitalintelligence.com/software/disoftware/pdblock/
- Linux:
  - → Disable auto-mounting
  - → Mount drive as read-only
  - → Example: mount –t <fs-type> –o **ro**,noexec,noatime,loop <image> <directory>
    - » ro: Do not write to disk, not even for root
    - » noexec: Do not execute files from this disk
    - » noatime: Do not change access time on access
    - » loop: Loopback device, i.e. opening an image as a file system
- See http://www.cftt.nist.gov/software_write_block.htm for test reports of dedicated software!

Duplication issues

- Read errors: What to do when encountering erroneous sectors on the source media
  - → Try to get the data nevertheless (several retries)
  - → If really not accessible, then it wasn't for the suspect as well!
    - » When still suspected → Hardware investigation (platter surface)
  - → Write zeros ('0x00') to the destination instead
    - » This will cause the least harm and not introduce other material
    - » Additionally, mark it as "BAD" externally or within (not **pure** 0x00)
- Wiped destination disk
  - → Ideally the destination disk should be wiped before acquiring
    - » This means all zeros, not just a (fast/complete) formatting!
    - » Reason: Read errors, larger size, … precaution
  - → Not needed when acquiring to an image **file** (common today)
- Large disks may require multiple destination volumes
  - → Splitting the image into several image files
  - → Care required on analyzing: Seams!

Forensic duplication file formats

- Raw: Bit-by-bit copy of the source (".dd", ".bin", …)
  - → Every program can work with this format
  - → There is no compression and no metadata
    - » Compression only for transfer possible, not for working with it!
  - → Integrity check must be external (separate file with hash)
- AFF/AFF4: Advanced Forensic Format (".AFF", ".AFD")
  - → Open format: Documented, no royalties, BSD-licensed code
  - → Supports arbitrary metadata
  - → Includes metadata, compression, chain-of-custody recording, encryption, image signing
- EnCase: "Standard" in law enforcement (".E01“, “.E02”, …)
  - → Proprietary file format, certain metadata
  - → Supports compression and encryption
    - » Requires more CPU power to work with, but less space
- Several other exist: http://www.forensicswiki.org/wiki/Forensic_file_formats

Creating a forensic duplication: dd

- dd = Data Dump; Used to create binary copies
- Example: dd if=/dev/hdb of=SuspectHD.bin conv=notrunc,noerror,sync bs=1024
  - → if: Input device
  - → of: Output device; just a normal file here
  - → notrunc: Don't truncate output on errors
  - → noerror: Do not stop on read errors
  - → sync: Write zeros on read errors instead of skipping sector
  - → bs: Block size. Default = 512; better performance with larger values, but read errors always affect complete block
    - » Use the physical size if possible; usually 512 (or 4096)
  - → count: Number of blocks to copy
    - » Must be multiplied by "bs" value to get bytes!
  - → skip: Number of blocks skipped before copying starts
- Make sure that "of" is mounted, but "if" is **not**!

Creating a hash of the whole image

- Important to assure the identity of the image and the source
- Therefore two hashes should theoretically be built
  - → One of the source drive
  - → One of the image
- Actually, usually only a single one is calculated, as reading the source again would not be different from image creation!
  - → Still important: Later modifications of the image can be detected easily
  - → Additionally, in case of doubt, the original can be re-read and hashed and compared to the image which was analyzed
    - » Helps against swapping images or malicious modifications
- Typically SHA-1, SHA-256, or MD5 is used
  - → MD5 should not be used any more, as it is known to be susceptible to attacks (not yet broken completely)

Creating a hash of the whole image: Example

- Example for creating a MD5 hash:
  - → chmod 444 SuspectHD.bin
  - → md5sum –b SuspectHD.bin >md5sum.txt
  - → chmod 444 md5sum.txt
- Example for checking:
  - → md5sum –c md5sum.txt
    - » File need not be specified – stated in md5sum.txt!
- Content of md5sum.txt:
  - → 3be6330d9da0db04d45ef96c86bd7afc SuspectHD.bin

See "sha1sum" for calculating SHA-1 hashes

- → "shasum" calculates other versions as well
  - » Algorithm: 1, 224, 256, 384, 512

- Note: chmod is only there for "security": Read-only files!

Duplication + Hashing: dcfldd

- Slight enhancement of "dd", the disk duplication SW
  - → Open source program
  - → Created by the DoD Computer Forensics Lab (DCFL)
- Features:
  - → Hashing of the data on the fly (=during duplication)
    - » Not only for whole file but also for smaller blocks
  - → Status output (progress bar)
  - → Supports disk wipes with special patterns (not just zeros)
  - → Multiple and split output possible
  - → Produces raw images only

Duplication + Hashing: dcfldd

- Example: dcfldd if=/dev/hda of=/mnt/evidence/disk_a.dd conv=sync,noerror hashwindow=1024 hashlog=hash.txt
  - → Parameters similar to dd
    - » if: Input device
    - » of: Output device
    - » sync: Write zeros on read errors instead of skipping sector
    - » noerror: Do not stop on read errors
    - » bs: Block size. Default = 512; better performance with larger values, but read errors always affect the complete block
      - – Use physical size if possible; usually 512
  - → Additional parameters (hashing):
    - » hashwindow=1024: Separate hash for every 1024 bytes
      - – Practice: Use 1000000 or larger
    - » hashlog=hash.txt: Where to write the hash values
- Windows: if=\\.\PhysicalDrive3
  - → >= Vista: Application must run as Administrator

Partition and file system information

- "Volume": Careful, it can mean many things!
  - → Collection of addressable sectors
    - » Not necessarily on one physical device or consecutive sectors
    - » Must only look to the OS/application as if it were cons. sectors!
  - → Single accessible storage area within a single file system
    - » Typically within a partition
  - → An entity that has a drive letter mapped to it
    - » Therefore applicable only to Windows, not Unix
- Physical disk organization can be complex
  - → Several disks can be grouped to create a single "volume"
    - » Example: RAID-0 (Striping), RAID-…
  - → This volume can then be split in several partitions
    - » Within an partition there can be more partitions
  - → Each partition has a single file system
  - → Not the whole disk must be assigned to partitions

Partition and file system information
Forensic considerations

- On complex or uncommon systems, copying the physical disk may not be very useful
  - → String search is always possible
    - » Unless partitions are compressed or encrypted!
  - → But recreating the file systems may be impossible
    - » Depends on the OS used, which is perhaps not available
- Sometimes it may therefore be better to do a "live" copy
  - → Start the system and copy all files to another computer with a "common" file system
  - → Note: All slack space, deleted files etc. are lost!
- Best, but most expensive/time-consuming approach:
  - → Create two full physical copies
    - » One for physical-drive-analysis and an "original" as evidence
  - → Boot from one copy and create a file system duplicate
    - » If possible, use VMWare → Snapshot allows reverting changes!

DOS partitions

- The most common type of disk organization
  - → DOS, Windows, Linux, BSD; most multi-boot systems
    - » 32 Bit versions only; 64 Bit versions are often different!
- Basic layout: See file systems!
- A DOS partitioned hard disk can only contain 4 partitions
    - » These are called "primary partitions"
  - → But one can also be an "extended partition"
    - » This can contain several "logical" ("secondary") partitions
      - – In theory, only two: A normal and again an extended one, …
  - → Any of the sub-partitions could be from a different OS and be organized differently within!
  - → One partition may be marked as "active" or "bootable"
    - » This will be the one the system boots from
    - » Note: The code in the MBR record may decide otherwise, perhaps based on user input, or change the markings!

MBR / Partition table example

- MBR = Master Boot Record
  - → 0-445: Boot code (to be executed on booting the system)
    - » 440-443: Windows ≥ NT: NT Drive Serial Number
      - – Also used by Linux 2.6 to determine boot volume location
  - → 446-509: Partition table (space for describing 4 partitions)
  - → 510-511: Magic number: 0x55, 0xAA
- Partition table:
  - → 0: Bootable Flag (0x80 = Boot partition)
  - → 1-3: Start CHS address
    - » Cylinder-Head-Sector; Only for old/small hard disks
  - → 4: Partition type
    - » E.g. 0x06 (FAT16, 32MB-2GB, CHS), 0x0c (FAT32 LBA), 0x83 (Linux), 0x84 (Hibernation), 0x86 (NTFS Volume Set), …
  - → 5-7: Ending CHS address
  - → 8-11: Starting LBA address
  - → 12-15: Size in sectors

MBR example

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000000000	33	C0	8E	D0	BC	00	7C	FB	50	07	50	1F	FC	BE	1B	7C
000000010	BF	1B	06	50	57	B9	E5	01	F3	A4	CB	BD	BE	07	B1	04
000000020	38	6E	00	7C	09	75	13	83	C5	10	E2	F4	CD	18	8B	F5
000000030	83	C6	10	49	74	19	38	2C	74	F6	A0	B5	07	B4	07	8B
000000040	F0	AC	3C	00	74	FC	BB	07	00	B4	0E	CD	10	EB	F2	88
000000050	4E	10	E8	46	00	73	2A	FE	46	10	80	7E	04	0B	74	0B
000000060	80	7E	04	0C	74	05	A0	B6	07	75	D2	80	46	02	06	83
000000070	46	08	06	83	56	0A	00	E8	21	00	73	05	A0	B6	07	EB
000000080	BC	81	3E	FE	7D	55	AA	74	0B	80	7E	10	00	74	C8	A0
000000090	B7	07	EB	A9	8B	FC	1E	57	8B	F5	CB	BF	05	00	8A	56
0000000A0	00	B4	08	CD	13	72	23	8A	C1	24	3F	98	8A	DE	8A	FC
0000000B0	43	F7	E3	8B	D1	86	D6	B1	06	D2	EE	42	F7	E2	39	56
0000000C0	0A	77	23	72	05	39	46	08	73	1C	B8	01	02	BB	00	7C
0000000D0	8B	4E	02	8B	56	00	CD	13	73	51	4F	74	4E	32	E4	8A
0000000E0	56	00	CD	13	EB	E4	8A	56	00	60	BB	AA	55	B4	41	CD
0000000F0	13	72	36	81	FB	55	AA	75	30	F6	C1	01	74	2B	61	60
000000100	6A	00	6A	00	FF	76	0A	FF	76	08	6A	00	68	00	7C	6A
000000110	01	6A	10	B4	42	8B	F4	CD	13	61	61	73	0E	4F	74	0B
000000120	32	E4	8A	56	00	CD	13	EB	D6	61	F9	C3	55	6E	67	81
000000130	6C	74	69	67	65	20	50	61	72	74	69	74	69	6F	6E	73
000000140	74	61	62	65	6C	6C	65	00	46	65	68	6C	65	72	20	62
000000150	65	69	6D	20	4C	61	64	65	6E	20	64	65	73	20	42	65
000000160	74	72	69	65	62	73	73	79	73	74	65	6D	73	00	42	65
000000170	74	72	69	65	62	73	73	79	73	74	65	6D	20	6E	69	63
000000180	68	74	20	76	6F	72	68	61	6E	64	65	6E	00	00	00	00
000000190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000001B0	00	00	00	00	00	2C	48	6E	BE	C0	BE	C0	00	00	80	01
0000001C0	01	00	07	FE	FF	FF	3F	00	00	00	0E	E3	CA	04	00	00
0000001D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000001F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	55	AA

```
3À▮Ð¼ |ûP P ü¾ |
¿  PW¹å ó¤Ë¾¾ ±
8n | u ▮Å âôÍ ▮õ
▮Æ It 8,tö µ ´ ▮
ð¬< tü»  ´ Í ëò▮
N èF s*þF ▮~  t
▮~  t  ¶ uÒ▮F  ▮
F  ▮V  è! s  ¶ ë
¼▮>þ}Uªt ▮~  tÈ
· ë©▮ü W▮õË¿  ▮V
 ´ Í r#▮Á$?▮▮Þ▮ü
C÷ã▮Ñ▮Ö± ÒîB÷â9V
 w#r 9F s ¸  »  |
▮N ▮V Í sQOtN2ä▮
V Í ëä▮V `»ªU´Aí
 r6▮ûUªuööÁ t+a`
j j ÿv ÿv j h |j
 j ´B▮ôÍ aas Ot
2ä▮V Í ëÖaùÃUng▮
ltige Partitions
tabelle Fehler b
eim Laden des Be
triebssystems Be
triebssystem nic
ht vorhanden


   ¸Hn¾À¾À  ▮
  þÿÿ?    ãÊ


                Uª
```

- Boot code
- Error messages
- NT Drive Serial Number
- Partition table
- Magic number (Signature ID)

- Text for error messages is at the end of the code
- The three bytes before the serial number are the relative offsets of the individual messages
  - Allows translations of different length without changing the code

See http://www.geocities.com./thestarman3/asm/mbr/Win2kmbr.htm

NTFS Partition Boot Record example

0x0B – 0x054: BIOS parameter block

```
0000: EB 52 90 4E 54 46 53 20 20 20 20 00 02 08 00 00   .R.NTFS     .....
0010: 00 00 00 00 00 F8 00 00 3F 00 FF 00 3F 00 00 00   ........?...?...
0020: 00 00 00 00 80 00 80 00 0D E3 CA 04 00 00 00 00   ................
0030: 00 00 0C 00 00 00 00 00 10 00 00 00 00 00 00 00   ................
0040: F6 00 00 00 01 00 00 00 9E D1 A3 28 0A A4 28 AC   ...........(..(.
0050: 00 00 00 00 FA 33 C0 8E D0 BC 00 7C FB B8 C0 07   .....3.....|....
0060: 8E D8 E8 16 00 B8 00 0D 8E C0 33 DB C6 06 0E 00   ..........3.....
0070: 10 E8 53 00 68 00 0D 68 6A 02 CB 8A 16 24 00 B4   ..S.h..hj....$..
0080: 08 CD 13 73 05 B9 FF FF 8A F1 66 0F B6 C6 40 66   ...s......f...@f
0090: 0F B6 D1 80 E2 3F F7 E2 86 CD C0 ED 06 41 66 0F   .....?.......Af.
00A0: B7 C9 66 F7 E1 66 A3 20 00 C3 B4 41 BB AA 55 8A   ..f..f. ...A..U.
00B0: 16 24 00 CD 13 72 0F 81 FB 55 AA 75 09 F6 C1 01   .$...r...U.u....
00C0: 74 04 FE 06 14 00 C3 66 60 1E 06 66 A1 10 00 66   t......f`..f...f
00D0: 03 06 1C 00 66 3B 06 20 00 0F 82 3A 00 1E 66 6A   ....f;. ...:..fj
00E0: 00 66 50 06 53 66 68 10 00 01 00 80 3E 14 00 00   .fP.Sfh.....>...
00F0: 0F 85 0C 00 E8 B3 FF 80 3E 14 00 00 0F 84 61 00   ........>.....a.
0100: B4 42 8A 16 24 00 16 1F 8B F4 CD 13 66 58 5B 07   .B..$.......fX[.
0110: 66 58 66 58 1F EB 2D 66 33 D2 66 0F B7 0E 18 00   fXfX..-f3.f.....
0120: 66 F7 F1 FE C2 8A CA 66 8B D0 66 C1 EA 10 F7 36   f......f..f....6
0130: 1A 00 86 D6 8A 16 24 00 8A E8 C0 E4 06 0A CC B8   ......$.........
0140: 01 02 CD 13 0F 82 19 00 8C C0 05 20 00 8E C0 66   ........... ...f
0150: FF 06 10 00 FF 0E 0E 00 0F 85 6F FF 07 1F 66 61   ..........o...fa
0160: C3 A0 F8 01 E8 09 00 A0 FB 01 E8 03 00 FB EB FE   ................
0170: B4 01 8B F0 AC 3C 00 74 09 B4 0E BB 07 00 CD 10   .....<.t........
0180: EB F2 C3 0D 0A 41 20 64 69 73 6B 20 72 65 61 64   .....A disk read
0190: 20 65 72 72 6F 72 20 6F 63 63 75 72 72 65 64 00    error occurred.
01A0: 0D 0A 4E 54 4C 44 52 20 69 73 20 6D 69 73 73 69   ..NTLDR is missi
01B0: 6E 67 00 0D 0A 4E 54 4C 44 52 20 69 73 20 63 6F   ng...NTLDR is co
01C0: 6D 70 72 65 73 73 65 64 00 0D 0A 50 72 65 73 73   mpressed...Press
01D0: 20 43 74 72 6C 2B 41 6C 74 2B 44 65 6C 20 74 6F    Ctrl+Alt+Del to
01E0: 20 72 65 73 74 61 72 74 0D 0A 00 00 00 00 00 00    restart........
01F0: 00 00 00 00 00 00 00 00 83 A0 B3 C9 00 00 55 AA   ..............U.
```

- Jump to code start + NOP
- Producer and/or type name+version
- Bytes per sector
- Sectors per cluster
- Reserved sector count
- Media descriptor
- Sectors per track
- Number of read/write heads
- Hidden sectors (start of volume)
- Disc unit number
- Signature byte (Magic number)
- Sectors in volume
- Start cluster of MFT
- Start cluster of MFT mirror
- Clusters per MFT record
- Clusters per index Block
- NTFS volume serial number
- Boot code
- Error messages

See: http://www.geocities.com./thestarman3/asm/mbr/NTFSBR.htm
and http://homepages.tesco.net/J.deBoynePollard/FGA/bios-parameter-block.html

Logical Volumes

- An alternative/add-on to DOS partitions
  - → Typically used on top of DOS partitions (one full-size partition per disk), but also works on “raw” disks
- Several physical volumes (partitions/disks) are combined to a volume group, which is then split into logical volumes
- Advantages:
  - → Physical volumes can easily be added
  - → Logical volumes can easily be resized (typ. while in use!)
  - → Logical volumes can span multiple physical volumes
  - → Disks identified by UUID → Physical disks can be moved
- Problems:
  - → Bootloaders/other OS might have problems with it
    - » Therefore typically a "DOS" boot partition + LVM for rest!
- Supported by many OS (notably not by Windows!)

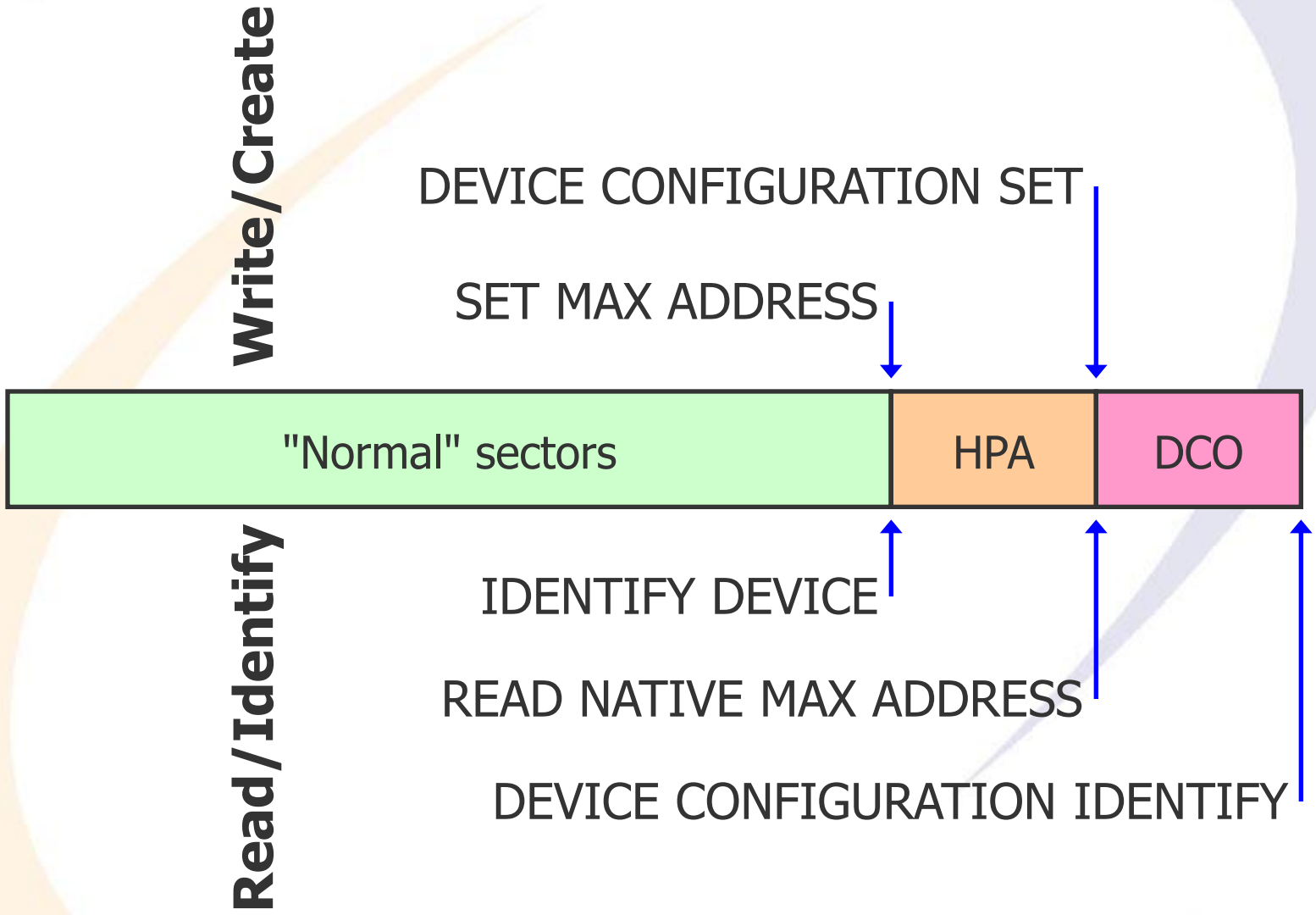
HPA – Host Protected Area

- Hidden area on the disk invisible to the OS
  - → Introduced with ATA-4 standard
  - → Usage e.g.:
    - » Repair info for OS (copy of installation DVD)
    - » Theft recovery and monitoring services
    - » Reducing capacity of disks to match existing ones
- Properties:
  - → Survives formatting the disk
  - → No access by user, OS, or BIOS
  - → Accessible through directly issuing ATA commands
    - » "READ NATIVE MAX ADDRESS (EXT)": Read physical size
    - » "SET MAX ADDRESS (EXT)": Set maximum addressable size
  - → "Volatile" bit: Changes revert on next power up/reboot
    - » Useful for imaging: The disk itself remains unchanged!
      - – Careful with write blockers: They may block the necessary ATA commands as they change the disk, although only temporarily!

DCO – Device Configuration Overlay

- Modifications/hidden area invisible to the OS
  - → Reduce disk capacity to exactly match existing ones
  - → Remove special (optional) features of the controller
  - → Introduced with ATA-6 standard
- Properties:
  - → Survives formatting the disk
  - → No access by user, OS, or BIOS
  - → Modifications are always permanent
- Access directly through ATA commands
- DCO and HPA can exist on the same disk
  - → First set DCO, then reduce size through HPA!
    - » READ NATIVE MAX ADDRESS will return the reduced size!
    - » DEVICE CONFIGURATION IDENTIFY shows the actual size

Combining HPA and DCO



HPA + DCO vs. computer forensics

- Both are simple to detect manually:
  - → Read number of sectors from physical drive (Internet, sticker)
    - » But a label on the drive need not necessarily be original …
  - → Compare to information obtained on the computer
- HPA (DCO) can also be detected through software
  - → Retrieving physical size and comparing it to the maximum addressable sector
- Changing the DCO is always permanent!
  - → No "volatile" bit → Image first, the remove DCO and image all/the rest again. **The disk is modified through this!**
- Imaging without HPA/DCO will produce incomplete copies
  - → And return different hashes than copies including them!
- Good forensic programs handle HPA (automatically), but DCO seems to be still a problem
  - → Or they automatically remove it but do not recreate them
  - → This applies to disk wiping programs as well …

Checklist for creating a forensic copy

- General preparation: Write blockers, disks (disk space) available, disks are wiped, software is available on CD-ROM
- Acquiring the forensic copy:
  - → Identify evidence disk (serial number, type, size, …)
  - → Connect evidence disk to write blocker
  - → Switch on evidence disk/write blocker
  - → Find evidence disk (which "name" is it on this system)
  - → Find target disk/space
  - → Start acquisition software
  - → Create image
  - → Create hash value & save it (unless done by software)
  - → Document other information electronically
  - → Unmount both disks & switch off write blocker
  - → Package evidence disk & add printout of summary (serial number, hash, person, which tools used, …)

Removing known files

- Usually a disk under investigation will contain an operating system, i.e. several thousands of uninteresting files
  - → But you can't be sure that everything within "C:\Windows" is from Microsoft and completely unchanged …
  - → Applies to any other "known" files/locations as well
- Simple approach: Compare file names and contents with installation media and securely downloaded updates
- Better: Create hash values of all files and compare them to public libraries of hash values of known files
  - → These exist for OS, applications, malware etc.
  - → Private libraries also exist for illegal files, e.g. child porn
    - » This is legal: Only the hash value is stored, not the file!
  - → md5deep calculates hash values on numerous files, esp. also recursively (has some other nice features as well)!

Hash libraries

- National Software Reference Library
  - → Contains hash values of known and traceable software applications, but none of illegal data
  - → Currently 33.058.182 unique hash values from 111.011.929 files (1.6.2013)
- Four main CDs
  - → Split according to the SHA-1 hash value)
    - » 00…-3F…, 40…-7F…, 80…-BF…, C0…-FF…
- File format: CSV
  - → SHA-1, MD5, CRC32, Filename, Filesize, Product + OS code
  - → "0000004DA6391F7F5D2F7FCCF36CEBDA60C6EA02", "0E53C14A3E48D94FF596A2824307B492","AA6A7B16", "00br2026.gif",2226,228,"WIN",""
    - » Corel Gallery 750.000, English, Windows

hashdeep

- Similar to md5deep (which, unlike its name suggests, also calculates other algorithms!), but advanced "set" features
- Compares against a set (list) of hash values
  - → Matched, missing, moved, and new files
- First run: Creates a file with size, <several hash algorithms as configured, default: MD5 & SHA256>, path+filename
  - → Can be run recursively on a whole file tree
    - » Example: `hashdeep -r myDir > log.txt`
- Matching mode: Positive and negative is possible
  - → List all those with matching/unknown hashes
- Audit mode: Verifies hashes and lists moved, changed and inserted files
  - → Details can be controlled with "-v", "-vv", and "-vvv"
    - » Example: `hashdeep -r -a -k log.txt myDir`
    - » This will list only success or failure (-v for details)

Identifying file types

- Important to identify files intentionally misnamed
  - → Changing the name from "drugs.doc" to "cmd.com"
  - → See also temporary office files: ".doc", ".xls" → ".tmp"
- Also important after undelete or file carving
  - → The filename may no longer be available, but the content is
- How it works:
  - → Most file formats include some kind of header or footer with specific value at certain positions: "Magic numbers"
  - → Linux: "file" command
- Example:
  - → `# MS Access database`
  - → `4  string  Standard\ Jet\ DB Microsoft  Access Database`
  - → At position 4 the string "Standard Jet DB" is expected
  - → Format: Position Type Value Document-type

"Magic number" examples

- "JFIF" JPEG images

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	FF	D8	FF	E0	00	10	4A	46	49	46	00	01	02	01	00	48	ÿØÿà JFIF H
00000010	00	48	00	00	FF	ED	0B	D8	50	68	6F	74	6F	73	68	6F	H ÿí ØPhotosho
00000020	70	20	33	2E	30	00	38	42	49	4D	03	ED	00	00	00	00	p 3.0 8BIM í

Note: Not immediately at the start of the file!

- "GIF8" GIF images

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	47	49	46	38	39	61	EE	00	D3	00	F7	00	00	00	00	00	GIF89aî Ó ÷
00000010	80	00	00	00	80	00	80	80	00	00	00	80	80	00	80	00	▌ ▌ ▌▌ ▌▌ ▌
00000020	80	80	C0	C0	C0	C0	DC	C0	A6	CA	F0	04	04	04	08	08	▌▌ÀÀÀÀÜÀ¦Êð

Also "magic": FF D8

- 0x89"PNG" PNG images

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52	▌PNG IHDR
00000010	00	00	02	80	00	00	01	E0	08	06	00	00	00	35	D1	DC	▌ à 5ÑÜ
00000020	E4	00	00	00	09	70	48	59	73	00	00	0B	13	00	00	0B	ä pHYs

- MS Access Database

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	00	01	00	00	53	74	61	6E	64	61	72	64	20	4A	65	74	Standard Jet
00000010	20	44	42	00	01	00	00	00	B5	6E	03	62	60	09	C2	55	DB µn b` ÂU
00000020	E9	A9	67	72	40	3F	00	9C	7E	9F	90	FF	85	9A	31	C5	é©gr@? ▌~▌▌ÿ▌▌1Å

Identification example

- Example file: "cmd.com"
  - → Note: Both "command.com" and "cmd.exe" do exist in "C:\Windows\System32" (Windows command line)!
- Output on a Linux machine:
  [user@host ~]# file cmd.com
  cmd.com: PDF document, version 1.4
- Suggested actions:
  - → Make a copy to a different disk
    - » Keep original disk and file unchanged!
  - → Rename extension to PDF
  - → Open with Acrobat Reader

cmd.com

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	25	50	44	46	2D	31	2E	34	0A	25	C7	EC	8F	A2	0A	31	%PDF-1.4 %Çì▮¢ 1
00000010	20	30	20	6F	62	6A	0A	3C	3C	0A	2F	54	79	70	65	20	0 obj << /Type
00000020	2F	43	61	74	61	6C	6F	67	0A	2F	4F	75	74	6C	69	6E	/Catalog /Outlin

Creating a timeline

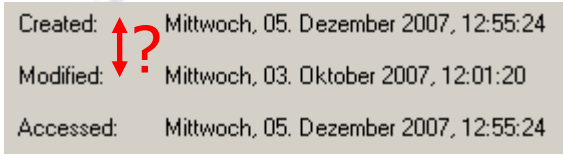
- Timeline: When any/certain actions were taken
  - → Take care: Usually all you get is computer local time!
- May contain various elements
  - → When the computer was started/stopped
    - » Use of company resources outside working hours
  - → When certain files were created/deleted/modified/accessed
    - » Creation: E.g. rootkit installation
      - – Note: Linux typically has no creation time!
      - – The "C" time is the time of the last change of the inode; this might be the creation time, but can be modified later as well through various other actions!
    - » Modification: Modification date past the date stated within
      - – Example: Backdating letters, modifying balance sheets, …
    - » Deletion: After notice of proceedings → Evidence of destruction
  - → When a certain user was logged in/active

Creating a timeline

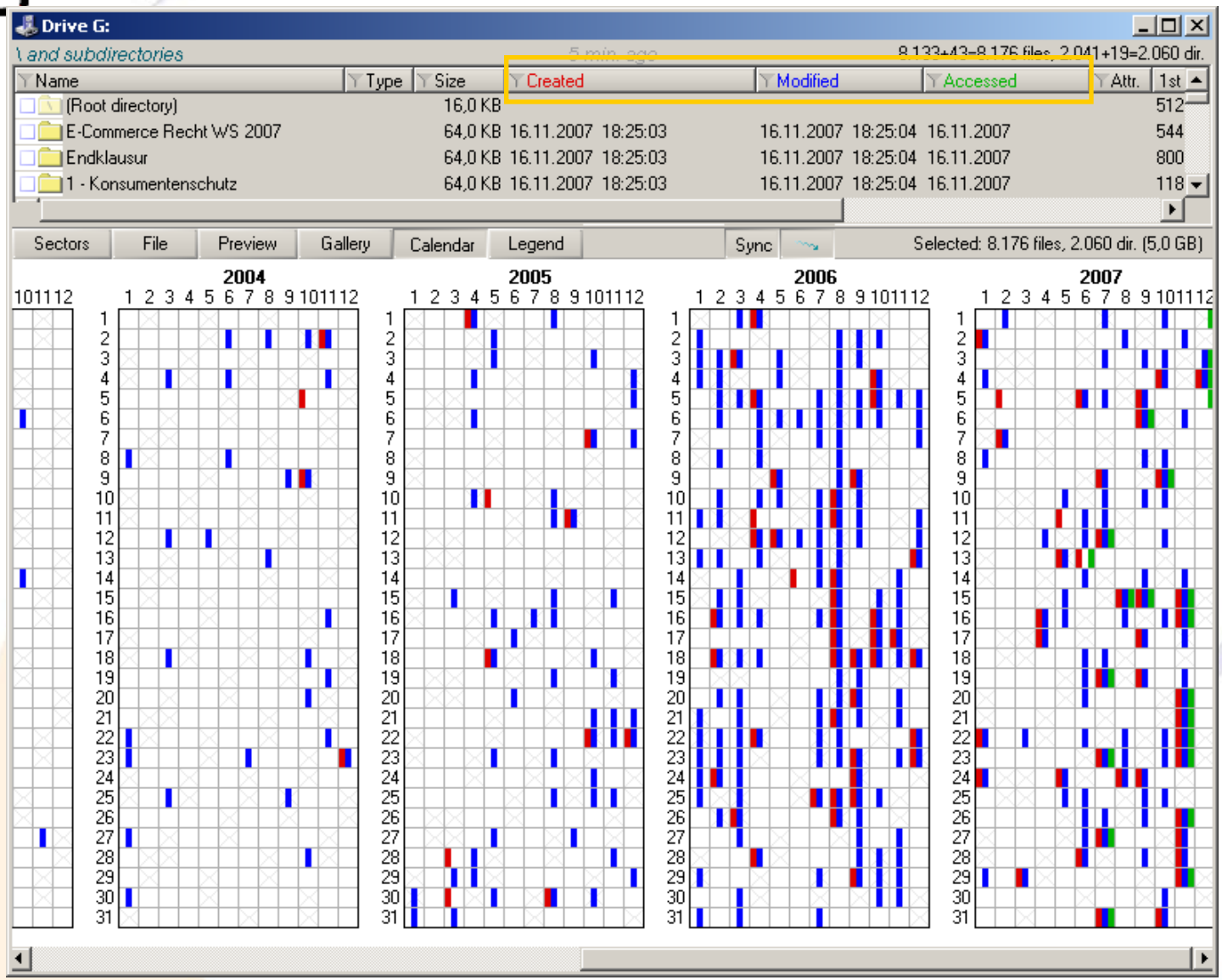
- Sources: MAC time of files, log files / Registry
  - → HKLM\System\CurrentControlSet\Control\Windows\ ShutdownTime: 64 Bit Hex datetime value
- Hints:
  - → Compare e.g. web cache files to their timestamps to detect clock skew!
  - → Look for inconsistencies in the naming of system restore points (which are created in increasing numbering and are timestamps, as they are files, directories, etc.)

MAC

- MAC = Modification, Access, Creation time
  - → Some file systems have other metadata as well!
  - → Access time is fragile: Most actions on a file will change it!
    - » Usually not: Appearing in a directory list
    - » Should: Open for display, copy (source & destination)
  - → Modification: When the file was written to
- Note: Modifying these values depends on the OS
  - → On most systems changes can be forbidden
    - » HKLM\SYSTEM\CurrentControlSet\Control\FileSystem\NtfsDisableLastAccessUpdate → 1 (Default in Vista!)
    - » Linux: mount –o noatime
- Windows specialty: Copying files retains M, but sets new C!
  - → Creation after modification:
    A hint that the file was copied here
  - → Not reset on extracting files from an archive!

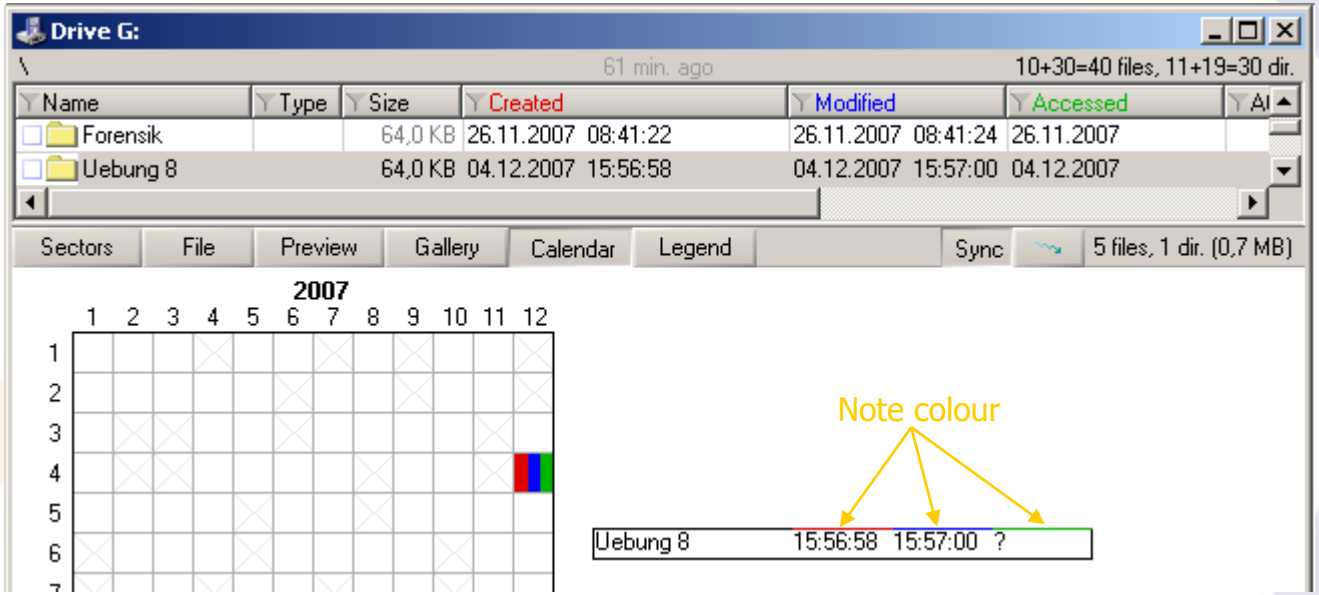


Timeline based on MAC: Example



- Access dates only after 13.6.2007
- Creation dates are rather recent, compared to modification dates
- Files must have been copied there
- Older C dates: Probably extracted from ZIP files!
- Notice the "blue line" in 8/2006: Continuous work over the weekends!
- Gray crosses: Weekends!

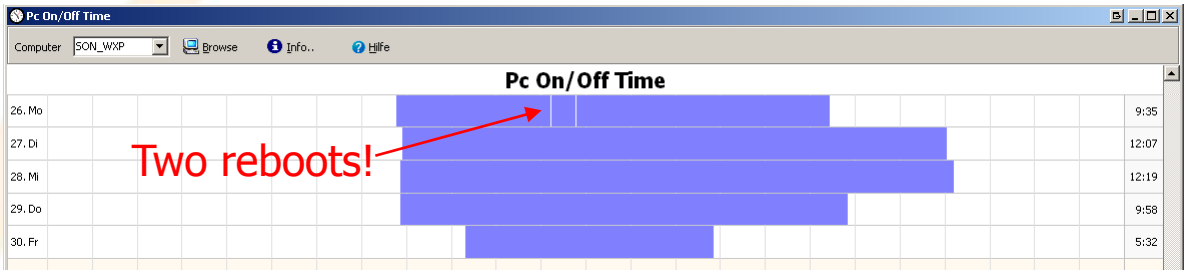
Timeline based on MAC: Example



- File information on hovering the mouse
  - → Note: The hovering is not quite correct: The access date is not shown in the popup, although marked in the calendar and shown in the directory view above!

Startup and shutdown information

- Recorded in the system log: Detailed time
  - → When this is cleared, the information is gone!
    - » Traces may remain on disk → partial information



- Based on MAC times of all files and all log entries
  - → Results only in vague times: When the computer was definitely on (single last shutdown time: Registry time)
    - » But it might have been on at other times as well …
- Manipulating the local clock allows falsifying such data
  - → But this is difficult: All file times must match these values too!
- Linux is similar to Windows: Specific entries/MAC+whole log

log2timeline

- Tool for creating a timeline from various sources
  - → Originally written in Perl, now a new backend in Python is under development
- Scans data from many sources, including:
  - → Apache/IIS/Squid logs, browser history/bookmarks (Chrome, Firefox, IE, Opera), Windows event log/Syslog, EXIF, MAC-time, OPenXML documents, pcap (packet sniffer), PDF metadata, prefetch files, recycle bin, restore points, setupapi log, XP firewall log, …
- Separated into backend (collecting data) and frontend (searching and presenting data)
- Stores data typically in SQLite (but exports to CSV, … too)

Conclusions

- The first and most important aspects of forensic are the
  Three "P's" of evidence: "preserve, package, protect"
  - → This especially includes using write blockers
- Computer forensics is not only undeleting files
  - → There are many small but important areas as well, e.g.
    - » Partition table examination
    - » E-Mail / Web browser forensics
    - » Recognizing files
    - » Creating timelines
    - » Investigating the Windows registry
    - » Recycle bins, LNK files, …
- What is therefore needed: Caution
  - → And a good list of where what information might be found, to acquire knowledge/expertise in this area if needed!

Questions?

Thank you for your attention!

© Michael Sonntag 2013

Links

- http://tech.groups.yahoo.com/group/hashkeeper/
- http://www.nsrl.nist.gov/
- http://www.utica.edu/academic/institutes/ecii/publications/articles/EFE36584-D13F-2962-67BEB146864A2671.pdf
- http://www.foi.se/upload/rapporter/foi-computer-forensics.pdf
- http://log2timeline.net/ http://log2timeline.kiddaland.net/