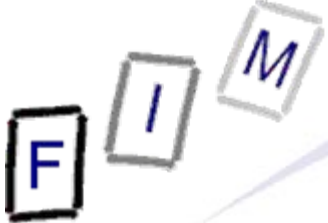Mag. iur. Dr. techn. Michael Sonntag

Web-browsing history

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

- The elements of web-browsing history and intentionality
- HTTP – Hypertext Transfer Protocol
  - → Cookies
- Internet Explorer
  - → File locations
  - → The index.dat file format
  - → Example
- Date/Time formats
- Firefox
  - → File locations
  - → Cookies, history, cache
- Webmail reconstruction example

The elements of web-browsing history

- History
  - → The list of URLs visited (at which time, …)
  - → Provides general information on time and location of activity
    - » URL's may also contain information: GET requests
      - – Example: Google searches
- Cookies
  - → Which websites were visited when + additional information
  - → May allow determining whether the user was logged in
  - → Can survive much longer than the history
    - » Depends on the expiry date of the Cookie and the configuration
- Cache
  - → The content of the pages visited
    - » Incomplete: E.g. ad's will rarely be cached (No-cache headers)
  - → Provides the full content of what was seen, e.g. Webmail
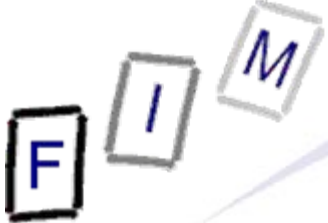   - » More exactly: What was delivered by the server

- Did the user visit the webpage intentionally?
  - → In general: If it's in the cache/history/cookie file: Yes
  - → See also: Bookmarks!
- BUT:
  - → What about pop-ups?
    - » E.g.: Pornography ads (no one sees them intentionally ☺)!
  - → Password protected pages?
    - » But images/JavaScript can easily supply passwords as well when opening a file!
- Investigation of other files, trying it out, content inspection … needed to verify, whether a page that was visited, was actually intended to be visited ("intentionality")
  - → Usually this should not be a problem:
    - » Logging in to the mail
    - » Visiting a website after entering log-ins
    - » Downloading files

Web browsing procedure

1. User enters the URL
2. Browser determines the IP address for the host part
3. Browser connect to the IP address (+port if specified)
4. Sends request
   → With additional information, e.g. what compression is allowed
   → May contain cookie(s)
5. Retrieves response
   → Headers and actual content
      » Header may contain cookie
   → Saved to memory (and perhaps the disk in the cache file)
      » Depends on headers, settings, …
6. Connection is closed
   → Note: HTTP 1.1 may keep the connection open for further requests (incl. pipelining). This is especially useful for images from the same site!
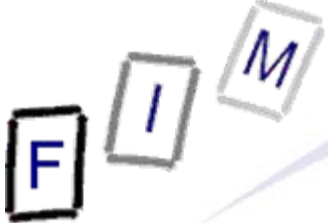
- Basis of HTTP is a reliable stream protocol (usually TCP)
- The HTTP state diagram is very simple
  - » With some exceptions, e.g. authorization
  - → There is only a single request
  - → There is only a single response
- HTTP request methods:
  - → GET: Retrieve some content
    - » Should never change the state on the server!
      - – Especially important if caching takes place somewhere
    - » Parameters (optional) are encoded in the URL
  - → POST: Send data for processing and retrieve result
    - » To be used for requests changing the server state!
    - » Parameters are sent in the request body
  - → HEAD, PUT, DELETE, TRACE, OPTIONS, CONNECT
    - » Of less importance

- The response always includes a status code
  - → 1xx Informational
  - → 2xx Success
  - → 3xx Redirection (request should be sent again differently)
  - → 4xx Client side error (e.g. incorrect request, not existing)
  - → 5xx Server side error (should not be retried)
- Caching of HTTP: Commonly performed through proxies
  - → Must either be validated with the source
  - → Or it is "fresh enough" according to client, server, and cache
  - → Note: Browsers often ignore this
    - » E.g. IE can be configured to never check for a newer version even if the cached page is already expired!
    - » This has no influence on what proxies do!

- Local (=browser) caches
  - → If a page is expired, it is not necessarily deleted from the local cache → It might remain there for much longer
  - → Can store even pages marked as "no-cache" and "no-store"
    - » "no-cache": Should not be cached for future requests
      - – But might still be written to disk (e.g. Mozilla)
    - » "no-store": Should only be held in memory
      - – Users are still allowed to use "Save As"!
  - → This cache can be very large and contain very old files
    - » Very important for computer forensics!
    - » Manual deletion or cleaner programs are simple and effective
      - – But must be used every time after surfing
      - – Attention: Many such programs just delete the files, only the more serious ones overwrite them securely!
      - – Also, fragments of files might remain in unused areas, so all free sectors and slack spaces would have to be cleaned every time!
      - – See also swap file/partition, hibernation file
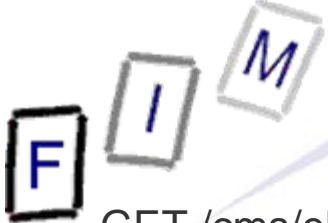
The HTTP protocol example:

http://elearning.fim.uni-linz.ac.at/cms/elearn_ebiz.phtml

GET /cms/elearn_ebiz.phtml HTTP/1.1
Host: elearning.fim.uni-linz.ac.at
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.7) Gecko/20070914 Firefox/2.0.0.7
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.7,de-at;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive

HTTP/1.1 200 OK
Date: Mon, 08 Oct 2007 13:36:22 GMT
Server: Apache/1.3.34 (Debian)
Set-Cookie: hashID=22d68c8b5698827d57f071f43d818456; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT ← Page, not Cookie!
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=iso-8859-1
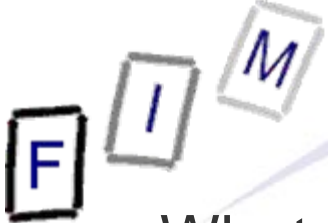
- What is a "cookie"?
  → Small (max. 4 kB) text file with information
    » Originates form the server
    » Stored locally
    » Transmitted back to server on "matching" requests
  → Content (with exemplary data):
    » Name: "session-id"
    » Value: "303-1195544-4348244"
    » Domain: ".amazon.de" } Sent to all requests ("/") of
    » Website path: "/" } subdomains of ".amazon.de"
    » Expiry date and time: 15.10.2007, 00:02:22   None → Till browser is closed ("session cookie")
    » Secure(https): * Will be sent also on non-HTTPS connections
- The data may have any meaning
  → Very rarely this is some "plain-text data"
  → Some part of it might be the IP address or the user name
  → But usually it is just a (more or less!) random unique number
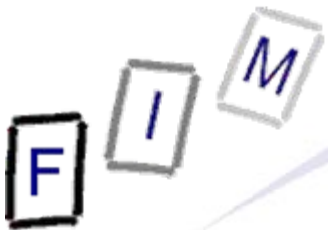
- Where can we find information on what users did with IE?
  - » Att.: Locations change slightly with OS version/language!
  - → <User profile>\Local Settings\Temporary Internet Files\ Content.IE5 ← Also later versions of IE
    (This is the version of the file format, not of the software)!
    - » Cache (webpages, images, applets, flash-files, …)
  - → <User profile>\Local Settings\History
    - » Where the user had been (URLs);
    - » Subdirectories for various time spans
  - → <User profile>\Cookies
    - » Cookies
- Note: Data is deleted from these locations independently!
  - → What is (was) present in one, is not necessarily available any more in the other locations
    - » We must search all three locations and assemble the results

- Each cookie file contains all cookies for a single domain
  - → The information is stored line-by-line; 9 lines = 1 cookie
- Example:

  __utma *Name*

  36557369.378120483.1187701792.1189418701.1190710388.4 *Value*

  hotel.at/ *Domain*

  1088 *Flags*

  2350186496 ⎫
  32111674 ⎬ *Expiration time (UTC; LoVal","HiVal)*

  2116717664 ⎫
  29884241 ⎬ *Creation time (UTC; format as above)*

  * *Secure (here: False)*

  __utmb

  …

- Note: Additional information on the cookies is in the index.dat file in the same directory!
  - → Number of hits, suspected as advertisement
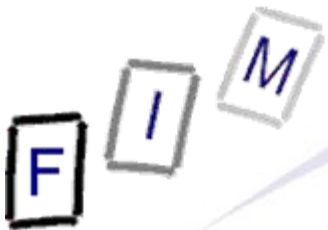
- This structure is the same for cookies, cache, and history
- Overall structure:
  - » Remember: File has bytes in reverse order (little endian)!
  - → Header: Magic number (text), file size, hash table offset, subdirectory names (cache only)
    - » Subdirectory names are referred to by index (0 = first)
  - → Hash table: Length of table, pointer to next hash table, 8-byte hash entries
    - » Entries: 4 bytes flags, 4 bytes record offset
  - → Activity records: Type, length, data (dependent on type)
    - » Type can be REDR, URL, or LEAK
      - – URL: Website visit
      - – REDR: Redirection to another URL
      - – LEAK: Purpose unknown (Possibly: Cache entry deleted, but file couldn't be deleted)
    - » Each record is a multiple of 128 bytes long

Source: http://odessa.sourceforge.net/

- URL records
  - → Last modified time: When the information was modified on the web server
    - » Filetime format; All zero if unknown
  - → Last access time: When the URL was visited
    - » Filetime format!
  - → URL offset
    - » URL itself is Null-terminated; no Unicode – ASCII only!
  - → Filename offset
    - » The name in the cache directory
  - → Cache directory index
    - » In which cache directory the file is stored (index; 0 = first dir)
  - → HTTP header offset
    - » The response headers only; not always present
  - → Hit count: How often visited

- REDR records
  - → Flags: Exact meaning unknown
  - → URL offset
    - » Null-terminated
- LEAK records
  - → Structure similar to URL record; purpose unknown
    - » See above: file couldn't be deleted (open in browser/editor)
- Not all records are necessarily present in the hash table
  - → When deleted, sometimes a record remains and only the hash entry is removed
    - » "Delete history" → Mark as deleted in hashtable
  - → As all records are block-sized (see before), "undelete" is possible without too many problems!
    - – A kind of file system within a file ☺ !
    - » Especially as each record starts with the type, and destroyed records are filled with well-known values (0x0BADF00D)

● Screenshot of header:

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	43	6C	69	65	6E	74	20	55	72	6C	43	61	63	68	65	20	Client UrlCache
00000010	4D	4D	46	20	56	65	72	20	35	2E	32	00	00	00	0B	00	MMF Ver 5.2
00000020	00	50	00	00	80	15	00	00	A0	06	00	00	00	00	00	00	P ▮
00000030	00	00	40	01	00	00	00	00	00	B0	F4	03	00	00	00	00	@ °ô
00000040	00	00	00	00	00	00	00	00	04	00	00	00	83	00	00	00	▮
00000050	35	58	39	54	4E	58	34	45	83	00	00	00	50	4B	38	30	5X9TNX4E▮ PK80
00000060	32	33	51	46	83	00	00	00	4A	54	4A	4E	36	35	58	32	23QF▮ JTJN65X2
00000070	82	00	00	00	42	52	4E	4F	4E	41	54	4D	00	00	00	00	▮ BRNONATM

☐ Magic "number"

☐ File size (0x000B0000 = 704 kB)

☐ Hash table offset (0x00005000)

☐ Cache directory names

Note byte order!

● Screenshot of (start of) hash table :

☐ Magic "number"

☐ Table length (0x0000020 → 32*128 Byte = 4096 Bytes long)

☐ Next hash table offset (0x00014000; absolute from start of file!)

☐ Cache entry (example)

● Screenshot of detail record:

□	Type
□	Record length (3*128B)
□	Last modified time
□	Last access time
□	URL offset
□	Filename offset
□	Cache directory index
	3 → BRNONATM
	Missing; non-cacheable!
□	HTTP header offset
□	Hit count
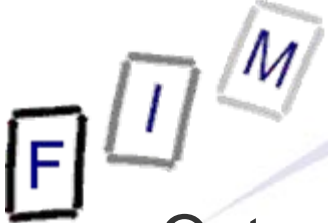
- Output from Pasco:
    - → Type: URL
    - → URL: http://www.amazon.de/Computer-Forensics-Library-Boxed-Set/dp/0321525647/ref=sr_1_14/302-3061595-9808016?ie=UTF8&s=books-intl-de&qid=1191921357&sr=8-14
    - → Modified time: \<Not present in file\>
    - → Last accessed time: 10/09/2007 11:18:48   9.10.2007, 9:18:48 UTC (!!!)
    - → Filename: 302-3061595-9808016[2].htm
    - → Directory: BRNONATM
    - → HTTP headers:
      HTTP/1.1 200 OK
      Content-Length: 120986
      Content-Type: text/html
- Other data:
    - → Record length: 3 (=3*128 = 384 bytes = 0x180)
        - » From 0x035800 to 0x35980

- Filetime: Number of ticks since 1.1.1601
  - → UTC; 100 ns resolution
  - → Usually stored as 8 hexadecimal numbers
- Unix time: Number of ticks since 1.1.1970
  - → UTC, 1 s resolution
  - → May appear as hexadecimal or decimal value (take care!)
    - » Hex: 9940F039
    - » Dec: 971815414
- Attention:
  - → Big endian or little endian?
  - → UTC or a different time zone? Which?
    - » Windows NT stores everything as GMT (according to its own time zone as configured)
  - → Difference of system time to actual time?

Nice tool: http://www.digital-detective.co.uk/freetools/decode.asp

- index.dat example: Filetime – Little endian
  - → B02D8366550AC801 = Tue, 09 October 2007 09:18:48 UTC
    - » Actually: Di, 09 Oktober 2007 11:18:48 +0200
- Cookie example (expiration time; Windows Cookie time):
  - → 2350186496,32111674 = 25.9.2007 08:53:07 UTC
- Firefox cookie (Unix numeric timestamp):
  - → 1192658552 = 17.10.2007 22:02:32 UTC
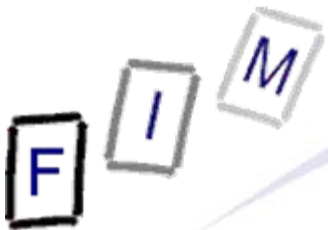
- Time zone issues:
  - → Identify time zone from installation
    - » Alternative: Geographical area of usage of the system
- Delta: Identify delta between computer time and UTC
  - → Attention: This might not necessarily be the same delta as when the timestamp was created!
    - » Manual corrections, time drift (important for longer timespans)
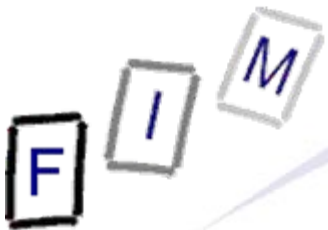
- Attention: Summer time ("daylight saving time", "DST")!
  - → Sometimes its UTC+1, but at the other dates it's UTC+2 !
    - » Austria: 29.3.2009-25.10.2009 → UTC+2; Rest of year: UTC+1
    - » Note: Dates of start/end changed over the years
      - – Was the corresponding patch applied to the computer? When?
      - – Windows: Registry stores start/end date, …
    - » Usually defined by certain weekends, not dates!
      - – Current AT: Last Sunday in March to last Sunday in October
  - → Does the system account for this?
    - » Timestamp stored as UTC or local time?
      - – NTFS: UTC; but FAT: Local time
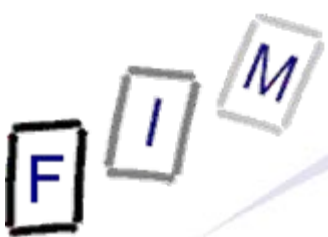
- Information is stored in the file "index.dat"
  - → File format see before!
  - → Again: Content is not necessarily the same as in other files
- Additionally: Several subdirectories for the actual files
  - → Note: These receive "random" filenames to avoid collisions
    - » Typically with "[1]", "[2]", … added at the end
    - » The files itself are NOT modified; URL's are kept the same!
      - – Recreating pages: Must "load" the URLs from the cache too
      - – "Transparent proxy" is needed
- The URL also contains GET parameters
  - → These might also be interesting!

    http://www.hotel.de/Booking.aspx?h_rooms=1&h_fbrs=1&h_step=3&h_departure=9/1/2007&h_arrival=8/26/2007&h_rmod=0&h_sbl=/Search.aspx?hs_arrival=8/26/2007&hs_destination=Lübeck&hs_circum=0&hs_landisoa3=DEU&hs_locationnr=37547&hs=2&hs_departure=9/1/2007&hs_ltype=1&hs_validate=2&hs_llat=53,86626&lng=EN&hs_llong=10,67468&lng=EN&h_persons=1&h_validate=1&h_hmid=50727&h_persons_total=1

    Booking a hotel for one person in Lübeck (Germany) from 26.8.2007 till 1.9.2007
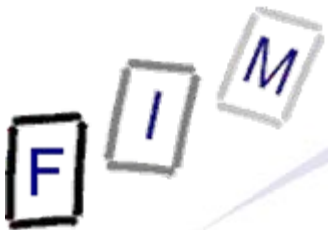
- Directory 5X9TNX4E:
  - → nav_logo3[1].png
- Directory BRNONATM:
  - → google[1].htm
  - → logo_plain[1].png
  - → google[1]
    LEAK record: Some binary data
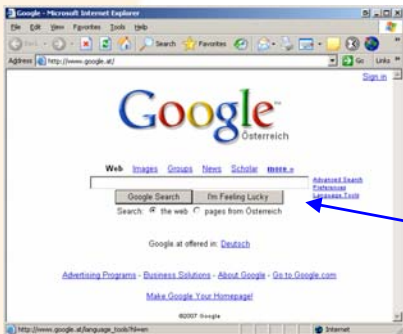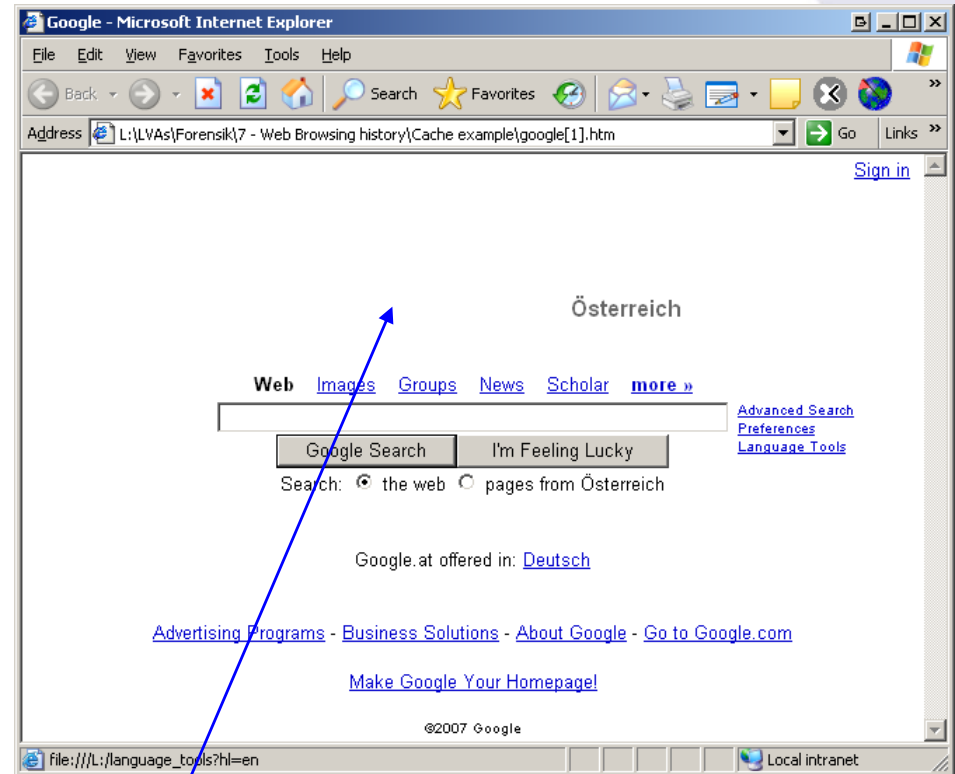
Directly opened as a file in the cache directory

Note: Images are missing because of modified file names (original ones in index.dat)!

Webpage
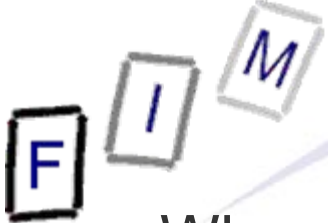
● Where can we find data on what users did with Firefox?
  » Profile ID is a random string generated once
  → \<User profile>\Local Settings\Application Data\Mozilla\ Firefox\Profiles\<Profile ID>\ Cache
    » Cache (webpages, images, applets, flash-files, …)
  → \<User profile>\Application Data\Mozilla\ Firefox\Profiles\<Profile ID>\ history.dat
    » **Extremely** strange file format ("Mork"); 2.x (3.x → SQLite!)
    » There does exist an exporter ("Dork")
  → \<User profile>\Application Data\Mozilla\ Firefox\Profiles\<Profile ID>\ cookies.txt
    » Cookies; Tab-delimited text file

● Easy cache access: URL "about:cache"
  → Also extensions available for directly viewing cached files
    » Should only be used on write-protected disks/images!
  → Firefox has two caches: In-memory and on disk
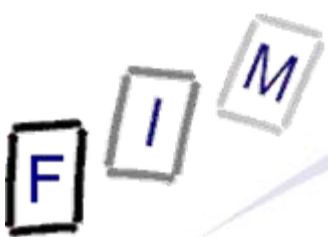
- Simple text file with tab-delimiters: Single line per cookie
- Format:
  - → Domain: ".amazon.de"
  - → Domain access: "TRUE"
    - » Probably a security setting
  - → Path: "/"
  - → Secure : "FALSE" (= Sent over any type of connection)
  - → Timestamp: 1192658552 (=17.10.2007 22:02:32 UTC)
    - – Local time: Do, 18 Oktober 2007 00:02:32 +0200 (Sommerzeit!)
    - » Format: Unix numeric value
  - → Name: "session-id"
  - → Value: " 302-0868837-0800841"
- Example:
  - → .amazon.de   TRUE   /   FALSE   1192658552 session-id 302-0868837-0800841

- Stored in a "strange" format, a kind of database
  - → Replaced in version 3 with a real DB (SQLite)!
    - » Examination quite simple: SQL queries!
- V2: Very difficult to parse, except through Firefox
  - → But there is an exporter, generating a tab-delimited file
- Example:
  - → C7D0D   3   2007-10-08 14:44:44   2007-10-08 14:47:07
    http://www.amazon.de/ref=rd_www_amazon_at/?site-redirect=at
  - → ID of the visit: C7D0D
  - → Visit count: 3
  - → First visit date: 2007-10-08 14:44:44
  - → Last visit date: 2007-10-08 14:47:07
  - → URL: http://www.amazon.de/......
  - → Take care of timezone of dates!
    http://www.forensicswiki.org/wiki/Mozilla_Firefox_3_History_File_Format

- The cache consists of 4 files plus the data files
  - → 1 cache map: Hash table for entries ("_CACHE_MAP_")
    - » Header plus 8192 records of cached elements
      - – Record: Hash number, eviction rank, data / metadata location
    - » Data may be saved within cache block file (below) or separately
  - → 3 cache block files ("_CACHE_00?_")
    - » Bitmap header and some cache content and/or metadata
    - » Varying block sizes: cache 1 = 256, 2 = 512, 3 = 1024 Bytes
      - – Maximum block count per data: 3 (→ up to 3072 Bytes)
  - → Data files: If the content doesn't fit into the cache blocks
    - » Filename = <Hash number><type><generation number>
      - – Type: d = cache, m = metadata (rare!)
      - – Generation number: Lowest byte of location
    - » No filename extension! → Filename doesn't tell file type!
      - – If known → Rename → Original file

● Cookies:

→ www.gmx.net/de/

  » Visits   1

  » moveinBrowser

  new%20MoveinData%28%29%2Eunpickle%28%7B%22viewed%22%3A%201%2C%20%22closed%22%3A%20false%2C%20%22latest%22%3A%20new%20Date%281192174225718%29%7D%29

    – Decoded: new MoveinData().unpickle({"viewed": 1, "closed": false, "latest": new Date(1192174225718)})

    – Decoded date (Unix): Fr, 12 October 2007 07:30:25 UTC

→ gmx.net/

  » GUD

  bMDEpJi1JPF9xN0JINkUyQkExJSIhJxweJBkeGyAvLjcsLDQpKzJCSzEIIiEnHB8dGRwcIC83Ny8tNC0uMktBMSMtSzksIh0gGw==

    – Mime encoded, but is just a binary value

    – Probably a unique ID for session handling

→ logout.gmx.net/

  Sa, 13 Oktober 2007 07:33:24 UTC

  » POPUPCHECK   1192260804812
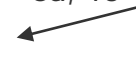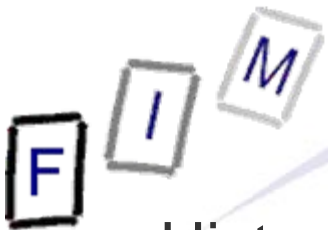
- History (pasco; adjusts for local time zone!): = 12.10.2007 7:30-7:33 UTC!
  → Modified/access time: 10/12/2007 09:30 until 09:33
    » Local time of event: Western European DST (=+2)
      – But shown according to the time zone set at the moment of the analysis; physically stored as UTC time!

- URLs (selection):
  → sonntag@http://www.gmx.net/de
    » User visited GMX homepage
  → sonntag@http://service.gmx.net/de/cgi/login
    » User logged in to GMX
  → sonntag@http://service.gmx.net/de/cgi/g.fcgi/mail/index?CUSTOMERNO=10333901&t=de1690301692.1192174366.c35ea10d&FOLDER=inbox
    » User visited his inbox
  → sonntag@http://service.gmx.net/de/cgi/derefer?TYPE=2&DEST=http%3A%2F%2Fwww.gmxattachments.net%2Fde%2Fcgi%2Fg.fcgi%2Fmail%2Fprint%2Fattachment%3Fmid%3Dbabgehj.1192174412.25124.s9vnnjbfon.74%26uid%3DKxs5Dm8bQEVsw%252FqY9HVpw45KNTg2NcIR%26frame%3Ddownload
    » User opened an attachment
  → sonntag@http://www.gmxattachments.net/de/cgi/g.fcgi/mail/print/attachment:/filename/Lebenslauf.doc?mid=babgehj.1192174412.25124.s9vnnjbfon.74&uid=Kxs5Dm8bQEVsw%2FqY9HVpw45KNTg2NcIR&frame=attachment
    » User downloaded an attachment called "Lebenslauf.doc"
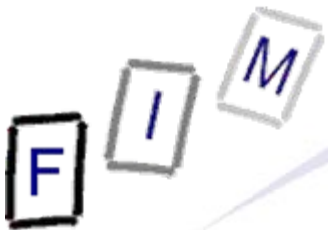
- Cache:
  - → 282 entries
    - » Images (GIF, JPG)
    - » Stylesheets (CSS)
    - » JavaScript (JS)
    - » HTML files (HTML)
      - – Only static files, login screen, etc.!
- What is missing are the actual E-Mails
  - → These are not cached on disk
    - » In previous versions they might have been cached
      - – Probably depending on server, not version of Internet Explorer!
  - → So webmail is not necessarily recoverable, but perhaps in some instances
- Note: The cache only contains, what is sent to the computer
  - → Locally drafted E-Mail is "form input" which is **never** cached!

- Typed URLS: Visited sites
- Form history and stored passwords
  - → For identifying visited sites and accessing them
  - → Often encrypted, but decryption programs exist
- Search history: What was the person looking for?
- Blocked sites: If the popup-host of a site was blocked, the site itself was probably visited!
  - → Manually unblocked sites obviously interesting!
- Certificate store: To identify secured sites visited often
  - → Might include client certificates, which act as a kind of key
- Download history: What file(names) were downloaded
  - → And where they were stored locally (name; for searching)
- Installed add-ons (browser controls)
- Language preferences and all other configuration options

- Allows Browsing without leaving traces (but see below!)
- Additional feature: Prevent Sites from sending data to other sites (InPrivate Filtering)
  - → IE traces third party content; if it appears on more than 10 (can be modified from 3 to 30) sites visited, it is blocked in InPrivate Browsing mode ("Ads" or similar)
    - » Must be activated manually each time (works per-session)!
    - » Can also be activated in non-private browsing mode
  - → Complete blocking (no third-party content) can be set manually; exceptions can be configured as well
- InPrivate Browsing does not store:
  - → New cookies (existing can still be read!), history entries, form data, passwords, typed URLs, search queries, visited links
  - → Toolbars and extensions are disabled
- Will keep: Bookmarks, downloaded files, Flash cookies

- InPrivate Browsing still stores files in the cache on the disk, but deletes them when closing the window
  - → This means, traces **WILL** remain on the disk!
  - → Which can be found through careful investigation
- Data might remain on disk because of swapped memory
- Reconstructing the history:
  - → Not available directly (not stored!)
    - » Article unclear about this; some parts might remain
  - → But possible through the cache, which contains the last access time of every stored element!
- No advantage regarding:
  - → Eavesdropping: ISP, … can still copy the connection
  - → Servers: May even be able to identify that this mode is used

- Firefox does not store
  - → History entries (incl. intelligent address bar), search queries, download history, form data, cookies, cache, typed URLs, passwords, visited links
- Will keep: Bookmarks, downloaded files, Flash cookies
- Same features as IE8
  - → Except third party elements
    - » Cookies can be filtered
    - » Images too, but not through the UI!
      - – about:config → permissions.default.image=3 (no third party images)
    - » Scripts etc.: NoScript or other extensions
- Extensions remain active!
  - → Configuration (e.g. third party images) is the same

Alternative: Network forensics

- Copying the network traffic allows reconstructing the page
  - → This requires live access on a router, intercept station etc. at the moment the user browses the web
    - » Wiretapping → Very difficult to do legally!
  - → Only very limited usability!
- Result: Trace of the individual packets
- Requires:
  - → Reassembly of the TCP connection (difficult → tool needed!)
  - → Splitting into the individual requests (HTTP 1.1 pipelining!)
  - → Manual reassembly to a "viewable" local page
    - » Inspection of the HTML code is quite simple
- Following pages: Wireshark example
  - → Left out: IPv6 DNS query, redirect to actual homepage, detailed analysis of the individual packets (not interesting!)

Follow TCP Stream _ □ x

Stream Content

```
GET /content HTTP/1.1
Host: www.jku.at
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; de; rv:1.9.2.18) Gecko/20110614 Firefox/3.6.18
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Cookie: sselect=people; jku_query_config=%7B%22pagesize%22%3A5%2C%22history%22%3A%5B%223%3Awildling%22%2C%221%
3Airrgeher%22%2C%221%3Anussbaumer%22%5D%7D
DNT: 1
Connection: keep-alive

HTTP/1.1 200 OK
Date: Wed, 17 Aug 2011 14:17:39 GMT
Server: Apache
Content-Length: 22187
Vary: X-language,X-AJAXable
Pragma:
Cache-Control: public, max-age=0, s-maxage=3600
Content-Type: text/html; charset=utf-8
Connection: close


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- HTML -->
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">


<head>
```
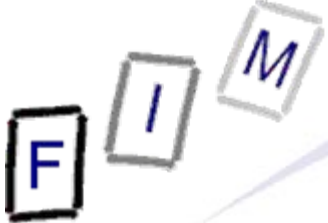
Find Save As Print Entire conversation (22982 bytes) ▼   ○ ASCII ○ EBCDIC ○ Hex Dump ○ C Arrays ● Raw

Help   Filter Out This Stream   Close

- What a user did with a web browser can usually be reconstructed quite good
  - → Especially Internet Explorer: Deleting the index.dat files is almost impossible
    - » Dedicated "cleaner" programs are needed
    - » Information may be stored multiple times
- Reconstructing the content of web-based E-Mail is difficult
  - → That, which, … can be done
  - → But content is typically not cached and therefore unavailable
- A variety of programs exist to investigate these files
  - → Few of them are free
  - → File formats are often not at all/badly documented
- Timestamps are very important, but many formats exist
  - → Identifying delta and timezone are paramount!

Questions?

Thank you for your attention!

- Anderson, Keith: Firefox history exporter:
https://bugzilla.mozilla.org/show_bug.cgi?id=241438
    (Entry at 2006-03-17 09:10:47 PDT)