Christian Praher, Michael Sonntag

Windows Forensics – Exercises

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

Overview/Schedule

- Introduction to the tools and the (Cygwin) environment
- Recycle bin forensics
- Case study I – Thumbs.db
- Case study II – Prefetch and event log
- Case study III – WLAN forensics
- Case study IV – Timeline forensics

Start

- The system is a virtual machine
- Windows XP is installed, but not activated
  - → This is not necessary for the tasks we are doing here!
  - → Please just cancel the reminder upon logging in
- Administrator account login:
  - → User: “Administrator”; password: “admin”
- Useful tools are installed and icons on the desktop

Environment

- More **incident response** than forensics
  - → No clear separation between the suspect system and the investigation environment
    - » Windows system is host of the forensics analysis tools
    - » At the same time the very same Windows system is also the subject of the investigation
  - → Real world scenarios could be e.g.:
    - » System administrator or boss asks about an incident that happened at the company
    - » Examination of the own system regarding a suspected malware infection
- Uses free and/or open source tools for the analysis
  - → Tools are mostly simple applications or scripts written in C, Perl, and/or Python
- **Cygwin** environment for running Linux/Unix tools on Windows
  - → Simple applications can be compiled directly as Windows binaries due to the Windows POSIX 1003.1 subsystem
  - → For more sophisticated applications Cygwin offers the most important Linux/Unix APIs on Windows in form of a shared library (.dll), which applications can link against
  - → Additionally, Cygwin provides a tool chain and most important a powerful shell (bash) for Linux/Unix look and feel on Windows
    - » Attention: in the Cygwin shell the Windows paths are modelled as Unix paths (incl. the drive letters!) and are translated: `/cygdrive/<drive_letter>`
    - » `E`.g. `C:\` becomes `/cygdrive/c/`)

Sidetrack: Date/time formats

- Filetime: Number of ticks since 1.1.1601
  - → 8 byte structure that stores time in UTC with 100 ns resolution
  - → Usually stored as 8 hexadecimal numbers
  - → MSDN: http://msdn.microsoft.com/en-us/library/windows/desktop/ms724284(v=vs.85).aspx
- Windows System Time
  - → 32 byte structure that specifies a date and time, using individual members for the month, day, year, weekday, hour, minute, second, and millisecond.
  - → Either in coordinated universal time (UTC) or local time, depending on the function that is being called.
  - → MSDN: http://msdn.microsoft.com/en-us/library/windows/desktop/ms724950(v=vs.85).aspx
- Unix time: Number of ticks since 1.1.1970
  - → 4 byte structure that stores time in UTC with 1s resolution
  - → May appear as hexadecimal or decimal value (take care!)
    - » Hex: 9940F039
    - » Dec: 971815414
  - → MSDN: http://msdn.microsoft.com/en-us/library/1f4c8f33(v=vs.71).aspx
  - → Unix Time and Windows Time: http://blogs.msdn.com/b/mikekelly/archive/2009/01/17/unix-time-and-windows-time.aspx

Sidetrack: Date/time formats

- 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

- Tools / Useful Links
  - → Linux date command
    Timestamps can be converted with the `@` sign,
    e.g. `date -s @1321877486`
  - → Only Unix timestamp converter
    - » http://www.gaijin.at/olsutc.php
  - → Time converter tool
    - » http://www.digital-detective.co.uk/freetools/decode.asp
  - → FileTimeConverter
    - » http://www.silisoftware.com/tools/date.php

Recycle Bin

- To get started we will examine the contents of the recycle bin that are stored in Windows XP under `C:\RECYCLER\<USER_SID>\INFO2`
- Since we are working with SIDs in the recycler directory, identify all users and their SIDs via the Windows registry
  - → Open the graphical registry editor `regedt32.exe` and navigate to
    - » `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList`
  - → What users can be found there?
    - » Hint: For the meaning of special SIDs, have a quick look at http://support.microsoft.com/kb/243330
- Which users have (already) recycled items at least once on the system at some previous time?

Recycle bin

- The INFO2 file structure
  - → Binary file
  - → Contains the file name twice: ASCII and Unicode
  - → 20 Byte file header; Bytes 12-13 (-15?) are record size
    - » Record size is usually 2003 = 0x0320 = 800 Bytes
- Record structure
  - → 260 Bytes: Original file name (ASCII), including path
  - → 4 Bytes: Record number (starting at 0)
  - → 4 Bytes: Drive number (00 = A, 01 = B, 02 = C, …)
  - → 8 Bytes: Deletion time (FILETIME format, UTC)
  - → 4 Bytes: Physical file size (=Bytes on disk!)
    - » Therefore always multiples of cluster size
    - » Actual file size: See directory entry of the file itself
  - → 520 Bytes: Original file name (Unicode), including path

Recycle Bin

- To be able to view the hidden INFO2 files, we have to list them with either cygwin with `ls`, or in the normal windows command shell with
  `dir /a`
- Have a look at one of the INFO2 files with the HxD hex editor, either from within Cygwin or the standard windows command shell
  - → `HxD.exe C:\RECYCLER\<SID>\INFO2`
- Analyse file manually
- Analyse recycler files with rifiuti tool
  - → `rifiuti.exe C:\RECYCLER\<SID>\INFO2`

Case Study I: Thumbs.db

- Some of the users of the machine under investigation are suspected of having viewed illegal images
- You as an investigator have the original illegal images (or at least hashes thereof!)
- Usually it should be enough to compare the hashes of the illegal contents with hashes produced from all (image) files found on the suspect machine
- Unfortunately the images may have already been deleted from the suspects machine (home directories)
- But there still exist preview image database files (`Thumbs.db`[1]) which can help proving that illegal content was viewed

[1] Further infos: http://accessdata.com/media/en_us/print/papers/wp.Thumbs_DB_Files.en_us.pdf

Case Study I: Thumbs.db

- With the help of the still existing `Thumbs.db` files, it can still be shown that the illegal contents have been viewed
  - → With special tools it is possible to extract the thumbnail images from the `Thumbs.db` file
- It is of course not possible to create hashes of the extracted images and compare those hashes directly with the original forbidden contents
  - → The images in den `Thumbs.db` file are completely different from their originals
- Solution: We have to create a `Thumbs.db` file of the illegal images we have, extract those images and compare their hash values with the hashes of the found `Thumbs.db` pictures!

Case Study I: Thumbs.db

- In the directory `C:\forensics\classified_images` (→ `/cygdrive/c/forensics/classified_images`) you find some “illegal” images
- Create a `Thumbs.db` file of these images by viewing them as thumbnails
- Use Cygwin and the tools `vinetto` and `md5deep` to extract the thumb pictures of the `Thumbs.db` and create MD5 hashes for the images
  - → Open Cygwin shell
    - » You find the contents of the Windows drives under `/cygdrive/<drive_letter>`, so go to /`cygdrive/c/forensics/classified_images`
    - » Create a directory for the extracted images and the created extraction report, e.g. `thumbs_extracted`
  - → Extract the `Thumbs.db` with vinetto
    - » `vinetto -o thumbs_extracted -H Thumbs.db`
      - – (you must be within the folder where the `Thumbs.db` file is)
    - » Have a look at the extracted images and the generated report
  - → Create md5 hashes of the extracted images with md5deep
    - » Go to the just created directory `thumbs_extracted`
    - » Therein you find a directory `.thumbs`
    - » Create a file of hashes for these files with: `md5deep -r .thumbs > hashes.txt`

Case Study I: Thumbs.db

- Now search through every single system user and identify any `Thumbs.db` files
  - → You can restrict yourself to the files found in `C:\Documents and Settings\<username>\My Documents\My Pictures` for each user
- Extract each `Thumbs.db` in the same fashion as described before
  - → `vinetto -o thumbs_extracted -H Thumbs.db`
    - » You must be within the folder where the Thumbs.db file is and the direcotry `thumbs_extracted` needs to be created before
- Now, the tool `md5deep` allows you to create hashes of these just extracted images and compare them on the fly to a file of existing hashes (which are of course the hashes of the illegal image thumbs)
  - → `md5deep -m /cygdrive/c/forensics/classified_images/thumbs_extracted/hashes.txt -r .thumbs`
- The output of the `md5deep` hash comparison is a list of files for which the hash values match
  - → Note down the users and the images that matched the search
  - → Which users were found to have viewed which illegal images?

Case Study II: Prefetch File / Event Logs

- In this scenario we need to identify which user most likely used a certain application found on the suspect machine
- On the machine there is an application named "`Putty.exe`", which employees are forbidden to use during work time
  - → Putty allows administering remote machines
- All employees claim that this application was already installed and was not used by them
- By analyzing the Windows prefetch[1] files and the security event log, try to confirm or invalidate the allegations of one employee having used the application during work time

1) For more info see, e.g.: http://msdn.microsoft.com/en-us/magazine/cc302206.aspx

Case Study II: Event Logs

- On a Windows XP machine three event logs exist by default
  - → Application
    - » Logs application specific things, determined by application developer.
  - → Security
    - » Logs security related events, e.g. (un)successful logon/logoff, object access, …
  - → System
    - » Logs events concerning the Windows system, like e.g. failed drivers, etc. Contents are determined by Windows.
- You can view the event logs with the standard Windows event viewer GUI. (Start -> Control Panel -> Administrative Tools -> Event Viewer)
- However processing large amounts of log data can become quite cumbersome with this graphical tool.
  A non graphical, in terms of query possibilities, very powerful alternative is the tool Log Parser (`LogParser.exe`, available as download from Microsoft)
  - → http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=24659

Case Study II: Event Logs

- `Logparser.exe` allows fine grained analysis of all kinds of (event) logs via a SQL like query language
  - → For more information on log parser, see e.g.
    - » http://www.stevebunting.org/udpd4n6/forensics/logparser.htm
    - » http://www.msexchange.org/tutorials/using-logparser-utility-analyze-exchangeiis-logs.html
    - » http://www.codinghorror.com/blog/2005/08/microsoft-logparser.html
    - » http://technet.microsoft.com/en-us/library/bb878032.aspx
    - » http://support.microsoft.com/kb/910447/de
- Tables that can be queried for Windows event logs
  - → Application
  - → Security
  - → System
- Schema of these tables (columns)
  - → EventLog
  - → RecordNumber
  - → TimeGenerated
  - → TimeWritten
  - → EventID
  - → Strings
  - → SID
  - → Data
  - EventType
  - EventTypeName
  - EventCategory
  - EventCategoryName
  - SourceName
  - ComputerName
  - Message

Case Study II: Event Logs

- **Installed on the system in the directory C:\Program Files\Log Parser 2.2**
  - → No path entry; needs to be run from there
- **Example query**

```
LogParser.exe "SELECT TimeGenerated, EventID, Message FROM Application WHERE
TimeGenerated >= '2011-11-14 11:55:00'"
```

- **Result**

```
TimeGenerated       EventID Message
------------------- ------- ----------------------------------------------------
---------------------------------------------------
2011-11-14 11:55:25 1006    Starting logon task.
2011-11-14 11:55:25 1002    Starting interactive setup.
2011-11-14 11:55:25 1004    Starting user task.
2011-11-14 11:55:27 1005    User task exiting. result code = 0x800704c7, message
 = The operation was canceled by the user.
2011-11-14 11:55:27 1003    Interactive setup exiting. result code = 0x800704c7,
 message = The operation was canceled by the user.
2011-11-14 11:55:27 1007    Logon task exiting. result code = 0x800704c7, messag
e = The operation was canceled by the user.

Statistics:
-----------
Elements processed: 135
Elements output:    6
Execution time:     0.07 seconds
```

Case Study II: Event Logs

- Finally, what we need to know to analyze the logon / logoff events of the users, are the respective event IDs
  (Event type IDs are Windows version specific and considerably changed between XP and Vista. For more information see e.g.: http://www.ultimatewindowssecurity.com/securitylog/encyclopedia/Default.aspx)
  - → Category Logon/Logoff (EventCategory = 2)
    - » successful local logon → 528
    - » successful network logon → 540
    - » user logoff → 538
    - » user initiated logoff → 551
    - » Logon Failure - Unknown user name or bad password → 529
    - » …
  - → There are problems with the logging of the logoff events in various Windows versions
    - » Especially, the “user logoff” event 538 will not be captured many times (e.g. after a restart)
    - » So, always make sure to also capture 551 “user initiated logoff”
    - » See e.g. http://support.microsoft.com/kb/828857

Case Study II: Prefetch Files

- With the help of the prefetch file, it should now be possible to identify
  - → Was the application in question run recently?
  - → If so, which user's login times fit the time determined from the prefetch file best? This is then our suspect user!
- The following MAC times contained in a prefetch file are interesting
  - → Dates of file itself
    - » Created
      - – When was the application first run?
    - » Modified
      - – When was the application run the last time?
    - » Accessed
  - → Inside the prefetch file there is a "last run" timestamp (Filetime format)
    - » When was the application run the last time?
  - → Runs
    - » How often has the application been called (7-bit)
- We use the graphical tool "Windows File Analyzer" to analyze the prefetch files tored in `C:\WINDOWS\Prefetch`
  - → `C:\forensics\tools\WFA\WFA.exe`
    - » Attention: The timestamps of the file (created, modified, accessed) are given in UTC and the last run timestamp inside the file is given in local time (UTC+1)!

Case Study II: Prefetch File / Event Logs

- Now, with the knowledge about the Windows event logs and the prefetch files, try to identify the user(s) who are likely to have used the application `putty.exe`
  - → First, identify when `putty.exe` was used by analysing the Windows prefetch files. From the prefetch files we do at least know when the application was first run and when the application was last run
  - → With the knowledge of the application runs of `putty.exe`, try to identify the users which come into consideration for having run the application, given their logon times
    - » What is an effective query to nail down the users?

Case Study II: Hints

- The Message column for the events with the IDs 528, 538 and 551 contain a very helpful value "Logon ID"
  - → Logon ID is a number (specified as hex value) that associates a logon with the respective logoff
    - » Both share the same logon ID (e.g. "Logon ID: (0x0,0x1D6417)")
- With the knowledge of this logon ID, it is possible to track down one specific logon session
  - → Search for logon events that occurred before the given timestamp
  - → Search for logoff events that occurred after the given timestamp
  - → Associate logons to logoffs with the unique logon ID, where the logon occurred before the timestamp and the logoff occurred afterwards

Case Study II: Hints

- Especially the Message column can be a very rich source of information, by searching through with wildcards (like queries)
  - → E.g. every logon is associated with a numeric logon-ID which connects both a logon and a logoff and can be queried by like

```
LogParser.exe "SELECT TimeGenerated, EventID, Message FROM Security WHERE EventCategory = 2
   AND TimeGenerated >= '2011-11-14 13:00:00' AND MESSAGE LIKE '%0x11e20%'“

TimeGenerated       EventID Message
------------------- ------- ----------------------------------------------------
--------------------------------------------------------------------------------
---------------------------------------------------------------------
2011-11-14 13:37:58 528     Successful Logon: User Name: Doris Domain: WINXP-FOR
ENSICS Logon ID: (0x0,0x11E20) Logon Type: 2 Logon Process: User32 Authenticatio
n Package: Negotiate Workstation Name: WINXP-FORENSICS Logon GUID: -
2011-11-14 13:47:59 551     User initiated logoff: User Name: Doris Domain: WINX
P-FORENSICS Logon ID: (0x0,0x11e20)
2011-11-14 13:48:03 538     User Logoff: User Name: Doris Domain: WINXP-FORENSIC
S Logon ID: (0x0,0x11E20) Logon Type: 2

Statistics:
-----------
Elements processed: 1715
Elements output:    3
Execution time:     0.23 seconds
```

Case Study II: Solution

- Timestamp “Creation” of PUTTY.EXE: 14.11.2011 12:38:21 (GMT)
- Query “All logins before the timestamp”
  - →
    ```
    C:\Program Files\Log Parser 2.2>LogParser.exe "SELECT
    TimeGenerated, EventID, Message FROM Security WHERE
    EventID = 528 AND TimeGenerated <= '2011-11-14 13:38:21'"
    -o:CSV
    ```
    - » Note down the closest Logon IDs: Doris - 0x11E20
- Query “All logins after the timestamp” with the given Logon ID
  - →
    ```
    C:\Program Files\Log Parser 2.2>LogParser.exe "SELECT
    TimeGenerated, EventID, Message FROM Security WHERE
    (EventID = 538 OR EventID = 551) AND TimeGenerated >=
    '2011-11-14 13:38:21' AND Message LIKE '%0x11E20%' " -
    o:CSV
    ```
  - → Output
    ```
    TimeGenerated,EventID,Message
    2011-11-14 13:47:59,551,"User initiated logoff: User Name:
    Doris Domain: WINXP-FORENSICS Logon ID: (0x0,0x11e20) ‟
    2011-11-14 13:48:03,538,"User Logoff: User Name: Doris
    Domain: WINXP-FORENSICS Logon ID: (0x0,0x11E20) Logon
    Type: 2 ‟
    ```
- Use the script “`who_was_logged_in.py`”:
  - → In cygwin `/cygdrive/c/forensics/tools`:
    ```
    python who_was_logged_in.py ‘yyyy-mm-dd hh:mm:ss’
    ```
    - » User “Doris”

Case Study II: Solution

- Timestamp “Embedded” (=“Written”-10s) of PUTTY.EXE: 14.11.2011 14:07:35 (GMT)
- Query “All logins before the timestamp”
  - →
    ```
    C:\Program Files\Log Parser 2.2>LogParser.exe "SELECT TimeGenerated, EventID, Message FROM Security WHERE EventID = 528 AND TimeGenerated <= '2011-11-14 15:07:35'" -o:CSV
    ```
    - » Note down the closest Logon IDs: Doris - 0x35688
- Query “All logins after the timestamp” with the given Logon ID
  - →
    ```
    C:\Program Files\Log Parser 2.2>LogParser.exe "SELECT TimeGenerated, EventID, Message FROM Security WHERE (EventID = 538 OR EventID = 551) AND TimeGenerated >= '2011-11-14 15:07:35' AND Message LIKE '%0x35688%' " -o:CSV
    ```
  - → Output
    ```
    TimeGenerated,EventID,Message
    2011-11-14 15:09:21,551,"User initiated logoff: User Name: Doris Domain: WINXP-FORENSICS Logon ID: (0x0,0x35688) “
    2011-11-14 15:09:25,538,"User Logoff: User Name: Doris Domain: WINXP-FORENSICS Logon ID: (0x0,0x35688) Logon Type: 2 “
    ```
- Use the script “`who_was_logged_in.py`”:
  - → In cygwin `/cygdrive/c/forensics/tools`:
    ```
    python who_was_logged_in.py ‘yyyy-mm-dd hh:mm:ss’
    ```
    - » User “Doris”

Case Studies III and IV

- In these case studies we want to identify illegal activities conducted through attaching USB devices to the computer
- Two USB related incidents should be identified and investigated on the subject machine
  - → WLAN USB dongle (case study III)
    - » Who was probably using the device?
    - » Which WLAN SSID was used?
    - » What was done with the WLAN connection?
      - – Visited web pages
  - → Mass storage USB thumb drive (case study IV)
    - » Who was probably using the device?
    - » Is there evidence that files were illegally copied to the Windows host via that device?
    - » Is it possible to identify if sensitive data has been copied from the Windows host to the USB drive (e.g. theft of company data)?

Case Study III: WLAN

- Identify all USB devices that have been attached to the computer with the tool USBDeview
  - → Launch the tool graphically from `C:\forensics\tools\usbdeview195\USBDeview.exe`
- Which of the users have been using these devices?
  - → What devices are listed?
  - → Interesting columns (local time, not GMT!)
    - » CreatedDate
      - – Time of first use of this very device. E.g. installation time for a WLAN adapter
    - » Last Plug/Unplug Date
      - – Device currently plugged in: Time of plugin
      - – Device currently not plugged in: Time when it was removed
    - » InstanceID
      - – Unique identifier of the device for mapping connection data to the dongle in the registry

Case Study III: WLAN

- Find the user(s) who have been logged in while the dongle was plugged in
  - → In Cygwin: `/cygdrive/c/forensics/tools $ python who_was_logged_in.py 'yyy-mm-dd  hh:mm:ss'`
- Identify connection data of the dongle (e.g. SSID, IP-Address, …) and map the dongle to the one listed by usbdeview
  - → When accessing a WLAN, its SSID is stored: HKLM\Software\Microsoft\WZCSVC\Parameters\Interfaces
    - » Subkeys look like GUIDs with values for "ActiveSettings", "Static#000?", …
    - » The values for "#Static000?" contain the SSIDs at offset 0x14
  - → Note down the GUIDs of the interfaces and search for a link between these GUIDs and the USB device in question (intentified by InstanceID from USBDeview)
    - » Search in the registry for the "InstanceID" of the USB dongle and match the given GUID
  - → IP address information for this connection (last only): HKLM\System\ControlSet00?\Services\Tcpip\Parameters\Interfaces
    - » Look for the same "GUID" key as of the WLAN!
    - » Dhcp*: Data on DHCP server, assigned address, netmask, default gateway, domain, nameservers, …
    - » LeaseObtainedTime/-TerminatesTime: Unix 23 Bit Timestamp
      - – When the Address was received and what is the definite last time it could have been used (but not: **was** used!)
    - » See: "What are Control Sets"?
      - – http://support.microsoft.com/kb/100010

Case Study III: WLAN

- We now know
  - → Who used the dongle
  - → When it was used
  - → Basic connection settings, like e.g. the SSID used
- What is of interest next is what the user did with the Internet connection
  - → In case of this Internet connection, a good starting point is to investigate artifacts left from web browser usage
  - → Every browser has its own way of storing files
    - » In our scenario we restrict ourselves to the Internet Explorer
    - » In practice the browser(s) actually used would have to be identified and then all of them investigated

Case Study III: WLAN

- The Internet Explorer browser stores the 25 most recently manually typed URLs in the registry
  - → HKCU\Software\Microsoft\InternetExplorer\TypedURLs
- We cannot examine this key directly in the `regedit.exe` tool, because only the values (=hive) of the currently logged in user is linked in (see next slide)
- We need to use a third party tool to analyse this user's hive "offline"
- A powerful open source Perl tool to analyse registry hives offline is "RegRipper"
  - → Extendable framework for adding registry-based forensic analysis as Perl scripts
  - → List available plugins:
    - » `C:\forensics\tools\carvey_tools>rip.exe -l`
  - → Run certain analysis against one particular hive
    - » `C:\forensics\tools\carvey_tools>rip.exe -r "Path\To\Registry\Hive" -p "name of plugin"`
  - → Get typed URLs
    - » `rip.exe -r "C:\Documents and Settings\<username>\NTUSER.DAT" -p typedurls`

Case Study III: WLAN – The Windows Registry

- 5 root keys exist:
  - HKLM: HKEY_LOCAL_MACHINE (Computer-specific data)
  - HKU: HKEY_USERS (User-specific data)
  - HKCR: HKEY_CLASSES_ROOT (application settings, file associations, class registrations for COM objects)
    - Link to HKLM\Software\Classes
  - HKCC: HKEY_CURRENT_CONFIG (Current hardware conf.)
    - Link to HKLM\System\CurrentControlSet\Hardware Profiles\Current
  - HKCU: HKEY_CURRENT_USER (Current user's data)
    - Link to HKU\<SID of current user>

- File locations:

HKLM\SAM	%SYSTEMROOT%\System32\config\SAM
HKLM\Security	%SYSTEMROOT%\System32\config\SECURITY
HKLM\Software	%SYSTEMROOT%\System32\config\software
HKLM\System	%SYSTEMROOT%\System32\config\system
HKLM\Hardware	Stored in memory only – non on disk!
HKU\.Default	%SYSTEMROOT%\System32\config\default
HKU\SID	%USERPROFILE%\NTUSER.DAT
HKU\SID_Classes	%USERPROFILE%\Local Settings\ Application Data\Microsoft\Windows\UsrClass.dat

Case Study III: WLAN – Regripper

- Some interesting RegRipper modules

```
> rip.exe -l "list plugins"
> rip.exe -r "C:\Documents and Settings\<username>\NTUSER.DAT" -p typedurls
> rip.exe -r "C:\Documents and Settings\<username>\NTUSER.DAT" -p regtime
> rip.exe -r "C:\Documents and Settings\<username>\NTUSER.DAT" -p ie_main
> rip.exe -r "C:\Documents and Settings\<username>\NTUSER.DAT" -p ie_settings
> rip.exe -r "C:\Documents and Settings\<username>\NTUSER.DAT" -p logonusername
```

Case Study III: WLAN

- With the typed URLs we now know what the user actively typed into the progress bar of the browser, but we do not know exactly when this happened
  - → We only know the time the most recent entry was written, through the write time of the registry key
- To get more information about the browsing activities we need to get information from the browsing history

Case Study III: WLAN – 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

Case Study III: WLAN – Web-browsing history

- 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 e.g. pop-ups?
    - » E.g.: Pornography advertisements!
- 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

Case Study III: WLAN – Internet Explorer: Interesting files/locations

- 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.IE5\
    - » 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

Case Study III: WLAN – Internet Explorer: index.dat structure (1)

- 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/

Case Study III: WLAN – Internet Explorer: index.dat structure (2)

- 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

Case Study III: WLAN – Internet Explorer: index.dat structure (3)

- 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)

Case Study III: WLAN – Pasco

- The open source tool "pasco" (/cygdrive/c/forensics/tools/pasco/bin) can be used to parse index.dat files
  - → Pasco is a Unix command linked against cygwin.dll, so you can run it again from within the Cygwin shell
  - → `$ ./pasco.exe -t ';' /cygdrive/c/Documents\ and\ Settings/Brian/Local\ Settings/Temporary\ Internet\ Files/Content.IE5/index.dat`
  - → `$ ./pasco.exe -t ';' /cygdrive/c/Documents\ and\ Settings/Brian/Local\ Settings/History/History.IE5/index.dat`
  - → `$ ./pasco.exe -t ';' /cygdrive/c/Documents\ and\ Settings/Brian/Cookies/index.dat`
- After the analysis with Pasco, we have a pretty good understanding of what the user did and when this was
  - → Here with CSVed, but normally with a spreadsheet or DB

Case Study III: WLAN – Pasco

- Sample 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

Case Study IV: Timeline Forensics

- Based on an example of Harlan Carvey
  - → Author of the books (amongst others)
    - » Digital Forensics With Open Source Tools
    - » Windows Registry Forensics: Advanced Digital Forensic Analysis of the Windows Registry
    - » Windows Forensic Analysis DVD Toolkit
    - » Perl Scripting for Windows Security: Live Response, Forensic Analysis, and Monitoring
  - → Slides and tools accompanying the books freely available
    - » http://code.google.com/p/winforensicaanalysis/
  - → Filesystem tool added by us

Case Study IV: Timeline Forensics

- Timelines may provide a more comprehensive and more holistic view of the actions on a suspect's machine than simple single timestamps
- The goal of a timeline is to aggregate events from different sources and arrange them in a chronological order
- The type of considered inputs depends amongst others heavily on the goal of the examiner and the available resources, but typically include
  - Registry key writes
  - File system changes (MAC)
  - Event logs
  - Other logs (web server, DHCP, applications etc.)
  - …
- Finally, the aggregated events of a timeline analysis have to be formatted nicely
  - Textual as a list of chronological events
  - Graphical as a time line

Case Study IV: Timeline Forensics

- We will aggregate the following sources into our timeline
  - → Event log
  - → Prefetch files
  - → Recycle bin INFO2 structures
  - → Registry
    - » Key write times
    - » User settings (NTUSER.DAT)
  - → Filesystem information
    - » Files that have been created, modified, or accessed in a particular period of time

- This output file will then be parsed to represent a chronological timeline of actions

Case Study IV: Timeline Forensics

- First, create a directory where the output contents are stored
  - → E.g. `C:\forensics>mkdir tln`
- Event Log data
  - →
    ```
    C:\forensics\tools\carvey_tools>evtparse.exe
    -d "C:\WINDOWS\system32\config" -t >>
    ..\..\tln\tln_raw.txt
    ```
- Prefetch data
  - →
    ```
    C:\forensics\tools\carvey_tools>pref.exe -d
    "C:\WINDOWS\Prefetch" -s localhost -t >>
    ..\..\tln\tln_raw.txt
    ```

Case Study IV: Timeline Forensics

- All user’s personal registry information

  →
  ```
  C:\forensics\tools\carvey_tools>rip.exe -r
  "c:\Documents and Settings\Anna\NTUSER.DAT" -u Anna -s
  localhost -p userassist_tln >> ..\..\tln\tln_raw.txt
  ```

  →
  ```
  C:\forensics\tools\carvey_tools>rip.exe -r
  "C:\Documents and Settings\Brian\NTUSER.DAT" -u Brian -
  s localhost -p userassist_tln >> ..\..\tln\tln_raw.txt
  ```

  →
  ```
  C:\forensics\tools\carvey_tools>rip.exe -r
  "C:\Documents and Settings\Charly\NTUSER.DAT" -u Charly
  -s localhost -p userassist_tln >> ..\..\tln\tln_raw.txt
  ```

  →
  ```
  C:\forensics\tools\carvey_tools>rip.exe -r
  "C:\Documents and Settings\Doris\NTUSER.DAT" -u Doris -
  s localhost -p userassist_tln >> ..\..\tln\tln_raw.txt
  ```

  →
  ```
  C:\forensics\tools\carvey_tools>rip.exe -r
  "C:\Documents and Settings\Edgar\NTUSER.DAT" -u Edgar -
  s localhost -p userassist_tln >> ..\..\tln\tln_raw.txt
  ```

Case Study IV: Timeline Forensics

- Times of most recent registry changes
  - `regtime.exe -r "C:\forensics\registry_backup\system" -m HKLM/System -s localhost >> "..\..\tln\tln_raw.txt"`
  - `regtime.exe -r "C:\forensics\registry_backup\software" -m HKLM/Software -s localhost >> ..\..\tln\tln_raw.txt`
- Recycle bin information for all users
  - `recbin.pl -i C:\RECYCLER\S-1-5-21-1409082233-746137067-1060284298-1003\INFO2 -s localhost -u Anna -t >> ..\..\tln\tln_raw.txt`
  - `recbin.pl -i C:\RECYCLER\S-1-5-21-1409082233-746137067-1060284298-1005\INFO2 -s localhost -u Charly -t >> ..\..\tln\tln_raw.txt`
  - `recbin.pl -i C:\RECYCLER\S-1-5-21-1409082233-746137067-1060284298-1007\INFO2 -s localhost -u Edgar -t >> ..\..\tln\tln_raw.txt`

Case Study IV: Timeline Forensics

- All file system changes (takes a long time!)
  - →
    ```
    Administrator@winxp-forensics
    /cygdrive/c/forensics/tools
    $ python files_changed.py -a -m -c
    '/cygdrive/c/' '2011-11-21 10:15:00' '2011-
    11-21 10:25:00' >> ../tln/tln_raw.txt
    ```
  - → Date: 21.11.2011 10:17 → USB thumb drive plugged in
    - » Will be shown in timeline a 09:17 Z (Z = GMT!)
- Finally, parse the aggregated event file into a chronological timeline and analyze it with a text editor
  - →
    ```
    C:\forensics\tools\carvey_tools>parse.pl -f
    ..\..\tln\tln_raw.txt
    >..\..\tln\tln_formated.txt
    ```