# Crime Scene & Hardware Inspection

## 1. Introduction

In this exercise, you are going to use some of the knowledge that you should have acquired from Part 1 in order to process a potential crime scene and recover any PDE sources (or other PE) that may be present. This exercise will be done in teams of 3 or 4 and each member of the team should take care to record their own contemporaneous notes as they may be called upon to give evidence at a tribunal or in court.

Once you have collected the PDE source, you are required to carry out an initial lab. hardware examination to record the conditions of the machine along with relevant information about storage devices, configuration etc.

## 2. Scenario

A member of staff has reported seeing unusual activity on the screen of a PC in the hardware lab. (CSE/168). The machine has been isolated, but not interacted with, since the report was received.

Because you are the nearest team with knowledge of how to handle potential evidence, you have been asked to examine the scene to record and recover anything which may be significant. The dept.'s own police force has already cordoned off the area of interest to protect it until you deal with the scene.

## 3. Task 1 - Collection

In your group, decide a plan of action.

Assign appropriate roles to each member of the team. (Don't forget to include an exhibits officer to check continuity information, catalogue exhibits and arrange for them to be transported for examination).

Carry out your plan, taking appropriate steps to identify and collect any potential evidence in a way which makes it more likely than not that any evidence you produce can be used in court if necessary.

## 4. Task 2 – Hardware Examination

The suspect device has now been transported to your lab. for further examination. Your task is to carry out an initial physical examination in order to record the condition of the machine, obtain details of internal storage and configuration, and baseline data from the BIOS/EFI installed in the machine. Care should be taken to ensure that continuity of evidence is maintained at all times.

You will be provided with a basic toolkit with which to open the machine and remove components if required.

N.B. BEFORE carrying out the BIOS/EFI check please re-install the machine case and ask a member of staff to check that it is safe to apply power to the machine.

Once you have completed your hardware examination, reassemble the device so that it can be returned to the evidence store.

## 5. Timing

You have

- Up to 10 minutes for initial scene assessment & planning.
- Up to 40 minutes to prepare and carry out your plan for PDE collection.
- Up to 40 minutes to carry out the lab. hardware examination.
- 10-20 minutes of debriefing at the end of the exercise.

**Sample forms which may assist in your tasks are provided below.**

# SAMPLE HARDWARE LOG

Case #: _____ Case Title: _____

Examiner: _____ Case Agent: _____

Date Evidence Seized: _____/_____/_____ Location Seized: _____

_____

Date Exam Started: _____/_____/_____ Location of Exam: _____

Date Exam Ended: _____/_____/_____ _____

## EVIDENCE DETAILS

| | Evid # / File # | Manufacturer | Model | Serial Number |
|---|---|---|---|---|
| **CPU** | | | | |
| **Monitor** | | | | |
| **Keyboard / Mouse** | | | | |
| **HDD 1** IDE SCSI SATA MASL CS | | Capacity(c/h/s) | | |
| **HDD 2** IDE SCSI SATA MASL CS | | Capacity(c/h/s) | | |
| **HDD 3** IDE SCSI SATA MASL CS | | Capacity(c/h/s) | | |
| **Floppy Disks** | | | | |
| **ZIP Disks** | | | | |
| **DVD/CD-ROM** | | | | |
| **Other** | | | | |
| | | | | |

## BIOS/EFI reported configuration

| | |
|---|---|
| BIOS type & version | |
| Date/Time | |
| “Wall clock” Date//Time | |
| CPU | |
| RAM | |
| Boot device sequence | |
| HD1 | |
| HD2 | |
| HD3 | |
| HD4 | |
| DVD/CD-ROM 1 | |
| DVD/CD-ROM 2 | |
| Other storage | |

SAMPLE Evidence Item Recovery Log Incident #:

| Item # | Description | Recovered By | Date/Time |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | Page _____ of _____ | |

SAMPLE PDE source record at scene Incident #:

Date/Time _____ Record produced by _____

Location _____

| **PDE source type** | |
|---|---|
| State | On / Off |
| Identifying mark(s) & location(s) | |
| If on – record of anything visible on screen | |
| Media present & location | |
| Physical connections (sketch or photograph if possible) | |

THE UNIVERSITY of York

# SAMPLE CRIME SCENE ENTRY LOG SHEET

***ALL PERSONS ENTERING THE CRIME SCENE MUST SIGN THIS SHEET***

**AGENCY:**
**SCENE LOCATION:** _____

**INCIDENT #:** _____

**NOTE: Officers assigned to maintain scene security must also log in and out on this sheet and should state their reason as "Log Officer".**

| | | | IN | OUT | |
|---|---|---|---|---|---|
| **NAME & TITLE** | **INITIALS** | **AGENCY** | **DATE / TIME** | **DATE / TIME** | **REASON FOR ENTERING** |
| | | | / | / | |
| | | | / | / | |
| | | | / | / | |
| | | | / | / | |
| | | | / | / | |
| | | | / | / | |
| | | | / | / | |
| | | | / | / | |

| | | | | | |
|---|---|---|---|---|---|
| | | | / | / | |
| | | | / | / | |
| | | | / | / | |
| | | | / | / | |
| | | | / | / | |
| | | | / | / | |
| | | | / | / | |
| | | | / | / | |

EHAC - Forensic

Some notes about the CAINE-YE USB toolkit.

It is based on an accepted toolkit, CAINE, with the addition of the Win-UFO forensic toolkit in Windows and an installation of the Autopsy forensic toolkit in Linux.

## General note

The “caine” user (in Linux) is unprivileged, in order to minimise the potential for damage. Some tools run in a privileged state, but the write blocker and their implementation also minimise the potential for damage.

Occasionally, you may need to mount a partition in order to use certain tools. To do this, you need to use “sudo” to temporarily elevate privileges. E.g. “sudo mount /media/sda1” will mount a simple system’s main windows partition (/dev/sda1) under the directory “/media/sda1”. Normal filesystem protections will apply to access to it – which means it is write-protected for unprivileged users.

## Storing data

We have a dedicated Data partition on the machines, which can be accessed from all installed O/S. It already contains files for some lab. Exercises and you can use it for intermediate results and any data that needs to be transferred from one O/S to another for analysis.

In Windows, this directory should be visible and writable automatically.

The Linux distro., however, automatically blocks writes to all devices so you will need to perform a little housekeeping to make it (and any other device or partition) writable each time you boot into Linux. The data partition should be /dev/sda6.

One recipe for this follows:

Boot the Linux distro and log in to the “caine” user.

- Open the “Unblock” app. on the desktop, and toggle the state of /dev/sda6 from Read Only to Writable by clicking on the radio button next to its name. Close the Unblock app.
- Open Mate Terminal from the System Tools menu.
- Use sudo and mkdir to create the mount point for the Data partition. “sudo mkdir /mnt/data” should do it.
- At the command line type “sudo mount /dev/sda6 /mnt/data” to mount the partition under /mnt/data. Check that all files are visible by using ls or the file manager app.

When working with removable devices, you may need to perform similar actions to access their contents. It’s worth creating a /mnt/src directory as a mount point for them, too.

# Imaging and Image Verification

This exercise is intended to introduce you to the use of a simple Imaging tool (Guymager) and why we use hashing/verification functions so often in forensic work.

This exercise is best done in pairs.

You will be provided with an old promotional USB device which contains a FAT32 filesystem containing some files.

Start by booting the workstation into into CAINE and ensuring that the data partition is mounted and writable.

1. In Guymager, Right click on the USB stick and select "Acquire image" to start the image creation process. Use the "Linux (dd)" image format to make it easier to clear up after the exercise. (EWF can be a more useful format, but it creates a LOT of files).
2. Fill in sufficient details to allow your image to be identified if necessary (think back to the crime scene exercise and consider what else you should be recording, and where).
3. In the Image Directory, select the Windows disc that you have made writable.
4. Make sure that at least one hashing algorithm is enabled, but please DISABLE verification after imaging is complete (it takes too long) and ensure that the "Re-read" option is also disabled.
5. Once imaging has completed, make a note of the hash values calculated for your image and compare your results with those of your neighbours. Are they the same? If not, why not?

Now, make sure that the USB device is writable and mount the filesystem. ("sudo mount /dev/sdf1 /mnt/src" might do it). Open a few of the files to view their contents, but do not edit them or add anything to the filesystem. When you are done, unmount the USB device.

Repeat the imaging steps listed above.

Has anything unexpected happened? Why do you think you get this result?

# Time Metadata- Worksheet

The purpose of this worksheet is to re-enforce previous lecture notes on times in file systems, and to explore how timestamps change when files are deleted to the 'recycle bin' and then recovered.

This subject is rather important; it is often misunderstood (e.g. how can 'create' time be later than 'modify' time?) and a proper understanding often helps piece together a document's history, especially when several copies are available within the same system, or are cached in different machines due to user roaming.

For this exercise, boot the workstation into Windows. You will be using the FTK Imager tool to examine the contents of the Windows disc as you carry out the tasks. The username is filled in by default and the password is "student_0987" (or maybe "student-0987")

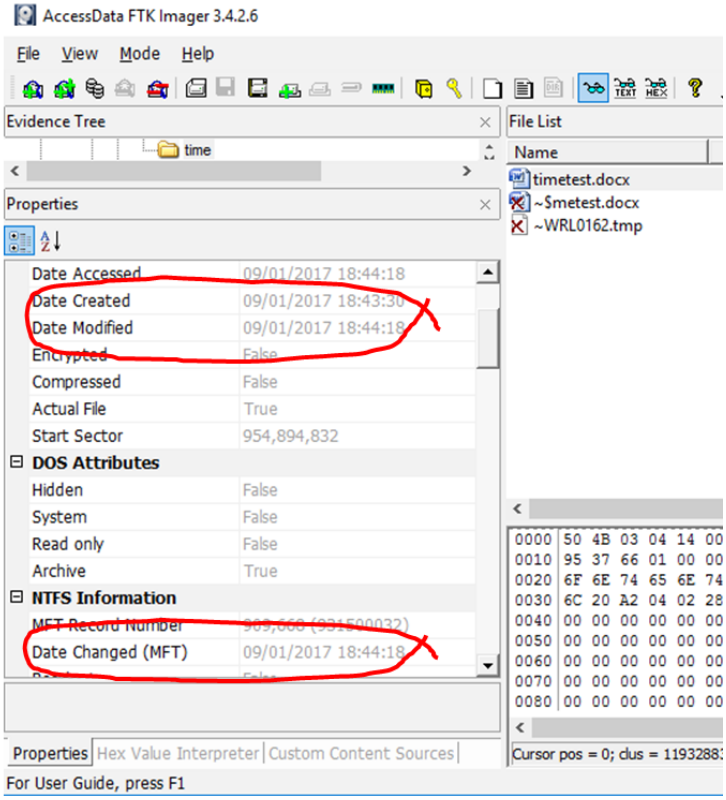
## 1. Viewing Date-Time Information.

Make a temporary directory on the Desktop, and create a new word document (e.g. timetest.docx). Close the document - then, after a short time, open, modify it and save it.

As noted in the lecture on file systems, the NTFS file system maintains three times for each file: created, modified and accessed, the last of which is not normally used in current Windows systems. It also maintains a timestamp for any time that the MFT record is modified.

For this experiment, make a note of the Created, Modified and MFT changed time after each operation. To obtain the MFT changed time it is necessary to use a forensic tool, such as FTK imager.

Open FTK Imager, Add your first NTFS physical disk partition as an evidence item then navigate to the file and view the metadata:




WARNING: FTK Imager does not refresh if you change the system. You will need to 'remove evidence item' and add it again to see any changes.

## 2. Review Time Patterns

(It will help if the recycle bin is empty before you begin.)

Work out which times change, and how in the following operations:

a) **Copy the file to a different directory.**

b) **Drag & drop the file to a different directory.**

c) **Edit the file and save the results.**

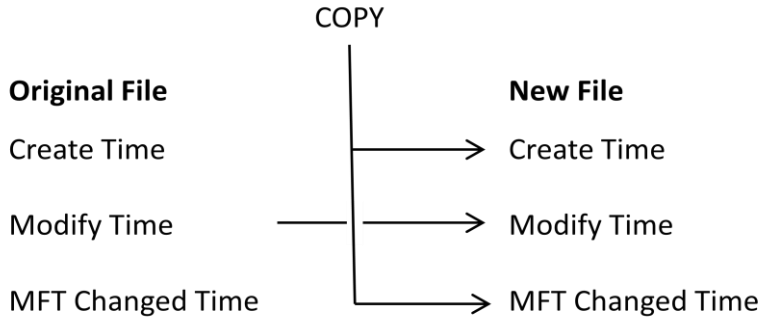
d) **'Delete' the file (ie select and just use 'delete' which places the file in the recycle bin). Map the times for both the associated recycle files, see below.**

e) **Restore the file from the recycle bin (open the recycle bin, right-click and select 'restore').**

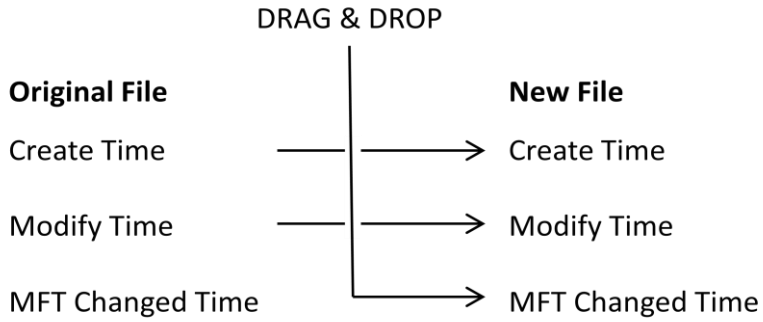
Note for the recycle bin: you will need to find the recycle-bin files associated with your test file. This is a $RECYCLE folder for your file system, and a $I and $R file associated with each recorded file. The $I file records the original path, the $R file is the original file itself (.docx files are zipped, so have a signature that starts 'PK...')

# Time Metadata - Answers

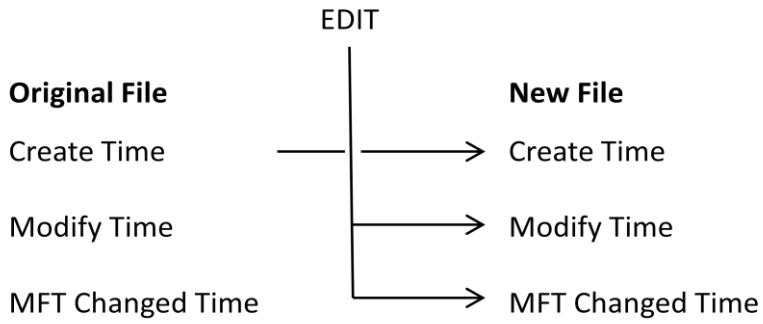
**a) Copy the file to a different directory.**



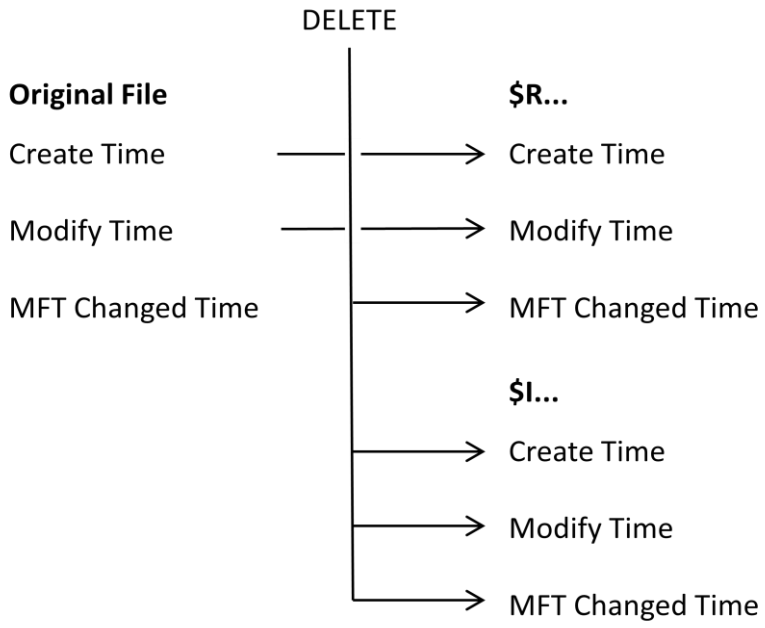

**b) Drag & drop the file to a different directory.**



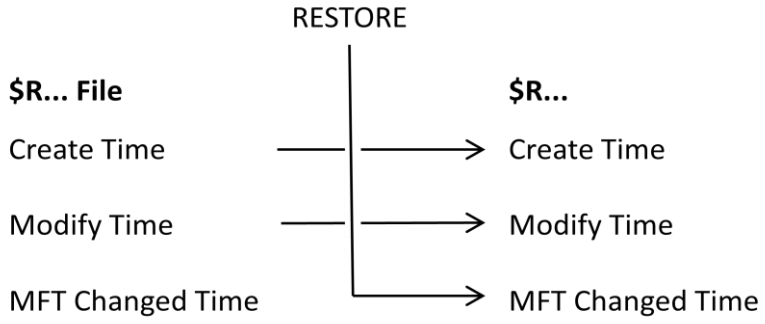

**c) Edit the file and save the results.**



**d) 'Delete' the file (ie select and just use 'delete' which places the file in the recycle bin). Map the times for both the associated recycle files, see below.**




**e) Restore the file from the recycle bin (open the recycle bin, right-click and select 'restore').**



# Data Encoding - Worksheet

The purpose of this session is to refresh your familiarity with low-level computer data representation.

The exercises should be straightforward, intended to practice reading low level data, but the notes are important - please read the sheet carefully.

## Contents



## 1. Numbers and Endian

At the lowest level, data in computer systems are binary - just a long string of 0's and 1's. In practice the smallest unit of data that is usually encountered is a **byte** - a string of 8 bits at a specific address.

*For example:*

| | | | | |
|---|---|---|---|---|
| *byte address:* | *0* | *1* | *2* | *3* |
| *binary data:* | *01000011* | *10001100* | *11110110* | *00101001 ...* |

To make this readable, it is usually written in hexadecimal, which uses 0..9, A..F to write the 16 possible combinations of 4 bits:

| Binary | Hexadecimal | Decimal |
|---|---|---|
| 0000 | 0 | 0 |
| 0001 | 1 | 1 |
| 0010 | 2 | 2 |
| 0011 | 3 | 3 |
| 0100 | 4 | 4 |
| 0101 | 5 | 5 |
| 0110 | 6 | 6 |
| 0111 | 7 | 7 |
| 1000 | 8 | 8 |
| 1001 | 9 | 9 |
| 1010 | A | 10 |
| 1011 | B | 11 |
| 1100 | C | 12 |
| 1101 | D | 13 |
| 1110 | E | 14 |
| 1111 | F | 15 |

A byte is therefore usually written as two hexadecimal characters, e.g 'A8', in text this is usually written as 0xA8 or A8H to warn the reader that it is a hexadecimal number, not decimal. . The string of bytes above would be written:

| | | | | |
|---|---|---|---|---|
| *byte address:* | *0* | *1* | *2* | *3* |
| *binary data:* | *0100\|0011* | *1000\|1100* | *1111\|0110* | *0010\|1001 ...* |
| *hexadecimal:* | *0x43* | *0x8C* | *0xF6* | *0x29* |

When we usually write numbers the leftmost digit is the most significant (ie represents the biggest number). However this is just a convention and computers can order their bytes in either direction:

Lowest address (first address) stores most significant (big) byte **big-endian**

Lowest address (first address) stores least significant (little) byte **little-endian**

common use:

| | |
|---|---|
| little-endian | Windows Operating System, file system, and application data |
| big-endian | network data (also called 'network byte order') |
| | records in many types of database |

**Exercise**

Use a hex editor (wxHexEditor in Linux, under Programming tools is preferred), and start a new file with at least 100 bytes for experimenting. Spend a little time getting familiar with the menu options and the display panes available, then:

**Problem: Convert the decimal number 1234567890 to hexadecimal, enter it into the hex editor in little-endian and check that the data interpreter gives the correct value integer.**

(If you need to check your conversion to hex, the Calculator found under Accessories, when put into Scientific mode, may help)

## 2. Text

The most basic form of text is to represent the latin characters, digits and a few special characters as single bytes, the most common standard is ASCII:

| Dec | Hx | Oct | Char | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 000 | NUL (null) | 32 | 20 | 040 |   | Space | 64 | 40 | 100 | @ | @ | 96 | 60 | 140 | ` | ` |
| 1 | 1 | 001 | SOH (start of heading) | 33 | 21 | 041 | ! | ! | 65 | 41 | 101 | A | A | 97 | 61 | 141 | a | a |
| 2 | 2 | 002 | STX (start of text) | 34 | 22 | 042 | " | " | 66 | 42 | 102 | B | B | 98 | 62 | 142 | b | b |
| 3 | 3 | 003 | ETX (end of text) | 35 | 23 | 043 | # | # | 67 | 43 | 103 | C | C | 99 | 63 | 143 | c | c |
| 4 | 4 | 004 | EOT (end of transmission) | 36 | 24 | 044 | $ | $ | 68 | 44 | 104 | D | D | 100 | 64 | 144 | d | d |
| 5 | 5 | 005 | ENQ (enquiry) | 37 | 25 | 045 | % | % | 69 | 45 | 105 | E | E | 101 | 65 | 145 | e | e |
| 6 | 6 | 006 | ACK (acknowledge) | 38 | 26 | 046 | & | & | 70 | 46 | 106 | F | F | 102 | 66 | 146 | f | f |
| 7 | 7 | 007 | BEL (bell) | 39 | 27 | 047 | ' | ' | 71 | 47 | 107 | G | G | 103 | 67 | 147 | g | g |
| 8 | 8 | 010 | BS (backspace) | 40 | 28 | 050 | ( | ( | 72 | 48 | 110 | H | H | 104 | 68 | 150 | h | h |
| 9 | 9 | 011 | TAB (horizontal tab) | 41 | 29 | 051 | ) | ) | 73 | 49 | 111 | I | I | 105 | 69 | 151 | i | i |
| 10 | A | 012 | LF (NL line feed, new line) | 42 | 2A | 052 | * | * | 74 | 4A | 112 | J | J | 106 | 6A | 152 | j | j |
| 11 | B | 013 | VT (vertical tab) | 43 | 2B | 053 | + | + | 75 | 4B | 113 | K | K | 107 | 6B | 153 | k | k |
| 12 | C | 014 | FF (NP form feed, new page) | 44 | 2C | 054 | , | , | 76 | 4C | 114 | L | L | 108 | 6C | 154 | l | l |
| 13 | D | 015 | CR (carriage return) | 45 | 2D | 055 | - | - | 77 | 4D | 115 | M | M | 109 | 6D | 155 | m | m |
| 14 | E | 016 | SO (shift out) | 46 | 2E | 056 | . | . | 78 | 4E | 116 | N | N | 110 | 6E | 156 | n | n |
| 15 | F | 017 | SI (shift in) | 47 | 2F | 057 | / | / | 79 | 4F | 117 | O | O | 111 | 6F | 157 | o | o |
| 16 | 10 | 020 | DLE (data link escape) | 48 | 30 | 060 | 0 | 0 | 80 | 50 | 120 | P | P | 112 | 70 | 160 | p | p |
| 17 | 11 | 021 | DC1 (device control 1) | 49 | 31 | 061 | 1 | 1 | 81 | 51 | 121 | Q | Q | 113 | 71 | 161 | q | q |
| 18 | 12 | 022 | DC2 (device control 2) | 50 | 32 | 062 | 2 | 2 | 82 | 52 | 122 | R | R | 114 | 72 | 162 | r | r |
| 19 | 13 | 023 | DC3 (device control 3) | 51 | 33 | 063 | 3 | 3 | 83 | 53 | 123 | S | S | 115 | 73 | 163 | s | s |
| 20 | 14 | 024 | DC4 (device control 4) | 52 | 34 | 064 | 4 | 4 | 84 | 54 | 124 | T | T | 116 | 74 | 164 | t | t |
| 21 | 15 | 025 | NAK (negative acknowledge) | 53 | 35 | 065 | 5 | 5 | 85 | 55 | 125 | U | U | 117 | 75 | 165 | u | u |
| 22 | 16 | 026 | SYN (synchronous idle) | 54 | 36 | 066 | 6 | 6 | 86 | 56 | 126 | V | V | 118 | 76 | 166 | v | v |
| 23 | 17 | 027 | ETB (end of trans. block) | 55 | 37 | 067 | 7 | 7 | 87 | 57 | 127 | W | W | 119 | 77 | 167 | w | w |
| 24 | 18 | 030 | CAN (cancel) | 56 | 38 | 070 | 8 | 8 | 88 | 58 | 130 | X | X | 120 | 78 | 170 | x | x |
| 25 | 19 | 031 | EM (end of medium) | 57 | 39 | 071 | 9 | 9 | 89 | 59 | 131 | Y | Y | 121 | 79 | 171 | y | y |
| 26 | 1A | 032 | SUB (substitute) | 58 | 3A | 072 | : | : | 90 | 5A | 132 | Z | Z | 122 | 7A | 172 | z | z |
| 27 | 1B | 033 | ESC (escape) | 59 | 3B | 073 | ; | ; | 91 | 5B | 133 | [ | [ | 123 | 7B | 173 | { | { |
| 28 | 1C | 034 | FS (file separator) | 60 | 3C | 074 | < | < | 92 | 5C | 134 | \ | \ | 124 | 7C | 174 | | | \| |
| 29 | 1D | 035 | GS (group separator) | 61 | 3D | 075 | = | = | 93 | 5D | 135 | ] | ] | 125 | 7D | 175 | } | } |
| 30 | 1E | 036 | RS (record separator) | 62 | 3E | 076 | > | > | 94 | 5E | 136 | ^ | ^ | 126 | 7E | 176 | ~ | ~ |
| 31 | 1F | 037 | US (unit separator) | 63 | 3F | 077 | ? | ? | 95 | 5F | 137 | _ | _ | 127 | 7F | 177 | &#127; | DEL |

Source: www.LookupTables.com

**Problem: Type some English text into LibreOffice and save it as "text". Save it again, 2 or 3 times, as "text-choose encoding" making sure that you at least save it as UTF-8 and UTF-16. Set the hex editor to load "Code for Information Interchange – ASCCI" and load the straight text file. Then set the hex editor to load UTF-8 and load the UTF-8 copy.**

How are non-latin characters represented? The most basic approach is to use the values 128-255 (0x80 - 0xFF) which are not defined in ASCII to represent extra characters. (See Appendix - codepages).

A better solution is UNICODE, in which each character is allocated a **code point** numbered from 0 to 0x10FFFF (about one million possible characters).

At the byte level a range of possible **encodings** are possible (i.e. how the code point is stored on disk or in memory): **utf-8** allocate a single byte to the lowest numbered code points, adding more bytes for higher code points. This is compact and has the advantage that the first 127 code points and their encoding are the same as ASCII - one byte per Latin character.

utf-16 allocates two bytes for every character, with more 16bit words added for the highest numbered codepoints. 16 bits is sufficient for the core letters in all languages (the 'basic multilingual plane'), and so usually the observed encoding is 2 bytes per character.

common use:

utf-8 a common format for the exchange of unicode data between applications

utf-16 (little endian) the standard format for most Microsoft System and Application data

**Problem: Load the remaining copies of your text into the hex editor again, making sure to change the options to match your file's encoding (as closely as possible) before loading each one. Compare the coding used against the other copies of the same text. If your text is not immediately readable, try changing the Options to use a Windows CP1252 character set.**

## 3. Date and Time

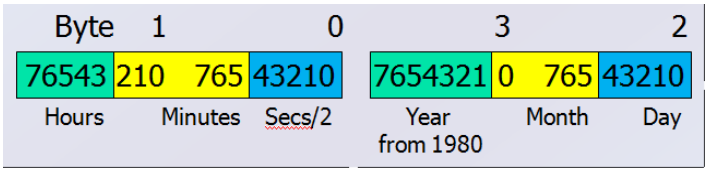
There are two main ways of encoding date and time (or date time groups: dtg) in computers:

- **Fielded;** in which certain bits within the value are assigned to the year, month, day, hour, etc
- **Relative**; in which the date-time is an integer, which gives the number of time intervals since a specific date.

Fielded Time Encoding

An example of a fielded date is the DOS timestamp found in FAT file system directories and some older Microsoft applications

:




For example, 0900 on 3rd January 2011 is:

| 9 | 0 | 0 | 31 | 1 | 3 |
|---|---|---|---|---|---|
| 01001 | 000000 | 00000 | 0011111 | 0001 | 00011 |

re-arrange into bytes:

0100 1000 0000 0000 0011 1110 0010 0011

converting to hex:

48 00 3E 23

Now, note the byte order: this is presented as 2 little endian 16bit numbers, so we need:

00 48 23 3E

Relative Time Encoding

An example of a relative time representation is Windows FILETIME, which is used extensively, including within the NTFS file system. It is usually stored as a 64bit integer in little endian format; the number represents the number of 100 nanosecond (1/10,000,000 second) intervals since 00 UTC (GMT) on 1 January 1601.

Of course, getting from such a number to an actual date time requires full knowledge of the calendar - including leap days, so is best left to software with the appropriate tables.

**Problem: What date is represented by the hexadecimal string: 00 00 00 E3 55 25 D0 01 ?**

**(This string is already in system byte order.) (Hint: you may find it useful to locate an online timestamp converter to check your answer).**

You may meet other time formats, for example the UNIX (POSIX standard) time format (aka EPOCH time) is similar to FILETIME, but the number represents the number of seconds since midnight UTC on 1 January 1970, not counting leap-seconds.

Time zones and Daylight Saving

Time varies around the world, and even local time is adjusted during some parts of the year (e.g. British Summer Time). This results in two main approaches to encoding times:

> *Record the local time*, having already taken account of time zones and daylight saving times in the current setting of the computer clock, or by some software function.
>
> *Record Co-ordinated Universal time (UTC)* - (In the UK still called Greenwich Mean Time.) The times presented to the user are adjusted to show local time, but the time stored is UTC.

These have different implications for forensics, since we generally need to know the local time at which an event occurred. If we can be assured that the system clock is accurate, then Local Time recording (e.g. 'DOS' times in ZIP files and the FAT file system) is unambiguous. On the other hand, times recorded in UTC require translation to local times, using the daylight saving and the *time zone set in the computer when the time was recorded.*

**In forensic work it is vital that you establish a basis for time evidence, and know how the time zone is handled for any times that you need to report.**

# User Actions - Worksheet

The purpose of this session is to:

- practice extracting evidence of user activity from the Registry
- link this evidence to times and dates in the file system
- merge evidence from several sources
- focus on 'evidence' extracted rather than 'ideas' about what might have happened.

**The Problem**

You have a full forensic image (Forensic_Workshop_1).

You are required to determine a timeline or description of events surrounding the document '**RecycleTestDocument.rtf**' on 2 January 2015.

1. Obtain evidence, in particular date and time information from the following sources:
   - Registry: User Assist
   - Registry: MRU
   - File system: Recent (lnk) documents
   - File system: metadata associated with the document of interest.

2. Arrange the evidence in the form of a timeline

**Hints**

Different tools are good at different jobs. You may find that FRED is better at generally searching for registry entries (since it can search the whole registry hive) but that you need to use RegistryRipper if you wish to extract UserAssist records, since this tool will decode the ROT data. (Note – there seems to be bug in the CAINE installation of RegRipper, so you might like to stick with FRED and search for ROT13ed strings instead).

Please refer to the "Using the CAINE USB toolkit" document, provided last week, for more information about using the tools.

## Appendix - Non-latin characters and codepages

The first approach to encoding non-latin characters was to use Extended ASCII. The ASCII character set is defined only for the first 128 values in any byte (0x00 - 0x7F), leaving the remaining 128 bits available for extra characters. Of course, it isn't possible to fit every language into just 128 characters, so to cover a range of languages different **codepages** were used.

Each codepage specifies how to interpret the values 0x7F -0xFF.

For example the Euro currency symbol, €, is not defined in the standard ASCII code, but Windows Codepage 1252 (Western Europe) reserves the value 0x80 (decimal value 128) for this symbol - so if the computer is using that codepage this is the value that will represent the Euro symbol on disk or in memory.

Codepages are manufacturer-specific and not fully standardised.

Common codepages include:

1250: Windows Latin 2 (Central Europe)

1251: Windows Cyrillic

1252: Windows Latin 1 (ANSI)

1253: Windows Greek

1254: Windows Latin 5 (Turkish)

1255: Windows Hebrew

1256: Windows Arabic

1257: Windows Baltic

1258: Windows Vietnamese

874: Windows Thai

Ideographic character sets (e.g. Chinese) have too many characters to encode in this way, a variety of manufacturer and regional approaches using 2 or more bytes per character exist.

Most current Western systems now use UNICODE, either with the ability to represent all codepoints, or in a limited 16bit form.

# Forensic Analysis (bonus)

User activity hints

# More Evidence...

Operating Systems help their users by providing prompts for previously used files and directories.

So user actions are recorded...

# Why

Most investigations are concerned with understanding what one or more computer users did, and when.

File systems provide valuable evidence, e.g.:

- Documents.
- Deleted Documents.
- Recent Documents.

Internet activity is also important (see later lecture).

The registry provides a further source of information about how the computer was used.

corroboration:
mutually supportive evidence.

# Contents

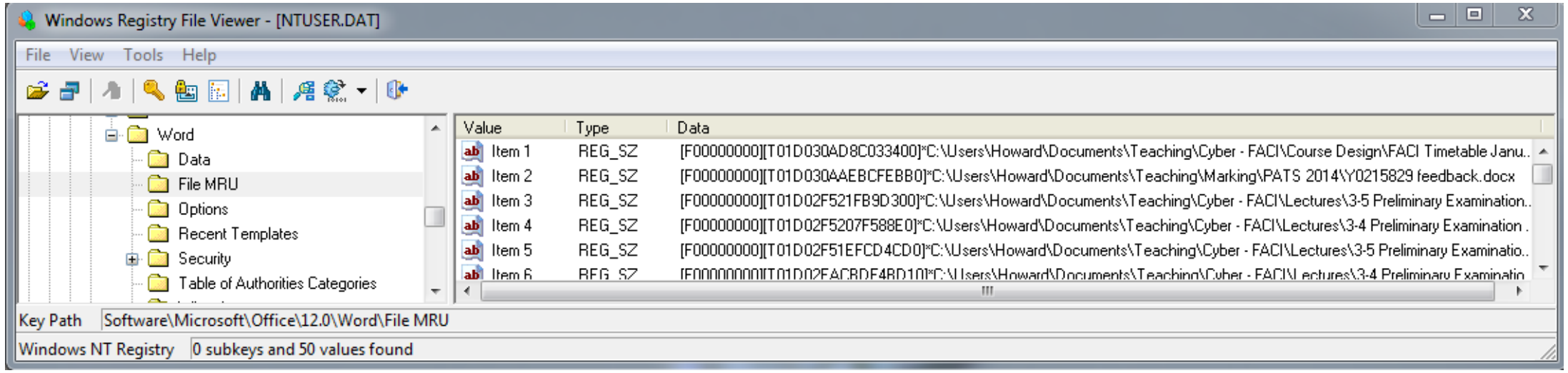
MRUs (Most Recently Used) lists for files opened by:

- Applications.
- Dialogue boxes.
- Explorer (‘Recent Documents’).

Recently launched applications (User Assist Keys).

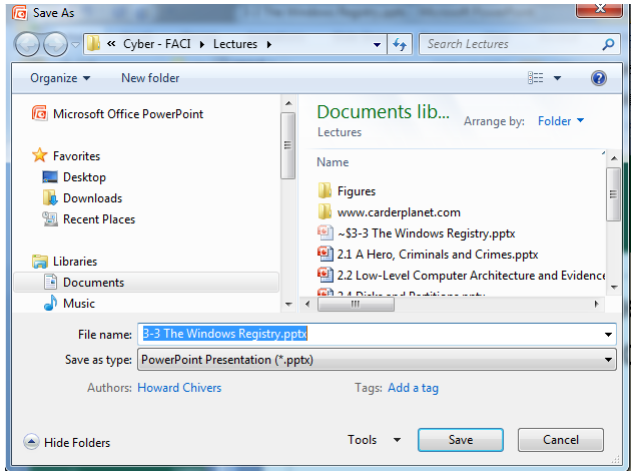
# Most Recently Used

- Allows applications to list recently accessed documents.
  - Located in the user hive (NTUSER.DAT) in an application sub-key under the Software Key, e.g.
    NTUSER.DAT\Software\Microsoft\Office\12.0\Word\File MRU



# Common Dialogue

- Common dialogues provide standard services to applications, such as ‘Save As’.



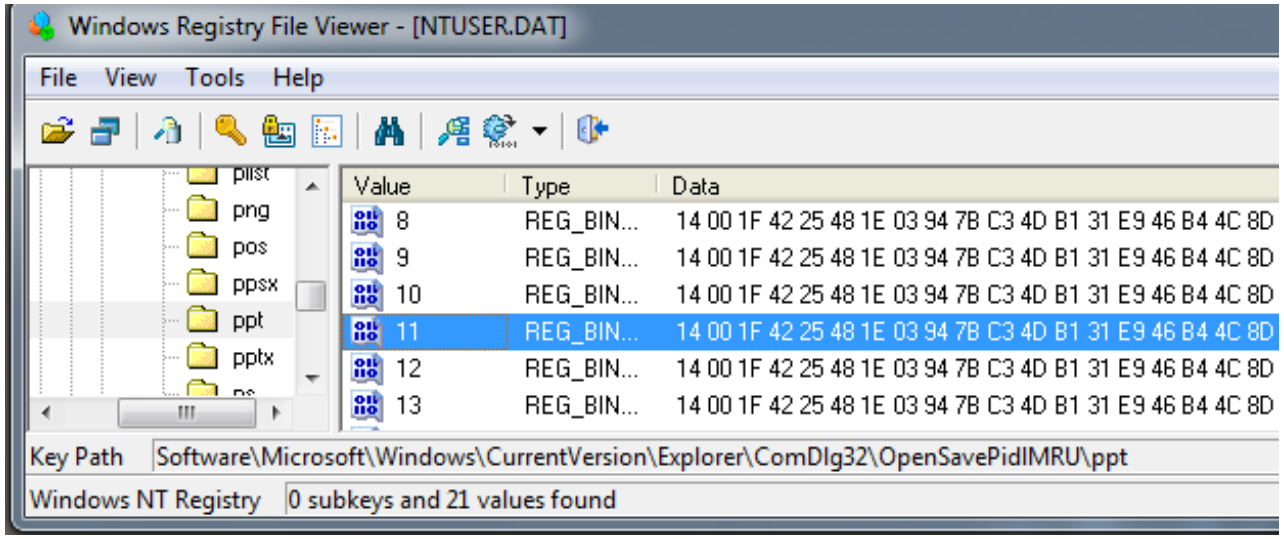

- The registry maintains separate MRU lists for these functions.

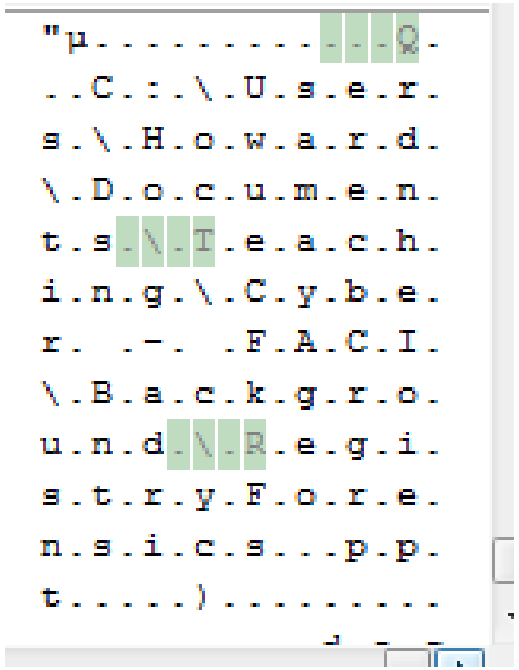
  NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSavePidMRU

- Under this key there is a separate list for each file extension.

# Common Dialogue Binary Data

- Common Dialog Entries are in a complex binary format.
- However, the relevant file path is usually present in UNICODE.

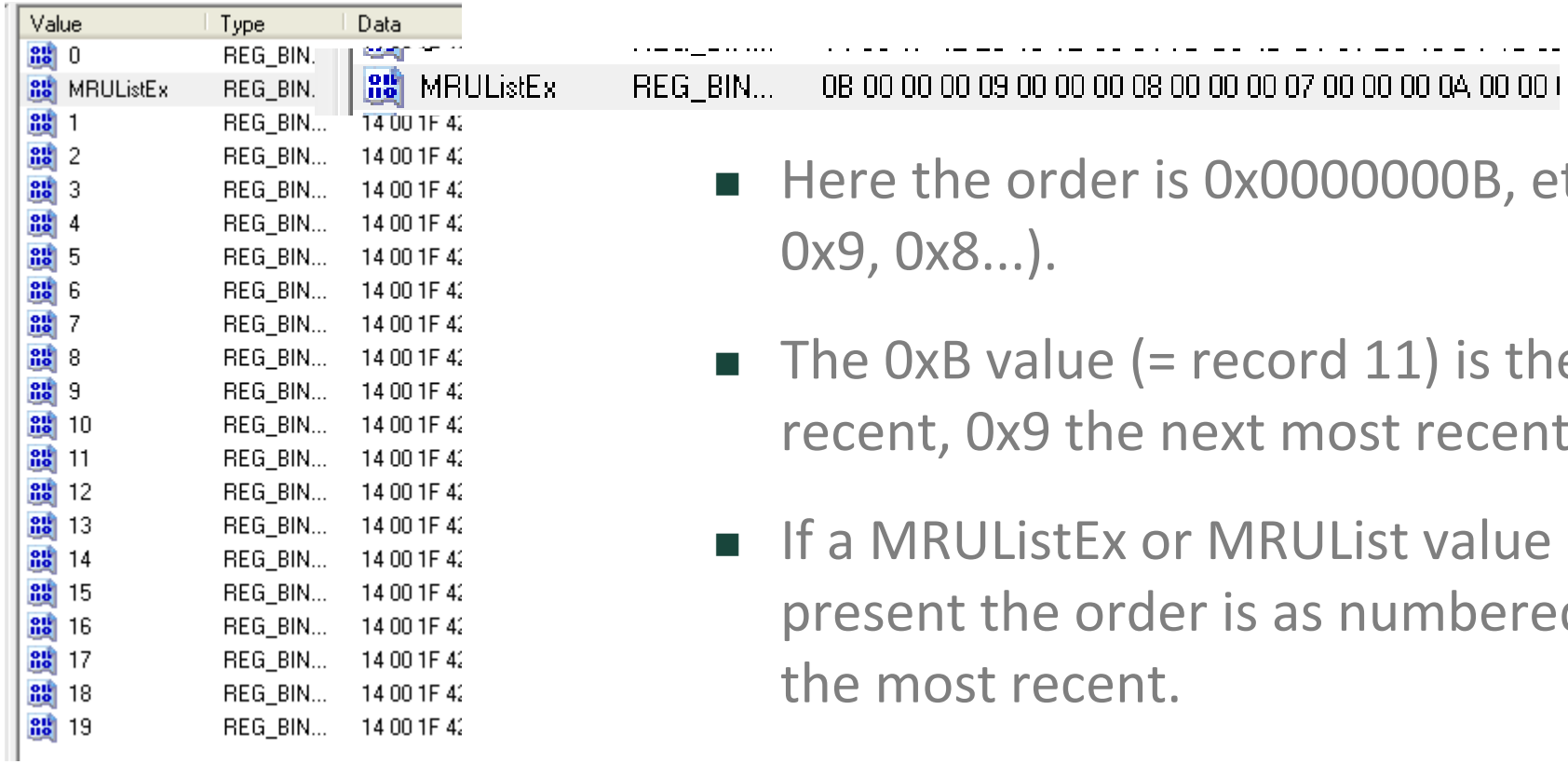



(From Mitec Data View)

# Access Order

- ComDlg MRUs include a **MRUListEx** value which lists the order of use of the MRU values.




- Here the order is 0x0000000B, etc (0xB, 0x9, 0x8...).
- The 0xB value (= record 11) is the most recent, 0x9 the next most recent, etc.
- If a MRUListEx or MRUList value is not present the order is as numbered with 0 the most recent.

# MRU list time and date

- Recall: dates and times are associated with registry **keys**. (Not with individual values)
- So there is a single time for all MRU values in a list.
- This provides the time of the most recent value.
  - Of course there is no information here about previous values, other than their historical order.

# Recent Documents

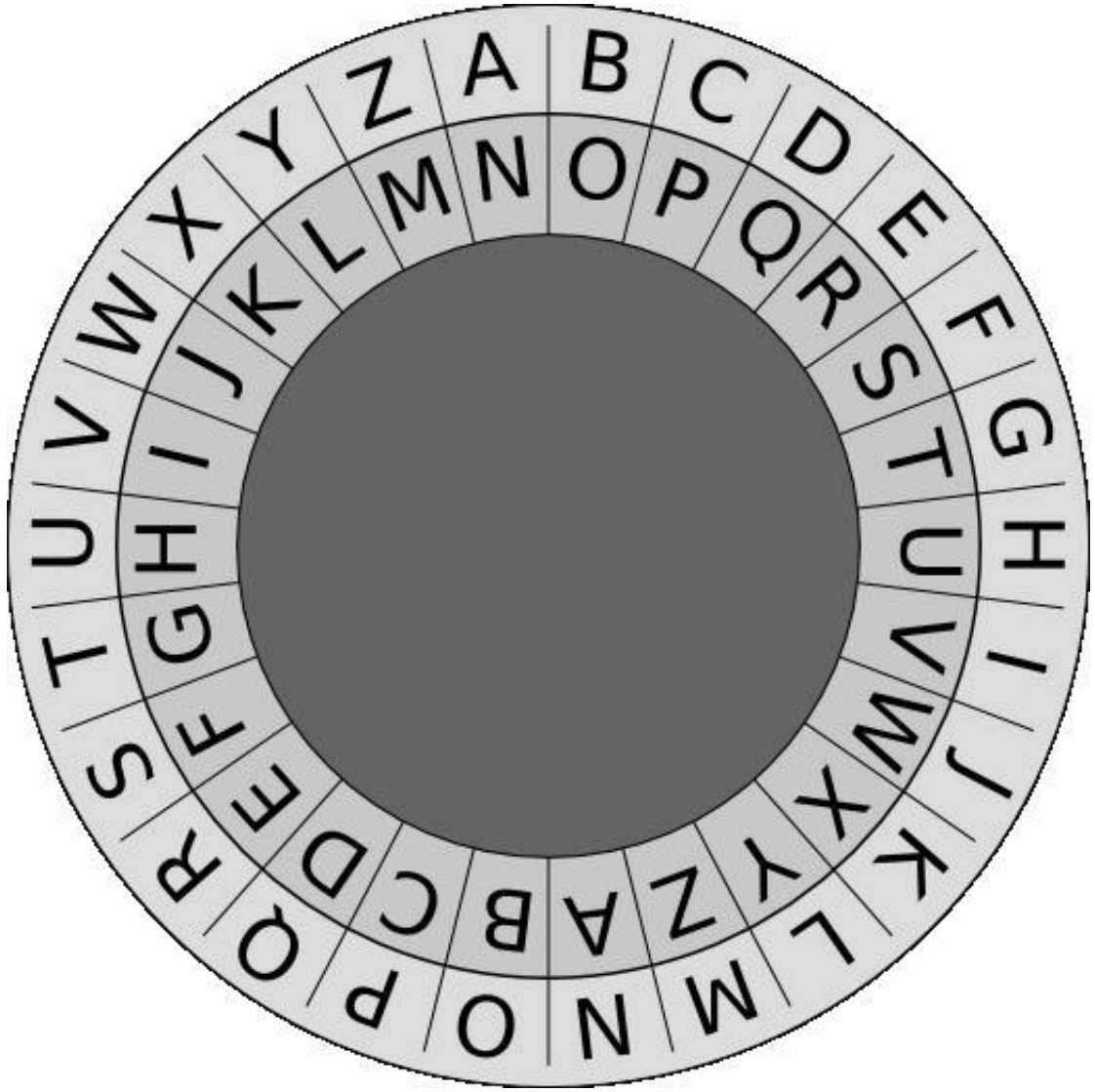
- Windows Explorer (the file browser) also records when documents are accessed.
- The format is similar to the Common Dialogue with subkeys corresponding to file extensions.
- The data are usually file paths in UNICODE.
- e.g.:

  NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\.ppt

# User Assist

- Stores information about software run using menus and windows explorer.
- This key often contains extensive information about a user’s actions.
  - Binary format.
  - File Paths Obfuscated using ROT13.
  - Includes access count and time information with each value.

# ROT13




Obfuscate text by rotating the letters 13 places. (Caesar Cipher / ROT-13)

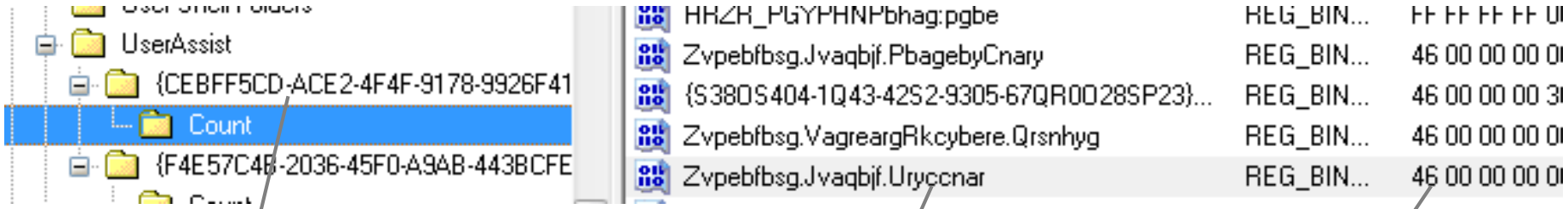
For unknown reasons popular with Geocache and Defcon puzzles.

Some Windows versions use a different form of encryption.

# User Assist

## Registry Path:

NUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{...}\Count




ROT13 Encoded values

Under keys:
{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}
{F4E57C4B-2036-45F0-A9AB-443BCFE33D9F}

Data Includes count and timestamp

# Viewing/Decoding

The Mitec registry viewer is able to show the values using ROT13, but does not correctly decode counts or times (probably using the obsolete Windows XP binary encoding offsets).

One solution is to use RegRipper:

```
UserAssist
Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist
LastWrite Time Thu Dec 16 17:13:25 2010 (UTC)

{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}
Thu Jan 15 11:13:31 2015 Z
  E7CF176E110C211B (87)
Thu Jan 15 11:13:30 2015 Z
  {7C5A40EF-A0FB-4BFC-874A-C0F2E0B9FA8E}\Microsoft Office\Office12\OUTLOOK.EXE (68)
Thu Jan 15 11:11:57 2015 Z
  {7C5A40EF-A0FB-4BFC-874A-C0F2E0B9FA8E}\AccessData\FTK Imager\FTK Imager.exe (11)
Thu Jan 15 11:07:56 2015 Z
```

# Comment

- Reminder – you need several items of evidence to justify conclusions.
- Registry User Histories are an important additional resource.
- Real systems contain thousands of files: evidence of recent actions are often very helpful in providing the starting point for an investigation.

# Additional Information

- Carvey, H., *Windows Registry Forensics*, Syngress
- *Stevens, D., Windows 7 User Assist Registry Keys*, Into the Boxes, January 2010 http://intotheboxes.files.wordpress.com/2010/04/intotheboxes_2010_q1.pdf

# Example Registry Keys

- Application MRU under software key, e.g.:

  NTUSER.DAT\Software\Microsoft\Office\12.0\Word\File MRU

- Common Dialog MRU list for each function:

  NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32*<function>**<extension>* e.g.: \OpenSavePidMRU\ppt

- Recent Documents

  NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\<extension> e.g.: \.ppt

- User Assist

  NUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{...}\Count

  {...}: {CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}

  {F4E57C4B-2036-45F0-A9AB-443BCFE33D9F}



# Event Logs – Worksheet

In this session, you're going to look at Windows event logs, specifically, to try to determine the answers to the following questions :

On 17 November 2011:

- Which users logged on?
- When were the related user sessions?
- Were the sessions online or offline?"

The event log from the relevant machine has been extracted as stored as Security_log_machine_12.evtx on the shared Data partition, in the usual folder.

This exercise is probably best done on the Cyber machines, using Windows' own Event Viewer (found under Windows Administrative Tools (usually)).