# Cyber Taster – Image Forensics

Image evidence is commonly used in digital forensics investigations. This may be because the images are themselves illegal, or because they contain information which can be used to inform a wider investigation. e.g. a selfie in front of a store the suspect was accused of stealing from.

This lab is comprised of three parts (and some extras):

Part A – Explores tools for viewing the metadata of JPEG images.

Part B – Briefly explores pixel level analyses which are useful in the absence of any metadata or to confirm which parts of an image have been manipulated.

Part C – Introduces the analyses of Windows thumbcache artefacts, which are an important part of any investigation involving illegal media.

**The resources for this practical session can be downloaded from** [**https://github.com/smck1/taste\_of\_cyber/tree/master/05\_digital\_forensics**](https://github.com/smck1/taste_of_cyber/tree/master/05_digital_forensics)

**Download** and **unzip** the **afternoon\_evidence.zip** file to somewhere convenient, such as the desktop

# Part A – JPEG Metadata Analysis

## Tools of the trade

File metadata is used widely even outside of forensics. Operating systems often use embedded metadata for organising files (particularly music on Windows), which is often accessible via the right click properties menu (see right).

Photographers and Photoshop enthusiasts will also be very familiar with JPEG metadata (particularly EXIF and XMP formats) as this can contain useful information when editing the image or provide copyright/authorship information.

As such, there are a variety of tools available online which can be found with a simple Google search.

Each tool will display the results differently and may be better for a particular task. For instance, it is easier to see attached thumbnails in “Jeffrey's Image Metadata Viewer” than it is in “JPEGsnoop”. You may wish to try out each tool or switch between them to get an idea of their strengths, however the lab is largely written with JPEGsnoop in mind.

**Metadata tools:**

* **JPEGsnoop** (Windows application, download)
  + <https://www.impulseadventure.com/dl.php?file=JPEGsnoop_v1_8_0.zip>
* **Jeffrey's Image Metadata Viewer** (Web)
  + <http://exif.regex.info/exif.cgi>
* **Metapicz** (Web)
  + <http://metapicz.com/#landing>

## Warm-up Exercise

1. If you haven’t obtained the images for this lab, download and extract **afternoon\_evidence.zip** from the Github page and **load a1.jpg** into an EXIF viewer.
2. The first thing to note is that there is a lot of **EXIF** information embedded in this file, mostly contained in a single APP1 marker, but there are two APP1 and two APP13 markers total.
   * Notice that the **make and model** of the camera are available in the EXIF information (specifically EXIF IFD0)
   * The existence of the APP13 markers suggests that the image has been edited with Photoshop, however other programs can use this APP marker as well. Note that a1.jpg was edited using the **Snapseed** application, rather than Photoshop.
3. Where was the image taken? Try entering the EXIF GPS coordinates on [Google Maps](https://www.google.co.uk/maps/) (even if the tool does it already)
   * Note: the easiest form to enter in to Google is the decimal form, e.g.: “12.12345, 13.12345”. Enter Latitude (positive for North, negative for South) first, then Longitude (positive for East, negative for west)
   * Example: **North 27.746789, West 15.574869** is equivalent to **27.746778,   
     -15.574861** while **South 27.746789, East 15.574869** would be **-27.746778, 15.574861**

Using one or more of the metadata tools, answer the following questions:

**Q1:** What is the full name of the camera used to take **a1.jpg**? (Include the camera series, which will require a visit to Google)

**Q2:** There are multiple timestamps present in **a1.jpg**, however they are different. Based on these timestamps, when was the photo **taken**, and when was it **edited**?

**Q3:** What are the dimensions of the embedded thumbnail in **a1.jpg**?

**Q4:** The next image, **a2.jpg,** has been cropped – it was originally created/edited using Windows Photo Gallery 6.0. What software has been used to do the most recent cropping?

**Q5:** Roughly how many years have passed between **a2.jpg** being modified in Windows Photo Gallery and it being cropped?

**Q6:** Which clubbing/holiday destination was **a3.jpg** taken at?

# Part B – Pixel Analysis

Sometimes metadata is not present and the forensic analyst must analyse pixel information to determine if image modifications have occurred. This section will introduce the concept as well as a few open source tools for doing this kind of analysis.

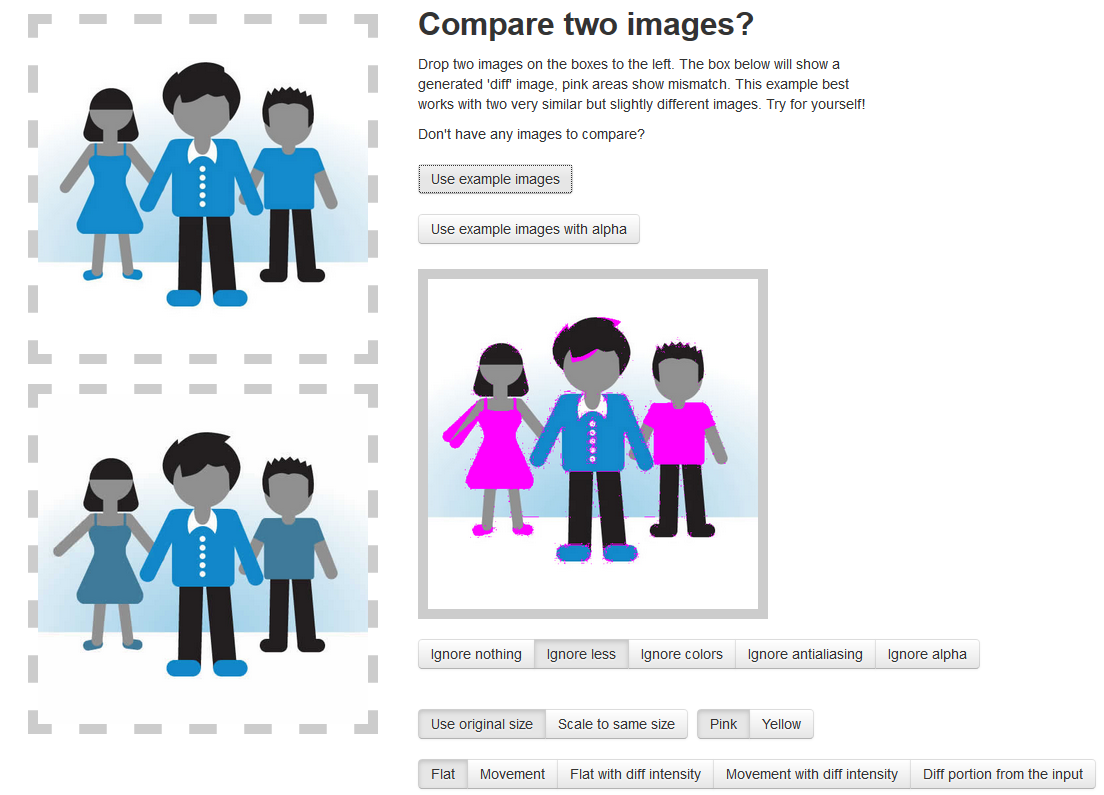
## Image Differences

Sometimes there are multiple copies of an image and we want to find out exactly what is different.

Fortunately, there are tools readily available online for this purpose, such as:

* <https://rsmbl.github.io/Resemble.js/>
* <https://online-image-comparison.com>
* <https://www.diffchecker.com/image-diff>

The first tool using Resemble.js is particularly good as it has several options which help to highlight changes in the files (particularly the “Ignore colors” and “Ignore antialiasing options). The interface simply requires that both images be uploaded separately and usually allows for the “difference” values to be shown, as below for the first link:



Use one of these tools and the files b1.jpg, b1\_edit.jpg, b2.png and b2\_edit.png to answer the following:

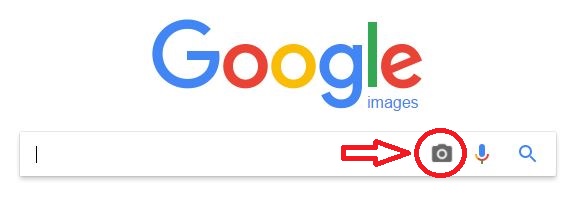
**Q7:** What has been added to **b1.jpg in b1\_edit.jpg**?

**Q8:** What has changed between **b2.png (not .jpg)** and its edited version, **b2\_edit.png**?

**Q9:** b2.png was also saved as a JPEG. **Compare b2.png and b2.jpg with the “ignore nothing”** setting on the resemble.js page. What causes so many pixels to be different? Does this explain some of the differences in b1.jpg and its edited version?

## Reverse Lookups – Where did this image come from?

It’s useful to be able to identify the source of an image if it came from the Web. This can be used to identify problem sites or to verify that an image has been taken from somewhere else. An example situation may be that of fake daring profiles which are often used in scams. If a suspect profile is using a publicly available stock image it indicates that the profile is likely a fabrication.

Reverse image searching can easily be performed in Google’s image search by dragging an image into the search bar, or by clicking the camera icon and using the file selection menu.

The results then show similar images which have been found on the Internet using some kind of perceptual matching algorithm (more on this next week!).

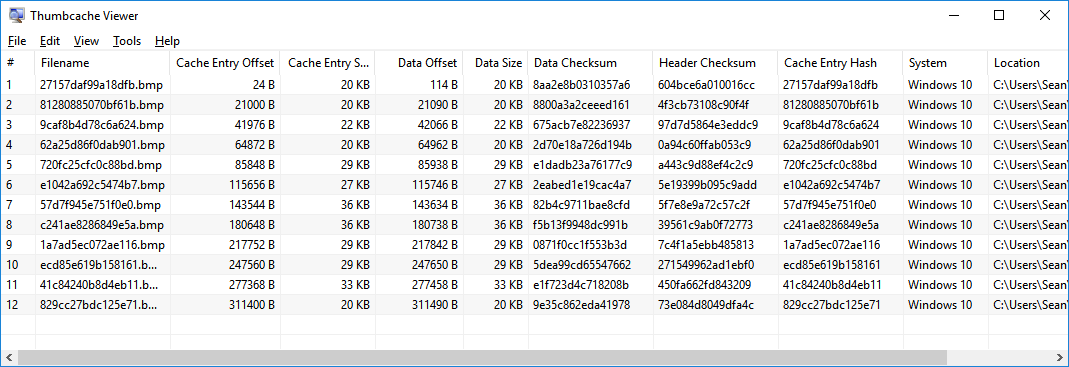
**Q10:** Drag and drop **b3.jpg** into Google image search. Who is the person in the photograph?

**Q11:** Using the same technique – what is the Flickr URL which **a1.jpg** was obtained from?

# Part C – Windows Thumbcache

Thumbnail analysis is an important part of image forensics. We have seen above that thumbnails can be embedded in regular JPEG files and that they may contain an old version of the image before it was modified.

We will now look at the Windows 10 thumbnail cache (thumbcache) to see how image previews for Windows Explorer are stored.

1. **Download the Thumbcache Viewer** from: <https://thumbcacheviewer.github.io/>
2. Find **thumbcache\_96.db (**from **afternoon\_evidence.zip),** and open it in the viewer. This is done by clicking File->Open and navigating to the db file.
3. Once the images are loaded you should be presented with something like the following:
   1. Click on each image in turn and view its contents in the pop-up preview window (it may be hidden behind the main window at first)
4. **Export all thumbnail images** by pressing **Edit->Select All** and then **File->Save all** and select a folder to save them in.

**Q12:** The list of images is almost identical to those found in the afternoon\_evidence.zip file for sections A and B – what is the internal file name of the new image?

**Q13:** Which other field (column) is this file name derived from?

**Q14:** Who is the man in the additional thumbnail? (Even if you know it, use reverse image search and the exported thumbnails to find out!)

The Windows thumbcache files are located at:

C:\Users\[**username]**\AppData\Local\Microsoft\Windows\Explorer\

* Try opening the different databases on the computer you are using to do the lab and see how they are different.

# Bonus: More Fun with JPEGs

If you are finished early, or just want to learn a bit more, try out the extra exercises below!

## Magic Numbers

As noted above, some data can come after the JPEG End of File (EOF) marker (0xFFD9), which is only intended to notify the decoder that it does not need to continue reading any data. However extra data may be appended after this marker which will not cause any errors when the file is opened.

**a3.jpg** has some data appended after the EOF marker, as noted at the bottom of the JPEGsnoop output:



**EX1:** What URL is present after the EOF marker in **a3.jpg**? (This is viewable with a hex editor, or even Notepad++)

We’ll come back to data hiding later in the module, but it’s worth noting that this kind of appended data does not invalidate the JPEG.

The lecture also mentioned the file header, or “magic number” which appears at the beginning of many files types. For JPEG this is 0xFFD8. While Windows often makes use of file extensions to determine what to do with a file, an application processing the file will use this magic number.

1. Change the file extension of **a1.jpg** to png (i.e. rename the file **a1.png**)
2. Note that Windows still displays a thumbnail for the file
3. Open the file in a metadata viewer – notice that the file loads without any issue.
4. **a1\_headerchanged.jpg** has had its magic number altered, such that even though the remainder of the file is unchanged, it no longer displays in Windows Explorer and opening it with any application causes an error.

**EX2**: What file type has the magic number for a1\_headerchanged.jpg been changed to?

**EX3:** How would you fix this file so that applications could decode it properly?