Using Facial Recognition to gather Social Media Intelligence

Jack Neilson

March 6, 2018

Contents

1 Literature Review

1.1 Background

1.1.1 SOCMINT

Social media intelligence (SOCMINT) is an emergent field in intelligence gathering where data is gathered from social media profiles. Massive amounts of data are added to social media services every day (Omand et al., 2012), much of it personal, making social media sites a potentially valuable resource when gathering information about groups or individuals (Ruiz, 2018). Social networks have also been used as a means of communication between persons of interest to the security services, making mining intelligence from their profiles a high priority (Omand et al., 2012)(Kilburn and Krieger, 2014).

After the 2011 riots in London that were organised in large part on social media, Her Majesty's Inspectorate of Constabulary stated that the police services were "insufficiently equipped" to effectively use SOCMINT in their response (Antonius and Rich, 2013) which suggests that social media intelligence sources may be woefully underutilised (Omand et al., 2012). This is not to say that the value of SOCMINT is not realised however, as many intelligence agencies are investing in tools to effectively gather and analyse SOCMINT (Antonius and Rich, 2013) or are performing case studies in to potential uses (Klontz and Jain, 2013).

While traditional human intelligence (HUMINT) focuses on building rapport and a foundation of trust in order to extract information from people of interest (Russano et al., 2014), users of social networking websites are much more likely to divulge personal information due to a misplaced senes of privacy (Livingstone, 2008). This makes SOCMINT attractive when attempting to gather data with little investment. The amount of data available to gather is vast in comparison to HUMINT sources (Omand et al., 2012), making mass collection and analysis viable (Unknown, 2013). The nature of SOCMINT makes it easier to analyse than HUMINT, which relies on "tells" and small social cues (Russano et al., 2014).

1.1.2 Uses of SOCMINT

As previously stated, SOCMINT has seen some emergent use particularly in the security services. The Greek Ministry of Defence has developed a framework to identify individuals fitting certain psychiatric profiles from their social media accounts to allow for early identification of potential insider threats (Kandias and Stavrou, 2015). By identifying factors that multiple intelligence agencies agree make a person more likely to pose an insider threat or negatively influence society (See appendix A), they were able to map usage habits (intensity, content, popularity) to these factors to draw conclusions about clusters of users. So far, the research has been helpful in insider threat prevention, delinquent behaviour prediction and forensic analysis support.

1.1.3 Facial Recognition

Facial recognition is a much more mature area of research than SOCMINT with many examples of industry usage. Facebook uses facial recognition to automate "tagging" photos with the identity of the persons pictured (Becker and Ortiz, 2008), and large companies are now releasing datasets such as YouTube Faces (Cui et al., 2013) in an effort to advance the field.

This is not to say that facial recognition is not without controversy however, as many privacy advocates have pointed out that accurate face regonition could infringe on their right to privacy (Ruiz, 2018). David Wood and Lucas Introne have posed that accurate facial recognition could lead to increased levels of surveillance, with no way to "opt out" (Introna and Wood, 2002)(Bowyer, 2004).

1.1.4 Uses of Facial Recognition

Facial recognition has many practical applications that are already being realised. As noted previously, Facebook uses facial recognition when "tagging" photos. This is presumably done to allow advertisers to more effectively target individual users - for example, a person identified in a photo with a barbeque may receive adverts for propane gas.

Facial recognition is also enjoying a heavy amount of attention from the security services due to it's use in identifying persons of interest. Case studies have been performed using images released to the public to ascertain how effective facial recognition is when looking for a specific person. In particular, Joshua Klontz and Anil Jain performed a case study using the images of the "Boston Bombers" against a set of test data (Klontz and Jain, 2013). Their approach was successful in recognising one of the perpetrators from a picture taken from his social media account (See appendix B).

1.1.5 Constrained vs Unconstrained

While facial recognition software has come a long way, achieving accuracy rates of up to 99% on small, consistent data sets (Best-Rowden et al., 2014), it is still in it's infancy when it comes to identifying people in "unconstrained" images. Images taken in the wild may have large variations in pose, facial occlusion and ambient lighting. This makes it difficult to identify facial features or markers (such as iris distinace, nasal distance, blemishes) which in turn has a negative impact on accuracy rates (Klare et al., 2015). When looking at applications of face recognition software with unconstrained datasets, matches are typically achieved when the test image has similar pose, facial occlusion and lighting as the sample image (See appendix B).

1.2 Prior Knowledge Attacks

1.2.1 Social Engineering

Social engineering in the context of information security refers to the ability of a person to gain access to information or control systems through a user or administrator, rather than through any technical oversight (Bakhshi, 2008). It is a popular technique against "hardened" targets as technical prevention measures have proven to be ineffective (Krombholz et al., 2015) making the human users and administrators the weakest link in the proverbial chain. In addition, it is difficult to train users in defenses against social engineering attacks. People believe they would not fall for such a trick despite research showing that humans perform poorly when attempting to detect deception (Krombholz et al., 2015)(Bakhshi, 2008).

Social engineering using e-mail as a medium is ubiquitous (Bakhshi, 2008). As a generel strategy, the person wishing to gather information will send one or more people an e-mail in hopes of a response (whether that be in the form of visiting a website, replying with their username or password, opening a file etc). Recent studies in realistic environments have had success rates of up to 23% (Bakhshi, 2008) - this is especially worrying given that even 1 repsondant could compromise an entire company.

Examples of social engineering are not hard to find. In his book *The Art of Deception* Kevin Mitnick describes a young Stanley Rifkin using social engineering to make the biggest bank heist of all time, stealing over 10,000,000 USD (Mitnick and Simon, 2011). Social engineering attacks are certainly not difficult to perform - recently, a 13 your old child used social engineering to breach the private e-mail of the then-chief of the CIA, John Brennen (Timm, 2015)

1.2.2 Spearphishing

Spearphish attacks are a subsection of social engineering attacks wherein an attacker sends a malicous e-mail to a user that has been crafted using information that would make it seem authentic (Caputo et al., 2014). For example, a normal social engineering attack using e-mail might send boilerplate messages to all personnel in a department to attempt to gain access to a network account. By contrast, a spearphish

attack may use the names of specific people in the department, the location of the department, a spoofed e-mail header to make it seem as if the message came from within the department, and so on.

Social engineering attacks have proven to be effective at accessing sensitive information even when significant effort has been expended to secure it. A recent spearphishing campaign was conducted against 500 US military cadets - over 80% clicked the link in the e-mail (Caputo et al., 2014). Spearphish attacks are even more effective than "blind" phishing because of the additional information included in the e-mail which lulls the user in to a false sense of security (particularly when the information could be wrongfully considered "sensitive", for example including a manager's name and phone number).

Spearphish attacks in particular are difficult to mitigate against. They target the human element in information security making technological defenses insufficient (Caputo et al., 2014). Education on spearphishing is not adequate to mitigate the potential threat either, as shown by William Pelgrin's exercise. He sent 10,000 New York State employees a phishing e-mail, and had a success rate of 15% (that is, 15% of users clicked the link in the e-mail then attempted to enter their passwords). The experiment was repeated after four months, with some success shown as the experiment had a success rate of 8% (Parmar, 2012). It must be remembered however that even a single successful spearphish attack could lead to a breach of information security.

1.3 Intelligence Gathering

1.3.1 SOCMINT

Social media intelligence (SOCMINT) refers to information gathered from social media profiles hosted on social networks. It is an emergent field in open-source intelligence (OSINT), which relies on gathering information users divulge about themselves in the public domain. It is characterised by the massive amount of data available (Omand et al., 2012) and the difficulty of analysis (Bartlett and Reynolds, 2015). An overview of subjects relating to social media intelligence gathering follows below.

1.3.2 HUMINT

Human intelligence (HUMINT) pertains to the gathering of intelligence from individual human subjects. Information may be divulged non-consensually e.g. in the case of interrogation (Evans et al., 2010), or consensually in the case of clandestine information gathering (Musco, 2017).

Non-consensual information gathering via interview or interrogation has only recently become a subject of study for the general public (Russano et al., 2014).

Consensual information gathering sits in a much more grey area. Presenting yourself as somebody else may not be illegal depending on the circumstances, however it poses several moral questions. Clandestine intelligence gathering is still an extremely effective strategy, particularly when attempting to gather sensitive information which may be more heavily protected e.g. airgapped, firewalled (Musco, 2017). This makes it attractive during wartime or times of civil unrest (Charters, 2018)(Gioe, 2017)

Using a human intelligence approaches when gathering information has several downsides. It is a high risk strategy, as should a person be found out the consequences can be severe (Charters, 2018). The potential reward of sensitive information may be deemed to not be worth the risk. It goes without saying that HUMINT does not scale particularly well - it is a useful tool when attempting to extract information from a single person or small group, but it is much less useful when gathering information about larger groups. It relies on trust being built and may be ineffective when attempting to gather information from targets trained in tradecraft (Charters, 2018)(Musco, 2017)(Gioe, 2017).

1.3.3 Individual vs Group Data

Much of the research done on social media intelligence has focused on group trends or finding subsets of a population (Antonius and Rich, 2013)(Omand et al., 2012). Comparatively, fairly little research has been done on identifying a single person of interest from a social network service.

A practical example of where SOCMINT has been used in the real world is the framework created by Kandias and Stavrou, in which features of a social media profile such as number of friends, "friend hops" standard deviation, psychiatric profiling and usage intensity are used to predict groups of interest that may become radicalised (Kandias and Stavrou, 2015). While this is certainly very useful, it focuses more on identifying groups by some common factor rather than identifying a single person of interest for further analysis.

1.3.4 Quantity of Information

Having limited information is not a problem when gathering social media intelligence. Rather, the opposite is true - 250,000,000 photos are added to Facebook every day (Omand et al., 2012). The vast amount of data is what makes searching for social media so difficult (and nigh on impossible in real time).

While the total amount of data added to social networking websites is extremely large, the information about a single user may be fairly small especially if the user has been trained to release as little information as possible (Kandias and Stavrou, 2015). This may make targeting single persons of interest less valuable, however should information "leakage" occur the potential payoff is high. It should be remembered that scanning social media profiles incurs little to no risk, something that cannot be said of human intelligence.

1.3.5 Accessibility of Data

As discussed previously the amount of data pushed to social networking websites is massive. This does not mean that all social networking data is readily accessible. Users may set privacy settings to disallow unauthorised parties from viewing their profile, or particular parts of it. This can be mitigated by having the service allow access, however acquiring access as a party of one may prove difficult.

Several social networking services also impose rate limits on their interfaces (for example, Facebook imposes a limit of 100 API calls per user of an application as well as limits on CPU time used). These limits obviously make searching through the entire data set of profiles unfeasible without allowances made on the side of the social network.

1.3.6 Uses

Usage of social media intelligence has been limited in large part due to the difficulty in analysing such large amounts of data for small snippets of useful information (Omand et al., 2012). There are, however, some interesting case studies available.

In April 2013, Edward Snowden leaked a deck of slides used by the NSA to brief people on the "PRISM" program (Unknown, 2013). These slides detailed how the NSA uses mass surveillance with cooperation from companies including Google, Facebook, and Apple, to conduct surveillance against persons of interest. As a government agency enabled by the oversight committee, the NSA has the power to require large social networking services like the ones listed to allow them access to their data.

As mentioned previously, Kandias and Stavrou have developed a framework for using social media intelligence for predicting and mitigating insider threats while working at the Information Security and Critical Infrastructure Protection Lab at the University of Athens (Kandias and Stavrou, 2015). The framework focuses on analysing the psychology of the subjects using their social media posting habits. It shows that seemingly inconsequential data such as posting frequency may still be useful when analysing social media intelligence. An example graph to find the amount of "klout" a social media user may have in the Nereus framework is referenced in Appendix C.

1.3.7 Challenges and Constraints

There are several challenges to consider when collecting and analysing data for social networking services. First is the issue of privacy. Many people see social media profiles in a similar setting as a

meeting between friends, and as such post things which they perhaps would not say or show in public (Livingstone, 2008). The chilling effect of public scrutiny, or even direct surveillance, does not seem to put a damper on this.

There are some that feel that police surveillance of social networks is too pervasive (Mateescu et al., 2015), and encroaches on their right to privacy. Despite the effectiveness of social media intelligence gathering, it may be harmful to law enforcement's public image if it continues to utilise SOCMINT.

As far as the law is concerned, information posted on social networking sites is considered to be in the public domain. This makes it legal for law enforcement to gather social media intelligence, provided they do it the same way an unprivileged user would. Several social networks have clauses against mass data collection in their Terms of Service, however it is theorised that this is to prevent competition from other companies when selling said information to advertisers.

1.4 Facial Recognition

1.4.1 Current Applications

Police and military uptake of face recognition technology is widespread. For instance, police in the United Kingdom recently used cameras to scan people's faces at the "Download" music festival (Gallagher, 2005). These images were then compared with a database of custody images across Europe, to identify known criminals for the purposes of crime prevention and outstanding warrant execution.

Facebook also uses face recognition on photos uploaded to its service (Facebook, 2018). By comparing face mappings to the known face mappings of yourself and immediate friends, it can suggest "tags" of who is in the image which presumably makes the information contained within the image more valuable to advertisers.

1.4.2 Unconstrained Facial Recognition

Unconstraned facial recognition is a subset of facial recognition which focuses on identifying faces in images with variations in pose, lighting, and facial occlusion. It is significantly more difficult than facial recongition which imposes constraints on these conditions.

Unconstrained facial recognition is typically used in situations where a subject is unaware or unwilling. As already mentioned, it was used in a case study by Klontz and Jain where they compared the images released by the FBI of Tamerlan and Dzhokhar Tsarnaev (the "Boston Bombers"). They found that by comparing the released images taken from security cameras to images on social media in an unconstrained setting, they could positively identify one of the brothers (Klontz and Jain, 2013) (see Appendix B).

2 Methodology

2.1 Overview

The software implemented will be in the form of a Python command line program, which allows users to specify an input image and data set to check it against. Additional identifying information such as name, date of birth etc. should also be utilised if given. The data sets the software references must come from social networking websites (or in the case of this proof of concept, a data set which is an accurate representation).

As its output, the software must show the five most similar faces in the data set to allow for human verification. They should be presented with information that was gained from their social media profiles such as age, gender, name etc., as well as the similarity to the test image. In the case that the software is being ran headlessly, the results should be available in a file format which includes an image of the match and a representation of their social media profile.

2.2 Functional Requirements

2.2.1 Face Recognition

The software must implement a method of facial recognition, to rank a set of faces by similarity to a given test image. The test face must be extracted from the input image. The top 5 matches should be outputted, along with relevant identifying information.

2.2.2 Social Media Integration

The software must have the ability to use social networking websites as it's data set when comparing a test face. It would be desirable for the software to have the capability to use other identifying information alongside the test image to narrow the search space. Due to limited time and resources, test accounts may be used as a proof of concept.

2.2.3 Headless

The software should be able to function without a GUI, since the requirement of being able to use extremely large data sets may require deployment to a large server cluster or to a cloud.

2.2.4 Multithreaded

Where applicable, the software should make full use of parallelisation to increase its speed and resource utilisation. This is particularly important when working with large data sets.

2.3 Non-functional Requirements

2.3.1 Accuracy

The software should have a *relatively* high success rate when identifying a person from a social media collection. Note that typical facial reocgnition performance in unconstrained environments achieve a maximum success rate of 30%.

2.3.2 Usefulness

The information gained by using the software should help when crafting prior-knowledge attacks, increasing their effectiveness.

2.3.3 Cross-Platform

The software must be cross-platform to allow for easy deployment to servers or a cloud.

2.3.4 Imprecision Tolerance

The software should attempt a best guess when using an input image with less than ideal pose, facial occlusion or lighting. Inferior results when given a test image like this are acceptable to a degree.

2.3.5 Usability

The software is being developed for use by information security professionals in the form of a command line tool. This is ubiquitous in the industry, so no more than a cursory glance over the usage information should be needed to begin using the tool.

2.3.6 Facial Recognition Libraries

Due to time constraints, reimplementing facial recognition capabilities is not feasible. Therefore, the software should make use of already existing facial recognition libraries, in particular dlib or OpenCV.

2.3.7 Language

The software should be written in Python to allow for easy use of face recognition programs and rapid prototyping.

2.3.8 Time Spent

The software must be able to complete the above task in the same time or less than a human.

2.3.9 Documentation

The software should have at a minimum a man page and usage information. More documentation would be beneficial.

2.3.10 Testing

The software will be tested using a comination of publicly available face recognition datasets including Labeled Faces in the Wild, the CMU Face in Action database, and the SCFace database.

Include final list of databases

3 Implementation

3.1 Face Recognition

The application has been developed using the face recognition module created by A. Geitgey:
    `https://github.com/ageitgey/face\_recognition/`.

It is an easy to use facial recognition library build on top of dlib. It exposes a python API, making it ideal for this project. It also boasts an accuracy rate of up to 99.38% on the LFW data set.

Alternate libraries (such as dlib) that were investigated were found to either have a much higher barrier to entry, or a much lower accuracy.

3.2 Test Server

3.2.1 nginx

To test the application, nginx is used to serve static content as well as as a reverse proxy. Nginx was chosen over Apache or another native Python webserver due to its capability at handling massive concurrent requests, its flexibility, and its speed at server static content. Since the application makes several connections simultaneusly and requests large amounts of images, nginx was ideal.

3.2.2 cherrypy

To serve a dynamic description of where profiles and their respective images are located, a cherrypy server was implemented which expands a glob from the test directory (which contains the test data sets) then returns a json representation of them. As noted earlier, nginx acts as a reverse proxy to this service

3.3 Portable Module

The application has been developed in a modular fashion. To this end, a run file has been provided alongside a python module. The python module includes the logic for retrieving profiles from a web service and setting up multiple threads to compare face mappings, whereas the run file comes with some pre-processing and default arguments. Ideally, the module should be re-usable for other web services.

3.4 Threading

The application has been developed to make use of multiple threads. The main thread in *face-gather.py* controls these threads and their results using locks, thread-safe queues, semaphores, and the thread.join() method.

The producer / consumer consumer model was useful when developing the application. In this context, the producers retrieve profiles from the social networking service and the consumers compare the retrieved profiles to the test data given and store the result.

3.5 Command Line Interactivity

Rather than use a graphical user interface (GUI), the application makes use of command line arguments (see Appendix D). There are several reasons for this; it cuts down on needless computational cost by eliminating the overhead of a GUI, it allows the OS the application is running on to be "headless" (i.e. run with only a command line or remote connection for user interactivity) again cutting down on overhead, and it allows for easier integration in to scripts (for example, appending logs to a log file using the POSIX ¿¿ operator).

One potential issue when using an application without a GUI is presenting the results. The application persists its results in a JSON format for use by other applications or for direct consumption by the end user.

`results`

3.6 Social Network Interactivity

Since this project has been done on a fairly small scale in a short time frame, it was not possible to gain access to a large social media platform to perform testing (ignoring the obvious ethical issues with such an approach). Instead, the test environment has been set up in such a way as to mirror the way in which a social media API may be presented - with an array of profiles, each one having a link to some related resources.

3.7 Challenges

3.7.1 Loading Images from Memory

Since the application retrieves images remotely over the internet, it makes sense to manipulate the images directly from memory. This avoids filesystem reads and writes, which are several magnitudes slower than memory reads and writes. The time save is massive, especially when repeated over thousands of images.

Unfortunately the face recognition library does not support loading images from memory, instead providing a method (load_image_file(uri)) to load images from the local file system. Upon inspection this method loads an image file from disk in to a Python Imaging Library (PIL) Image object, then casts it to a numpy array to allow for encoding operations. To mirror this method for use with images loaded to memory, the application uses the PIL Image.open() method on the raw response stream from the social network web server to create an Image object, then casts it to a numpy array. This achieves the same result as loading the image from the local file system significantly more efficiently.

```python
# Load an image in to memory from local or remote sources
def load_images(uri, remote=True):
    if remote:
        print('Retrieving image from ' + uri)
        resp = requests.get(uri, stream=True)
        resp.raw.decode_content = True
        img = Image.open(resp.raw)
        return fr.face_encodings(numpy.array(img))
    else:
        print('Retrieving image from ' + uri)
        img_array = fr.load_image_file(uri)
    return fr.face_encodings(img_array)
```

Above: A function with an optional keyword parameter to load an image from either a local or remote source.

3.7.2 Maximum Loaded Images

A potential problem when using the producer / consumer model is that producers may produce too much data, filling all available memory and leading to a crash. To solve this a maximum loaded images parameter can be supplied when running the application, which will cap the number of face encodings loaded into memory at that number. This has been implemented using a semaphore initialised with the maximum number of face encodings to be loaded. Each producer thread calls acquire() before loading a set of encodings in to memory, and each consumer thread calls release() when the encoding has been compared and the memory can be freed.

```
# Producer thread
    self.waiting_counter.acquire()
        if profile_queue.empty():
            return
        profile = profile_queue.get()

# Consumer thread
        img = img_queue.get()
        ---
        Comparison happens here
        ---
      self.counter.release()
```

Above: The producer thread calls acquire() on the semaphore before loading mappings in to memory, and the consumer thread calls release() on the semaphore after the memory used for mappings has been freed.

3.7.3 Multiple Threads for Producing and Consuming

As it stands, the application has been developed with mulitple threads in mind. The user can define how many threads should be used for downloading images and producing face mappings via the –cthreads argument, and how many threads should be used to compare face mappings via the –pthreads argument. This allows for much higher processor utilisation than a single threaded application, as all CPU cores can be utilised.

While the application processes images orders of magnitude faster than it would had it been single-threaded, it may still be too inefficient for use on particularly large data sets. It is also difficult to scale due to its singleton nature. In the future it may be worth refactoring to make use of a discrete GPU, or to make use of specialised libraries for big data processing such as MapReduce.

3.7.4 Thread Object Access Control

To control object access between threads two Queue objects are used. One contains a JSON representation of all of the profiles in the search spaces and is only accessed by the producer threads, and the other is used by producer threads to send face mappings of profiles to consumer threads. These objects are guaranteed to be threadsafe, meaning that both the put() and get() operations are atomic. To control access to the result object that multiple consumer threads may try to access, a Lock() is passed in the consumer constructor. Before a consumer accesses the result object it must first call lock.acquire() which will block if another thread has acquired the lock first. After a consumer has finished writing to the result object it releases the lock by calling lock.release(), allowing other consumer threads access.

```
        self.result_lock.acquire()
            self.result.add([distance, img[1]])
            self.result_lock.release()
```

Above: The result object is not thread-safe, so must have a lock around it.

3.7.5 Result Retrieval

Having a "top 5" result as was initially planned would require many comparison and list traversal operations. It would also make having multiple consumers pointless, as only one thread would be able to do comparisons at a time. Rather than having the top 5 matches, the result returned is now all faces that match the test face within a given threshold (by default, face distance < 0.6).

3.7.6   Command Line Arguments

The application can take many command line arguments to customise its use. Having the user fill out and remember every one of these arguments would make the application unintuituve and difficult to use. Therefore, the decision was made to make heavy use of optional keyword arguments both when running the program (e.g. --cthreads=1) and in functions within the application. This allows the user to specify some, all, or none of the optional parameters and still have the application function correctly.

```python
def search(test_face_location,
           uri,
           no_producer_threads=None,
           no_consumer_threads=None,
           max_loaded=None,
           threshold=None,
           name=None):
```

Above: The search function uses several keyword arguments with default arguments, then checks and loads with defaults if not supplied before continuing.

3.7.7   Inter-Thread Communication

The only inter-thread communication required by the application is for the consumer threads to be signalled when there are no more face mappings to compare. To this end, the main thread which spawns both the producer and consumer threads blocks by calling producer.join() on all producer threads. One all producer threads have returned, the main thread pushes a sentinel object on to the end of the face mapping queue. It then blocks by calling consumer.join() on all consumer threads. Once a consumer thread pops the sentinel object, it adds another sentinel to the end of the queue (for further consumer threads to consume) then returns. This guarantees that there is no unconsumed data, and that the consumer threads do not terminate prematurely.

```python
# Main thread
for processor in processor_list:
    processor.join()
img_queue.put(sentinels.NOTHING)

for recogniser in recogniser_list:
    recogniser.join()

# Consumer thread
if img == sentinels.NOTHING:
    img_queue.put(sentinels.NOTHING)
    return
```

Above: The main thread blocks until all producer threads are finished, then pushes a sentinel object to the end of the queue, then blocks until all consumer threads are finished.

4   Testing

5   Results

6   Evaluation

7   Conclusion

References

Antonius, N. and Rich, L. (2013), 'Discovering collection and analysis techniques for social media to improve public safety', **3**, 42.

Bakhshi, T. (2008), 'A practical assessment of social engineering vulnerablities'.

Bartlett, J. and Reynolds, L. (2015), the state of the art 2015 a literature review of social media intelligence capabilities for counter- terrorism, Technical report, Centre for Analysis of Social Media.

Becker, B. C. and Ortiz, E. G. (2008), Evaluation of face recognition techniques for application to facebook, *in* 'Automatic Face & Gesture Recognition, 2008. FG'08. 8th IEEE International Conference on', IEEE, pp. 1–6.

Best-Rowden, L., Han, H., Otto, C., Klare, B. F. and Jain, A. K. (2014), 'Unconstrained face recognition: Identifying a person of interest from a media collection', *IEEE Transactions on Information Forensics and Security* **9**(12), 2144–2157.

Bowyer, K. W. (2004), 'Face recognition technology: security versus privacy', *IEEE Technology and Society Magazine* **23**(1), 9–19.

Caputo, D. D., Pfleeger, S. L., Freeman, J. D. and Johnson, M. E. (2014), 'Going spear phishing: Exploring embedded training and awareness', *IEEE Security & Privacy* **12**(1), 28–38.

Charters, D. A. (2018), 'Professionalizing clandestine military intelligence in northern ireland: creating the special reconnaissance unit', *Intelligence and National Security* **33**(1), 130–138.

Cui, Z., Li, W., Xu, D., Shan, S. and Chen, X. (2013), Fusing robust face region descriptors via multiple metric learning for face recognition in the wild, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 3554–3561.

Evans, J. R., Meissner, C. A., Brandon, S. E., Russano, M. B. and Kleinman, S. M. (2010), 'Criminal versus humint interrogations: The importance of psychological science to improving interrogative practice', *The Journal of Psychiatry & Law* **38**(1-2), 215–249.
URL: *https://doi.org/10.1177/009318531003800110*

Facebook (2018), 'How does Facebook suggest tags?', `https://www.facebook.com/help/122175507864081?helpref=faq_content`. Online; accessed 2nd March 2018.

Gallagher, P. (2005), 'Download festival: Facial recognition technology used at event could be coming to festivals nationwide'.

Gioe, D. V. (2017), the more things change: Humint in the cyber age, *in* 'The Palgrave Handbook of Security, Risk and Intelligence', Springer, pp. 213–227.

Introna, L. and Wood, D. (2002), 'Picturing algorithmic surveillance: The politics of facial recognition systems', *Surveillance & Society* **2**(2/3).

Kandias, M. and Stavrou, V. (2015), 'Personal traits analysis as a means to predict insiders'.

Kilburn, M. and Krieger, L. (2014), 'Policing in an information age: The prevalence of state and local law enforcement agencies utilising the world wide web to connect with the community', *International Journal of Police Science & Management* **16**(3), 221–227.
URL: *https://doi.org/10.1350/ijps.2014.16.3.341*

Klare, B. F., Klein, B., Taborsky, E., Blanton, A., Cheney, J., Allen, K., Grother, P., Mah, A. and Jain, A. K. (2015), Pushing the frontiers of unconstrained face detection and recognition: Iarpa janus benchmark a, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 1931–1939.

Klontz, J. C. and Jain, A. K. (2013), 'A case study on unconstrained facial recognition using the boston marathon bombings suspects', *Michigan State University, Tech. Rep* **119**(120), 1.

Krombholz, K., Hobel, H., Huber, M. and Weippl, E. (2015), 'Advanced social engineering attacks', *Journal of Information Security and applications* **22**, 113–122.

Livingstone, S. (2008), 'Taking risky opportunities in youthful content creation: teenagers' use of social networking sites for intimacy, privacy and self-expression', *New Media & Society* **10**(3), 393–411.
URL: *https://doi.org/10.1177/1461444808089415*

Mateescu, A., Brunton, D., Rosenblat, A., Patton, D., Gold, Z. and Boyd, D. (2015), 'Social media surveillance and law enforcement', *Data Civil Rights, October* **27**, 2015–1027.

Mitnick, K. D. and Simon, W. L. (2011), *The art of deception: Controlling the human element of security*, John Wiley & Sons.

Musco, S. (2017), 'The art of meddling: a theoretical, strategic and historical analysis of non-official covers for clandestine humint', *Defense & Security Analysis* **33**(4), 380–394.

Omand, S. D., Bartlett, J. and Miller, C. (2012), 'Introducing social media intelligence (socmint)', *Intelligence and National Security* **27**(6), 801–823.
URL: *https://doi.org/10.1080/02684527.2012.716965*

Parmar, B. (2012), 'Protecting against spear-phishing', **2012**, 811.

Ruiz, J. (2018), Gchq and mass surveillance, Technical report, Open Rights Group.

Russano, M. B., Narchet, F. M., Kleinman, S. M. and Meissner, C. A. (2014), 'Structured interviews of experienced humint interrogators', *Applied cognitive psychology* **28**(6), 847–859.

Timm, T. (2015), 'The cia director was hacked by a 13-year-old, but he still wants your data', Published in the Guardian newspaper.

Unknown (2013), 'Prism slides', Leaked to several newspapers by Edward Snowden in late 2013.

Appendices

A Threat Graph
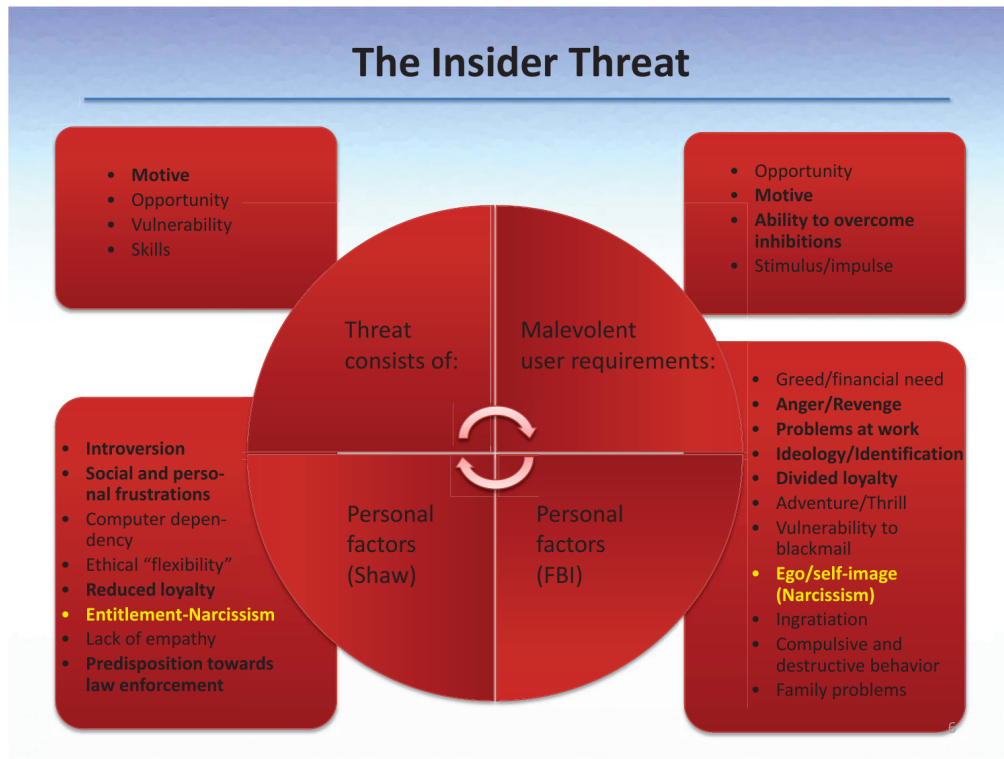
Graph of insider threat factors (Kandias and Stavrou, 2015).

B   "Boston Bomber" Identification
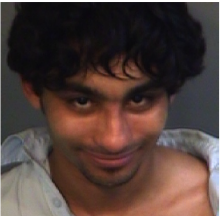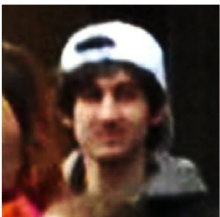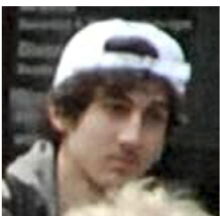
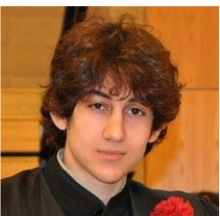| Probe	Rank 1	Rank 2	Rank 3

Table of potential matches, note the correct identification from the picture taken from social media with similar pose and lighting (Klontz and Jain, 2013).

C Connection Graph

Category	Influence valuation	Klout score	Usage valuation
Loners	0 - 90	3.55 - 11.07	0 - 500
Individuals	90 - 283	11.07 - 26.0	500 - 4.500
Known users	283 - 1.011	26.0 - 50.0	4.500 - 21.000
Mass Media & Personas	1.011 - 3.604	50.0- 81.99	21.000 - 56.9000

Graph of social media connections and klout score in the Nereus framework(Kandias and Stavrou, 2015).

D Command Line Arguments

```
jack@seven:~/Documents/hons_project$ python3 run.py -h
usage: run.py [-h] [--pthreads PTHREADS] [--cthreads CTHREADS]
              [--maxload MAXLOAD] [--name NAME]
              test_face service

A python3 program to gather social media intelligence using facial recognition

positional arguments:
  test_face             Location of the test face
  service               Social network service to search

optional arguments:
  -h, --help            show this help message and exit
  --pthreads PTHREADS   Number of image processing threads
  --cthreads CTHREADS   Number of image comparison threads
  --maxload MAXLOAD     Maximum number of images pre-loaded into memory
  --name NAME           Name of person of interest
```

A help page showing the command line arguments that can be supplied.