REDLIGHT
AN EFFICIENT ILLICIT IMAGE DETECTION
APPLICATION FOR LAW ENFORCEMENT
BY
SEAN P ALVAREZ

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER SCIENCE AND STATISTICS

UNIVERSITY OF RHODE ISLAND
2012
ABSTRACT

Law enforcement investigators manually sort through hundreds of thousands of images when investigating a hard drive for the presence of illicit images. This is error prone, as many items may be missed due to the volume of images investigated. This is also very time-consuming for the investigator, creating backlogs of cases needing to be processed that can stretch for months or even longer. This thesis presents a user friendly application that is immediately available for use by law enforcement that can be used to increase the efficiency of searching suspect storage media for illicit content in both images and video. By using this application, an investigator can process new investigations in a much shorter period of time than can be done manually, allowing for extra time to get caught up on any backlog of cases. The application is approximately 85% accurate at detecting illicit images, and is able to process at least 38 images per second, which is significantly faster than can be done manually. The application is multi-threaded in such a way that processing speed will go up depending on how many processing cores are available in the machine. The application requires less than 30MB of memory to start, allowing it to run on any relatively modern machine. This enables any department or division of law enforcement to use the application without incurring any costs of upgrading or replacing workstations.
TABLE OF CONTENTS

ABSTRACT .................................................................................................................... ii

ACKNOWLEDGMENTS ................................................................................................. iii

TABLE OF CONTENTS ............................................................................................... iv

LIST OF TABLES ........................................................................................................ vii

LIST OF FIGURES ....................................................................................................... viii

CHAPTER 1: INTRODUCTION ..................................................................................... 1

CHAPTER 2: BACKGROUND ......................................................................................... 4

2.1 Law Enforcement Criteria ...................................................................................... 4

2.1.1 The tool should be fast in processing images .................................................. 4

2.1.2 The tool should have a low memory footprint ................................................. 5

2.1.3 The tool should be reasonably accurate ....................................................... 6

2.1.4 The tool should be easy to use ..................................................................... 6

2.1.5 The tool should have a feature set that facilitates law enforcement investigations ................................................................................................................ 7

2.2 Related Work ....................................................................................................... 7

2.2.1 Other Techniques and Tools ........................................................................... 8

2.2.2 Machine Learning ........................................................................................... 9

2.2.3 Linear Discriminant Analysis ........................................................................ 9

2.2.4 Viola-Jones Facial Detection ......................................................................... 10

2.2.5 URI Developed Classification Program ......................................................... 10
4.2 Criteria: The tool should have a low memory footprint, enabling it to be run on lower end machines.

4.3 Criteria: The tool should be reasonably accurate.

4.4 Criteria: The tool should be easy to use and the tool should have a feature set that facilitates law enforcement investigations.

CHAPTER 5: CONCLUSION

APPENDIX A: SAMPLE MD5 HASH OUTPUT REPORT

APPENDIX B: SAMPLE CSV FILE OUTPUT REPORT

APPENDIX C: SAMPLE HTML OUTPUT REPORT

BIBLIOGRAPHY
LIST OF FIGURES

Figure 1: Application Design ................................................. 13
Figure 2: Main GUI ................................................................. 14
Figure 3: Search Options by File Path or Logical Drive .................. 15
Figure 4: Advanced Options Dialog ......................................... 16
Figure 5: Runtime Results Display .......................................... 21
Figure 6: Business Logic Tier ................................................... 24
Figure 7: Directory Search Routine .......................................... 25
Figure 8: Scanning Thread State Diagram .................................... 29
Figure 9: Engine Functions ...................................................... 30
Figure 10: Application Data Flow ............................................. 33
Figure 11: Save Report Options ............................................. 36
application needs a user interface that is intuitive for law enforcement investigators to use regardless of their technical skill level.

This thesis presents an application that has been designed and implemented in such a way that will allow law enforcement investigators to use the URI developed classification program through an easy to use graphical user interface (GUI). In order to allow the application to be useful to law enforcement, the following criteria have been met:

1) The tool should be fast in processing images.
2) The tool should have a low memory footprint, enabling it to be run on lower end machines.
3) The tool should be reasonably accurate.
4) The tool should be easy to use with minimal or no training needed.
5) The tool should have a feature set that facilitates law enforcement investigations.

The outcome of this thesis is an automated application for illicit image detection that meets the criteria needed by law enforcement investigators. The application can process a minimum of 38 images per second, with speeds increasing as more processing cores are made available due to the multi-threaded nature of its design. The application requires less than 30MB of memory in order to start, and can display over 27,000 positive results on a 32-bit machine. Scanning is approximately 85% accurate when detecting illicit images, and has many features specifically designed for law enforcement use. The application is user-friendly, commercial in quality, and is also extensible towards the integration of future improvements. One feature that is
CHAPTER 2: BACKGROUND

This section will describe law enforcement criteria and how it will be met by the application, followed by background information on machine learning and the program developed at URI for automated image classification.

2.1 Law Enforcement Criteria

In working with law enforcement investigators it has been determined that the following are the most important characteristics of a good tool to assist them in investigations.

2.1.1 The tool should be fast in processing images

In order to determine if a suspect computer contains illicit images, an investigator must first find images within the computer’s file system. In most computers these images are located in common places, such as a user’s 'Picture' folder, or the location of temporary files used by an internet browser. However, in some cases these images may not be located in these folders, either due to a particular application or an intentional attempt to hide them. Through rough estimation, navigating to a common location within the file system can take as much as 30 seconds. Knowing that images on a computer can be located in multiple areas, it can be estimated that as much as three minutes will be spent simply searching the drive to find images.
due to either budget or procedural constraints, are unable to update the hardware used by their investigators on a regular basis. The intention of this tool is to allow it to run on as many different computers as possible, provided the Microsoft Windows® operating system is being used (XP and later). In order to meet this requirement, an assumption is made that the majority of investigators are using computers manufactured no earlier than 2007. In order to meet this criterion, this thesis minimizes the use of memory across the overall design of the application.

2.1.3 The tool should be reasonably accurate

   When investigating a computer for the presence of illicit images, it may not be necessary for every single illicit image to be found in order to press charges or to indicate that a more thorough investigation should be conducted. Accuracy of the URI classification program is determined by the digital forensics research group, not directly by the design and implementation of the application. Thus, the primary accuracy criterion for this work is to not adversely affect the accuracy of the URI program that the tool will be using.

2.1.4 The tool should be easy to use

   Due to both budget and time constraints, it is not feasible for law enforcement agencies to invest in lengthy training for every tool that they have to use. This thesis has made the application very easy for anyone to use with little or no training, so that investigators can add it directly into their tool kit and start using it as quickly as possible.
2.2.1 Other Techniques and Tools

Some of the most prominent commercial illicit image scanners are:

1) Content Scanner (ContentPurity 2011)
2) Snitch (Hyperdine Software 2011)
3) Net Eraser (Web PC Tools 2011)
4) Media Detective (TapTap Software 2011)
5) Porn Detection Stick (Paraben Corporation 2011)
6) X-Ways Forensics (X-Ways Software Technology 2011)

Only X-Ways Forensics is directed solely towards law enforcement use.

However, X-Ways Forensics uses a skin tone only approach to detecting illicit images, which means it simply looks for the percentage area of the image that contains a color that could be interpreted as 'skin tone'. If this percentage exceeds a certain threshold, the image is returned as illicit. In addition, this is only available as part of a larger forensic analysis application, which may be much more than an investigator needs if all they want to do is run a scan for illicit images.

Porn Detection Stick does have some features for law enforcement as well, but this tool does not provide real-time display of detected images. The application must be run to completion (or cancellation) before any detected images can be viewed. It also has many features that are directed towards use by parents or schools, which would be unnecessary to an investigator.
line, as much as possible. Once this line has been generated, any new data presented to the LDA can be compared against this line and its class can be determined based on which side of the line the data falls on.

2.2.4 Viola-Jones Facial Detection

The URI classification program can optionally make use of an algorithm called the Viola-Jones Method (Viola and Jones 2001) for facial detection within an image. While the algorithm was originally designed to detect generic objects, it is frequently used for the purpose of detecting faces within images. The Viola-Jones method evaluates pixels within given rectangular sections of an image file, which it then uses to generate ‘classifiers’, or data that is indicative of the shapes in the rectangle. These classifiers are then fed through a ML algorithm in a cascading fashion, producing a model file known as a Haar Cascade. Viola-Jones facial detection is implemented as part of the OpenCV library as the function cvHaarCascades().

2.2.5 URI Developed Classification Program

The classification program works in two stages. First, an image is examined using computer vision techniques for features within the file that could indicate whether the image is illicit or non-illicit. Secondly, these features are presented to a trained Machine Learning algorithm to make a determination of the image’s class.

The computer vision techniques used to examine the features of an image are provided via the OpenCV (Open Source Computer Vision) library (Bradski, Pisarevsky and Ershimov 2011). Using this library, the image is divided into eight
2.2.6 Microsoft .NET™ Framework

In order to accomplish the stated design goals in a short period of time while maintaining the ability to produce a commercial quality application, the language C# with the Microsoft .NET Framework v3.5® will be used. The .NET Framework is the latest development library produced by Microsoft for use in the designing of Windows applications. It includes many built-in routines that can be used to create graphical user interfaces (GUIs), as well as all programming logic that is expected in an iterative programming language. It is freely available to all users of the Windows operating system, and Windows Vista and later versions are shipped with it built in (Microsoft Corporation 2010).

This should not be a constraint on the usability of the application by the maximum audience, as in a 2010 informal poll it was found that over 65% of users have the .NET Framework v3.5 installed (Hanselman 2010). In addition, most investigators will be sure to keep their computers updated at all times in order to prevent any security vulnerabilities. In the case that the .Net Framework is not installed on the system that will use this application, an application installer is included that will prompt the user to obtain it if not detected.
3.2 Graphical User Interface Tier

The first step in development of the application was to design the user interface. The interface is the only interaction users require with the application, as search options are configured, scans are started, and the results of a scan are all displayed within this single window. The GUI that is shown to the user upon application start-up is shown in Figure 2, and is designed according to conventions that users expect from a Microsoft Windows® application (Microsoft Corporation 2010).
additional checkboxes representing all logical volumes detected on the system. By providing these different search options, the investigator is able to perform on-site triage in accordance with any time or warrant limitations. In addition, other less commonly used options can be accessed via the Advanced Options button.

Figure 4: Advanced Options Dialog
used by operating systems to determine the default program for a file, even when file extensions are not used (Sammes and Jenkinson 2004). This type of search will take longer due to the need to read the first four bytes from disk, but it will identify all image files, even if they have been given an inappropriate file extension in an effort to hide the file. The file signatures that correlate to each extension are also listed in Table 1. When the application is launched, the search type defaults to file extension.

The checkbox Extract and Scan ISO files, if checked, will extract any ISO files located to a temporary directory on the local machine and scan the contents for illicit images. ISO files are uncompressed images of CD or DVD discs that are stored according to the ISO 9960 file format (ECMA International December 1987). Due to the size of these files, this option is disabled by default.

The next control in the Advanced Options dialog is the slider bar, and affects how the engine itself processes images to determine if an image is illicit or not. The two scan options (standard or with face detection) are described in the Engine section of this thesis.

The next four settings are exposed for testing purposes and will not be enabled in the commercial version of the application. *FD Max Area* and *FD Model* affect how the face detection performed in the Engine is run, to be described below. *Vid Rate* and *Vid Max* affect how video processing is performed in the engine. All values seen are the optimal values that have currently been found for these settings.

---

1 Testing of both facial detection and video scanning is under current research at the University of Rhode Island at the time of this writing. Published results on optimal values are expected to be released in the near future.
particular timeline of interest. The investigator can also select options here to expand the image scan to include video files, or files stored in a compressed archive.

The first two options can be used to expand the file search. Selecting the *Scan Video Files* option will cause the file search to include video files to be scanned by the engine. Extract and Scan Archive Files will cause the file scan to identify any compressed archive type, extract its contents to a temporary directory and then scan these extracted files for illicit content. As compressed files are used to minimize storage space, the investigator is warned upon selecting this option that a potentially significant amount of hard drive space could be required upon using this option. Both of these two options have the potential to slow down the scan, as the time required to decompress a file will be needed, and the time required to scan a video file is much longer than that to scan an image file. For this reason both of these options are deselected by default.

The final two sections of this dialog are used to filter down the search to only include files of particular interest. By selecting File Size, the investigator can cause the scan to only include files within a particular file size range, specified in KB. This could be used to exclude very small images, such as thumbnails, or exclude unnecessarily large images. By selecting a time option (created, accessed or modified), the investigator can cause the scan to only report those files created, accessed and/or modified during a particular date range. Investigators may need to use this option when establishing a time line during an investigation, or to comply with restraints imposed by a warrant.
content. The timer on the right will continue to increment until completion of the scan, at which point it will stop and display the total duration of the scan.

Below the scan timer are buttons that allow a user to pause and resume a scan, as well as cancel a scan completely. If the *Pause Scan* button is pressed during a running scan, all searching and scanning application threads are paused. The text of this button will then change to *Resume Scan*. This allows an investigator to stop the scan to determine if enough evidence has been gathered without actually canceling the scan altogether, at which time they can either resume the scan re-activating all searching and scanning threads, or cancel the scan. The scan can be cancelled at any time by pressing the *Cancel Scan* button. When pressed, a warning will be issued notifying the user that the scan will terminate and be marked completed at this point. Once the user confirms this, the scan will immediately terminate by stopping all active searching and scanning threads.

The actual results of the scan are displayed in the lower part of the window as seen in Figure 5. This display is updated in real time as images are processed. The results are displayed in a tabular format and include a thumbnail display of the image (non-illicit images intentionally displayed), the file name and path of the image, the created/modified/access time stamps of the image, and an MD5 hash of the image. The MD5 hash is included so that if investigators wish to copy the image from the suspect storage device, that copy can be verified using the hash value to have not changed in the process from the original. If videos have been selected for scanning, a separate *Videos* tab in the results window is shown that will contain video results, identical in appearance to the *Images* tab. The only difference in the result display of
engine to determine which (if any) images are illicit. Finally, these results are reported back to the GUI to display to the user. An overall view of the business logic tier can be seen in Figure 6.

![Business Logic Tier Diagram](image)

Figure 6: Business Logic Tier

### 3.3.1 Searching Thread

Upon receiving a request from the GUI for a search to begin along with the parameters specified by the user, the searching thread will create a queue for the purpose of storing the filenames (including the full path) of any images located during the search. If video files are also specified, a separate queue will be created for the purpose of storing filenames of video files. The searching thread will then spawn $p-1$ threads to perform the actual scan of these images for illicit content. If video files have been selected for scan, a single low-priority thread is created for the purpose of
- If the directory to examine is a symbolic or hard link, it is skipped so as to not examine the same directory twice.

- If file size filters have been applied by the user, the size of the current file is validated against these and the file is skipped if not within parameters.

- If file timestamps have been specified by the users, the appropriate timestamp(s) are compared and the file is skipped if not within parameters.

There is no difference to the algorithm used if the search begins at a folder or logical drive root. However, if the search is defined to start from the root of a logical drive or drives, and the logical drive has been determined to be the drive the operating system is installed on, the searching thread will do ‘smart searching’. This refers to the folder paths searched by the thread. Through feedback from law enforcement investigators, files of potential evidentiary value are often found in defined folder paths relative to Windows®. Conversely, some folder paths do not usually have image files of evidentiary value. The search algorithm is able to make use of this knowledge by searching paths likely to contain evidence first, and those not likely to contain evidence last, with all other paths in between. Using this, if a volume under examination is large, the investigator may be able to obtain evidence quickly, and then cancel the rest of the search saving valuable time. The likely and unlikely folder paths can be seen in Table 2.
responsible for the processes associated with the scanning threads. Once all scanning threads have terminated, the searching thread will notify the GUI that the scan is complete and terminate.

3.3.3 Optional Compressed File Searching

If the user has selected to search and scan for illicit files in compressed files, a small change is made to the searching process. Before the search is begun, a temporary directory will be defined using the Windows temp directory as defined by the operating system for use by applications, using the symbol %temp%. A sub directory in this temp directory will be made called %temp%/Redlight. As compressed files are encountered by the searching thread, the filenames of these files will be added to a queue for later processing. Once the search has been completed for all folders and subfolders as defined above, the searching thread will then process this compressed filename queue, decompressing the files into the temp directory. Once all files have been decompressed, the searching thread will then examine all files in the temp directory for images (and videos if defined), adding any found to the standard image and video queues for processing by the scanning thread. Any files determined not to be images or videos are deleted in order to minimize any storage space used by temporary directory.
scan result has been communicated, or if a file name is not found, the searching thread will be asked if the search is complete. If it is not, the process will be repeated until both the queue is empty and the search is complete. If both of these conditions have been met, the scanning thread will terminate.

3.3.5 Optional Video Scanning

If a search for illicit videos has also been specified, a single low-priority thread is launched to scan the video queue using the same state diagram seen in Figure 8 for images. If any image scanning threads complete before the video queue is empty, these threads will be switched to scanning the video queue. Once all image threads have completed, if the video queue is still not empty, the original video scanning thread will be changed to normal priority and all active scanning threads will continue processing the video queue until complete.

3.4 Engine Tier

```c
bool initScanner(const char* modelFileBase, const char* faceModelName);
int TestFace(const char* inputFilename, int videoSize, int videoWidth, void* userDatas, bool scanFaces, double minFactorXs, double factorYs);
int testPortrait(const char* imageFile, bool scanFaces, double minFactorXs, double factorYs);
```

Figure 9: Engine Functions

The third and lowest tier of the application is that of the engine, which is driven by the URI developed program as described in Chapter 2. The program itself is wrapped in a C++ Windows® DLL file (Microsoft Corporation 2007), which can be loaded into memory once as a function, but accessed independently by multiple
full path of the video file must be set. As the video scanner functions by processing
the video as a series of images, the scanning rate and maximum number of frames
must also be specified. The scanning rate determines how many frames in the video
should be skipped between each sampled frame. The maximum number of frames
specifies how many frames should be scanned as non-illicit before the video is
declared non-illicit as a whole. While current research has determined the optimum
values for these parameters to be 50 and 25 respectively, the values can be set
dynamically to facilitate further testing. The fourth parameter userData must be set to
NULL as it is currently not used. The function also allows for input of face detection
for further accuracy as in the function for image testing.

Of note as to how the engine processes images and videos for content is that
both a normal scan and a scan that includes testing for facial area are not mutually
exclusive. If facial area is to be tested, a standard scan is performed first, and then
only those images or video frames determined to be illicit are tested for facial area.
This means that running a scan using the standard parameters, and then running a
second scan including facial area will not produce different results than just running a
single test including facial area.
passed to the searching thread. Once the parameters have been received and the searching thread begins, an event notification will be sent to the GUI every time a directory search has begun so that the GUI can update the current directory being searched to the user. While the search is being performed, located image files are placed into the image queue so that they can be scanned by the scanning thread. Once an image is found, an event notification is also passed to the GUI so that the number of images found can be updated in the result display.

When the scanning thread de-queues an image from the images queue, it is then passed to the engine to determine if the image is illicit or non-illicit. Once the result is sent back to the scanning thread, the result of the scan is passed to the main GUI, along with the file name and path of the file. Once received by the GUI, the number of files scanned is updated in the result display. If the result is positive, the result display will also generate and display a thumbnail image in the result list, along with the path and file name, timestamps of the file as reported by the file system, and a calculated MD5 hash of the file. These operations are performed in the GUI as it minimizes the amount of data passed between the tiers. This also has very little performance impact because the file itself is cached into memory by the operating system when examined by the engine, so there is no disk I/O required. When the search is complete and all scanning threads have terminated, an event notification is sent to the GUI, and the user is notified that the scan is complete.
In the practice of digital forensics, there are three tools that are used by a majority of investigators:

- EnCase® by Guidance Software (Guidance Software, Inc. 2011)
- FTK by Access Data (Access Data Group, LLC 2011)
- X-Ways Forensics by X-Ways (X-Ways Software Technology 2011)
presented in court or to supervisors, the user can choose to filter what type of information is included as seen in the lower section of Figure 11.

The format of the report was developed in conjunction with the URI digital forensics research group, and is displayed based on formatting instructions contained in a style sheet that is saved alongside the report. The top of the generated HTML report displays the case name, number and investigator information, the time the scan was run and its duration, and the number of images found, scanned and identified as illicit. The program logo is also displayed so that the application used to generate this report is plainly evident.

All information chosen will be included in the HTML report in a tabular format, similar to that seen in the results display. If the user chooses to not include images, only the text information for each file will be included. If the user selects to use thumbnail or complete images, an images folder will be generated alongside the HTML report containing either thumbnails of each image or an exact copy of each image. In the report itself, standard HTML img tags are produced which refer to the image in the generated images directory, displaying thumbnails in the report in a manner similar to those in the result display. The only difference between the two options is the size of the image stored in the images directory. A sample of an HTML report is included in Appendix C. Please note that this report has been intentionally generated without images.
<table>
<thead>
<tr>
<th>Component</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Asus G74S</td>
</tr>
<tr>
<td>Operating System</td>
<td>Windows 7 Enterprise</td>
</tr>
<tr>
<td>Processor</td>
<td>Intel CORE™ i7 2.00 GHz</td>
</tr>
<tr>
<td>Logical Processing Cores</td>
<td>8</td>
</tr>
<tr>
<td>RAM</td>
<td>8GB DDR1333MHz SDRAM</td>
</tr>
<tr>
<td>Disk Storage</td>
<td>2x 500GB 7200rpm</td>
</tr>
</tbody>
</table>

Table 4: Test System Specifications

<table>
<thead>
<tr>
<th>Number of Scanning Threads</th>
<th>Number of Processing Cores</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>605</td>
</tr>
<tr>
<td>2</td>
<td>615</td>
</tr>
<tr>
<td>3</td>
<td>700</td>
</tr>
<tr>
<td>4</td>
<td>626</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Runtime Test Results (Seconds)
swapped without requiring significant contention. If a video thread is used, it is run at low priority so that contention is minimized on that thread.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Core</td>
<td>38.72</td>
</tr>
<tr>
<td>2 Cores</td>
<td>39.12</td>
</tr>
<tr>
<td>4 Cores</td>
<td>57.98</td>
</tr>
<tr>
<td>8 Cores</td>
<td>63.14</td>
</tr>
</tbody>
</table>

Table 6: Runtime (images/second)

Assuming that \( p-1 \) scanning threads are used to search and scan 23,425 image files, the runtime in images per second can be seen in Table 6. This demonstrates that regardless of the number of cores, the application can process at least 38 images per second. This also shows that the more the application is able to be multi-threaded, the faster it can run.

4.2 Criteria: The tool should have a low memory footprint, enabling it to be run on lower end machines

A test of memory usage was done using the Windows® Performance monitor application, perfmon.exe. The test was set to monitor the working set (Russinovich, Solomon and Ionescu 2009), or total memory used by the application and all shared dll’s called by the application. The test was originally done using the configurations seen in Table 5, but it was found that the memory usage of the application does not vary based on the test configuration. Instead, the memory used is entirely dependent
determining that images are illicit or non-illicit. To determine accuracy of images found, the application was run against the test data set looking for images by file name and by file header. The results of these tests can be seen in Table 8. These results show that all expected images were found, and therefore the accuracy of searching is 100%.

<table>
<thead>
<tr>
<th>Scan Type</th>
<th>Scan Results</th>
<th>Application Accuracy</th>
<th>URI Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Images Classified Illicit (No Face Detection)</td>
<td>10,405 of 12,180</td>
<td>85.4%</td>
<td>85.4%</td>
</tr>
<tr>
<td>Images Classified Non-Illicit (No Face Detection)</td>
<td>4,925 of 11,505</td>
<td>57.2%</td>
<td>57.2%</td>
</tr>
<tr>
<td>Images Classified Illicit (With Face Detection)</td>
<td>9,505 of 12,180</td>
<td>78.0%</td>
<td>78.0%</td>
</tr>
<tr>
<td>Images Classified Non-Illicit (With Face Detection)</td>
<td>10,280 of 11,505</td>
<td>89.4%</td>
<td>89.4%</td>
</tr>
</tbody>
</table>

Table 9: Classification Results

The accuracy of the number of illicit and non-illicit image classifications is entirely dependent on the results produced by the URI developed tool in the engine tier of the application. Thus for the purposes of this application, the accuracy of classification need only match that tested by the URI research team that developed the tool. The application was run against the entire dataset of images and videos, and the results can be seen in Table 9. Based on reports from the research team, the expected
In addition to the Beta feedback mentioned above, the software has been evaluated by the National Institute of Justice (NIJ) Electronic Crime Technology Center of Excellence (ECTOE). In this report, the ECTOE performed testing on all features of the software, including its usability and applicability to law enforcement investigations. In the conclusion of this report, it is stated that “RedLight would be a great asset to the investigative tool box ... [The application] could greatly expedite cases where an examiner is looking for pornography.” (NIJ Criminal Justice Electronic Crime Technology Center of Excellence January 2011). Based on the ongoing feedback mentioned above as well as this report, it can be concluded that this criteria has been met.
The application is extensible for future development to increase its speed and accuracy. By allowing for the dynamic entry of the model files and cascades used by the machine learning portion of the engine, new models and cascades can be developed and easily integrated to the application with little or no change. By having multiple layers in the application, any one layer can be changed independently of the others, minimizing the potential of breaking the entire application by modifying one tier.

The application that has resulted from this thesis is able to scan a minimum of 38 images per second with 85% accuracy. The application requires less than 30MB of memory in order to start, and can display over 27,000 positive results on a 32-bit machine. Once a scan is complete, a report can be generated for import into other tools used during a forensic investigation or as a professional HTML report. This application is available for any digital forensics investigator to use to make their investigation of a subject computer significantly more efficient.
APPENDIX B:

SAMPLE CSV FILE OUTPUT REPORT

Information File

| Case Name: | Thesis |
| Case Number: | Sean Alvarez |
| Investigator: | Sean Alvarez |
| Start Path: | C:\Users\Sean\Desktop\Thesis\data |
| Subfolders: | True |
| Scan Start Time: | 3/12/2012 5:42:46 PM |
| Scan End Time: | 3/12/2012 5:50:00 PM |
| Total Scan Time: | 7 Minutes 14 Seconds |
| Found: | 23685 |
| Scanned: | 23685 |
| Suspect: | 16985 |

CSV File

<table>
<thead>
<tr>
<th>File Name, File Path, Created Time Local (UTC-05:00:00), Created Time UTC, Accessed Time Local (UTC-05:00:00), Accessed Time UTC, MD5 Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>020009.jpg, C:\Users\Sean\Desktop\Thesis\data\Illicit\Kriss Test</td>
</tr>
<tr>
<td>PM, 66EB7DCA3443CED4E7B701A447D9D25</td>
</tr>
<tr>
<td>020011.jpg, C:\Users\Sean\Desktop\Thesis\data\Illicit\Kriss Test</td>
</tr>
<tr>
<td>PM, 3E3A6D00DE35176364348EE5A71E2C5</td>
</tr>
<tr>
<td>020024.jpg, C:\Users\Sean\Desktop\Thesis\data\Illicit\Kriss Test</td>
</tr>
<tr>
<td>PM, 84B6ED7CFC3C73422C6D025F2E024C9</td>
</tr>
<tr>
<td>020031.jpg, C:\Users\Sean\Desktop\Thesis\data\Illicit\Kriss Test</td>
</tr>
<tr>
<td>PM, 170B87EE37BE033B844AFF576B0FEF0</td>
</tr>
<tr>
<td>020039.jpg, C:\Users\Sean\Desktop\Thesis\data\Illicit\Kriss Test</td>
</tr>
<tr>
<td>PM, 72CF79E8E94CD25A63A67047083A34</td>
</tr>
<tr>
<td>020043.jpg, C:\Users\Sean\Desktop\Thesis\data\Illicit\Kriss Test</td>
</tr>
<tr>
<td>PM, 2AB41003FEE249987F453B630301F3F7</td>
</tr>
<tr>
<td>020055.jpg, C:\Users\Sean\Desktop\Thesis\data\Illicit\Kriss Test</td>
</tr>
<tr>
<td>PM, 5B3B54D71F9DE0698B4E0D4E6483</td>
</tr>
<tr>
<td>020056.jpg, C:\Users\Sean\Desktop\Thesis\data\Illicit\Kriss Test</td>
</tr>
<tr>
<td>PM, D8563D6795044022B5F537007371E12</td>
</tr>
</tbody>
</table>
BIBLIOGRAPHY


Hanselman, Scott. *How many PCs in the world have the .NET Framework installed?*. January 19, 2010.


