2008

Crawler 2.0: A search tool to assist law enforcement with investigations

Daniel Joseph Harkness
Iowa State University

Crawler 2.0: A search tool to assist law enforcement with investigations

by

Daniel Joseph Harkness

A thesis submitted to the graduate faculty

In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Co-Majors: Computer Engineering; Information Assurance

Program of Study Committee:

Doug W. Jacobson, Major Professor

Thomas Earl Daniels

Patricia A. Thiel

Iowa State University

Ames, Iowa

2008

DEDICATION

This thesis is dedicated first to my loving wife, Beth. Without your support, encouragement, and patience I never could have completed this. We have embarked on a journey of life together and I would not want anyone else as a companion on that journey. Let this usher us into the next stage of our journey.

To all of my friends and family, this thesis is also dedicated to you. Your encouragement and support has been a blessing throughout this process. I could not ask for a better family or better friends.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Over the past few years, the internet has been evolving rapidly and a new paradigm in web development has taken shape. Often referred to as Web 2.0, it is a shift in web development which focuses on sharing information and allowing user interaction. The sharing of information by users has resulted in a new location for law enforcement to discover evidence. However, the process of locating this evidence is often a tedious one.

Crawler 2.0 is a tool with law enforcement's needs in mind. It is a web crawler and parser with Web 2.0 technology in mind. Given a Web 2.0 page as a starting point, it will interpret known content types and provide a basis for keyword searches. Crawler 2.0 is intended to be expandable for the addition of new, updated, or custom sites and technologies.

CHAPTER 1. INTRODUCTION

As the number of internet users continues to grow rapidly, so does the technology that is used to provide content. One of the newest trends in web development is Web 2.0. While definitions of Web 2.0 vary, the general concept of it is fairly well agreed upon. Web 2.0 refers to using the Internet as multi-part platform. Often it means taking pieces from multiple sources on the fly to create a better product. Additionally, it often embraces the idea of user participation. These dynamic and user-oriented principles are part of what is driving much of the interest in internet use today.

However, as Web 2.0 continues to grow it creates new risks and challenges. The sharing of content and code leaves portions of webpages outside of the designer's control. Malicious code on a seemingly benign site may install a virus on the users' machine, or enroll it in a botnet [1]. In addition to the risks brought about by the code itself, there are also mental and physical risks to users caused by the psychological aspects of Web 2.0. The very nature of Web 2.0 is to involve the users and help allow people to connect and share. This can come in the form of sharing facts about people, places, events, etc. However, it also comes in the form of sharing personal and private information. Both of these can open users to additional risk. Because of all of the information and "facts" being posted by other users, it is very easy for a user to obtain inaccurate (or even intentionally falsified) information on which to base a decision. The risks of sharing personal data can range from humiliation to job loss to personal harm [2][3][4][5][6][7].

From a law enforcement standpoint, Web 2.0 also creates new challenges. It creates new ways that computers can relate to crime and adds new sources of digital evidence. Since computer crime is virtually guaranteed to occur, it is important for law enforcement to stay on top of new technology, such as Web 2.0 and to be aware of both the risks that it presents and the benefits that it provides. One major need of law

enforcement is to recognize and locate evidence in the user-centric environment of Web 2.0.

Unfortunately, there has been very little research on Web 2.0 in the context of law enforcement. Most Web 2.0 research has focused on feature and usability enhancements for Web 2.0 (which will not be discussed here) or on security and privacy risks in Web 2.0. The little research that I have found which may be applicable (but doesn't appear to have been done with law enforcement in mind) involves Web-based Inference Detection [8] and specialized web crawlers [9][10][11].

I propose a tool, Crawler 2.0, to assist law enforcement with data gathering in Web 2.0. With all of the user-centered Web 2.0 sites popping up, there is a lot of potential information to be found about suspects and/or victims. Crawler 2.0 is a portable web crawler which investigators can use to gather and search data from Web 2.0 sites. It is envisioned to focus on Web 2.0 sites and regular sites which are one link away from a Web 2.0 site.

CHAPTER 2. THE PROBLEM

As mentioned previously, the rapid emergence of Web 2.0 sites and technology has resulted in new security and privacy risks. These new dangers have already begun to show up in the news. This has resulted in a need for law enforcement to use Web 2.0 within their investigations. Unfortunately, while some providers have begun to cooperate with law enforcement, there have been few (if any) tools to assist them. Further complicating the situation is that Web 2.0 creates a new and unique environment for the investigation of most common crimes.

2.1. Notable Cases Involving Web 2.0

All of the user-centric design in Web 2.0 is leading to a rapidly growing user base which posts information to the web. This information may be in the form of facts and data, beliefs and opinions, or even personal experiences. Unfortunately, a lot of users fail to recognize the potential impact of the information that they share or of the trust that they put in the information of others.

There have been many cases where sexual predators have used information found online about teenagers to lure them, trick them, or attack them. In one case, the information used was a work address posted by a 16-year old girl [2]. In many cases they use false information about themselves to gain the trust and friendship of their victims [2][3].

In another case, the presence of a woman's photo on a man's social networking profile resulted in her being targeted for murder. Upon seeing the woman's photo on the profile, the man's girlfriend attempted to hire someone to kill her. The person she attempted to hire turned out to be a police detective, however [4].

In another case, inappropriate use of MySpace contributed to the suicide of a teenage girl. In October, 2006, Megan Meier hung herself after being told on MySpace that the world would be better off without her. The message was sent by someone

pretending to be a 16-year old boy to get information about Megan's relationship with another teenage girl. The account had been created by the other girls mother, but was used by multiple people [5][6].

In Novato, California, two teenage boys were arrested after an incriminating video of them was found on MySpace. The video showed the boys throwing homemade firebombs at an abandoned airplane hangar. The Novato police regularly search MySpace for evidence of criminal behavior in the area [7].

Not only are users put at risk by information they post themselves and false information that they choose to trust, but by information posted about them by others. Because many of the social networking sites involve the concept of "friends," other users may post information on a user's profile, or they may post pictures or other information about the user. Because of the linking between friends' profiles, others may be able to obtain personal information about a user from their friends' pages.

2.2. Web 2.0 Providers' Assistance

In January 2008, MySpace and the Attorney Generals of almost every state in the U.S. announced that they had reached an agreement in efforts to protect users of social-networking sites, children in particular. The agreement focused primarily on proactive measures to include identification and removal of known sex offenders, automatically making underage user profiles private, and improving age verification procedures [12][13]. In May 2008, Facebook came to similar agreement and joined the task force with MySpace and the Attorney Generals [14].

Unfortunately, these announcements have fallen short of assisting law enforcement with general investigation. MySpace did promise in the agreement to respond to complaints about inappropriate content within 24 hours [12]. However, this requires first finding the inappropriate content and also seems to continue to focus primarily on issues of sexual content. This is only one criminal area which may require

investigation in Web 2.0. Evidence of all types of crimes can be found on Web 2.0 pages. The previously cited cases are examples of this.

2.3. Law Enforcement's Needs

The security and privacy risks in Web 2.0 are generating a need for law enforcement to prepare to deal with Web 2.0 in investigations, as can be seen in the multiple examples above. In 1995, David Carter introduced four categories of computer crime [15]. The first category is where computers are the target (i.e. theft of computers or files, vandalism of web sites, etc.). Next is where computers are an instrument for crime (i.e. hacking, network scanning, etc.). His third category was computers being incidental to other crimes (i.e. crimes which can be committed without computers, such as money laundering, but where computers are used to make it easier). Finally, there are crimes due to computer prevalence (such as software piracy).

Web 2.0 will produce new vectors for computer crime. Cases of stalking through personal information posted on social networking sites [2][3] provide an example of new ways that computers are being used in an incidental manner. Security risks caused by loss of control over code in Web 2.0 will certainly add to the evolution of crime as well [1]. For example, specialized Web 2.0 worms could be considered new crime caused by the prevalence of the new technology. However worms and malicious code are technical crimes which are not that different from current computer crimes except in the manner that they are spread or executed. The major frontier for Web 2.0 and law enforcement to collide is in the privacy issues. The user-centric design of Web 2.0 creates a new place for digital evidence to be found.

The evidence of most common crime investigated by law enforcement has traditionally been found in physical form (written documents, fingerprints, etc.) or a digital form which can be isolated and preserved prior to the investigation (hard disks, cds, etc.). However, investigators will now need to look towards the dynamic content in

Web 2.0 for evidence as well. Not only is the evidence not guaranteed to be static, but typically the actual hard drives where the information is being kept will be outside of the reach of investigators. This means investigators will need new processes and technologies to locate and capture relevant information from Web 2.0 content. Crawler 2.0 is an exploratory tool developed to prompt research and development targeted at solving these issues.

2.4. Data Quantity

Adding to the problem for law enforcement is the sheer quantity of information that they need to explore for potential evidence. I will illustrate this with an example. Assume that the average MySpace user has 25 friends, 20 photos (10 of which have comments on them), and 0 videos. To process one user will involve looking at 13 web pages: The profile page, the "view all friends" page, the "view all pictures" page, and the 10 individual picture pages. To look at the comments that the user may have made on their friends pages will involve review of 338 web pages (13 for the user's profile plus 13 for each of their 25 friends).

CHAPTER 3. RELATED WORK

Before beginning any development project, it is important to look for related work. It may be that a solution already exists, or it may be that parts of the solution already exist. Unfortunately, I have been able to find little to solve law enforcement's Web 2.0 investigative needs. The related work that I have found is presented here.

3.1. Web-based Inference Detection

Chow, et al. introduced the idea of web-based inference detection at the Web 2.0 Security & Privacy conference in May, 2007 [8]. In their presentation, they gave an example of how the web could be used to test inferences of the form (set of terms A) IMPLIES (set of terms B). As an example they deduce the inference {sibling Saudi magnate} → {Osama Bin Laden}. They present two methods for testing these inferences, both of which use search engines.

The first method is to compare two searches side-by-side and see how many of the results appear in both searches. The first search is of the form *set of terms A* (sibling Saudi magnate). The second search is of the form *set of terms A set of terms B* (sibling Saudi magnate Osama Bin Laden). If the number of entries appearing in both is high (taking into account the number of entries relating to just *set of terms A*), then it gives strong support to the inference.

Unfortunately the second search in the above method may have results in a different order, so a side-by-side comparison may be difficult (especially if the search results in a large number of hits). So a second method is presented. Instead of comparing the hits side-by-side, the number of hits can be used. Since the second search (of the form *set of terms A set of terms B*) should be a refinement of the first search, taking the number of hits in the second search and dividing by the number of hits in the first search yields a probability. This probability is the confidence of the inference rule.

This use of data from Web 2.0 content could be useful to investigators looking for connections between individuals or between individuals and events. However, the investigator will need some data to use as a starting point to determine the potential inferences. Additionally, due to the large volume of data on the Internet, this process will probably only work for making connection between very unique or specialized terms or for making connections among well documented information.

3.2. Shark-search

In 1998, the Shark-search algorithm was introduced. Shark-search is an algorithm for web crawling which is focused on finding relevant information near a starting point [9]. It works on the principle that relevant data is usually located near other relevant data. Using this principle, it performs a smart search which focuses on searching in areas likely to contain results.

To determine the likelihood of success it scores a page based on its relevance to the query. It then applies this score and a decay factor to the links found on this page. The list of URLs to visit is maintained as a priority queue so that the links most likely to contain relevant information get visited first. It also makes use of the anchor text to determine relevance.

The purpose of this focused search is to help find relevant data fast. Unfortunately, I believe that in the case of law enforcement, finding all relevant data is more important. First, it is important because it is good practice to find both inculpatory and exculpatory evidence. Second, if the investigator does not know where at least some relevant data is located it may be difficult to point the algorithm in the right direction.

3.3. WebSPHINX

WebSPHINX is a Java toolkit developed at Carnegie Mellon University developed between 1998 and 2002 [10]. It is a modification/recreation of SPHINX

which was developed in the summer of 1997 at the Compaq System Research Center by Robert Miller and Krishna Bharat [11].

WebSPHINX consists of two parts, an application which allows a user to build a web crawler through a graphical environment, and a set of Java libraries. The Java libraries were of interest to me as a basis for creating a web crawler for Web 2.0 content. They allow for the creation of a crawler as an extended class implementing two functions. The first function takes a link and determines whether or not the crawler should visit the link. The second function processes a page.

Unfortunately, in order to recognize Web 2.0 technologies as compared to Web 2.0 sites (such as phpBB compared to MySpace) more information than the link itself may be needed to decide whether or not it is applicable to the specific crawler. Also, I believe the best approach is an exhaustive crawl (at least to a certain depth) so the crawler should visit all links even if it can't parse the information (this is explained further in section 4.4). WebSPHINX is also limited in its ability to perform form-based authentication which may be needed for some Web 2.0 content.

3.4. Law Enforcement Use of Web 2.0

Law enforcement itself has been researching ways to use Web 2.0. However most of the research appears to be focusing on how they can use it for collaboration and information dissemination rather than how they can use it to gather evidence.

3.4.1. Collaboration

In September 2007, Tom Looney wrote an article for Public Safety IT Magazine about the potential for the use of Web 2.0 technology in public safety and homeland security agencies [16]. In the article, he discusses how Web 2.0 can be used to meet some of the major collaboration needs of the agencies. Agencies can use the technology to share information while restricting it from access to the public.

He gives a theoretical scenario of using Web 2.0 for a gang task force. A patrol officer can use online maps to mark areas of gang influence, while an investigator can use a wiki to record information about gang members and activities. The two can be linked together, and other officers can use the information to determine where the greatest threats are and increase the police presence in those areas. Officers can note their findings in blogs, which can be updated on their homepages and available through RSS feeds to other team members and administrators, eliminating the need for maintaining special email lists.

He also gives a real-life implementation being used in Alabama. The Law Enforcement Tactical System (LETS) brings together systems from multiple agencies in the state (including motor vehicles, courts, and correctional facilities). It contains over 21 million records and has been in use since January 2003. The solution was developed in seven months by programmers from two universities and provides services to about 4,500 users. In one portal, it brings together data on 17 million registered vehicles, 4 million drivers, outstanding arrest warrants for 500,000 people, records of 25,000 inmates, and 5,000 abuse-related court orders.

3.4.2. Interaction with Public

In London, the Greater Manchester Police has begun to use Facebook as a tool to interact with and disseminate information to the public [17]. They use a Facebook application which has 452 monthly active users (as of 8:29 PM CST, November 10, 2008), which is down from the 750 users reported in the article. The application is used to post information about wanted persons, traffic news, and other important police information to the users. According to the article it also features a link to anonymously submit tips to the police department.

CHAPTER 4. SOLUTION: CRAWLER 2.0

As mentioned above, law enforcement investigators are in need of a tool which can help them gather relevant data from Web 2.0 content. Investigators are already using Web 2.0 content in investigations. However, as of yet the investigations involving Web 2.0 have been done manually (to the best of my knowledge). Crawler 2.0 is a tool which can help change this.

4.1. Description

In its simplest form, Crawler 2.0 is a web crawler. However, it is unique in that it is geared specifically towards user-centric Web 2.0 content, and has been built with law enforcement in mind. It doesn't only crawl the web, but it parses the information that it encounters into a common, meaningful data format. It then allows searching of the information in both the parsed and raw form.

4.2. Web 2.0 Parsing

With so much information in Web 2.0, it is not unlikely that information useful to investigators may be present. However, there are many different ways to present the information. This can result in a lot of wasted time searching through pages just to locate the data being sought. One of the things that Crawler 2.0 contains is a set of data structures that is used to hold data from a variety of sources in a common, comparable format. Crawler 2.0 can take a supported Web 2.0 page and parse its formatting for data which is then placed into the common data structures. The information can then be extracted from these data structures and presented in the same format as other data of the same type, regardless of the format of the original source.

4.3. Profile Searching

There is a lot of personal information to be found in social networking profiles of Web 2.0. Especially with the current generations, many people will talk about everything going on in their life online. Many people even keep online diaries and make much of

their communication public through postings on their own or friends' profiles. This means that there is a lot of information available on Web 2.0 profiles which may be of use in a law enforcement investigation. Furthermore, posts on public profiles have seemingly become accepted as public communication, and are subject to investigation by law enforcement [18]. Crawler 2.0 will allow an investigator to obtain the public content of a number of linked profiles, and then to search them for relative keywords. When a keyword is found, the program can return the entire data item which contained the keyword. A single data item refers to a single post, a single comment, a single photo caption, etc.

4.4. Hidden Information Extraction

During the writing of Crawler 2.0, I realized that users could use the html base of Web 2.0 for information hiding pretty easily. When viewing a web page, generally only text, images, web applications, etc. are seen. The tags (and their attributes) used to separate and format the content and HTML comments, are generally hidden from the user (at least hidden from what is seen when the page is viewed in a browser). Because of this, a user could use html tags to hide data from anybody viewing their profile that is not aware it is there. There are three ways a user can hide the information using HTML.

- Valid formatting tags can store hidden information in the attributes. (Ex.).
- Invalid tags can be used for storing hidden information. (Ex. <this tag is really hidden info></this>).
- HTML comments can be used to hide information. (Ex. <!-- This comment is being used for hidden info -->).

Which methods are available to a user may depend on what structures the Web 2.0 application allows a user to use. MySpace, for example, allowed the first two methods in user-added content when I first began writing Crawler 2.0, and now does not

allow any of the methods. Users could use these methods to pass messages to their associates about criminal matters, such as the time and location of a drug deal, for example.

These methods work because most common web browsers will just ignore malformed and unknown tags and attributes, as well as HTML comments. Crawler 2.0 could be used to extract this hidden information. Since the program operates on the raw html returned by the URL, it can provide a plain text search, which treats the entire webpage as a single string regardless of where keywords are found. This search can also be used to locate the information on pages which Crawler 2.0 retrieves, but which it does not know how to parse.

4.5. Crawling Links

Since Web 2.0 is so geared towards the sharing of information, linking sources together, and social networking, information tends to spread out like a web. To link relevant information together it is helpful to take a systematic approach, not available in most search engines. Crawler 2.0 uses a fixed set of starting points, and only gathering information from linked pages, pages linked to those, and so on. This results in a trail which can explain how the information that is found is linked back to the original site. By following this crawling pattern, Crawler 2.0 is able to avoid a lot of useless hits which have no relation to the subject of the investigation. This would be especially useful when searching a forum, or looking for information about a user not only on the user's profile but also on their friends' profiles.

The drawback is that the number of links to follow can grow rapidly as the depth increases. For this reason Crawler 2.0 has a user-specified maximum crawl depth. When links are parsed, they are assigned a depth. Before the page pointed to by the link is retrieved, the depth is checked.

CHAPTER 5. CRAWLER 2.0 DESIGN

Because Crawler 2.0 is meant to operate in Web 2.0, it needs to be ready for changes at any time. Also, because it is designed for use by those who do not necessarily have an inside knowledge of computer programming, it must have an intuitive control panel and display of the results. Additionally, it needs to be portable to run on any computer. Because Crawler 2.0 is for use by law enforcement, the work that it performs must be recordable and repeatable, and the information it gathers must be from public sources. These requirements were the basis for choosing the design approach for Crawler 2.0.

5.1. Modular Design

To satisfy the need for readiness for change, I chose a semi-modular design for Crawler 2.0. The crawling, data management, and interface are all in one application. The parsing instructions can be provided in two different manners: CSL (Crawler Specification Language) modules and source-code extension. CSL modules (which will be described in Chapter 9) provide a method to add very basic parsing functionality to Crawler 2.0 without an understanding of computer programming. A Crawler class exists which may be extended by those with programming experience to provide more intelligent parsing. Either method of adding parsing instructions is intended to add a single Web 2.0 category to Crawler 2.0 at a time. A category refers to either a commonly used Web 2.0 technology (phpBB, MediaWiki, and WordPress are just a few examples) or a common Web 2.0 site (MySpace, Facebook, and Bebo are just a few examples). Some examples of modules and an overview of the design concept can be seen in Figure 1.

The modular design also helps with satisfying the second design requirement for Crawler 2.0. By making a single application file with independent modules, it allows a

user to handle multiple types of Web 2.0 content without needing to find separate programs or handle each piece of content manually.

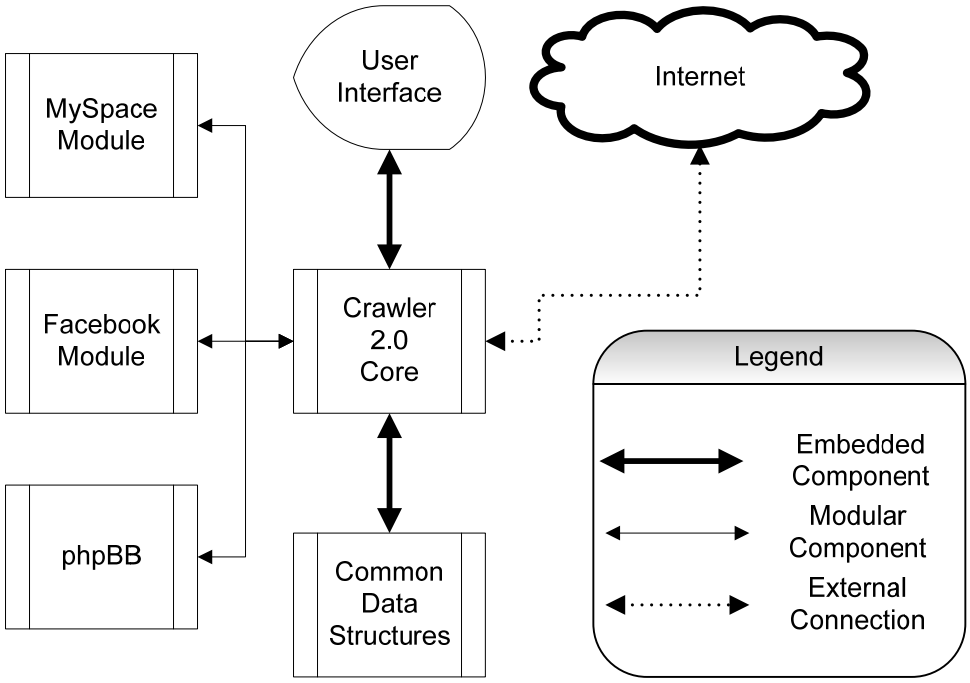

Figure 1. Overview of Crawler 2.0 Structure

5.2. Portability

To satisfy the portability requirement, a universal language will need to be used. In this case, universal means that it can run on Windows, *nix, and Mac OS X platforms without needing different code versions. Using a universal language will allow Crawler 2.0 to be used by an investigator regardless of what computer they are on.

The portability requirement also means that Crawler 2.0 must be able to do its work on the fly, without needing to access central databases or storage mediums. For this reason, while Crawler 2.0 will be able to output results to a file, it will not need to use files for any input (other than the modules which are optional and will be stored with the program file). This means that all information about formatting and structure will need to be stored in the modules themselves.

5.3. Accountability

To satisfy the needs of court, it is important to keep as complete a record as possible of the evidence being gathered. Because of these needs, Crawler 2.0 needs to keep a full account of its actions. It also needs to ensure that it is only accessing publicly available information.

To keep a full record, an entry is kept in data storage for every page downloaded. The entry records a timestamp indicating when the page was downloaded, the URL of the page, and the identifier for the data entry of the page where a link to this page was found. The URLs that are provided as the starting URLs for the crawl receive a default id and have a user provided flag set. Furthermore, the results of the request are hashed and stored locally with the hash as the filename. The hash algorithm used is SHA-1, which is a widely used hash in the computer forensics field. The information stored in the data item allows for a record of when and where the information was retrieved, while the cached copy of the page allows for the page to be reproduced even if it is no longer available on the Internet or if the Internet copy has been changed.

To ensure that information being accessed does not exceed that which the investigator could access through the internet, Crawler 2.0 acts as if it were a user. It can only follow links present on the webpage, and it will not attempt to bypass any security measures. It does however allow for login credentials to be provided for certain sites if the writer of the parsing instructions includes authentication information. Furthermore, by default it will obey the robots.txt files provided by server administrators. However, as this may not be necessary since Crawler 2.0 is acting under the control of the investigator, they may turn it off.

CHAPTER 6. IMPLEMENTATION MEETS DESIGN

The implementation of Crawler 2.0 will be done in Java to satisfy the portability requirement. Java was chosen as the language for implementation because it is universal however it is still very versatile and powerful. Additionally, it has the advantage of containing support for GUI development so that a user-friendly interface for Crawler 2.0 can be developed in the same language as the rest of the system. Finally, it is a popular, well documented, and heavily supported language. This should provide some benefit for the requirement of change readiness.

6.1. HTTP Connection Handling

Since Crawler 2.0 will be accessing Web 2.0 content through the same manner as a user, this means it will connect to servers via the HTTP and HTTPS protocol. The Apache Software Foundation has put together a Java library package for client-side HTTP communications called HttpClient [19]. The libraries are licensed under the Apache License, Version 2.0. I chose to use these libraries to handle the connection to servers and the retrieval of the Web 2.0 content.

6.2. URL Normalization

In order to avoid retrieving the same web pages over and over, Crawler 2.0 implements some URL normalization and a user-defined expiration time period. URL normalization is based on the IETF RFC 3986 for URI Syntax [20]. Only transformations guaranteed to refer to the same document are used. There are additional transformations proposed by other sources [21], however they may vary from web server to web server or from one web application to another. In order to reduce the chance that Crawler 2.0 misses evidence, these non-guaranteed normalization methods are not used.

6.3. HTML Handling

Having chosen to use Java, I decided to use an HTML library for parsing rather than writing the functions from scratch. I chose to use HTML Parser version 1.6. This is

a set of Java libraries for parsing HTML documents. The libraries are written by D. Oswald, S. Raha, I. Macfarlane, and D. Walters and distributed through SourceForge.net [22]. The libraries have been published under the GNU Lesser General Public License.

6.4. Interface Design

The Java Swing libraries are designed for creating graphical user interfaces. I chose Swing because it allows me to easily create a form-based interface. Swing contains ready-to-use components including file choosers and tree structures. What Swing does not do well is work in the background. In order to provide feedback to the user while a crawl is being performed, and in order to allow the user to interact with the application (to stop, resume, or save the crawl), I needed to perform some of the functions in the background. Foxtrot 3.0 by Simone Bordet is a Java library aimed at running long or computationally heavy tasks in the background within a Swing-based application [23]. It is licensed under the BSD license, and was used in Crawler 2.0 for opening additional windows (to view the results for example) and while performing a crawl to keep the user interface responsive.

6.5. Searching

Since one of the major goals of Crawler 2.0 is to provide law enforcement with useful searching capabilities for Web 2.0 content, it wouldn't be complete without a search feature. Before implementing a search feature, I considered ways to improve the results of the searches. Stemming was a topic that was mentioned in both the parsing section of a book chapter about web crawling and in the future work section of one of the first articles about Google [24] [25]. Furthermore, it is a technology that has since been implemented by Google [26]. Stemming is the conversion of words to their root forms before comparison, and allows for a search for "vandalism" to also find "vandal" or "vandalized" which would not have been found using a substring search. However, the usefulness of stemming in the English language has been debated [26], so I decided to

add it to Crawler 2.0, but make its use optional. In many instances, the Porter stemming algorithm was mentioned and it appears to have many existing implementations [24] [26]. One implementation that I found for this algorithm was in the libstemmer Java library which is publicly available under a BSD license as part of the Snowball project [27]. I chose to use this implementation in Crawler 2.0.

Another concept that I came across which seemed useful for searching was the ignoring of stop words (also known as stoplisting) [24]. Stop words are common words, which provide little context to a search and may result in many irrelevant search hits. Again, however, I give the user the option of whether or not to ignore them. I chose to use the list of stop words which Ranks.nl believes to be the set of English stop words used by Google [28].

CHAPTER 7. IMPLEMENTATION: PACKAGE LAYOUT

The implementation of the Crawler 2.0 application was broken down into six Java packages, based on the purpose of the code within each package. The primary package, crawler2 contains only the Main class. The other packages are crawler2.gui, crawler2.core, crawler2.crawlers, crawler2.data, and crawler2.utils. The packages will be explained here. The package layout can be seen in Figure 2.

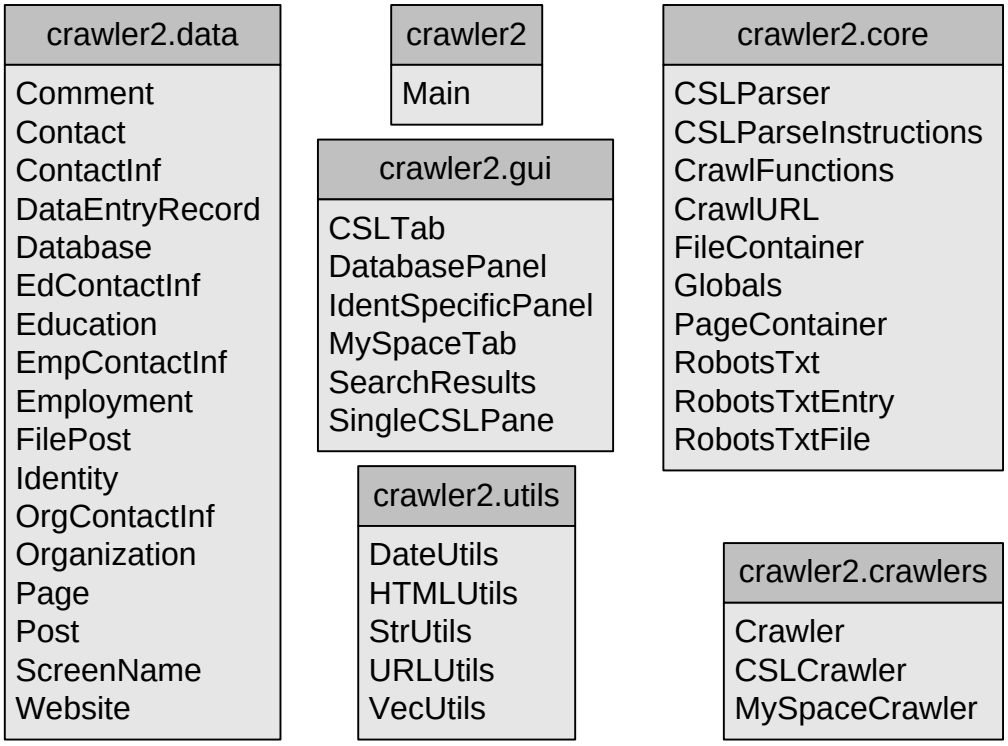

Figure 2. Crawler 2.0 Package Contents

7.1. Data Storage: crawler2.data

I began by designing the data storage component of the application. In order to ensure that data found could be identified again, I figured that the most important data element is the information about the webpage itself, such as when it was accessed, what the URL was, whether or not the URL was user provided and if not, then where it came from. By recording this information, the data items can then be tied together. This information is recorded in a Page object. The Page object is not seen in Figure 3 below,

but all other data items must reference an existing Page object to record where they were found

The next most significant data item is an Identity. The Identity data item contains the information identifying a user. The identity field is the unique identifier for an individual (in MySpace, for example, this is the friendID value). It also contains a field to record the name of the system that the identity is from (MySpace or Facebook, for example). There are additional fields for a birth date and name. All other data items must reference an Identity item.

There are three types of posts which can be recorded by the data items. A Post data item indicates text that is posted without reference to another data item. For example, any information which a user puts directly on their own social networking profile would be considered a post. There is also a FilePost data item which is similar to a Post (and can be attached as part of a Post). The FilePost data item records a file posted by a user. Posts and FilePosts are attributed to the user which posts the item only. The third kind of post is a Comment. A Comment is a textual post that is made in reply to some other piece of data. It could be in reply to a Post or FilePost, but could also be in reply to a Page, Identity (an entry posted on someone else's profile would be considered in reply to the Identity that the profile is for) or any other data item. The Comment data type is linked to two Identities (the one which made the comment, and the one whom the Page or other data item where the Comment was found is attributed two). It is also linked to the item which it is a reply to.

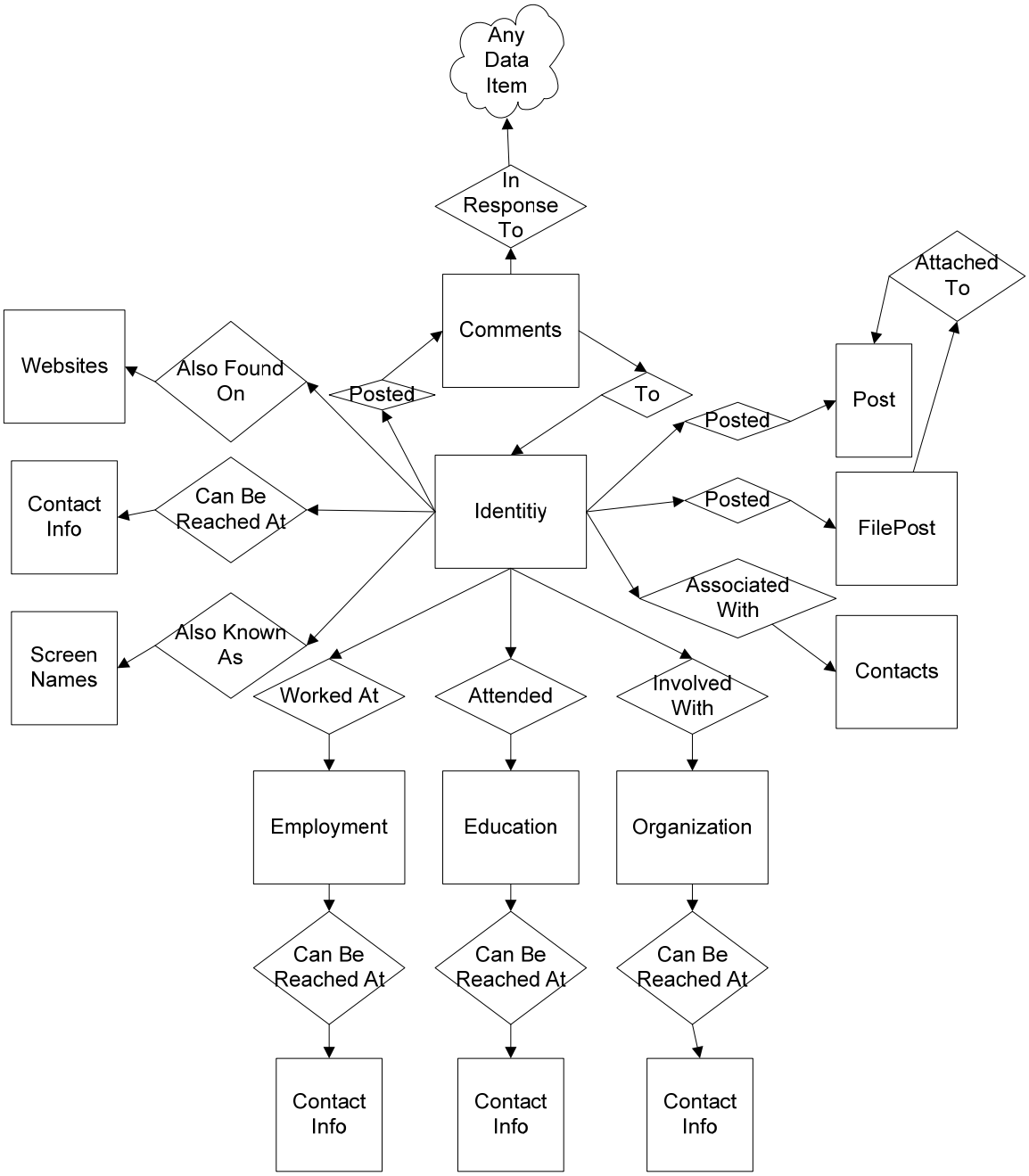

Figure 3. Data Storage Design

Additional data items for which classes exist are contact information (ContactInf), Contacts, and ScreenNames and additional Websites that are associated with an Identity. There are also data items to record Employment history, Education history, and Organizations that an Identity have been involved in and the respective contact

information pertaining to those records (EmpContactInf, EdContactInf, and OrgContactInf). All of the data items (except Page) and their relationships can be seen in Figure 3. Each data item is linked to the Page on which it was found upon creation.

Due to the complexity of the data storage portion, an additional class (Database) was developed to store and manage the data items. For easy identification of whether an item exists in the data store, a DataEntryRecord class was also developed. This class simply holds the key, type, and location of each item that is stored in the database. All data items contain a static TYPE field which is used by the Database class to determine where an item should be stored. The TYPE field and the id field in the DataEntryRecord are what is used to check for the existence of an item. When any new data item is created (other than a Page or Identity) an Identity ID and Page ID must be provided. DataEntryRecords are created for those IDs and types and their existence is verified before the object is created.

7.2. Utilities: crawler2.utils

As with any large programming project, it quickly became evident that there were several repetitive and common tasks which needed to be performed. These began popping up as early as while working on the crawler2.data package. The utility functions were broken up into separate classes based on the types of items that they assist with. This is a very reusable package and could easily be imported to other applications, which is part of the reason that it was kept separate.

7.3. Core Functions: crawler2.core

This package contains many of the classes that are essential to the operation of the application. It is a catch-all package to hold the classes which are essential to the operation of the application, but either don’t have the growth potential or aren’t specialized enough to call for their own package.

The CSLParseInstructions and CSLParser classes contain the information necessary to read and store the Crawler Specification Language modules. The CSLParseInstructions is the class that stores the information. The CSLParser is a static class which contains the functions needed to generate the instructions from a file.

The FileContainer and PageContainer classes are simply data structures which are used to return multiple pieces of information from some of the CrawlFunctions calls. CrawlFunctions is another static class. It contains functions related to the retrieval of pages and files from the Internet or from local storage.

The RobotsTxt, RobotsTxtEntry, and RobotsTxtFile classes are used for the retrieval, storage, and checking of robots.txt files. These files are a de-facto standard in use on the Internet today for server administrators to inform robots (a.k.a. crawlers) where they can and cannot go. Because Crawler 2.0 needs to obtain content in the same form that it would appear to the investigator if they were using a web browser, it identifies itself as "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.0.3705;)." However, it will recognize robots.txt entries identified as applying to all robots or which apply to "Crawler2.0," which is its name. Crawler 2.0 follows the rules for robots.txt found in the IETF Internet-Draft by M.Koster published on Dec 4, 1996 [29]. It does not handle any extensions and treats all unsuccessful HTTP response codes as a non-existent file (except for 3xx codes, which it will attempt to follow). It only caches the robots.txt files during the running crawl. Once the application is exited (even if the current crawl is saved) the robots.txt cache is cleared.

Finally, the core package contains a static Globals class. Breaking up the application into the numerous classes makes it much easier to recognize individual parts of it, however it makes the passing of information more difficult. For commonly accessed and/or static variables that are used by multiple classes and methods, it seemed that a central location would work best. This is the purpose of the Globals class. It stores

the common instance of the Database class, the HttpClient, and many user-specified options, such as storage locations, maximum crawl depth, and the time to wait before considering a page expired.

7.4. Crawlers: crawler2.crawlers

This package contains the classes that provide the parsing instructions for the application. No parsing is done except what is provided in these classes. There are always two classes present: Crawler and CSLCrawler.

The Crawler class is the base class for all other parsing classes. It is a very basic class which provides for storage of a name to identify the set of instructions, a set of identifiers to use to determine whether or not these instructions are applicable to a page, and a flag to indicate whether or not the set of instructions should be used. The default identifier types are: “HostEquals: ”, “URLContains: ”, “URLRegEx: ”, and “PageContains: ”. Host equals will match a URL based on an identical match to the webpage’s host. URL contains will match a URL based on a substring match to the URL. URL regex will match a URL based on a regular expression match to the URL. Page contains will match a webpage based on a substring match to the raw contents of the webpage. The Crawler class also contains very basic parsing capabilities which must be overridden by classes which extend it. The parsing provided by this class simply identifies links, and adds them to the list of links to be visited.

The CSLCrawler class is an extension of the Crawler class. It is the class which is used to implement Crawler Specification Language modules. On initialization, the application will create an instance of the CSLCrawler class for each module present in the CSL storage directory.

Currently, there is also a MySpaceCrawler class present. This class is an extension which handles the parsing of MySpace content. It extends the Crawler class by adding fields for a username and password, as well as a list of friendIDs to ignore. The

ignore list can be very helpful because there are sometimes friend accounts on a profile which are incredibly large and incredibly well linked, but which are unlikely to contribute to the investigation. An example of this would be the "Tom" account (friendID 6221) which is added as a friend to every new MySpace account by default. The login credentials and authenticate method allow an investigator to access content that is public as far as the MySpace network is concerned, but would be locked to a non-MySpace user.

7.5. Interface: crawler2.gui

Graphical user interfaces are fairly specialized parts of the application. Therefore, I felt that it would make sense to keep them all together in their own package. This was largely an organizational decision rather than one out of necessity. With the exception of the primary control panel, all customized gui components are stored in this package. This includes the classes to display and manipulate the results of the crawl. Further description of the gui is provided in Chapter 8.

7.6. Main: crawler2

This package only contains one class: Main. The Main class is an extension of a Java JFrame. It is the class which handles the initialization of the application and which controls the running of all other parts of the application. It also acts as the initial interface for the user and allows them to setup, initiate, and control Crawler 2.0.

The Main class initializes the graphical interface and sets some default values in the Globals class. It then checks the default storage location to see if there is an existing preferences file (named globals.dat). If the file exists, then it updates the applicable settings and updates their fields in the Preferences tab on the user interface. It then adds additional tabs to the user interface for Web 2.0 content types as needed. It then awaits user interaction. Further details on operation of the interface can be seen in Chapter 8.

CHAPTER 8. GUI OPERATION

The goal of Crawler 2.0 is to assist the law enforcement community, not the information technology community. This means that the target audience of the application cannot be assumed to have a strong technology background. Therefore, it is important that the application has an intuitive, easy to use interface.

To accomplish this, I first broke down the setup and operation of Crawler 2.0 into sections. In order to separate the different sections without hiding them, I chose to use a tabbed interface which can be seen in Figure 4. The tabbed interface was chosen because I believe it breaks apart the sections while keeping it clear that they exist (due to the tab headings).

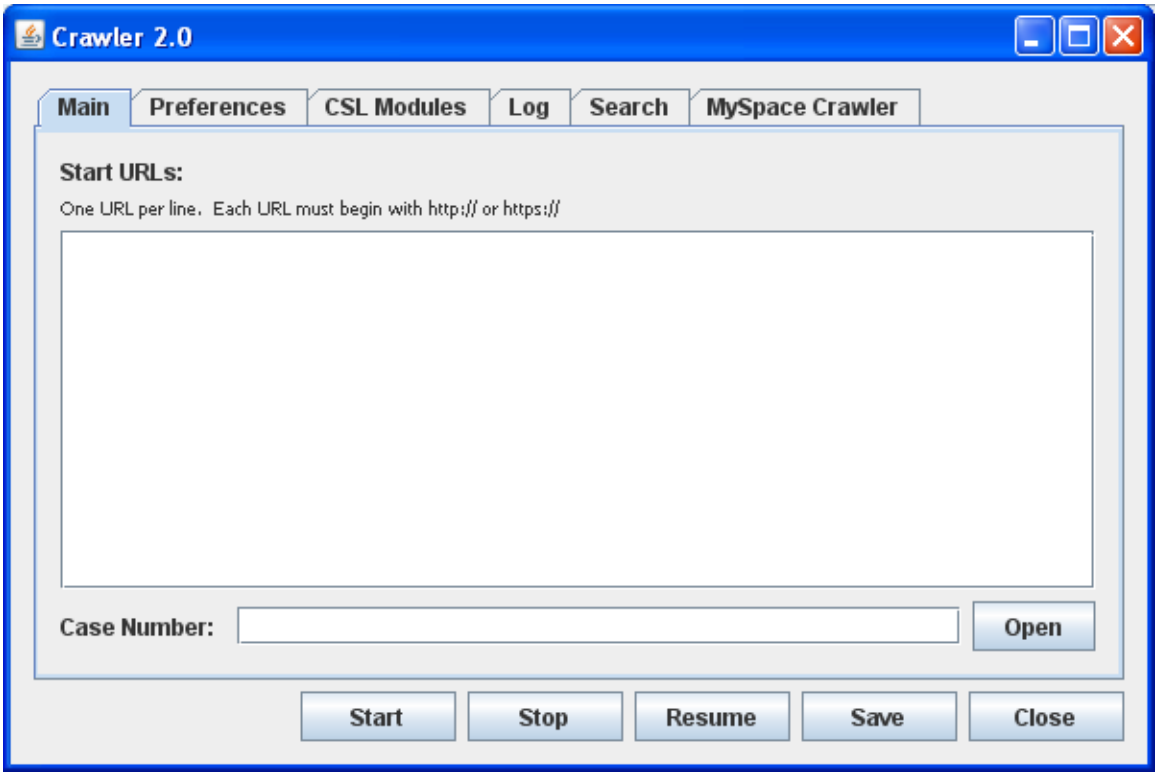

Figure 4. Crawler 2.0 Interface

8.1. Basic Interface

There are five tabs initially present in Crawler 2.0: Main, Preferences, Log, Search, and CSL Modules. In Figure 4, there is a sixth: MySpace Crawler. This tab is

present because I have implemented a MySpace class in the crawlers package for demonstration and testing of the application. The MySpace interpretation contains benefits from user-supplied options, and thus code was added for a MySpace Crawler tab. This is an example of a tab that has been added outside of the basic framework. The five buttons on the bottom of the interface are present regardless of the presently selected tab, and allow for control of application.

8.1.1. Main Tab

The Main tab is the tab which is shown initially upon loading the application. It is very simple and only contains three components. There is a text area for inputting the URLs with which Crawler 2.0 should begin the crawl. There is also an optional case number field which can be used to provide a name for the session. Folders with the session name will be created later if the session is saved, thus allowing for the future restoration of the session for further crawling or additional searches. Finally, the Open button allows the user to select a previous session to restore.

8.1.2. Preferences Tab

The Preferences tab, shown in Figure 5, is the most complex tab present in the gui. It allows the user to manipulate settings for the crawl and the storage locations for files. To help keep things easy to understand, the components are clearly labeled with the settings that the control. The preferences present on this tab effect the overall operation of the application. Additional preferences, which affect only one Web 2.0 content type may be added to individual tabs created by developers of extensions to the Crawler class.

The upper part of the Preferences tab contains fields for specifying the storage paths for pages, files, and data. The page storage path is where webpages are cached before they are parsed. The file storage path specifies the location to cache non-webpage downloaded content (pictures, videos, etc.). The program data path is where the session information, such as files containing the parsed data and the settings used for the crawl,

will be stored. The buttons next to the fields will open a new window which can be used to select the paths graphically, rather than typing them in manually.

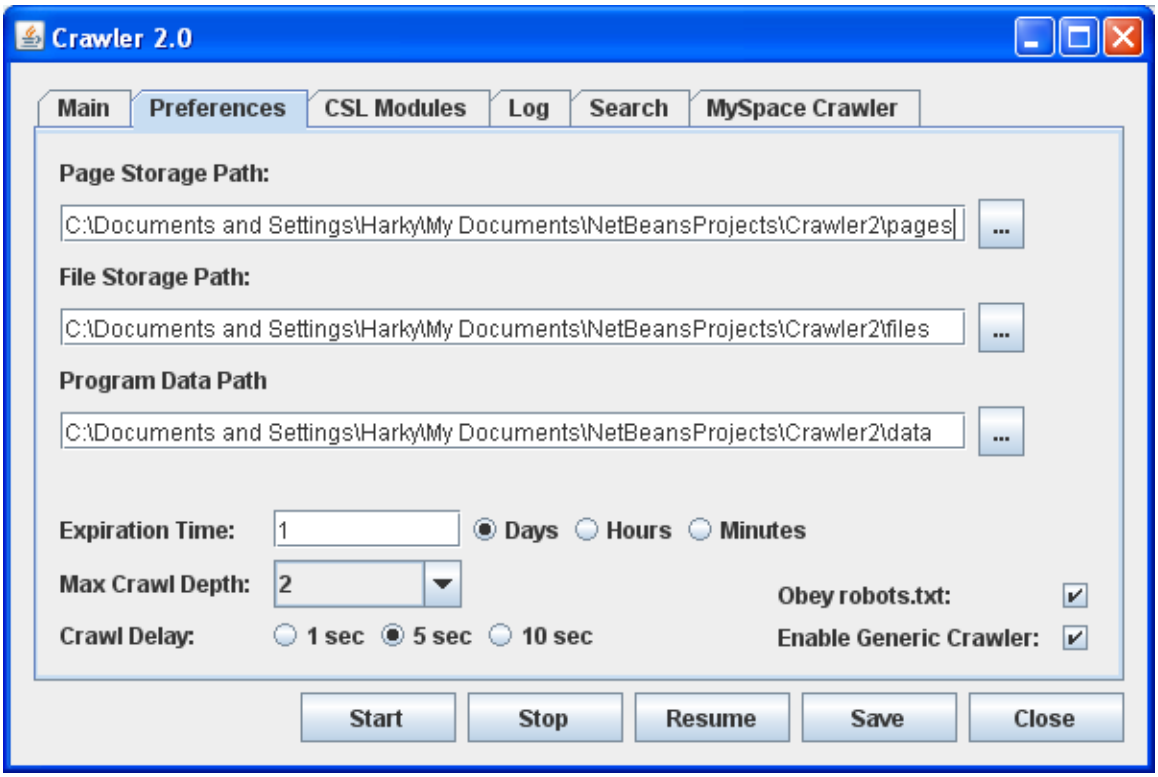

Figure 5. Preferences Tab

The lower portion of the Preferences tab contains the rest of the settings. The expiration time specifies the number of days, hours, or minutes that must have passed before retrieving the same URL over again if it is encountered multiple times. The crawl depth specifies the number of steps to take away from the starting URLs before stopping the crawl. The crawl delay specifies the number of seconds to wait in between HTTP GET requests. This is in addition to the time that it takes to retrieve and process the previous request. This is intended to help avoid excessive bandwidth use. The obey robots.txt checkbox enables the use of robots.txt files to recognize areas of websites which the application is not supposed to visit. The final checkbox enable the Crawler class to parse pages solely for links thus allowing pages more than one link away that cannot otherwise be parsed to continue to be obtained as long as they are within the depth

specified. Figure 5 represents the default options (except the paths which default to the pages, files, and data subdirectories of the directory that the program is running from).

8.1.3. CSL Modules Tab

This tab lists any existing CSL modules and allows the user to set whether or not they are enabled. By default all CSL modules will be enabled. Because the only parsing instructions that have been developed thus far have been implemented through the use of the more powerful Crawler class extension, there are no CSL modules, and this tab simply reports "There are currently no CSL modules loaded." However, the framework is all there for the automatic loading (on load of the application) of any CSL modules placed in the csl subdirectory of the application directory.

8.1.4. Log Tab

This tab simply reports the progress of the crawl. Each time that a page is finished being processed a line is printed to the text area. If there is a problem retrieving a URL or parsing the page obtained from the URL, then this is reported as well. There are also two buttons present, which allows the user to either clear the log, or save a copy of the log. The save button will open a new window for the user to select a path to save the file to. Entries are also posted to the log to indicate when the crawl has been completed and when saving has been finished (if the crawl has resulted in large amounts of parsed data, or if the crawl is paused while there is a large queue of links to process, the save process may take a while).

8.1.5. Search Tab

Upon completion of the crawl, the search tab allows the user to do two things. They can either view the parsed data in its entirety, using the "View Database" button, or they can initiate a search of the data. When initiating a search, there are two options (which are selected by default). One option controls the use of stemming, and the other option controls the use of stop words. With the options selected, the user can also bypass

them for specific words or phrases by enclosing the applicable word or phrase in quotes. Both the “View Database” and “Search” buttons will open new windows, which will be discussed later in this chapter.

8.2. MySpace Crawler Tab

The MySpace Crawler tab, shown in Figure 6 is an example of an extension tab. An extension tab would be any tab which is not part of the basic framework. These are tabs that are created outside of the basic framework. The code must be written and added to the application manually by the developer and then the application must be recompiled.

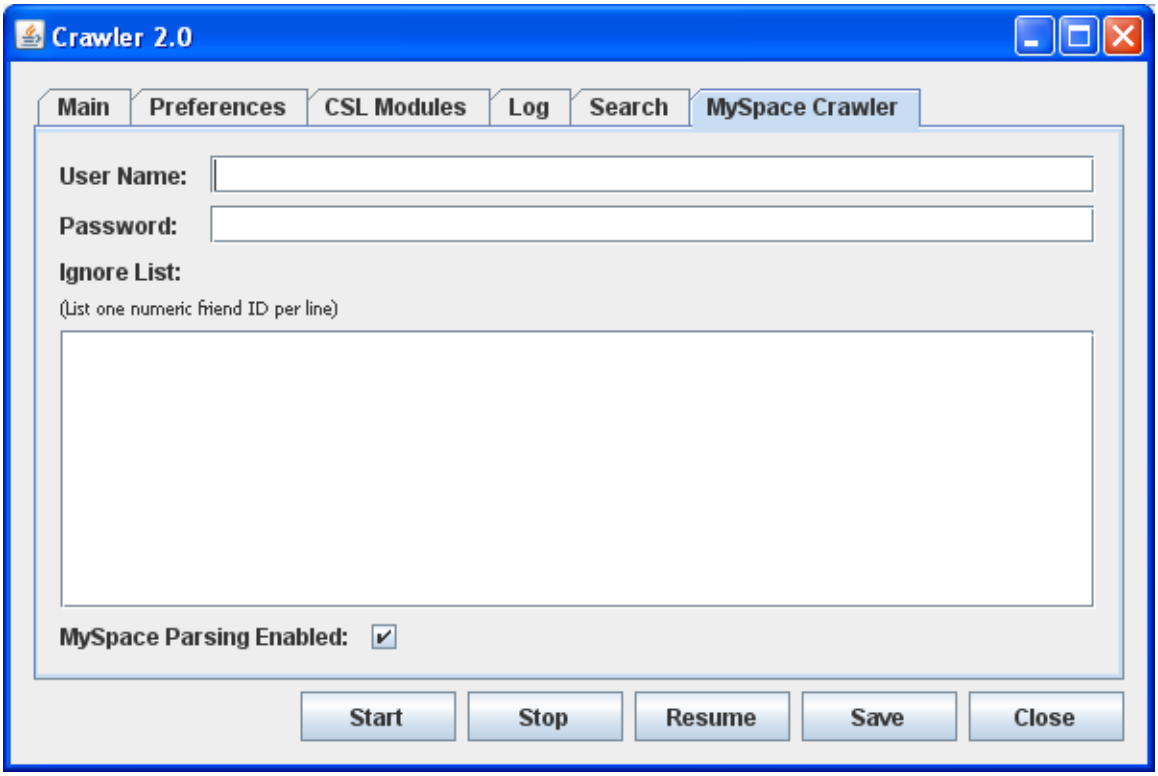

Figure 6. MySpace Crawler Tab

The MySpace Crawler tab consists of three fields and one option. The first two fields allow a user to input their login credentials. The password field hides the text so that someone walking by cannot read the user’s password. The third field is a text area which is used to input the IDs of profiles which the parser should ignore. This is useful

for friends which are known to have massive amounts of contents and are unlikely to contribute evidence to the search. An example of this may be a college mascot, a popular band, or a celebrity. Finally there is a checkbox which can be used to disable the crawler if the user knows that they do not want to parse MySpace content (or if there were to be two different versions of the parser with different capabilities).

8.3. Database Viewing Windows

8.3.1. Database – Entire View

When the "View Database" button on the Search tab is clicked, it brings up the "Database – Entire View" window. This window can be seen in Figure 7. It consists of seven tabs, which cover all of the parsed data (only the page entries where data was found are displayed and they are displayed with their applicable data items).

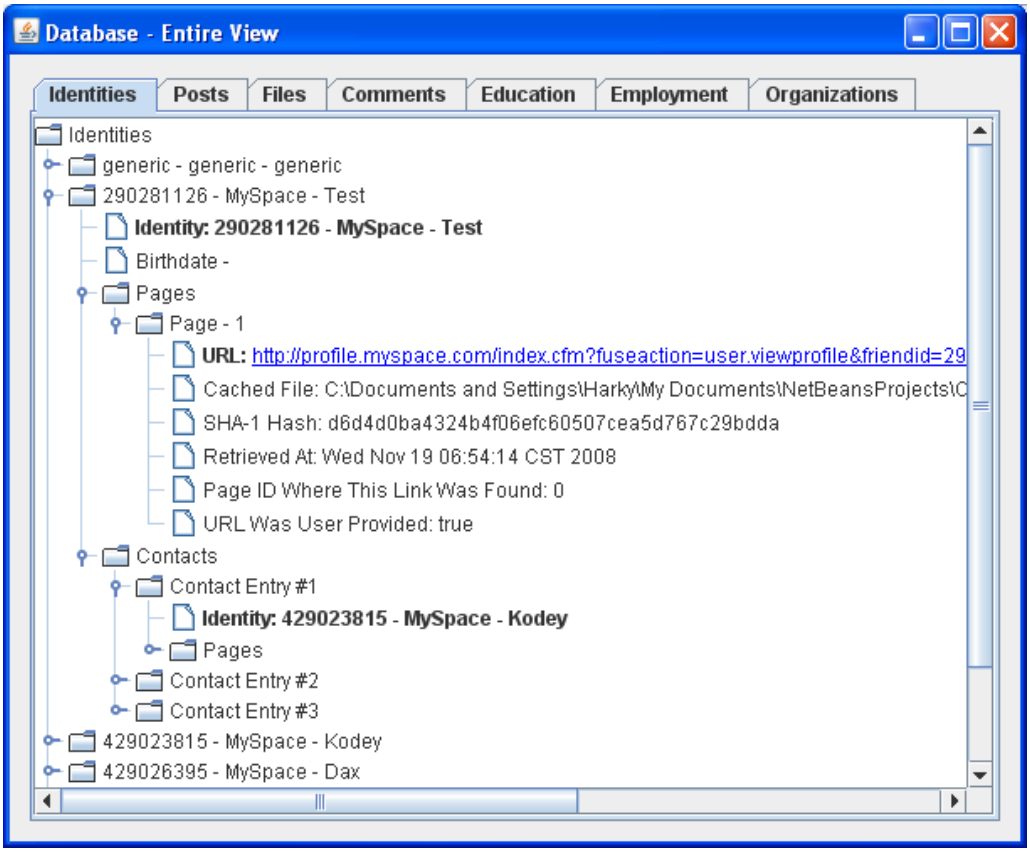

Figure 7. Database - Entire View

Each tab displays data as a tree structure. The contact information, contacts, screen names, and alternate website data items are all wrapped up into the Identities tab with the identity object that they are related to. Likewise, the education contact information, employment contact information, and organization contact information data is wrapped up with their associated education, employment, and organization data items on the applicable tabs.

Within these tabs, there are two types of clickable fields which will open new windows for the user. The URL fields for the page and file entries are clickable. They will open the default web browser and go to the specified URL. Additionally, the bold faced identities will open a new window.

8.3.2. Database – Filtered To Identity

The “Database – Filtered To Identity” window will display exactly what it says. Seen in Figure 8, it is similar to the window displaying the entire database, but it only displays entries related to the specified identity.

It also has an additional pane on the right hand side which displays the information about the identity that it has been filtered to. This information will include the contact information, screen names, and additional websites for the filtered identity here as well (if applicable). It also replaces the Identities tab with a Contacts tab where it displays information about the contacts associated with the filtered identity. In Figure 8, you can see the filtering at work with the Comments tab, where the numbering contains large gaps where comments that were not written either to or by the filtered Identity have been excluded.

Similar to the other database window, it will allow for clicking of links within the tree data and for clicking bolded identities. Rather than changing the currently filtered identity, clicking one of the bolded identities will open an additional filtered window so

that the user does not lose their current view. The URL links in the pane on the right-hand side are also clickable, and will open the URL in the user's default web browser.

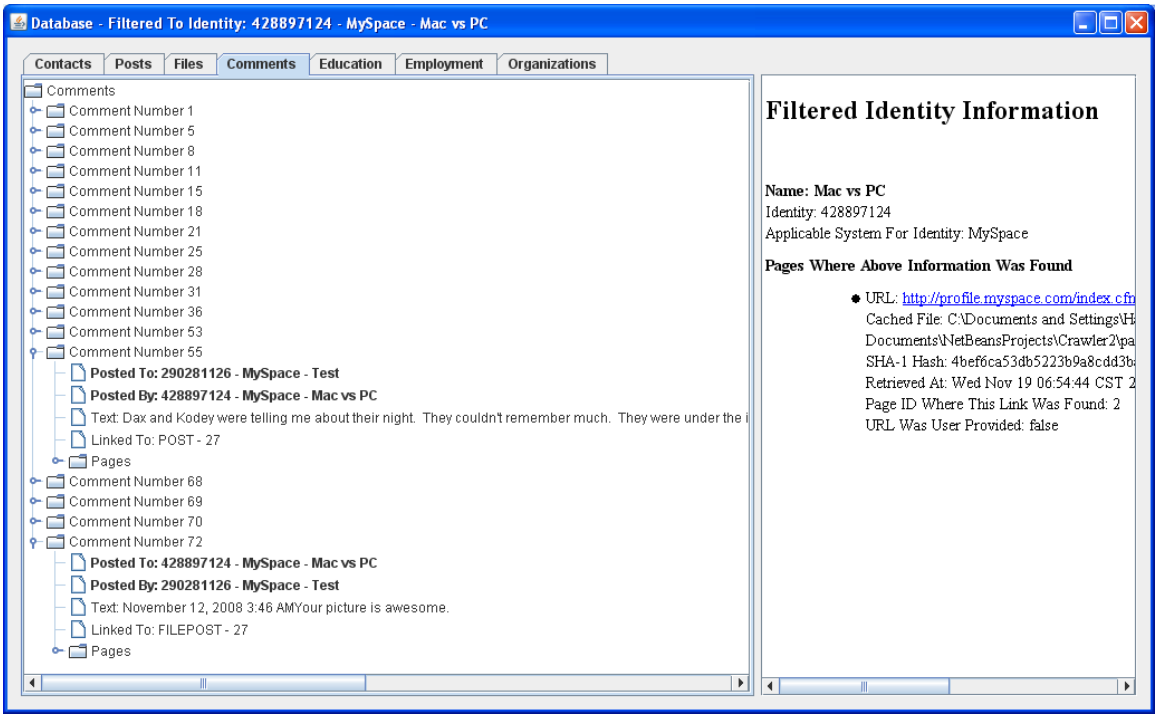

Figure 8. Database - Filtered To Identity Window

8.4. Search Results Window

The final window that is part of the user interface is the "Search Results" window. This window is displayed when the user clicks the "Search" button in the Search tab. It consists of three parts: the search terms, the search results, and the result details. Examples of this window can be seen in Figures 9 and 10, with the search results themselves explained in Chapter 11.

The search terms are displayed at the bottom of the window in a non-editable text field. These are displayed to remind the user of the terms that they searched for, and to show them if any words were omitted (because they were stop words and "Ignore Stop Words" was checked on the Search tab). They are not stemmed, so as not to confuse the user since some of the stems due not always make sense (such as "sprai" being the stem for "spray") but still work since the related words will be reduced to the same stem.

The search results are displayed on the left half of the window in a tabbed pane with two tabs. Each of the tabs contains a list of items, where each item (one item per row) represents one search hit. The first tab is the Parsed Data Search tab. This tab lists search hits amongst the parsed data. The second tab is the Plain Text Search tab and lists search hits resulting from a substring search of all cached pages, regardless of their ability to be parsed or not.

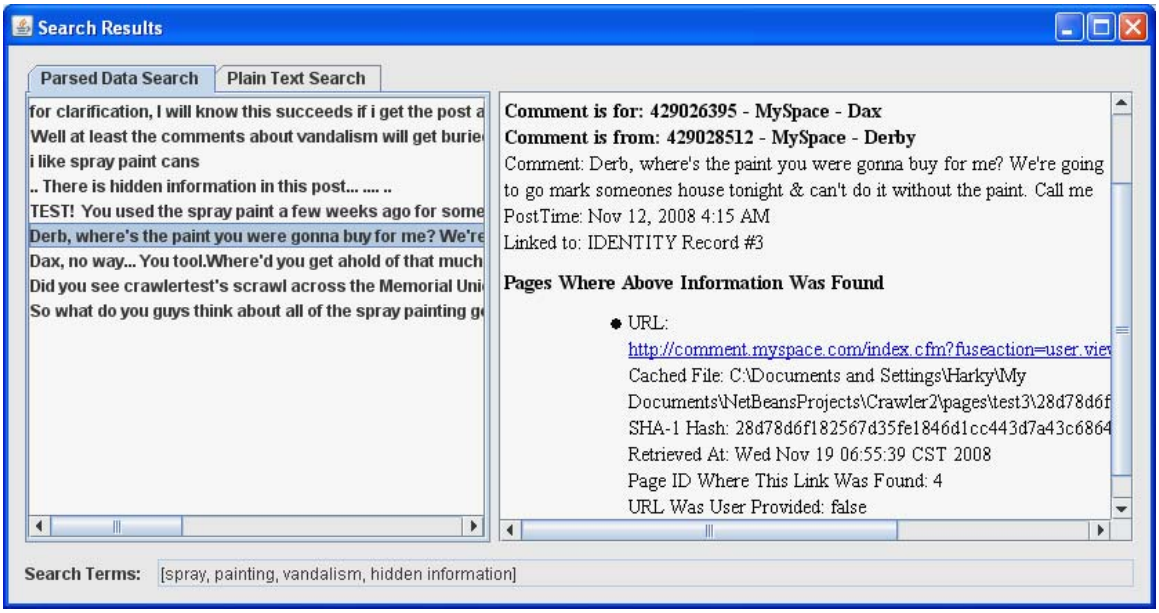

Figure 9. Search Results Window - Parsed Data

When searching in the parsed data, only some of the fields are searched. First of all, FilePost, Page, and Contact data items are not searched at all. The only string-based fields in those data items are the file extension and the URL fields. In the Identity data item, the birth date field is ignored, and in Post and Comment data items, the post time and link/attachment fields are ignored. Likewise, the start and end dates are ignored in the Education, Employment, and Organization data items. In all types of contact information data items, the phone and fax information is ignored.

When an item in the list of parsed data results is selected, the information about that item is loaded in the right-hand pane. Similar to the database windows, the URLs are

clickable and open the user's default web browser. However, the bolded identities are not clickable.

Figure 10. Search Results Window - Plain Text

When searching plain text, all pages are searched. The purpose of this is to search for information which may not have been parseable, and also to locate hidden information according to Section 4.4. In the plain text search, although stop words are ignored if the option had been selected, stemming is not performed. A substring search is performed for each keyword (or quoted phrase). If a search hit is found, the text surrounding the search hit (up to 15 characters on either side) is added to the results and removed from the page (in memory, not the cached copy). This prevents the same section of the webpage from being shown multiple times if more than one search term appears in close proximity, but doesn't present the page from showing up multiple times if different search terms are found in different areas of the webpage.

When an entry is clicked in the plain text results list, it loads the page information into the information pane on the right. The link is clickable and opens the user's default browser.

CHAPTER 9. CRAWLER SPECIFICATION LANGUAGE

Crawler Specification Language (CSL) is a language which can be used to write external parsing modules (Crawlers) for Crawler 2.0. It is designed so that users without a significant programming background can still add functionality to the application. It does however require basic understanding of HTML.

A CSL module is a text file which contains instructions (written in CSL) that Crawler 2.0 can use to parse applicable webpages. It consists of four sections: Priority, Name, Identifiers, and Parse. All sections begin with the section name enclosed in square brackets on a line of its own and end with one or more blank lines.

9.1. Priority Section

The priority section is optional and simply consists of the section name on the first line, followed by a single integer on a line by itself and then one or more blank lines to indicate that it is the end of the section. The integer is the priority value, and is used to order the set of parsing instructions in the list maintained by the application. Acceptable priority values are between 1 and 100, with 1 being the highest priority. If multiple sets of parsing instructions have overlapping identifiers, whichever set of instructions has the highest priority will be used to handle pages matching the overlapping identifiers. If there are multiple sets of instructions with the same priority that have overlapping identifiers, the order they will be placed in the list is not guaranteed, and any one of them may end up handling applicable pages.

9.2. Name Section

The name section is required and is used to identify the set of instructions in the CSL Modules tab. It consists of the section name on a line by itself, followed by a line for the name field, and one or more blank lines. The line for the name field must begin with “Name: ” and be followed by text

9.3. Identifiers Section

This section is required and contains the instructions telling Crawler 2.0 what web pages to use this set for. It consists of the section name on the first line (by itself), then any number of identifier lines, and then one or more blank lines to signify the end of the section. If there are no identifier lines, then it will be assumed to be applicable to all web pages. The possible identifier lines are: “HostEquals: ”, “URLContains: ”, “URLRegEx: ”, and “PageContains: ”. Host equals will match a URL based on an identical match to the webpage’s host. URL contains will match a URL based on a substring match to the URL. URL regex will match a URL based on a regular expression match to the URL. Page contains will match a webpage based on a substring match to the raw contents of the webpage. For each of these, a line consists of the type (i.e. “HostEquals: ” followed by the text that is to be matched).

9.4. Parse Section

This section is also required and contains the actual parsing instructions. It begins with the section name on a line by itself, then any number of data item instructions, and finally one or more blank lines to indicate the end of the section. A data item instruction is actually a subsection and contains the information necessary to parse a single type of data item (such as an Education record or a Post).

A data item instruction consists of the name of the data type surrounded by square brackets on a line of its own, then a set of instructions (one instruction per line), and finally a line containing only “END” to signify the end of the data item instruction. There are five possible instruction lines, three of which are required. The first line should be “StartTag: ” followed by the actual tag or a substring of the tag which indicates the start of this data type (for example, if <tr class=”contactInfo”> were the starting tag for a ContactInf data item, then a line reading StartTag: class=”contactInfo” would be acceptable). The second line should be “EndTag: ” followed by the ending tag (or

substring) for the data item (in the previous example, EndTag: </tr> would be acceptable). The third instruction is “HasFields: ” and contains either true or false, depending on whether or not separate fields for the data item are parseable. The final two instructions are required if HasFields was true. They are “FieldOrder: ” and “FieldDelimiter: ” and specify the information needed to parse the different fields. FieldOrder contains a list of the parseable fields for the data item separated by a semicolon (for example address;phone could be a valid entry if the data type were contact information). FieldDelimiter contains the character or string that serves as a delimiter for the parseable fields (going back to the previous example </td><td> might be an applicable delimiter).

For each data item type that can be parsed from an applicable webpage, there should be at least one data item instruction for that data type. Unfortunately, not all data types are recognized by Crawler 2.0 when reading the CSL module. Additionally, it not all fields of the data items are parseable either. These limitations will be discussed later in Chapter 12.

CHAPTER 10. TESTING

This work has been focused primarily on designing and implementing the framework application that is Crawler 2.0. To show that the framework can be useful, I have performed some testing. An extension of the Crawler class has been developed for MySpace profiles, and profiles have been set up to test it. Additionally, webpages have been created which contain hidden information per Section 4.4.

10.1. MySpace Profile Items

Before writing the MySpace crawler, I had to decide what information was of interest on a MySpace profile. To do this I looked at my own MySpace profile, as well as friends profiles. I also created a new profile to see what options were available that either weren't available when I created my profile, or that I had forgotten about. Doing this, I came up with the following as potential items of interest (their applicable data types are in brackets as well).

- Profile Owner <Identity> <ContactInf>
- Interests: General, Music, Movies, etc. <Post>
- Details: Hometown, Body type, Smoking/Drinking, etc. <Post>
- Schools attended / Organizations participated in <Education> <Organization> <EdContactInf> <OrgContactInf>
- Networking interests <Post>
- Employers <Employment> <EmpContactInf>
- Blog entries and applicable comments <Post> <Comment>
- Blurbs: About me, Who I'd like to meet <Post>
- Friends <Contact>
- Comments <Comment>
- Photos, captions, and applicable comments <Post> <FilePost> <Comment>
- Videos, descriptions, and applicable comments <Post> <FilePost> <Comment>

10.2. MySpace Structure

To parse a MySpace profile, determined the structure of a MySpace page. To do this studied the source of multiple MySpace pages. Since I studied the pages in my browser, I was familiar with the content and thus knew what data I was looking for. Once I had found an item, I began looking around the item to find what identified the item. I found that MySpace pages were heavily table-based (not surprising if you've looked at them). I also found that there is a lot of JavaScript in use. Luckily the JavaScript had little effect on the parsing of the contents of the data items. It did however make it difficult to get all of the items. However late in development this was overcome for some of the data items. Given more time, this should be able to be overcome for the remaining data items. Further information will be given in Chapter 12.

I also noticed that not all content appears on the profile page. Thus I determined the links to access the friends, comments, blogs, pictures, and videos independently. I defined all content related to a single profile as being at the same depth, thus a comment on the fourth page of comments would still be at the same depth as the profile that the comments are for.

However, some of those pages are only accessible to logged in users (whether or not the profile is locked down or not). Thus I had to develop an authentication method for the class to be run before beginning the crawl. Using a previously developed MySpace API from Gath Adams [30] as a guide, I was able to develop a POST form-based authentication method.

10.3. Profile Setup

I setup a MySpace profile with the name "Test" and put in information for every available field. I also had three friend profiles setup solely for testing purposes. Miscellaneous communications back and forth between the profiles was performed. Some of the communication was random, and other communication concerned supposed

illegal activity. Additionally I made comments which included links to external sites to test that the link parsing worked and to test plain text searching. Additionally, I had one of the profiles set up private to show that the authentication was working, and another profile set up as a friend of a friend to show that information can be gathered from nearby, but not directly-linked profiles.

10.4. Hidden Information

I created three plain web pages, each one to demonstrate one type of information hiding presented in Section 4.4. I then linked to these pages through comments on the MySpace profiles. Since the hidden information occurs on non-MySpace pages, it will not be parsed, but the pages should still be retrieved as long as they are within the specified crawl depth, and thus should be able to be located with a plain text search.

CHAPTER 11. RESULTS

After setting up the MySpace profiles and the webpages containing hidden information, I ran Crawler 2.0. I tested for three things: MySpace authentication, data gathering, and search functionality.

11.1. MySpace Authentication

To test MySpace authentication, I ran the application both with and without login credentials for the "Test" profile and compared the results. The first thing I noticed was that the crawl without authentication was over quicker than the one with authentication. I then looked at the number of comments found by each session. The results without authentication can be seen in Figure 11, and can be compared to the results in Figure 12, which used authentication. The reason for the difference is that most of the comments occur within the comments pages (as compared to responses to blogs, pictures, etc.), which are off limits if you are not logged in, regardless of the public/private status of the profile.

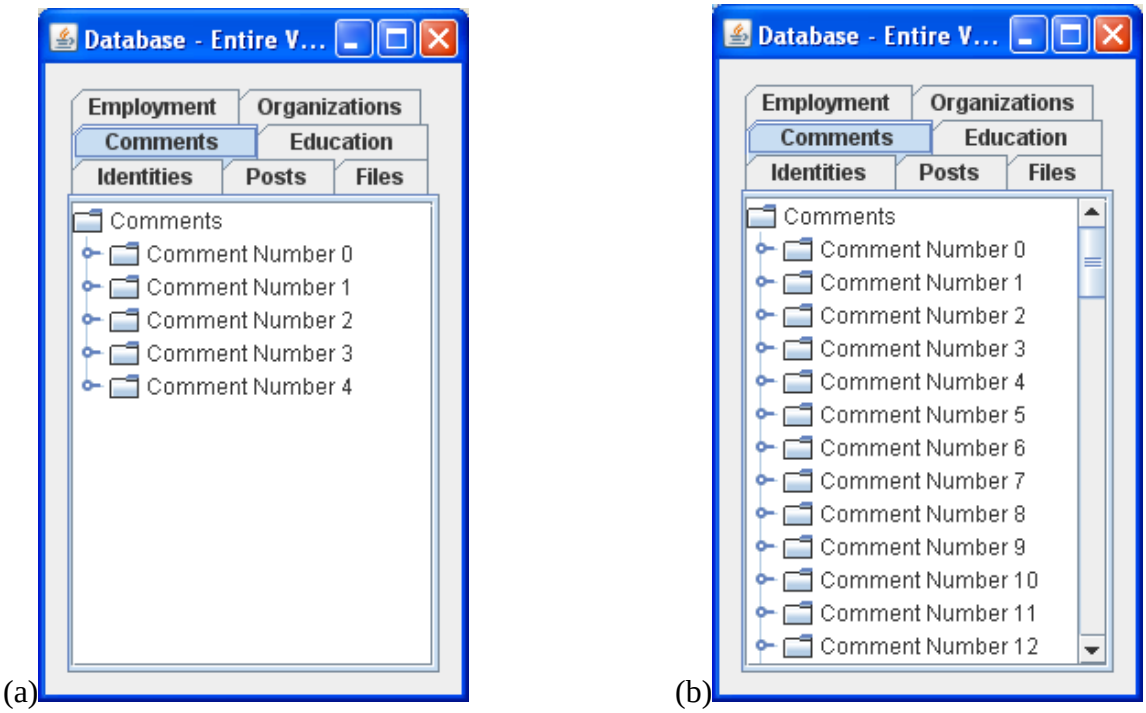

Figure 11. Comments Found (a) No Authentication vs. (b) With Authentication

11.2. Data Gathering

Since I am aware of all of the content on the MySpace profiles, I am aware of what data exists on them. To determine whether the data gathering was working, I used a crawl with authentication and then used the "View Database" option to see what information was parsed from the profiles. Upon reviewing the items, I was able to confirm that the all of the data that I expected to be parsed was indeed found.

11.3. Search Functionality

After viewing the database results, I performed a search to see if the searching was functional. I performed the search with the intent of finding two things: information related to vandalism (in particular spray painting) and hidden information. The search string used was *spray painting vandalism "hidden information"* and both stemming and the ignoring of stop words (none of which were present in the search terms) were enabled. The results previously shown in Figures 9 and 10 are the results of this search. They showed that the information posted in blog posts and comments about vandalism were found, and that they were also found in the plain text search. Additionally, the plain text search found all three types of hidden information.

CHAPTER 12. CONCLUSIONS

Web 2.0 is a rapidly growing technology. It is already a huge part of the lives of many Americans, especially the younger generations. Blogs, wikis, bulletin boards, and social networks are all over the place. As users become more comfortable posting their innermost secrets or providing personally identifiable information, risks to their safety go up. At the same time, they are becoming more and more likely to expose (intentionally or not) evidence of crimes or other misuse to the public. This has already been seen in numerous court cases.

As law enforcement is beginning to see Web 2.0 as a place where criminal information exists, they are beginning to use it more. However, even a small search can quickly grow painstakingly large. And while many Web 2.0 companies are trying to help combat the use of their systems for crime, they are not providing much in the way of investigative assistance for crime that is discussed or even promoted using their systems.

Crawler 2.0 is a tool developed with law enforcement in mind. It performs much of the data gathering for them, and then allows them to review the results in whole or based on searches. It records all of its activities and caches the pages that it visits locally to create the evidence trail. It works without jeopardizing privacy rights because it only crawls for information that an investigator would be able to see by themselves. It is built to be upgradeable, portable, and easy to operate.

Testing has shown that Crawler 2.0 can do what it was built to do. Testing was performed for the MySpace social network, and results were able to show that it can successfully retrieve information from the system. Additionally, test showed that some ways that information could be hidden in webpages could also be defeated by Crawler 2.0.

12.1. Limitations

Unfortunately, Crawler 2.0 also has some current limitations. While it is modular in design, the functional aspect of the modularity does not work as well. Additionally, it currently lacks robustness as it only knows how to parse one source of Web 2.0 content: MySpace, which has a few limitations of its own.

12.1.1. CSL Limitations

As reported previously CSL modules are limited when it comes to what they can parse. The first problem is that many of the data types that CSL modules could interpret do not have simple default fields that the data can be dumped into. This means that CSL requires that the HasFields instruction is true, thereby requiring that the fields are separated by a common delimiter. This issue affects Education, Employment, and Organization data types as well as all contact information data types. Unfortunately, if MySpace is an example, there is rarely an identical separation between different fields of an item.

The other problem is that it is very difficult for a CSL module to determine relationships between items. For this reason, items that require a link to another item must be specified with the item they are linked to. For example contact information data for Education, Employment, and Organization data types must be specified with the initial data items. To do this, a field with the name of the data type (EdContactInf, for example) must be specified as a field to the parent item. The item itself must then have fields (note that the contact information must still contain start and end tag instructions, but that they will be ignored).

Identity data items can only be recognized in an <a href… tag. This means that the only way to parse identities (if they cannot be parsed the generic default identity will be used) is if there is a link tag which can be uniquely identified. In this case, the URL pointed to in the <a href… tag will be used as the identity. If identification of identities is

enabled, then new identities found on a page will automatically be marked as contacts for the identity that the page is associated with.

ContactInf entries (not for Education, Employment, or Organization), Post entries, and ScreenName entries are simply linked to the identity that the page is assigned to. Post entries and ScreenName entries are limited in their field options. Each can have either no fields or exactly two fields. In the case of the Post, the two possible fields are text and postTime. If no fields are specified all of the contents are put into text. In the case of the ScreenName, the two fields are sName and proto. If there are no fields, then all contents are defaulted into sName.

Website data items and Comment data items simply cannot be parsed by the CSL module. Additionally, FilePost entries are automatically created by the CSL module. Anytime that a link is encountered which does not appear to be HTML (does not end in a recognizable web-related extension, or with "/"), it will assume it is a file and create an applicable entry.

12.1.2. MySpace Limitations

Crawler 2.0 knows how to parse MySpace content, however it only knows how to parse MySpace content. This is a current limitation that clearly affects is use as a robust tool. However this limitation is based primarily on time constraints. Furthermore, there are some limitations to its abilities with MySpace content, which were caused by time constraints. When there are many comments related to videos or photos, the comments will be spread across multiple pages. There is no known direct link to the additional pages, however they are accessible through JavaScript. At the time of initial development of the parsing instructions for these comments, it was not clear how to process this. Later work successfully did this for regular profile comments, so it is likely that the same process can be used for photo and video comments. Additionally, no way is currently apparent for downloading posted video files directly. However there are

other applications that supposedly do this, so it can be done. The method just needs to be found.

12.2. Future Work

Based on successful testing with Crawler 2.0, I do believe that the framework is fairly solid. The most crippling limitation is its lack of robustness in parsing. However this limitation is time based and future work should be able to alleviate it. Additionally, there is one new feature that I believe could be a useful addition.

12.2.1. Improve Robustness

With time spent on researching the format of other Web 2.0 content types, this could be improved greatly. There are many highly popular social networks and many Web 2.0 technologies for blogs, wikis, and bulletin boards that could be added to Crawler 2.0 to improve its usefulness to investigators. The main hurdle to overcome is the time it takes to figure out the inner workings of the technologies.

12.2.2. Use and Improve CSL

I believe that CSL does have a function. It could currently be used to parse Posts, Education records, Employment Records, Organizations and a few other data types if they are delimited. However, it is a matter of finding technologies and/or websites that present information in such a format. I believe that additional time spent on CSL may also be able to rework it to make it more robust in its capabilities as well.

12.2.3. Relationship Finding

A lot of times it is useful to find things that are common between multiple subjects related to a crime. Social networking technology may be able to help expose relationships between these individuals. Were they coworkers at one time? Did they go to school together? Do they have a mutual friend in common? These are just some of the things which may be found in social networks. However, the task of investigating this can be very time consuming and difficult, especially if there are many profiles to

compare and/or multiple sources of the profiles. A feature that Crawler 2.0 could be made to provide is the ability to either take multiple Web 2.0 profiles as input and find commonalities (common employers, groups, interests, etc.) or to look at the data it finds in a regular crawl and report back any commonalities that it finds.

REFERENCES

[1] Lawton, G. (2007). Web 2.0 creates security challenges. *Computer, 40*(10), 13–16.

[2] CBS Broadcasting, & Associated Press. (2006, February 3). *MySpace In Sex Assault Probe*. [Online]. Available: http://www.cbsnews.com/stories/2006/02/03/tech/main1277928.shtml [Accessed 2008, November 6].

[3] Williams, P. (2006, February 3). *MySpace, Facebook attract online predators*. [Online]. Available: http://www.msnbc.msn.com/id/11165576/ [Accessed 2008, November 6].

[4] Associated Press. (2006, September 14). *Woman accused of attempting MySpace hit*. [Online]. Available: http://www.msnbc.msn.com/id/14833529/from/RS.1/ [Accessed 2008, November 6].

[5] Associated Press. (2008, May 15). *Mom indicted in deadly MySpace hoax*. [Online]. Available: http://www.cnn.com/2008/CRIME/05/15/internet.suicide.ap/index.html [Accessed 2008, November 6].

[6] Rasch, M. (2008, May 22). *Anti-Social Networking*. [Online]. Available: http://www.securityfocus.com/columnists/473/1 [Accessed 2008, November 6].

[7] Associated Press. (2006, April 5). *Teens arrested after posting alleged firebombing video on Myspace.com.* [Online]. Available: http://www.usatoday.com/tech/news/2006-04-05-myspace-arrest_x.htm [Accessed 2008, November 6].

[8] Chow, R., Golle, P., & Staddon, J. (2007). Inference Detection Technology for Web 2.0. Presented at *Web 2.0 Security and Privacy 2007,* Oakland, CA. [Online]. Available: http://seclab.cs.rice.edu/w2sp/2007/ [Accessed 2008, November 19].

[9] Hersovici, M., Jacovi, M., Maarek, Y.S., Pelleg, D., Shtalhaim, M., & Ur, S. (1998). The Shark-search algorithm. An application: tailored web site mapping. *Computer Networks and ISDN Systems, 30*(1-7), 317-326.

[10] Carnegie Mellon University. (2002). WebSPHINX (Version 0.5) [Software]. [Online]. Available from http://www.cs.cmu.edu/~rcm/websphinx/ [Accessed 2008, August 1].

[11] Miller R., & Bharat, K. (1998). SPHINX: a framework for creating personal, site-specific web crawlers. *Computer Networks and ISDN Systems, 30*(1-7), 119-130.

[12] Garrett, R.T., & Associated Press. (2008, January 15). *Texas AG: MySpace agreement offers 'false sense of security'*. [Online]. http://www.dallasnews.com /sharedcontent/dws/dn/latestnews/stories/011508dnnatmyspace.2d6c721.html [Accessed 2008, November 7].

[13] Barnard, A. (2008, January 15). *MySpace Agrees to Lead Fight to Stop Sex Predators*. [Online]. Available: http://www.nytimes.com/2008/01/15/us/ 15myspace.html [Accessed 2008, November 7].

[14] Pennsylvania Office of Attorney General, (2008, May 8). Pennsylvania Attorney General Corbett Announces Multi-State Agreement With Facebook... Press release. [Online]. Available: http://www.reuters.com/article/pressRelease/ idUS209266+08-May-2008+PRN20080508 [Accessed 2008, November 7].

[15] Carter, D. (1995). Computer crime categories: how techno-criminals operate. *FBI law enforcement bulletin, 64*(7), 21.

[16] Looney, T. (2007, September). How Web 2.0 is Helping Public Safety. *Public Safety IT Magazine*. [Online]. Available: http://www.hendonpub.com/resources/ articlearchive/details.aspx?ID=4183 [Accessed 2008, November 10].

[17] Kirk, J, (2008, April 18). *British police use Facebook to gather evidence*. [Online]. Available: http://www.pcworld.ca/news/article/61e1ae570a01040801dd6a67117fad36/pg0.htm [Accessed 2008, November 10].

[18] Davis, W. (2006, May 15). *Teens' online postings are new tool for police*. [Online]. Available: http://www.boston.com/news/nation/articles/2006/05/15/teens_online_postings_are_new_tool_for_police/ [Accessed 2008, November 7].

[19] Apache Software Foundation. (2007). HttpClient (Version 3.1) [Software]. [Online]. Available: http://hc.apache.org/httpcomponents-client/index.html [Accessed 2008, August 4].

[20] Berners-Lee, T., Fielding, R., & Masinter, L. (2005, January). RFC 3986: Universal Resource Identifier (URI): Generic Syntax. [Online]. Available: http://tools.ietf.org/html/rfc3986 [Accessed 2008, November 10].

[21] Wikipedia. (2008, July 24). *URL Normalization*. [Online]. http://en.wikipedia.org/wiki/URL_normalization [Accessed 2008, November 10].

[22] Oswald, D. (2007). HTML Parser (Version 1.6). [Software]. [Online]. Available: http://htmlparser.sourceforge.net/ [Accessed 2007, November 10].

[23] Bordet, S. (2008). Foxtrot (Version 3.0). [Software]. [Online]. Available: http://foxtrot.sourceforge.net/ [Accessed 2008, November 5].

[24] Pant, G., Srinivasan, P., Menczer, F. (2003). Crawling the Web. In M. Levene and A. Poulovassilis (Eds.), *Web Dynamics*, Springer-Verlag. [Online]. Available: http://dollar.biz.uiowa.edu/~pant/Papers/crawling.pdf [Accessed 2008, June 27].

[25] Brin, S. and Page, L. (1998, April). The anatomy of a large-scale hypertextual web search-engine. *Computer Networks and ISDN Systems, 30*(1-7), 107-117.

[26] Wikipedia. (2008, November 15). *Stemming*. [Online]. Available: http://en.wikipedia.org/wiki/Stemming#Applications [Accessed 2008, November 18].

[27] Porter, M. & Boulton, R. (2002). libstemmer Java library. [Software]. [Online]. Available: http://snowball.tartarus.org/index.php [Accessed 2008, August 1].

[28] Ranks.nl. (n.d.). *English stopwords*. [Online]. Available: http://www.ranks.nl/resources/stopwords.html [Accessed 2008, November 18].

[29] Koster, M. (1996, December 4). *A Method for Robots Control*. [Online]. Available: http://www.robotstxt.org/norobots-rfc.txt [Accessed 2008, November 9].

[30] Adams, G. (2008, May 4). MySpace API. [Software]. [Online]. Available via email from G. Adams: http://gathadams.com/2007/05/04/myspace-api-2/ [Requested 2008, August 4]. [Received 2008, August, 4].

[31] Associated Press. (2007, April 28). *'Drunken Pirate' sues school that nixed degree* [Online]. Available: http://www.msnbc.msn.com/id/18372103/ [Accessed 2007, October, 27].

[32] Best, K. (2007, October 1). *Social networking has dangerous side* [Online]. Available: http://www.floridatoday.com/apps/pbcs.dll/article?AID=/20071001/LIFE/710010304/1005 [Accessed 2007, October, 27].

ACKNOWLEDGEMENT

I would like to take this opportunity to thank those who contributed to this thesis through their assistance, ideas and time. First of all, I thank Dr. Doug Jacobson, for your guidance and funding through my Graduate career. Your guidance and suggestions helped point me in the right direction and keep me on track so that I could complete this. Next I would like to thank Lt. Aaron DeLashmutt for your assistance as I have been learning how law enforcement operates. Your time and assistance has taught me a lot. Finally, I would like to thank Beth Harkness and Andy Viar for their assistance creating generic MySpace profiles and populating them with data for testing.