Developing a Process Model for the Forensic Extraction of Information from Desktop Search Applications

Timothy Pavlic
School of Computer and Information Science
University of South Australia

Jill Slay
Defence and Systems Institute
University of South Australia
Jill.Slay@unisa.edu.au

Benjamin Turnbull
Defence and Systems Institute
University of South Australia
Benjamin.Turnbull@unisa.edu.au

ABSTRACT

Desktop search applications can contain cached copies of files that were deleted from the file system. Forensic investigators see this as a potential source of evidence, as documents deleted by suspects may still exist in the cache. Whilst there have been attempts at recovering data collected by desktop search applications, there is no methodology governing the process, nor discussion on the most appropriate means to do so. This article seeks to address this issue by developing a process model that can be applied when developing an information extraction application for desktop search applications, discussing preferred methods and the limitations of each. This work represents a more structured approach than other forms of current research.

Keywords: Desktop search, digital evidence, extraction technique, process model.

1. INTRODUCTION AND BACKGROUND

Consumer electronics, especially computers and computer system components, have become more affordable and capabilities have increased, particularly in relation to storage. Smith (2004) shows that over four years, from April 2000 to April 2004, average hard drive capacities increased from 30GB to 250GB. This phenomenal increase in storage capacity has enabled users to store all of their data instead of deleting it to create space for new information they acquire. However, this increase in storage sees added complexity in the management of files and user-created documents, and the market has responded with increasingly powerful file search utilities to respond to this need.

The first generation of desktop searching utilities operated in real-time, which resulted in potentially lengthy search times. The second and current generation of desktop search utilities, in the form of Windows Vista search, Google Desktop Search, Spotlight and Beagle to name a few, are much like their internet search counterparts except they catalogue information on personal computers to allow fast searching of the indexed documents (Cole 2005). By pre-indexing user-data and storing this information in a database-type file (or files), these applications are able to search almost instantly. Information indexed by these applications includes file location information, file meta-data, and file content. As these devices are designed to store user-data, entire text files and image thumbnails may exist independently of their original creation area.

Unlike older desktop search utilities, the entire text of email, office documents, text files and PDF documents may be stored, as well as metadata associated with music and video files. Depending on which desktop search application is used, thumbnails may also be stored. As discovered (Turnbull, Blundell & Slay 2006), particular search utilities, such as Google Desktop Search, may also not purge their databases of a file when that file is deleted and keep a record of all files on a machine. Whilst this is not part of any standard and may be altered in more recent versions, it does give an insight into the value of forensic investigation of these systems as a component of a full analysis.

Since desktop search engines are still a new area of software development, there is little work that discusses these data stores as potential locations for digital evidence, but there is the possibility that the cached information stored by these applications can be of evidentiary use.

The extraction of data from desktop search utilities has the potential to be of great use to investigators due to the nature of the cache in many of the desktop search engines. The search applications continually index data on the machine as the user creates and modifies content. This can include emails sent and received from email clients, text documents, instant messaging conversations and web browser caches. All of this data is stored independently of the original electronic transaction, with the mechanisms for addition, updating and deleting specific to the application. It is therefore dependent on the application as to when such files are added and, as importantly, removed, from these file structures.

The only other research into this area found that it is possible to extract information from Google Desktop Search, a particular desktop search engine, but only in a work-around fashion, that is manual, cumbersome and inefficient

(Turnbull, Blundell & Slay 2006). The authors recommended that further work be conducted into either examining the databases structure, to allow tools to be developed to extract evidence directly, or to develop a plug-in for Google Desktop Search that can access the information via the program executables. It is important to note that this work was explicit to a specific desktop search application, and the access methodology created was not universal or generalisable. Each system differs in the paths it stores from, the parameters for stored information, the format of storage and the circumstances under which it will delete information.

The main problem found in this work was in the extraction of the information from the storage mechanism in a forensically sound manner, and ensuring that alteration has not further occurred, either through the application continually indexing new information or ensuring that the interface itself did not interpret data. In this work, it was discussed that allowing the applications to execute could potentially purge cached deleted items from the storage mechanism or overwrite older cached items with newer versions, and that this could potentially lose timestamp information or data.

This research endeavours to answer the question:

- Can information be data-mined from desktop search applications using a variety of methods so that the information can be used to locate evidence, and
- Can the process be verified to abide by forensic rules so that the evidence is considered admissible in a court of law?

Specifically, it seeks to determine different potential methods from which a desktop search application may be accessed, the appropriate order of attempting each of these methods, and the technical implications and ramifications that accomplish each process. This outcome is inline with other, similar work (Reith, Carr & Gunsch 2002).

The primary outcome is a process model that determines the most appropriate method of extracting data for forensic examiners, classified by search applications and versions. The architecture is comprised of a series of procedures detailing how to extract information from desktop search engines index databases. Each procedure of the architecture is accompanied by associated limitations and test plans.

It is hoped that a robust process model for the extraction of evidence from desktop search applications will enable a more simple and consistent approach for investigators handling these applications, as they are a new form of application and are possibly not currently searched as a potential form of evidence.

The next chapter of this work introduces the proposed process model, and discusses each stage in turn. The third and final chapter of this work discusses the outcomes, future research challenges and concludes this work.

2. PROCESS MODEL FOR EXTRACTION APPLICATION DEVELOPMENT

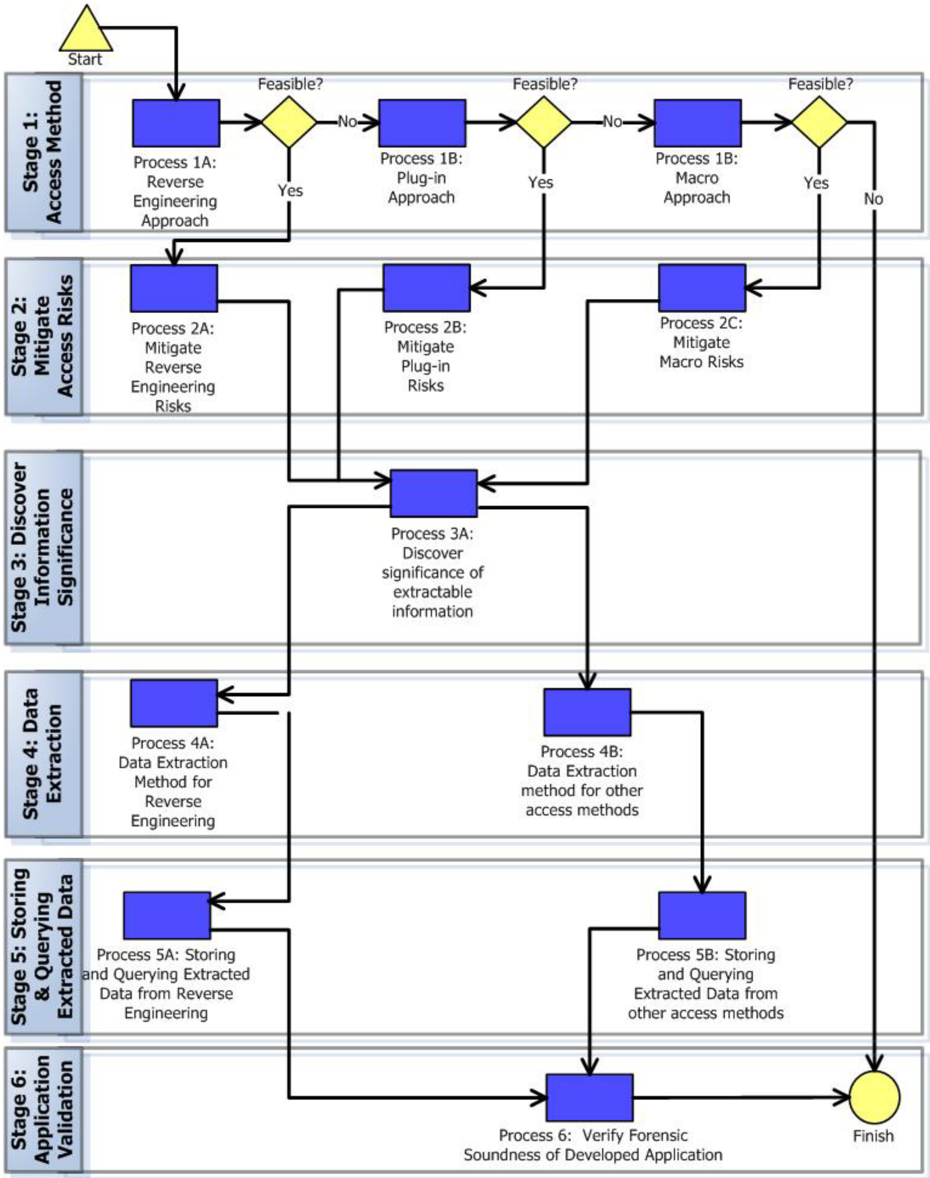
Start
Stage 1: Access Method
Process 1A: Reverse Engineering Approach
Feasible?
No
Yes
Process 1B: Plug-in Approach
Feasible?
No
Yes
Process 1B: Macro Approach
Feasible?
Yes
No
Stage 2: Mitigate Access Risks
Process 2A: Mitigate Reverse Engineering Risks
Process 2B: Mitigate Plug-in Risks
Process 2C: Mitigate Macro Risks
Stage 3: Discover Information Significance
Process 3A: Discover significance of extractable information
Stage 4: Data Extraction
Process 4A: Data Extraction Method for Reverse Engineering
Process 4B: Data Extraction method for other access methods
Stage 5: Storing & Querying Extracted Data
Process 5A: Storing and Querying Extracted Data from Reverse Engineering
Process 5B: Storing and Querying Extracted Data from other access methods
Stage 6: Application Validation
Process 6: Verify Forensic Soundness of Developed Application
Finish

Figure 1 - Process model for developing information extraction applications for desktop search engines.

It has been established at this point that no process model exists for the analysis of data from desktop search applications, but given the proprietary nature of these applications and the lack of standardisation, a methodology would be appropriate.

To answer the research questions, a process model defining how to develop extraction applications for desktop search engines was developed. The initial model, shown in Figure 1, represents the complete process followed to develop an extraction application for a given desktop search engine.

Stage 1 is the key stage of the process, as the method of accessing the information stored by the desktop search engine is selected, and all other actions performed in other stages are dependent on the method selected. In this model, three methods of data extraction from the desktop search application were considered:

- Reverse engineering the storage mechanism to allow for retrieval
- Interfacing with the search engine via an API
- Screen-scraping the application

Each of these was considered for forensic soundness and the completeness of information extracted. Whilst ease of use was considered, the most complete methods are given in order, from most to least desirable.

Stage 2 of Figure 1 examines the forensic risks of the selected access method in stage 1, and defines ways in which the risks can be mitigated or negated. This work is of importance as it highlights the strengths and weaknesses of each approach, and the limitations that may be in place for different investigators.

Stage 3 covers the exploration of the desktop search engine, documentation of the search engine and third party information about the search engine to determine the significance of extractable information, where initially the actual data values of the information can be ambiguous.

Stage 4 provides sub-processes that determine what procedures can be used to extract information from the search engine’s data store, based on the selected access method.

Stage 5 involves resolving the issue of storing the information that will be data mined from the search engine data store, and the subsequent querying of the data to locate evidence.

The final stage, stage 6, is applicable to all access methods, hence the merging of the individual flows into this single final sub-process. Stage 6 details how the developed extraction application requires testing and validation, to ensure that its operation maintains the integrity of data from the search engine data store once it has been extracted, and that the extraction process does not introduce risks to the forensic process.

2.1 Stage 1: Access Method

The first action to consider when developing a software application for the retrieval of information from a desktop search engine is the method in which the stored information will be accessed. The implementation of this stage of the process model will answer the first sub-question of the research: “For each desktop search engine, can the parallel storage mechanisms be accessed? If so, in what ways?”. The conceived options available for accessing the parallel storage mechanism of a desktop search engine are: reverse engineer the search engine data store (Process 1A), develop a plug-in for the application to utilise the native search abilities (Process 1B) and develop a macro application that simulates human user interaction with the application (Process 1C). Not all the options are applicable to every desktop search application and furthermore, each method does not have the same level of desirability.

2.1.1 Process 1A: Reverse Engineer the Search Engine Data Store

Reverse engineering, in a software engineering sense, is the process of deriving a base set of requirements for a system from a pre-existing, already deployed system by examining source code of the application, and other artefacts such as documentation, test results and file systems [1]. The accrued requirements from the reverse engineering process are then forward engineered to develop a system that essentially performs the same as the reverse engineered system. This process is employed to often bring legacy systems from non object oriented approaches into modern object orientated based systems, to aid continual development of the system. However, in this instance, investigators are attempting to determine the format of files associated with the applications so as to be able to analyse them directly without running a desktop search application. Such direct access could allow access via independent applications or as an additional feature of existing forensic examination applications. A proposed outline of the proposed processes associated with the reverse engineering sub-process are given in Figure 2.

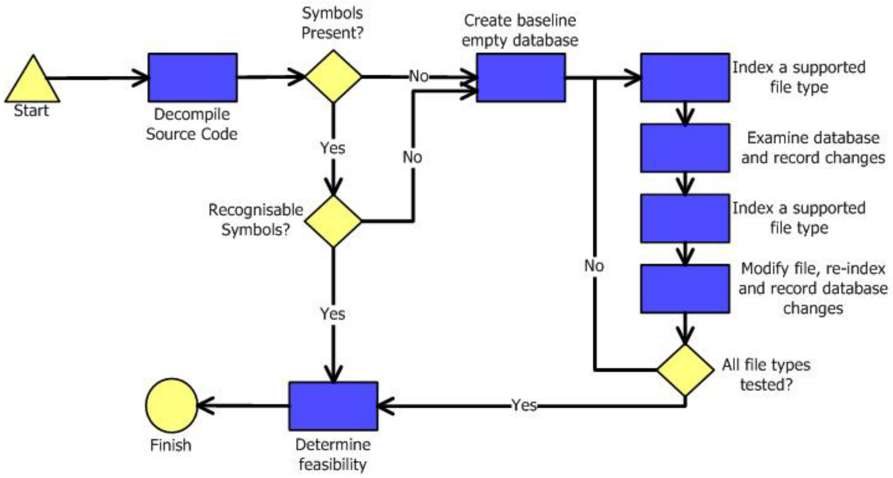
Start
Decompile Source Code
Symbols Present?
No
Yes
Recognisable Symbols?
No
Yes
Create baseline empty database
Index a supported file type
Examine database and record changes
Index a supported file type
Modify file, re-index and record database changes
All file types tested?
No
Yes
Determine feasibility
Finish

Figure 2 – Process model 1A: Reverse engineering a search engine data store.

2.1.1.1 Limitations of Reverse Engineering

The legality of reverse engineering is currently a highly debated topic. In the United States, law 17 USC §1201 allows the reverse engineering of applications for interoperability, similar to the Australian Copyright Act 1968, Part III Division 4A Section 47D (1b) which allows reverse engineering for developing interoperable applications. Whilst these rules would seem to allow reverse engineering to be pursued when developing an extraction application, the term interoperability could be debated in this case, since the extraction application does not intend to match the functionality of the search engine, but rather use the data for entirely different purposes. However, such legal issues are jurisdictional and dependent on the country of use, and therefore their use should not be immediately discounted, but approached with caution and knowledge of current legislation (Cifuentes & Fitzgerald 2000).

Analysing assembly code and determining what libraries the application may be using for its data storage is a time consuming task. Code analysis is also a specialist skill and therefore not all law enforcement agencies will have capacity or expertise to perform the task.

Another major limitation to reverse engineering is the potential encryption of the data store of the search engine. Decompilation of the search engine and data store may reveal the use of encryption by the search engine to encrypt information it caches (Henrard, Englebert, Hick, Roland & Hainaut 1998). The complexity of determining the encryption algorithms used from the decompiled executables will in most cases be the primary motive to develop an extraction application with one of the other access methods. By utilising one of these other access methods, the issue of propriety encryption on the data store can be eliminated. Since the extraction application will be retrieving information that is normally intended for the end user, it will be in human readable form, as decryption will have been performed before returning the results of a query.

One means of mitigating the legal issues associated with the decompilation and ensuring the accuracy of the outcome is through the support of the initial application developer. Not only will this company understand the exact nature of the files, but may be able to provide an application for extracting information from them without altering files. To the researcher's knowledge, no company has publicly discussed such work.

2.1.2 Process 1B: Interfacing with Search Engine

All search operations bound for the desktop search application are handled by the search engine component of the application. For some desktop search applications, this search engine may be available for interfacing outside of the native user interface of the application. As identified in Figure 3, access may be available as an operating system service (such as a Windows service), as a standalone component with a standard interfacing method (such as a web server running on the local device) or the company may release a software development kit (SDK) for developing applications that is able to interface with the search application's executables. If the search engine component is designed as a service, then an SDK may be on offer from the application vendor or the programming community may have developed a third-party one.

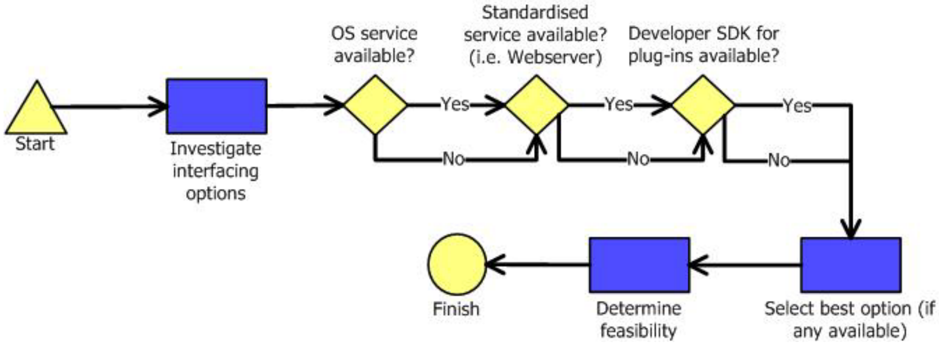
OS service available?
Standardised service available? (i.e. Webserver)
Developer SDK for plug-ins available?
Start
Investigate interfacing options
Yes
No
Yes
No
Yes
No
Finish
Determine feasibility
Select best option (if any available)

Figure 3 – Process model 1B: Interfacing with the search engine of the application.

If an SDK is available, an application can then be implemented utilising the SDK to retrieve information from the storage mechanism. If the search engine has a standardised interfacing method, then the standard should be discerned and an application should be developed following the protocols used to send and retrieve information.

Limitations of Interfacing with the Search Engine

The major limitation of this approach is the reliance on the original search application to extract and interpret data on behalf of the investigator. This presupposes two issues; that the search application will continue indexing whilst operational, and that, without direct access to the data store, there is the possibility that the desktop search application to filter or interpret results before presentation.

Since the executables of the search application are now responsible for retrieving requested information from the data store, the application needs to be running for the interaction to occur. This poses a potential threat to the forensic process, since the information retrieved could be modified by the application during retrieval. Extended testing in a controlled system setup would be required once the extraction application is developed. With the search application running, it also poses the threat that the application is free to continue its indexing policies. This could include deletion of cached items that no longer exist on the file system, meaning potential evidence could be lost. Some metadata could also be updated, such as timestamps, which would then loose some evidentiary value. By developing an application with this approach the risks would have to be assessed, managed and documented according to the rules of evidence, so that any evidence correlated from extracted information is admissible.

2.1.3 Process 1C: Interfacing with User Interface of Application via Macros

Although previously discussed that it is a time consuming task for an investigator to manually search for results from the application, the process can be partially automated through the use of scripting languages that are capable of replicating human interaction with system input devices through the use of macros. These scripts, coupled with screen-scraping can be used to simulate user input, and will attempt to capture search results returned from the application. Screen scraping is the process of capturing the output of a screen for use in another application and is traditionally used in computer systems re-engineering to link new systems into older legacy systems without the need to re-develop the old legacy system [2]. A proposed outline of events is given in Figure 4.

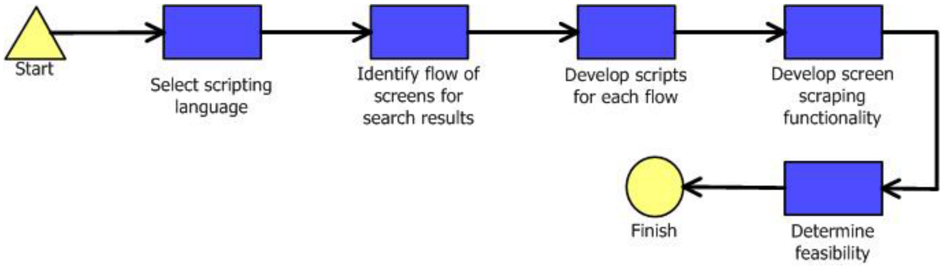
Start
Select scripting language
Identify flow of screens for search results
Develop scripts for each flow
Develop screen scraping functionality
Determine feasibility
Finish

Figure 4 - Process model 1C: Interfacing with user interface of desktop search engine

The developed application would require routines for navigating every screen that the target search application generates, and the order in which the routines flow, to match the application. This identification of screen flows and development of scripts to extract the information is shown in Figure 4. Once the navigation aspect is completed, data extraction methods (such as screen scraping) need to be developed for text, images and other data types supported by the application, also shown in Figure 4. Whilst the scripting languages are capable of selecting and copying text from interfaces, images will need to be handled via the incorporation of a screen capturing library into the extraction application. For other data types that may be returned by the application, intercepting and piping the output streams to file is one solution, however, this may not be necessary as the purpose of the application in many cases is to narrow the search field for manual inspection by an investigator.

2.1.3.1 Limitations of Interfacing with the User Interface

As with developing an application to extract information from the search engine via the actual executables of the search application, the information retrieved could be modified by the application. Again, if an extraction application is developed by utilising the user interface, the risks arising from this method will have to be managed and documented.

Developing a tool in this manner also requires a large amount of time, as every possible scenario of screen navigation has to be catered for. Information extraction will also be complex requiring a parser that is capable of determining different result layouts for the multitude of file types that are index-able by the application.

It is also noted that the issues related to running the desktop search application in order to interact with it are still present here – the desktop needs to be operational but it cannot still be indexing files, as doing so potentially reduces the integrity of the output.

2.2 Stage 2: Mitigate Forensic Risks of the Access Method

Once the method for accessing the data stored within the storage mechanism of the desktop search application has been determined, the forensic soundness of the method needs to be investigated. Since the focus of this stage is to mitigate forensic risks of the access method, following this process when developing an extraction application will partially answers the third sub-question of the research: “Are these access and extraction methods forensically sound, or can any risks to the forensic process be mitigated sufficiently?”. This stage involves the examination of key areas related to the desktop search application that can help to decrease the magnitude of any identified threats to the integrity of the information retrieved. As Figure 1 shows, each access method leads to a separate process for mitigating risks in stage 2. Whilst there are identical activities within the sub-processes of this stage there are also special cases that are to be considered for each access method and hence individual sub-processes were warranted.

2.2.1 Process 2A: Mitigate Reverse Engineering Risks

The only step of this process is the determining of legal issues that may be applicable to reverse engineering, and the subsequent resolving of these issues. As previously mentioned, the legality of using the data store of a desktop search engine for evidence is debated, as the usage may not be considered as allowing true interoperability between applications, since the function of the extraction tool is different from that of the search engine. The advice of a legal professional should be sought so that clarity on the matter can be reached, and the development of the extraction application can continue, or an alternative access method can be selected. If this legal issue is ignored, and the application is completed, then evidence obtained from the data store may be considered inadmissible since it was obtained through illegal means.

2.2.2 Process 2B: Mitigate Search Engine Interfacing Risks

The initial step of this process, as shown in Figure 5, is to discover which files and system entities (such as Windows registry entries) are used during the standard operation of the desktop search application. This can be achieved by using utilities that are part of the operating systems management features, or through the use of third party applications such as Filemon[1] and Regmon[2]. These utilities will reveal what files the application reads and writes to during start-up, whilst idle and when performing searches [3].

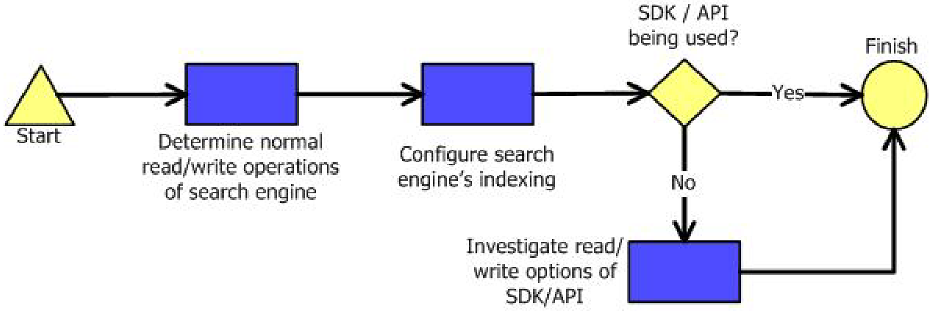

Figure 5 - Process model 2B: Mitigating forensic risks when interfacing with search engine.

Since the access method of interfacing with the search require it to be running, knowing which files are utilised by the application can help to manage the risk

[1] Available at http://www.sysinternals.com/Utilities/Filemon.html (October 2006)

[2] Available at http://www.sysinternals.com/Utilities/Regmon.html (October 2006)

to the integrity of the information retrieved. If the files are known, an operating system level change to the file permissions can be attempted to prevent any writing to occur.

The second step identified in Figure 6 step is the configuring of the search application's indexing options. Parallel to the reasons for determining which files the search application utilises, configuring the application can aid in minimising the risk to the integrity of the information retrieved due to the requirement of the application executables needed to be running.

Examination of the application's configuration menu will assist in determining what options could pose a threat to the information contained in the data store of the application. Most search applications have configuration options for deletion policies, specifying times when cached items are purged from the cache. If possible, deletion should be disabled or the time limit should be maximised. The indexing process itself should also be disabled, as files that are cached but have been modified since being cached could then be updated by the search application, overwriting valuable metadata such as timestamps.

The third step of the process defined in Figure 6 applies when developing an extraction application that will utilise an SDK, either supplied by the vendor or third-party, to interface with the search application. By thoroughly checking supplied SDK documentation and example code utilising the SDK, there may exist "read-only flags" for use in function calls to the search application. These would be beneficial in ensuring that any interactions between the developed extraction application and search application are guaranteed to leave all cached information unmodified in the data store.

2.2.3 Process 2C: Mitigate Risks Interfacing via Macros

This process, shown in Figure 7, has the same initial steps as process 2B, but has the third step of identifying the accuracy of the screen scraping methods employed for the access method. If the macro scripting language used for the access method does not have the ability to select and copy text from the search engine's user interface then an image capturing library should be in place to overcome this.

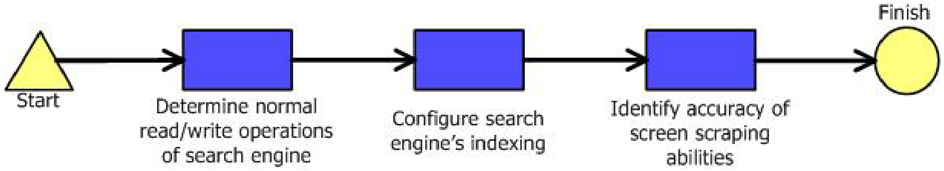
Finish
Start
Determine normal read/write operations of search engine
Configure search engine's indexing
Identify accuracy of screen scraping abilities

Figure 6 - Process model 2C: Mitigating risks when interfacing with user interface.

2.3 Stage 3: Determine Significance of Extractable Information

Once the method of accessing information has been selected and the forensic risks have been mitigated and documented, the third stage of this process is to determine the significance of the extractable data. Whilst it is known and appreciated that investigators must consider what information is to be extracted before beginning use of this process model, the particulars of what information is extractable is based upon the access method and what information is stored in the individual implementation, which is dependent on which application is used. What information is sought is therefore reliant on these factors, which would not be known before these stages.

In some circumstances, it may be ambiguous as to what the actual extracted information is related to. If the true meaning of the extracted information is not known, and its significance is incorrectly identified, then false conclusions in a forensic investigation may be drawn. This could lead to the discrediting of an investigator's testimony in court. If the significance is identified properly, this risk to the forensic process can be mitigated. In this stage, all three access approaches merge together, shown in Figure 7, as determining the meaning of ambiguous information draws upon information sources used by all three methods.

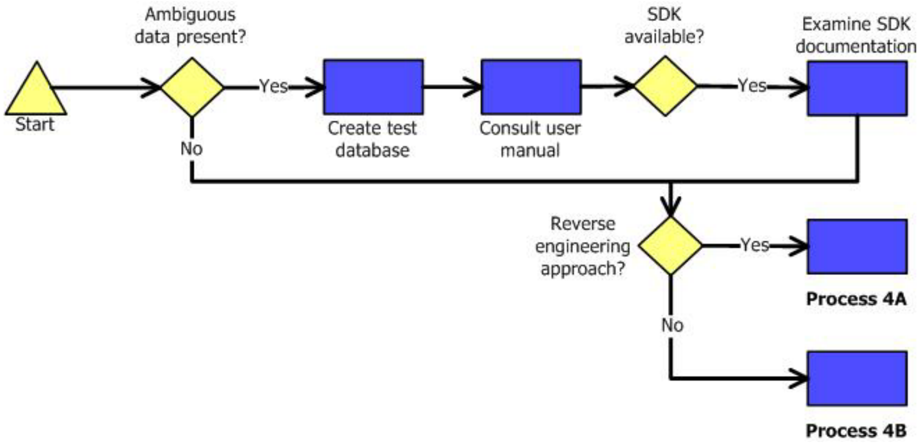

Figure 7 - Process model 3A: Determining the significance of extractable information.

2.3.1 Process 3A: Discover Significance of Extractable Information

If it is ascertained that ambiguous data is present and the meaning of the data can not be determined, the first action is to develop a controlled test database with the desktop search engine, as shown in Figure 7. Creating this test database is similar to the approach applied in process 1A (Figure 2) to determine the schema of the data store. By gradually indexing selected files of specific file types and documenting the search results returned by the desktop search engine, conclusions can be drawn about how the desktop search engine may alter some metadata of the indexed files such as timestamps. Timestamps can be ambiguous if they are not descriptively labelled internally by the search engine and the time value they contain could be in reference to the creation of the cached file in relation to the file system, or could be referring to the date the file was indexed by the application and stored in the cache.

2.4 Stage 4: Data Extraction

The fourth stage of the model deals with the actual extraction of data from the desktop search engine's parallel storage mechanism. Following the processes of this stage when developing an extraction application answers the second sub-question of this research: "Can cached information be data mined from parallel storage mechanisms created by desktop search engines?". By developing a method that can successfully extract information from a desktop search engine, the extracted information can then be used in stage 5 to search for evidence.

2.4.1 Process 4A: Data Extraction Method for Reverse Engineering Approach

Data extraction from desktop search engines where a reverse engineering approach has been selected is the most straightforward approach, consisting of only one option, to directly extract data. Since the structure of the data store is known due to reverse engineering, the extraction application can be developed to read directly from the files containing the cached information. This in effect simulates the approach the actual search engine would use to retrieve the information for the user.

2.4.2 Process 4B: Data Extraction Method for Interfacing with Search Engine

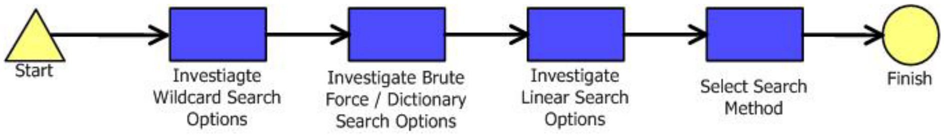
Start
Investiagte Wildcard Search Options
Investigate Brute Force / Dictionary Search Options
Investigate Linear Search Options
Select Search Method
Finish

Figure 8 - Process model 4B: Data extraction method for interfacing with search engine.

For an application that has not been reverse engineered, a data mining approach to information extraction will need to be selected. Since the information can not be extracted directly from the storage mechanism, but via the application itself, an automated process that queries the application is needed.

2.4.2.1 Wildcard Search

A wildcard search is the first option to consider, as seen in Figure 8. This is the simplest method of data mining the search application; however, it is not applicable to all search applications. By creating a query that utilises a wildcard operator, all results stored in the search applications data store will be returned. The exact formatting of the wildcard search term will vary between search applications, and will have been determined in stage three. Commonly, the * character is used as the wildcard operator and in some cases, may be used in conjunction with file types (i.e. *.txt).

2.4.2.2 Brute Force / Dictionary Search

This method of data mining the search application is more involved than wildcard searching but should be applicable to most search applications. Figure 8 shows this search option as the second option to be considered in process 4B. Dictionary search is a similar concept to a dictionary attack, where a dictionary of words are used by themselves, or concatenated together, in an attempt to gain access to a computer system by an attacker [4, p. 407]. By using a dictionary of search terms, brute force is applied to the search application and a large magnitude of search requests are made, many returning results previously returned. The dictionary is a plain text file, or series of plain text files, read by the extraction application. The files contain a list of words in the common language of the locale of the law enforcement agency (i.e. English, French, Spanish, etc.), a list of common misspellings for words from the locale, a list of file types and a list of case related terms such as suspect names and victim names. Depending on the scale of the crime and the people involved, multiple language files may be needed for searching. The extraction application would iterate through the words listed in the dictionary files and format them into queries for the search application, where it is processed and a result set is returned. The result set returned by each query may contain information on a cached file, already returned by a previous query. This is due to multiple search terms being present in the same cached file. This problem is dealt with in stage five.

2.4.2.3 Linear Search

This last devised data mining approach is not applicable to all desktop search applications. It is intended for use where as part of the metadata for a cached item, there exists a unique identifier. This identifier is usually used by the application to retrieve the item from the storage mechanism. The first identifier used by the application in most situations will be 0, but depending on the implementation and indexing operations that have occurred, it may be greater than 0 as the item at 0 may have been purged from the cache and the identifier 0 is not used again. The last identifier can be determined by checking the statistics of the application to acquire the number of items that have been cached. However as with the first identifier, this may not provide a true indication as to the last identifier due to indexing operations. If this is the case, then using an arbitrary large number much greater in magnitude than the number of cached items can be substituted. Using the first identifier value and last identifier value in an iterative loop, queries can be generated based on the identifiers and individual items will be returned. If the initial and final identifiers are not accurate due to the aforementioned reasons, returned results will have to be checked for empty or null values.

2.4.2.4 Selecting Search Method

The last step of process 4B is to select the search method after evaluating all the possible options in the previous steps. A combination of search methods may be selected in this process if it is decided that a single method will not return enough information in the query results.

2.5 Stage 5: Storing and Querying Extracted Data

Once the method for data mining has been selected, the fifth stage determines the method for storing the extracted information, so that it can be queried by an investigator or other forensic software tool to determine if there is any information of evidentiary value. This stage of the model attempts to answer the last part of the main research question: "Is it possible to extract information from desktop search applications parallel storage mechanisms, in a forensically sound manner, so that it can be used as evidence?" By developing the ability to query extracted information, it can then be analysed for evidence and used during a forensic investigation.

2.5.1 Process 5A: Storing and Querying Data from Reverse Engineering

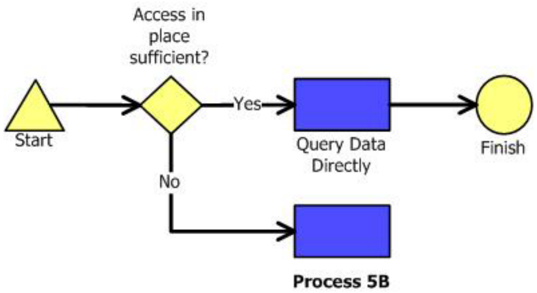

Figure 9 - Process model 5A: Storing and querying data from reverse engineering.

The first option to consider when storing and querying retrieved data from a reverse engineering approach, is whether or not querying the data in place is sufficient. The reverse engineering process will allow the information to be read directly from the storage mechanisms of desktop search engines. Therefore, the requirement to store extracted data may be redundant if a method for querying the data in place is available. If such a querying method does not exist, process 5B should be followed to select a storage and query method for extracted data. This process is modelled in Figure 9.

2.5.2 Process 5B: Storing and Querying Extracted Data from Other Access Methods

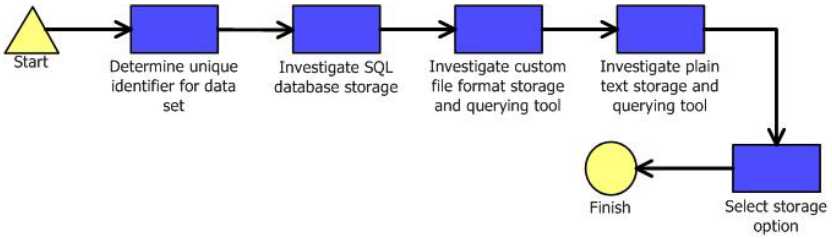
Start
Determine unique identifier for data set
Investigate SQL database storage
Investigate custom file format storage and querying tool
Investigate plain text storage and querying tool
Select storage option
Finish

Figure 10 - Process model 5B: Storing and querying extracted data from other access methods.

For non-reverse engineering based extraction applications, or when accessing the data store in place is insufficient for a reverse engineered approach, there are three conceived methods for storing and querying retrieved information as shown in Figure 10. The first step for all three is to determine a unique identifier for each search result. If the search method utilises a linear search through the use of item identifiers supplied by the search application, then this is likely to be the best unique identifier. If this is not available, a file's URL which will either be filename and path or URL of the website (if the search application caches visited sites), can be used to create the unique identifier, since two files with the exact same name and path can not exist.

There are three alternatives, each with its own advantages and disadvantages; Database, a custom file format, and outputting as plain text. The advantage of a database is that all data would be in an accessible format optimised for searching, but the disadvantage is that many forensic analysis tools may not be able to directly interface with a database for searching purposes. The use of a custom file format may be preferable for some situations, particularly where the file format is interoperable with other custom tools, however, the disadvantages are a lack of interoperability with known standards and, as information is not stored in a flat format, it might not be directly searchable by other applications. Plain text is searchable, but loses the speed and ubiquity of a database-stored system. The use of XML to provide structure to such a file would mitigate or reduce many of these disadvantages.

A solution to avoiding the disadvantages of each of these file formats would be to save any extracted information in multiple ways.

2.6 Stage 6: Verifying the Forensic Soundness of the Developed Extraction Application

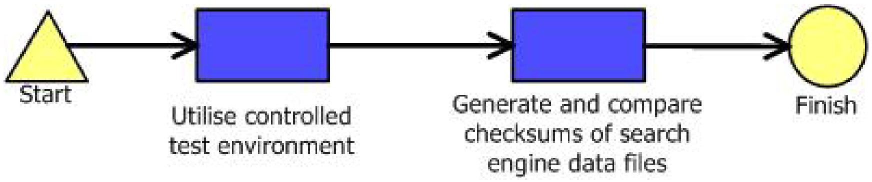
Start
Utilise controlled test environment
Generate and compare checksums of search engine data files
Finish

Figure 11 - Process model 6A: Verifying the forensic soundness of the developed extraction application.

With the functionality of the developed extraction application completed, the final step is to verify that the operations of the extraction application are forensically sound. This stage answers the main research question by proving that the developed extraction application does not modify the information read from the search engine data store. This validation is performed in two parts: creating a test environment to see what information is modified by the search application upon indexing and comparing generated hash values of the data store files from before and after the extraction application operates.

2.6.1 Controlled Testing Environment

This first test utilises a ‘garbage in, garbage out’ philosophy to determine if the information retrieved by extraction process is modified in any way from the original file that was cached [5, p. 13.3]. Creating a test environment will vary between search applications, but the main goal is to limit the indexing performed by the application to one test file of which the contents and metadata are known. With the search application configured and the file created, the indexing process should be started by the application. When completed, the developed extraction application should be executed to retrieve all the information available, which will be the one file.

Retrieved information then needs to be compared against the original file. Any differences in metadata or cached content need to be documented, and explanations for the differences need to be concluded. Metadata that differs may already have been documented in stage 3 of the process model, when determining the significance of the data. These changes should already be accounted for in the expected output of the retrieval application. Any unexpected modifications may require debugging of the developed application to resolve the problem, and if not possible due to utilisation of third-party code, the changes and their impacts on the integrity and meaning of possible evidence need to be documented.

2.6.2 Checksum Comparison

This essential test ensures that the developed extraction application does not modify any of the files of the search application’s data store. If the application modifies the cache files then it can be claimed that the evidence has been tampered with to pervert the course of justice, since incriminating information could have been injected into the retrieved data. Since the use of MD5 and SHA1 (either individually or in combination) are accepted methods of ensuring duplications of seized media are exact copies. Due to the near-impossible collision chance when generating unique hash values, they provide an accurate test to ensure the retrieval application does not modify the search application’s files (Carrier & Spafford 2004). Forensically, taking hashes of all database files used by applications before and after analysis may ensure that the act of accessing them (using the different methods as described within the process model) has not altered or tainted these in any way. The use of hash values is especially important in situations as described by [6] where a copy of Google Desktop Search was run on the database files from another machine, and examiners needed to ensure that the application itself was not updating files and indexing new data from the investigator’s machine.

3. FUTURE WORK AND CONCLUSION

3.1 Future Research

The process model developed is the first attempt at providing a series of stages to consider when developing an evidence extraction application for desktop search engines. As such, it is a high level focus on the major issues faced when developing said applications. The scope of the research limited the scope of initial model, which allows greater future work into reviewing and extending of the model. As new desktop search engines are released by developers, extraction applications will need to be developed by applying the model. Through the application of this model to new desktop search engines, it may be found that other access methods and associated stages can be added to the model, to extend the usefulness of the model for future use on desktop search engines. By developing new extraction applications with the model, new overlooked sub-processes can be added to the model derived from anecdotal evidence obtained during development of the extraction applications. These further contributions will aid in expedient future development of extraction applications when following the process model.

However, as noted by Turnbull et. al [6], the release of the Windows Vista operating systems, with built in comprehensive desktop search functionalities, may reduce the need for the model developed in this research. If powerful desktop search capabilities are provided by default with the popular operating system, the numbers of users of alternative products may be impacted. As it currently stands though, there are a variety of desktop search applications thriving, and implementing an extraction tool for each one can be beneficial for law enforcement agencies to have access to every piece of evidence that is possible.

3.2 Conclusion

Forensic computing is a reactive field that must keep up to date with the latest consumer technologies and software. Criminals find novel ways to use these technologies and if left unaddressed, they can conduct illicit business without fear of prosecution (Kruse II, Heiser, 2002; Marcella & Greenfield 2002). One such category of consumer software is desktop search engines. Functioning like Internet search engines, they index files on a hard disk to make locating the files easy. However previous research found that there could be forensic value in the information indexed by desktop search engines. Desktop search engines could contain cached copies of deleted files, which could be of great use to forensic investigators if a seized computer has a desktop search application installed. The research suggested a method of accessing one brand of desktop search engine, Google Desktop Search, but noted that better methods should be devised and for a variety of applications.

This research aimed to devise these better methods for extracting information from desktop search applications. The key output of this research is a process model that details the stages of development that should be undertaken when developing a desktop search evidence extraction application. The devised model focused on 6 stages of the development process, which define all methods of extracting data from this valuable and largely forgotten source.

REFERENCES

Carrier B & Spafford E, 2004, 'An Event-Based Digital Forensic Investigation Framework', *Digital Forensic Research Workshop*, August 11-13, 2004, Baltimore, Maryland, USA

Cifuentes C & Fitzgerald A, 2000, 'The legal status of reverse engineering of computer software', *Annals of Software Engineering*, Vol 9 no 1-4, May 2000, Springer Netherlands

Cole, B 2005, 'Search engines tackle the desktop', *Computer,* vol. 38, no. 3, pp. 14-17.

Comella-Dorda, S, Wallnau, K, Seacord, RC & Robert, J 2000, 'A survey of black-box modernization approaches for information systems'. *International Conference on Software Maintenance*, 11-14 Oct 2000, San Jose, California USA

Conklin, WA, White, GB, Cothren, C, Williams, D & Davis, RL 2004, *Principles of Computer Security*, McGraw-Hill, New York.

Henrard J, Englebert V, Hick J-M, Roland D & Hainaut J-L, 1998, 'Program understanding in databases reverse engineering', chapter of *'Database and Expert Systems Applications*', Springer Berlin

Jarzabek, S & Woon, I 1997, 'Towards a precise description of reverse engineering methods and tools', *First Euromicro Conference on Software*

Maintenance and Reengineering, 17-19 Mar 1997, Berlin, Germany

Kruse II W, Heiser J, 2002, *Computer Forensics; incident response essentials*, Addison-Wesley, Boston, USA

Marcella A & Greenfield R, 2002, *Cyber Forensics; a field guide for collecting, examining, and preserving evidence of computer crimes*, Auerbach Publications, New York, USA

Microsoft 2006, *FileMon for Windows v7.04*, Microsoft TechNet, viewed 2 Nov 2006, <http://www.microsoft.com/technet/sysinternals/utilities/Filemon.mspx>.

Reith M, Carr C & Gunsch, G, 2002, 'An Examination of Digital Forensic Models', *International Journal of Digital Evidence*, Vol. 1, no 3, available online at www.ijde.org

Shelly, GB, Cashman, TJ & Vermaat, ME 2001, *Discovering Computers 2002: Concepts for a Digital World*, Thomson Learning, Boston.

Smith, I 2004, *Cost of Hard Drive Space*, viewed 22 April 2006, <http://www.littletechshoppe.com/ns1625/winchest.html>.

Turnbull, B, Blundell, B & Slay, J 2006, 'Google Desktop as a Source of Digital Evidence', *International Journal of Digital Evidence,* vol. 5, no. 1.