Chapter 4

EXTRACTING EVIDENCE USING GOOGLE DESKTOP SEARCH

Timothy Pavlic, Jill Slay and Benjamin Turnbull

Abstract Desktop search applications have improved dramatically over the last three years, evolving from time-consuming search applications to instantaneous search tools that rely extensively on pre-cached data. This paper investigates the extraction of pre-cached data for forensic purposes, drawing on earlier work to automate the process. The result is a proof-of-concept application called Google Desktop Search Evidence Collector (GDSEC), which interfaces with Google Desktop Search to convert data from Google's proprietary format to one that is amenable to offline analysis.

Keywords: Google Desktop Search, evidence extraction

1. Introduction

Current desktop search utilities such as Windows Desktop Search, Google Desktop Search and Yahoo! Desktop Search differ from earlier tools in that user data is replicated and stored independently [1, 10]. Unlike the older systems that searched mounted volumes on-the-fly, the newer systems search pre-built databases, accelerating the search for user data with only a nominal increase in hard disk storage [5]. The replication of data in a search application has potential forensic applications – data stored independently within a desktop search application database often remains after the original file is deleted.

In previous work [9], we examined the forensic possibilities of data stored within Google Desktop Search; in particular, we discussed the extraction of text from deleted word processing documents, thumbnails from deleted image files and the cache for HTTPS sessions. However, the format of the extracted data files does not allow for simple interpretation and analysis; therefore, the only sure method of extracting data was via the search application interface. We also showed that it was possible to maintain the forensic integrity of the extracted data by disabling certain components of the Google Desktop Search application. But this data could only be accessed using manual keyword searches submitted via the application interface.

This paper presents a more efficient technique for extracting data from desktop search utilities. The discussion focuses on Google Desktop Search, but the concepts are applicable to other desktop search applications. The resulting proof-of-concept application, Google Desktop Search Evidence Collector (GDSEC), automates the data extraction process and enables investigators to copy data from Google Desktop Search files in a forensically-sound manner without having to conduct manually searches using the interface.

2. Google Desktop Search

Google Desktop Search was released in 2004. The original version was designed only for Windows XP. Currently, versions are available for Windows Vista, Linux and Mac OS X.

The Windows version of Google Desktop Search was designed for single users. However, when Google Desktop Search was installed and run by an administrator in a multi-user environment, the program would index and search all files regardless of their ownership. This potential security flaw received widespread media coverage [6, 8].

Security concerns have been raised about the integration of Google Desktop Search with Google's Internet search engine, but these vulnerabilities have not been exploited [3]. Attention has also focused on the privacy issues related to Google Desktop Search's approach of copying local data to external machines for faster search [2].

This work focuses exclusively on the Windows-based implementation of Google Desktop Search, the most widely used application. The Macintosh and Linux versions of Google Desktop Search operate very differently. Note that Google Desktop Search is executed under Windows NT/2000 and later versions because it uses libraries that are available only in more recent platforms.

Google Desktop Search has three executables, `GoogleDesktopIndex.exe`, `GoogleDesktopSearch.exe` and `GoogleDesktopCrawl.exe`. The `GoogleDesktopSearch.exe` executable is the main program of the search suite; it operates by setting up an HTTP server on local port 4664 and controls all user interactions. The `GoogleDesktopCrawl.exe` program traverses the file structure on the hard disk and reports changes to `GoogleDesktopIndex.exe`. `GoogleDesktopIndex.exe` interfaces with persistent storage files, `GoogleDesktopCrawl.exe` and the Microsoft Indexing Service. The Indexing Service sends notifications when files are changed; this information is used by `GoogleDesktopCrawl.exe` to determine the files that may require updating. Note that Google Desktop Search creates a registry key at `HKEY_USERS\SID\Software\Google\Google Desktop` where `SID` is the unique user SID. Several options are provided, including locations for file storage.

The Google desktop searching utility allows third-party additions to its software, which facilitates the customization of search parameters. However, third-party additions must use the Google API to customize all settings via the Google program, meaning that direct communication with the database that stores files is not permitted. Google provides a software development kit (SDK) for Google Desktop Search that contains five APIs. The SDK is based on the COM model, allowing any programming language supporting COM to be used to develop plug-ins that utilize the APIs.

Google Desktop Search supports the ability to encrypt the data store that contains cached items. However, further examination has revealed that the application merely invokes Windows NTFS encryption for the folder containing user data. Since the computer is being examined for forensic purposes, we assume that some measure of access is guaranteed.

3. Google Desktop Search Evidence Collector

This section describes the Google Desktop Search Evidence Collector (GDSEC) tool. It highlights the methods developed for accessing and extracting data, and for storing results. Also, it discusses how evidence collection can be conducted in a forensically-sound manner.

3.1 Accessing Data

Several methods are available for accessing data from desktop search applications. The ordering of access methods from a forensic integrity perspective (best to worst) are:

- Accessing files directly.
- Accessing files using an interpreter.
- Extracting data using API mechanisms provided by the original application.
- Extracting data using the API.
- Searching for data using the API.

Directly accessing and interpreting any files created by a desktop search utility is the preferred method from a forensic perspective because it ensures that all the stored information is available without using an intermediate system. In addition, the data is much more easily extracted using existing digital forensic tools.

The issue with accessing files directly or using an interpreter is that it is difficult to determine the format of the files, which is required to ensure that all the data can be extracted in its original form. Of course, the format can be reverse engineered, but unless the software developer is involved, reverse engineering may have to be performed repeatedly because the format often changes between releases.

Extracting data via an API is less preferable than accessing the data directly. Using an API requires the original Desktop Search program to execute in a forensically-sound manner. The primary advantage is that it permits more thorough extraction of data from the given file format than screen scraping or manual searching.

Our previous research [9] was unsuccessful at determining the file structure to an adequate level of detail. We were, therefore, unable to access the data directly from within Google Desktop Search. However, the following method can be used to access file data in a forensically-sound manner:

Data Access Method

1 Copy the Google Desktop Search storage folder (default is `c:\Documents and Settings\username\Local Settings\Application Data\Google\Google Desktop Search`) from the source machine to the Google Desktop Search folder on the analysis machine.

2 Rename the file `GoogleDesktopCrawl.exe` to `GoogleDesktopCrawl.exe2` on the analysis machine; this prevents the file from loading.

3 Open the Google Desktop Search program and ensure that no email programs are loaded on the analysis machine.

4 After the Google Desktop Search program has loaded on the analysis machine, navigate to the storage folder and change the file attributes of the files to read-only; this allows the Google Desktop Search program to close without editing any files.

This data access method is time consuming; the only options are to manually search for keywords using the user interface or to screen scrape the information to another search tool. In either case, there is no means to ensure that all the data has been extracted. The problem is acerbated by the fact that Google Desktop Search performs a strict search, i.e., the entire word being searched must be present for a hit to occur (searching for "bana" does not return results with "banana").

As mentioned earlier, Google Desktop Search provides several APIs to enable third-party applications to be used for data search and collection. Also of interest is Google Desktop's interface mechanism, which uses a web interface on a local host web server; this web server receives all user queries and functions as the main user interface to the application. Since a web server is a common service with a standardized access method, it provides another method for accessing data maintained within Google Desktop's storage mechanism. Thus, an HTTP-based extraction application can be used to submit queries to Google Desktop Search and retrieve results.

Extracting information from Google Desktop Search via an HTTP server was deemed to be the most effective method. Several APIs are available that enable data to be retrieved in raw HTML or XML formats. Our GDSEC prototype uses GDAPI, a Java-based API for querying the Google Desktop Search web server.

3.2 Analyzing Output Data

Google Desktop Search was used on a test database containing a variety of file types. Our analysis revealed that Google Desktop Search records file-type-specific metadata (e.g., movie lengths and bit rates, and image resolutions) in a common set of fields, which means that the value of the fields are ambiguous.

The SDK documentation supplied by Google [4] describes an option for viewing search results in an XML format. Specifically, by appending the string `&format=xml` to the end of a search result page, the results can be viewed as a formatted XML page; this helped us to understand the data that is retrieved for each filetype. Every search result has a standard set of XML elements. File-specific metadata is stored in the *snippet* element as a single string, which could be parsed if required.

Google Desktop Search (version 2) enables items to be viewed in a timeline format, which lists the files indexed on each day. Implementing this feature requires metadata (e.g., timestamps) to be stored. A *time* element (with date and time information) was discovered in the XML search results. Examination of the SDK revealed it to be the date/time that the item was indexed and cached by Google Desktop Search, rather than a timestamp extracted from the computer's file system metadata (e.g., file creation time or time of last modification).

3.3 Extracting Data

Google Desktop Search does not offer a wildcard search feature. A linear search requires an identifier for the indexed entries. However, although Google Desktop Search has identifiers, we were unable to format search requests based on item identifiers. In any case, item identifiers would have to be discovered by issuing queries before they could be used in queries; this doubles the computational requirements.

Consequently, our experiments used brute force search with a dictionary containing a small set of words designed to test the ability of the application to handle query results that contained references to files discovered by previous queries. The keywords in the dictionary were chosen to correspond to the test files used to evaluate the application and validate the extraction process.

3.4 Storing and Querying Extracted Data

GDSEC was developed as a proof-of-concept application for extracting data. Consequently, the results are simply stored in text files. The search application initially stores the retrieved results in memory as result objects before writing them to files. Each result object is simply an encapsulated collection of strings and integers used to represent every XML element available from a Google Desktop Search query result. A red-black binary tree is used to manage all the result objects with the *url* XML element (which points to a file on the file system or the Internet) of the search result used as the unique identifier. After a query is issued, result objects are created for each result and an attempt is made to add them to the tree based on their URLs. A file that has already been discovered in a previous query is not added to the tree.

The text files generated as output contain a list of all the elements extracted from the XML results along with the information related to the elements. Cached content is also appended to the end of the text output. The file names of output files are based on the last component of the URL (usually the file name and extension). For cached files with the same name that reside in different directories, an extra numerical character is appended to the file extensions of the output files to make them unique. Illegal file name characters such as "?" that appear in a URL (due to web pages with parameters) are replaced with the "_" character. The text files are generated in a separate folder on the file system. Each folder is given a unique name by using its creation time; this ensures that all subsequent output requests are written to different folders.

Table 1. Google Desktop Search data.

Filename	**Match**	**Filename**	**Match**
`dbc2e.ht1`	Yes	`Dbdam`	Yes
`Dbdao`	Yes	`Dbeam`	Yes
`Dbeao`	Yes	`Dbm`	Yes
`dbu2d.ht1`	Yes	`dbvm.cf1`	Yes
`dbvmh.ht1`	Yes	`fii.cf1`	Yes
`Fiid`	Yes	`fiih.ht1`	Yes
`Hp`	Yes	`hpt2i.ht1`	Yes
`rpm.cf1`	Yes	`rpm1m.cf1`	Yes
`rpm1mh.ht1`	Yes	`rpmh.ht1`	Yes
`uinfo_data`	No		

3.5 Verifying Forensic Soundness

It is important to verify that the GDSEC application is forensically sound and that the extracted data can be used as evidence. The verification process used a controlled indexing test and a hash value comparison.

The first test used a controlled indexing environment to verify that GDSEC retrieved data without modifying it. A partition was created on a test system with multiple files named `EVIDENCE.txt` containing the text string "criminal activity." Google Desktop Search was configured to only index this partition. After the indexing was completed, GDSEC was launched with instructions to perform the dictionary search and to write all the retrieved items to a text file. This text file contained all the XML search results and the cached content retrieved from the cache URL. The cached content that was recovered contained the strings "criminal activity," which proved that no data was modified during extraction.

Next, it was necessary to verify that no other data was modified during the extraction process. As part of the controlled indexing test, when the file was indexed, Google Desktop Search was terminated and MD5 hash values [7] were generated for all the data files used by the search application. The application was then re-executed and the remainder of the controlled indexing test was performed. When this was completed, Google Desktop Search was once again terminated and a second set of MD5 hash values was generated for the data files.

Table 1 shows the results of the hash value matching test. Only file `uinfo_data` was altered; all the other files had the same hash values before and after extraction and were, therefore, unaffected. The file `uinfo_data` stores user information about the search application and no actual cached content. Therefore, although this file was altered by

Google Desktop Search, the loss of integrity is known and explained, and does not impact the extraction of cached content.

4. Conclusions

Google Desktop Search Evidence Collector (GDSEC) is a prototype tool designed to collect data from the files used by Google Desktop Search in a forensically-sound manner. The current version of GDSEC interacts with Google Desktop Search to extract information. However, the preferred extraction technique from a forensic point of view is for the application to directly access files; future research will investigate this issue with the goal of implementing the capability in GDSEC. Other avenues for improvement include interfacing GDSEC with an SQL database to provide the ability to conduct additional searches of the retrieved information and implementing routines to retrieve cached content for items that have multiple cached versions (e.g., websites that are visited frequently).

References

[1] B. Cole, Search engines tackle the desktop, *IEEE Computer*, vol. 38(3), pp. 14–17, 2005.

[2] Electronic Frontier Foundation, Google copies your hard drive – Government smiles in anticipation (www.eff.org/press/archives/2006/02/09), February 9, 2006.

[3] T. Espiner, Google admits Desktop security risk, ZDNet UK (news.zdnet.co.uk/internet/security/0,1000000189,39253447,00.htm), February 20, 2006.

[4] Google, Google Desktop (desktop.google.com).

[5] S. Olsen, Google unveils Desktop Search, CNET News.com (www.news.com/2100-1024_3-5408765.html), October 14, 2004.

[6] B. Posey, Working with NTFS encryption (www.brienposey.com/kb/working_with_ntfs_encryption.asp), 2002.

[7] R. Rivest, MD5 message-digest algorithm, RFC 1321 (www.ietf.org/rfc/rfc1321.txt), 1992.

[8] T. Spring, Google Desktop Search: Security threat? PC World (blogs.pcworld.com/staffblog/archives/000264.html), 2004.

[9] B. Turnbull, B. Blundell and J. Slay, Google Desktop as a source of digital evidence, *International Journal of Digital Evidence*, vol. 5(1), 2006.

[10] X1 Technologies, X1 Desktop Search (pro.x1.com/?source=Yahoo).