CLOUD SIGNATURE: AN APPLICATION TO DETECT
CLOUD-COMPUTING APPLICATIONS ARTIFACTS

BY

JEREMY KOPPEN

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

UNIVERSITY OF RHODE ISLAND
2012
MASTER OF SCIENCE THESIS
OF
JEREMY KOPPEN

APPROVED:
Thesis Committee:
Major Professor          Dr. Victor Fay-Wolfe
Dr. Timothy Henry
Dr. Haibo He
Nasser H. Zawia
DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND
2012
ABSTRACT

During the process of investigating a computing device, law enforcement officers must analyze the device for any data that may be of evidentiary value. With the emergence of cloud-computing, the data may no longer be present on the physical device, but the data may exist on a remote server. Currently, there are no existing tools that are capable of detecting the presence of cloud-computing applications on seized devices. The lack of tools to support the presence of cloud-computing applications could mean that data with evidentiary value is not being detected.

This thesis project designed and implemented a prototype tool that automatically detects the presence of supported cloud-computing based applications, and then interprets the data so that it can be presented to a law enforcement officer in a format that facilitates their forming a warrant to the cloud service provider for the evidence.
ACKNOWLEDGMENTS

I would like to thank my advisor and friend Dr. Vic Fay-Wolfe. Throughout my entire graduate school career he has provided me with multiple opportunities and unlimited support. I honestly could not have asked for a better person to lead me through this process and the last two years of my life.

I also would like to thank Kevin Bryan and Sean Alvarez for their advice and guidance in my research, Travis Scarborough and James Brigden for their research assistance, and every other member of the Digital Forensic Center for their never-ending support. I would like to say thank you to Lorraine Berube and Catherine Robinson for their guidance over the past two years. They both have always been there to assist me with any question about school or the East Coast in general. They are truly amazing people.

In addition I would like to thank all of my family and friends for continuously motivating me to never give up. It was always refreshing to know whatever problems I would encounter; I could always call my local Iowans for support. I would like to explicitly thank Katie Felix for her amazing editing skills. The amount of time she put into helping me with this thesis is unimaginable, and it is impossible to thank her enough.
# TABLE OF CONTENTS

ABSTRACT ................................................................................................................................. ii
ACKNOWLEDGMENTS .................................................................................................................. iii
TABLE OF CONTENTS ................................................................................................................ iv
LIST OF TABLES .......................................................................................................................... vi
LIST OF FIGURES ........................................................................................................................ vii
CHAPTER 1: INTRODUCTION ...................................................................................................... 1
  1.1 Statement of Problem ........................................................................................................... 1
  1.2 Goals .................................................................................................................................... 2
    1.2.1 Requirements of Cloud Signature ................................................................................. 3
  1.3 Microsoft .NET Framework .................................................................................................. 6
  1.4 Organization of Thesis ......................................................................................................... 7
CHAPTER 2: REVIEW OF LITERATURE ....................................................................................... 9
  2.1 Definition .............................................................................................................................. 9
  2.2 WetStone Technologies’ Gargoyle Investigator ................................................................. 10
  2.3 JADsoftware Incorporated’s Internet Evidence Finder ....................................................... 10
  2.4 ATC - NY’s Cyber Marshal Dropbox Reader .................................................................... 11
  2.5 Digital Detective’s Net Analysis ......................................................................................... 11
  2.6 ForensicSoft’s SAFE Block ................................................................................................ 11
CHAPTER 3: METHODOLOGY .................................................................................................... 12
  3.1 Web Based Cloud-Computing Applications ...................................................................... 13
  3.2 Client Based Cloud-Computing Applications .................................................................... 20
    3.2.1 Dropbox Client Application .......................................................................................... 21
    3.2.2 Windows Live Mesh Client Application ....................................................................... 26
CHAPTER 4: FINDINGS .................................................................................................................. 32
  4.1 Triage Forensic Test Findings ............................................................................................. 34
    4.1.1 Windows XP 32-Bit Triage ........................................................................................ 35
4.1.2 Windows XP 64-Bit Triage ................................................................. 36
4.1.3 Windows 7 32-Bit Triage ................................................................. 37
4.1.4 Windows 7 64-Bit Triage ................................................................. 38
4.2 Disk Level Forensics Testing Findings .................................................. 39
  4.2.1 Windows XP 32-Bit Disk Image ....................................................... 40
  4.2.2 Windows XP 64-Bit Disk Image ....................................................... 41
  4.2.3 Windows 7 32-Bit Disk Image ......................................................... 42
  4.2.4 Windows 7 64-Bit Disk Image ......................................................... 43
4.3 Requirements Findings ......................................................................... 44
4.4 Next Steps ............................................................................................. 46
CHAPTER 5: CONCLUSION ........................................................................ 49
BIBLIOGRAPHY .......................................................................................... 51
# LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1. Windows 7 index.dat Storage Directories</td>
<td>14</td>
</tr>
<tr>
<td>Table 2. Windows XP index.dat Storage Directories</td>
<td>14</td>
</tr>
<tr>
<td>Table 3. index.dat File Structure</td>
<td>16</td>
</tr>
<tr>
<td>Table 4. Keyword Object Structure</td>
<td>19</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1. Cloud Signature User Interface</td>
<td>12</td>
</tr>
<tr>
<td>Figure 2. Bytes Represented In A URL Entry</td>
<td>17</td>
</tr>
<tr>
<td>Figure 3. Parsed URL Bytes</td>
<td>17</td>
</tr>
<tr>
<td>Figure 4. Keyword Text File</td>
<td>18</td>
</tr>
<tr>
<td>Figure 5. Web Based Report Title Page</td>
<td>19</td>
</tr>
<tr>
<td>Figure 6. Web Based Results Without File Name</td>
<td>20</td>
</tr>
<tr>
<td>Figure 7. Web Based Results With File Name</td>
<td>20</td>
</tr>
<tr>
<td>Figure 8. Config.db</td>
<td>22</td>
</tr>
<tr>
<td>Figure 9. File_Journal Table From File_Cache.db</td>
<td>23</td>
</tr>
<tr>
<td>Figure 10. Dropbox Config.db Report</td>
<td>24</td>
</tr>
<tr>
<td>Figure 11. Dropbox Report For Directories</td>
<td>25</td>
</tr>
<tr>
<td>Figure 12. Dropbox Report For Files</td>
<td>25</td>
</tr>
<tr>
<td>Figure 13. Windows Live Mesh Directory File Tree</td>
<td>28</td>
</tr>
<tr>
<td>Figure 14. Windows Live Mesh User Table</td>
<td>29</td>
</tr>
<tr>
<td>Figure 15. Windows Live Mesh MeshObject Table</td>
<td>30</td>
</tr>
<tr>
<td>Figure 16. Windows Live Mesh DataEntity_Enclosure</td>
<td>30</td>
</tr>
<tr>
<td>Figure 17. Windows Live Mesh Report</td>
<td>31</td>
</tr>
<tr>
<td>Figure 18. Main Title Page</td>
<td>34</td>
</tr>
<tr>
<td>Figure 19. Windows XP 32-Bit Triage Running Time</td>
<td>35</td>
</tr>
</tbody>
</table>
Figure 20. Windows XP 32-Bit Triage Resource Result........................................... 35
Figure 21. Windows XP 64-Bit Triage Running Time.............................................. 36
Figure 22. Windows XP 64-Bit Triage Resource Result........................................... 36
Figure 23. Windows 7 32-Bit Triage Running Time................................................ 37
Figure 24. Windows 7 32-Bit Triage Resource Result............................................. 37
Figure 25. Windows 7 64-Bit Triage Running Time................................................ 38
Figure 26. Windows 7 64-Bit Triage Resource Result............................................. 38
Figure 27. Windows XP 32-Bit Disk Image Running Time........................................ 40
Figure 28. Windows XP 32-Bit Disk Image Resource Result.................................... 40
Figure 29. Windows XP 64-Bit Disk Image Running Time........................................ 41
Figure 30. Windows XP 64-Bit Disk Image Resource Result.................................... 41
Figure 31. Windows 7 32-Bit Disk Image Running Time........................................... 42
Figure 32. Windows 7 32-Bit Disk Image Resource Result....................................... 42
Figure 33. Windows 7 64-Bit Disk Image Running Time........................................... 43
Figure 34. Windows 7 64-Bit Disk Image Resource Result....................................... 43
CHAPTER 1: INTRODUCTION

1.1 Statement of Problem

Cloud-computing refers to the process of having applications and data storage being provided to end users via the Internet. Internet users are beginning to operate a majority of their applications along with data storage using cloud-computing hardware. The list below shows the average percentage of Internet users utilizing different cloud-computing services (Horrigan, 2008).

- Webmail Applications (Gmail, Hotmail, Yahoo) - 56%
- Online Photo Storage - 34%
- Online Applications (Google Docs, Photoshop) - 29%
- Online Video Storage - 7%
- Online Paid Data Storage - 5%
- Online Hard Drive Backup - 5%

The popularity of these services is creating new challenges for state and local law enforcement. By using the cloud-computing applications, the geographic location and status of files with potential evidentiary value are much more difficult to detect and obtain. The data files are no longer contained on a local device, which law enforcement can seize. The data files could in fact be located on several different servers that may be located in several different jurisdictions; making locating and seizing the data files an unfeasible solution. In addition, the cloud-computing applications often support a multitude of various platforms. This allows users to modify the data files from devices such as a cell phone, tablet pc, gaming device, or...
additional computers. Because of these issues, it is important that law enforcement quickly becomes aware of the increasing usage of cloud-computing applications.

This thesis addresses these problems by researching and analyzing the storage patterns of several cloud-computing applications, and then develops a tool to automatically detect and report any relevant data to an investigator. The prototype proof of concept tool developed for this project supports the investigation of several prominent cloud applications such as Microsoft Window’s SkyDrive, Microsoft Window’s Live Mesh, Dropbox, and Google Documents.

1.2 Goals

The main goal of this thesis is to identify the ways that the cloud-computing based applications supported are interacting with the physical computing device. This research provides the necessary information needed to create a tool capable of detecting and parsing any of the supported cloud-computing application on the computational device. The tool that results from this thesis, which I will refer to as Cloud Signature, is created to be used as an application that law enforcement will be able to use through a simple intuitive graphical user interface (GUI). For Cloud Signature to be useful to state and local law enforcement, it was designed to meet the following requirements:

- It should support physical computer devices
- It should support Windows XP 32-Bit, Windows XP 64-Bit, Windows 7 32-Bit, and Windows 7 64-Bit
• It should support Microsoft’s Internet Explorer Web Browser Version 6, 7, 8, and 9
• It should have a low memory footprint
• It should be reasonably fast at returning results
• It should accurately find all supported cloud-computing applications
• It should report as much of the information required for a search warrant as possible
• It should contain a built-in reporting option
• It should be easily updated
• It should be easy to use and require minimal training

Another goal of the project is that once Cloud Signature is completed, it should provide the user with a list of the cloud-computing applications found on a device as a result of a search. It should develop a report with data extracted from the cloud-computing applications needed to create a search warrant.

The ability of Cloud Signature to meet these requirements is the criteria that this thesis uses to evaluate this project. A more in-depth explanation of the requirements chosen for this tool can be found in Section 1.2.1.

1.2.1 Requirements of Cloud Signature

In cooperation with the Rhode Island State Police I determined that the following are the desirable characteristics for a cloud-computing application detection tool.
The tool should support a physical computer device

Cloud-computing applications exist on a multitude of devices, such as cell phones, tablets, and computers. Due to the scope of this problem, I have narrowed my research to the physical computer device such as desktop computers and laptops.

The tool should support Windows XP 32-Bit, Windows XP 64-Bit, Windows 7 32-Bit, and Windows 7 64-Bit.

For my research, Cloud Signature supports two of the most popular versions of the Windows Operating System, which are Windows XP and Windows 7. The tool supports both the 32-bit and 64-bit version of these operating systems.

The tool should support Microsoft’s Internet Explorer Web Browser Version 6, 7, 8 and 9

Web based cloud-computing applications have the ability to run on a variety of web browsers. Each of these web browsers may store temporary Internet data in a different way. Due to the scope of this problem, this research focuses on Microsoft’s Internet Explorer Version 6, 7, 8, and 9. I chose these versions because Windows XP first launched with Internet Explorer Version 6, and Internet Explorer Version 9 is the most recent released version (“Support for Windows Internet Explorer 6, 7, 8, and 9”, 2012).

The tool should have a low memory footprint

As stated above, the tool supports the following operating systems: Windows XP - 32-Bit, Windows XP - 64-Bit, Windows 7 - 32-Bit, and Windows 7 - 64-Bit. From Microsoft’s documentation, I found that Windows XP - 32-Bit recommends 128 MB of RAM (“System Requirements for Windows XP”, 2007) Windows XP - 64-Bit
recommends 1 GB of RAM ("Microsoft Windows XP 64-Bit Edition", 2001)
Windows 7 - 32-Bit recommends 1 GB of RAM, and Windows 7 - 64-Bit
recommends 2 GB of RAM ("Windows 7 System Requirements - Microsoft
Windows, 2011"). After reviewing these requirements a goal was set to have my tool
to use under 50 MB of RAM. This allows Cloud Signature to run on all of the
supported operating systems.

**The tool should be reasonably fast at returning results**

Through several interviews with members of the Rhode Island State Police, It
was identified that the tool should be able to complete a search in less than 1 minute

**The tool should accurately find all supported cloud-computing applications**

When law enforcement is examining a seized computer in which a cloud-
computing application may have been used, the tool should be able to accurately find
the presence of any of the cloud-computing applications supported. The tool should
also parse out and display any data files that may be of evidentiary value.

**The tool should report as much of the information required for a search warrant as possible**

Cloud Signature should parse any information that would assist in the creation of
a search warrant. Through my interviews at the Rhode Island State Police, I have
identified this information to include, but not limited to: usernames, email addresses,
IP addresses, dates of use, and any applicable filenames. A report then is created in
order to present this information in a readable format.
The tool should contain a built-in reporting option

Cloud Signature has the ability to output a report in the form of an HTML web page. This allows the user to view the results at a later date.

The tool should be easily updated

The number of cloud-computing applications is increasing at an alarming rate. Since several cloud-computing applications are being released each month, it is important that this tool is easily updated to detect these new applications. This allows the University of Rhode Island to release update patches to ensure that the state and local law enforcement will have access to the most effective cloud-computing application detection tool.

The tool should be easy to use and require minimal training

Cloud Signature should be easy to use for the law enforcement officers without any special training.

1.3 Microsoft .NET Framework

To successfully complete the goals explained in section 1.2.1 Cloud Signature was programmed the C# language implementing .NET Framework v4.0. The .NET Framework v4.0 is the most recent version released by Microsoft. This framework contains many built in operations that allow for an easier creation of Window’s user interfaces, and also contains several advanced programming concepts to help create an optimal version of Cloud Signature. The framework is freely available to all users via download, and any Windows operating system after Windows Vista has the .NET Framework built in (“.NET Framework”).
By using the .NET Framework 4.0 the ability for this program to be supported on a multitude of machines is not impacted at all. This framework is supported on every operating system that the tool is being designed to support from section 1.2.1. If Cloud Signature is run on a computational device that does not have the .NET Framework 4.0 currently installed, the application then prompts the user to install this framework.

1.4 Organization of Thesis

Chapter 2 of this thesis provides a review of literature and related work. This includes a review of literature corresponding to the definition and implementation of cloud-computing services, and a description of similar tools currently being used. Also, it includes a description of tools that were used as a model for Cloud Signature.

Chapter 3 describes the processes followed to identify the way that the supported cloud-computing applications interacted with the physical computing device. This chapter then explains the various data structures implemented by these applications to store user-specific data. Finally, the process of how the tool parses the specific data structures in order to extract the evidence in a user readable format is explained.

Chapter 4 provides the results that were determined via the methodologies explained in Chapter 3. The performance of Cloud Signature on the supported operating systems is discussed in detail. Finally, the performance is evaluated against the requirements that were outlined in section 1.2.1.

Chapter 5 provides a final review of the conclusions obtained from this research and analysis. This chapter goes into detail about the successfulness of creating a tool
to detect the presence, along with parsing any relevant data of a cloud-computing based application. Then areas that could be explored with future research and software development are discussed.
CHAPTER 2: REVIEW OF LITERATURE

This section provides an overview of current forensic tools used by law enforcement; also it reviews current literature corresponding to the definition and implementation of cloud-computing services.

2.1 Definition

The National Institute of Standards and Technology, NIST, defines cloud-computing as “a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.” (Mell, Peter, & Timothy, 2011).

According to NIST, the five essential characteristics of a cloud model are defined as: on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. The three service models are defined as: Software as a Service, Platform as a Service, and Infrastructure as a Service. The four deployment models are: private cloud, community cloud, public cloud, and Hybrid cloud.

The applications that are being researched in this thesis, which includes Windows Skydrive, Windows Live Mesh, Dropbox, and various web based cloud-computing applications such as: Google Documents, Dropbox online, Google Mail, Microsoft’s Hotmail, Windows Live Mail, and others. All of the previously listed applications are a member of a Public cloud deployment model and a SaaS, Software as a Service,
model. The SaaS model refers to the fact that an application provides a public service to all consumers in which the service being provided is distributed via a web browser. In this model, the consumer has no control over the cloud structure. The application is entirely maintained and run on the application’s cloud infrastructure (Craig - Wood, 2010). If interested, the NIST publication provides more information regarding the other types of cloud deployment and service models (Mell et al., 2011).

2.2 WetStone Technologies’ Gargoyle Investigator

WetStone Technologies’ Gargoyle Investigator is a tool that was created to scan a computational device for the presence of malware, meaning malicious software (“Gargoyle”). The final reporting feature of Cloud Signature was modeled on the reporting feature found in Gargoyle.

2.3 JADsoftware Incorporated’s Internet Evidence Finder

JADsoftware Incorporated’s Internet Evidence Finder is a tool that was created to scan a computational device and identify web browser artifacts (“Internet Evidence Finder”). While Internet Evidence Finder is very effective at detecting and parsing web browser based artifacts, it does not support any client based installed cloud applications. The Graphical User Interface, GUI, of Cloud Signature was modeled off of the interface found in Internet Evidence Finder.
2.4 ATC - NY’s Cyber Marshal Dropbox Reader

NY’s Cyber Marshall has released a Dropbox Reader tool that has the ability to parse the Dropbox SQLite database files. While this tool is effective, it is a command line based tool comprised of Python scripts, which ultimately makes this tool difficult for the average person to use. Also, this tool only works with the Dropbox application and it does not have the ability to be updated (“ATC-NY’s”).

2.5 Digital Detective’s Net Analysis

Digital Detective’s Net Analysis is well known amongst the digital forensic community. The tool was originally developed in 2001, and has been used by numerous law enforcement entities around the world in order to parse Internet browser artifacts (“NetAnalysis”). Net Analysis was used to verify that Cloud Signature was accurately displaying all data that may be of evidentiary value. The running time and memory performance of Net Analysis was used as a benchmark to test against Cloud Signature.

2.6 ForensicSoft’s SAFE Block

ForensicSoft’s SAFE Block is a tool designed to facilitate software write blocking. SAFE Block prevents writing to any disk or flash storage media (“SAFE Block Features”, 2010). Cloud Signature was designed to work alongside of SAFE Block to preserve the evidentiary integrity of the item being analyzed. The user interface of Cloud Signature was also modeled off of the user interface found in SAFE Block.
CHAPTER 3: METHODOLOGY

This section provides an overview of the design process of Cloud Signature. First the interface is discussed, and then the specific operations that Cloud Signature performs are explained.

For this thesis project it was first necessary to design an application template. An application template refers to the way in which the user interacts with the application. The graphical user interface of Cloud Signature is designed to be very intuitive, and is comprised of three check boxes, one selection box, and one click button. The user must select which applications they wish to scan for, and which drive they would like to scan. When the user presses the Go button, a file dialog box pops up asking the user where the report should be saved. The user is not required to interact with the program after selecting the output directory. Figure 1 shows the user interface of the Cloud Signature application.

![Cloud Signature User Interface](image)

**Figure 1 - Cloud Signature User Interface**
The remaining portion of this methodology is split into two different cloud-computing applications categories: web based cloud-computing application, and client installed cloud-computing application. A web-based cloud-computing application refers to a cloud-computing application that can be run entirely via a web browser interface. Some popular examples a web-based cloud-computing application are Gmail, Amazon Web Services, and Microsoft’s SkyDrive. A client based cloud-computing application refers to a cloud-computing application that requires the user to install a piece of client software.

3.1 Web Based Cloud-Computing Applications

In order to detect the presence and use of a web based cloud-computing application, the use of a web browser’s temporary Internet data was required to be analyzed. For this research, Microsoft’s Internet Explorer web browser versions 6, 7, 8, and 9 were supported. Windows XP supports Internet Explorer versions 6, 7, and 8. Windows 7 supports Internet Explorer versions 7, and 8, and 9. All of these versions of Internet Explorer implement the index.dat file structure from version 5. Overall, all of the Internet Explorer versions share the same index.dat structure, but the location of where the temporary Internet files are stored may change.

When a user is operating Internet Explorer, user temporary Internet data is automatically saved on the physical computing device in index.dat files. The location of these index.dat files is dependent on the operating system that is being used and the version of the operating system. After researching Internet Explorer, it was
determined that the index.dat files were being stored in the following directories for each operating system:

Windows 7 Operating System index.dat Storage Directories 32 and 64-Bit

<table>
<thead>
<tr>
<th>Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{Drive Letter}:\Users\{User Name}\App Data\Roaming\Microsoft\Windows\Privacy\index.dat</code></td>
</tr>
<tr>
<td><code>{Drive Letter}:\Users\{User Name}\App Data\Roaming\Internet Explorer\UserData\Low\index.dat</code></td>
</tr>
<tr>
<td><code>{Drive Letter}:\Users\{User Name}\App Data\Roaming\Microsoft\Internet Explorer\DOMStore\index.dat</code></td>
</tr>
<tr>
<td><code>{Drive Letter}:\Users\{User Name}\App Data\Roaming\Microsoft\Windows\IECompatCache\index.dat</code></td>
</tr>
<tr>
<td><code>{Drive Letter}:\Users\{User Name}\App Data\Roaming\Microsoft\Windows\IECompatCache\Low\index.dat</code></td>
</tr>
<tr>
<td><code>{Drive Letter}:\Users\{User Name}\App Data\Roaming\Microsoft\Windows\IETidCache\index.dat</code></td>
</tr>
<tr>
<td><code>{Drive Letter}:\Users\{User Name}\App Data\Roaming\Microsoft\Windows\Cookies\index.dat</code></td>
</tr>
<tr>
<td><code>{Drive Letter}:\Users\{User Name}\App Data\Local\Microsoft\Feeds Cache\index.dat</code></td>
</tr>
</tbody>
</table>

Table 1 - Windows 7 index.dat Storage Directories

Windows XP Operating System index.dat Storage Directories 32 and 64-Bit

<table>
<thead>
<tr>
<th>Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{Drive Letter}:\Documents and Settings\{User Name}\Cookies\index.dat</code></td>
</tr>
<tr>
<td><code>{Drive Letter}:\Documents and Settings\{User Name}\UserData\index.dat</code></td>
</tr>
<tr>
<td><code>{Drive Letter}:\Documents and Settings\{User Name}\Local Settings\History\History.IE5\index.dat</code></td>
</tr>
<tr>
<td><code>{Drive Letter}:\Documents and Settings\{User Name}\Local Settings\Temporary Internet Files\Content.IE5\index.dat</code></td>
</tr>
</tbody>
</table>

Table 2 - Windows XP index.dat Storage Directories
In order to explain the method of Cloud Signature’s process for parsing an index.dat file, the file structure of these files must be described. The index.dat file contains information pertaining to web pages that were accessed by the user. The information Cloud Signature extracts from the index.dat files includes the last accessed time, the URL visited, and the filename. The last accessed time states when the website was visited by the user. The URL visited stores the ASCII string representation of the URL that was visited, and the filename stores the name of the file that was accessed and temporarily stored by the browser on the physical device. The filename may not be set in all of the entries of an index.dat file because not all entries correspond to a file that is being temporarily stored on the computing device. As noted before, Internet Explorer versions 6, 7, 8, and 9 all implement the same data structure to store temporary Internet files. These applications implement Client URL Cache Version 5.

In every index.dat file there exists a 32-bit integer value at offset 0x20, which indicates where the entries are being stored on the file. Once Cloud Signature has moved to the beginning of the entry storage, it starts to search for any valid entries. An entry can be one of three types:

1. REDR - This symbolizes a browser redirect.
2. LEAK - This symbolizes an error that was generated. Usually this is generated due to an error occurring during the deletion operation of a URL entry.
3. URL - This symbolizes every URL, Uniform Resource Locator, which the user visited.
Cloud Signature only scans for entries marked LEAK or URL. It does not search for REDR entries because the final URL that the user would arrive at would still be indicated as a URL entry.

The LEAK and URL entries are setup the exact same way. Table 3 shows the byte structure of the data locations from the individual entries in the index.dat file.

Once Cloud Signature has found a LEAK or URL entry, it begins to parse the Last Accessed Time Stamp, URL Offset, and Filename Offset.

<table>
<thead>
<tr>
<th>Information Name</th>
<th>Offset From Beginning of Entry</th>
<th>Size (in bytes)</th>
<th>Description of Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of Record</td>
<td>0x0</td>
<td>4</td>
<td>Indicates if REDR, URL, or LEAK</td>
</tr>
<tr>
<td>Last Modified Time Stamp</td>
<td>0x8</td>
<td>8</td>
<td>Contains the Last Modified Time Stamp. Usually this field is left blank with Internet Explorer</td>
</tr>
<tr>
<td>Last Accessed Time Stamp</td>
<td>0x10</td>
<td>8</td>
<td>Contains the last time the user accessed the corresponding URL</td>
</tr>
<tr>
<td>URL Offset</td>
<td>0x34</td>
<td>4</td>
<td>Contains the offset from the beginning of the entry, to where the URL is stored</td>
</tr>
<tr>
<td>Filename Offset</td>
<td>0x3C</td>
<td>4</td>
<td>Contains the offset from the beginning of the entry to where the filename is stored</td>
</tr>
</tbody>
</table>

Table 3 - index.dat File Structure
After Cloud Signature has parsed the data found in the table above, it moves to the URL and Filename offsets to parse the ASCII String representation of the bytes located in these data fields. After all of the data for the entry has been parsed, Cloud Signature determines if the entry has any evidentiary value. If the entry does contain evidentiary value, it is added to the report. The tool then continues searching for another iteration of a URL or LEAK entry. Below in Figures 2 and 3, a URL entry is shown after it has been parsed by a custom WinHex template.

<table>
<thead>
<tr>
<th>Offset</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0005100</td>
<td>55 52 4C 20 02 00 00 00</td>
<td>A0 78 09 E5 D2 ED CC 01</td>
<td>URL</td>
<td>x ñöíï</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0005110</td>
<td>50 78 09 E5 D2 ED CC 01</td>
<td>51 42 70 02 00 00 00 00</td>
<td>00 00 00 00 00 00 00 00</td>
<td>ñöíï 08p</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0005120</td>
<td>60 00 00 00 00 00 00 00</td>
<td>00 00 00 00 00 00 00 00</td>
<td>00 00 00 00 00 00 00 00</td>
<td>ñöíï 08p</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0005130</td>
<td>00 01 00 00 00 00 00 00</td>
<td>00 00 00 00 00 00 00 00</td>
<td>00 00 00 00 00 00 00 00</td>
<td>ñöíï 08p</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0005140</td>
<td>52 40 5B 02 04 00 00 00</td>
<td>00 00 00 00 00 00 00 00</td>
<td>52 40 5B 02</td>
<td>ñöíï 08p</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0005150</td>
<td>00 00 00 00 00 00 00 00</td>
<td>00 00 00 00 00 00 00 00</td>
<td>00 00 00 00 00 00 00 00</td>
<td>ñöíï 08p</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0005160</td>
<td>6F 70 70 65 6E 43</td>
<td>5F 6F 6B 69 65 3A 6E</td>
<td>ñöíï 08p</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0005170</td>
<td>6F 70 70 65 6E 43</td>
<td>5F 6F 6B 69 65 3A 6E</td>
<td>ñöíï 08p</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0005180</td>
<td>6F 70 70 65 6E 43</td>
<td>5F 6F 6B 69 65 3A 6E</td>
<td>ñöíï 08p</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0005190</td>
<td>00 00 00 00 00 00 00 00</td>
<td>00 00 00 00 00 00 00 00</td>
<td>00 00 00 00 00 00 00 00</td>
<td>ñöíï 08p</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00051A0</td>
<td>00 00 00 00 00 00 00 00</td>
<td>00 00 00 00 00 00 00 00</td>
<td>00 00 00 00 00 00 00 00</td>
<td>ñöíï 08p</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2 - Bytes Represented In A URL Entry

Figure 3 - Parsed URL Bytes
To determine if a parsed entry is of evidentiary value, it is compared against a list of user specified keywords. Cloud Signature builds a list of specific keywords in a text file located in the program’s directory. Throughout the research process, a list of keywords pertaining to various web based cloud-computing applications was created, such as the following: Google Documents, Google Mail, Google Plus, Personal Google Web Searches, Dropbox Web Client, Windows Sky Drive, Windows Live Services, Facebook, and MSN Services. Below in Figure 4, one sees the text file containing the keyword list that Cloud Signature uses to create a Keyword Object, which is used to compare the parsed data against to determine if there is evidentiary value. It is important to note that the keywords are not searched case sensitively.

![Figure 4 - Keyword Text File](image)

Each Keyword object contains a title, an array of specific keywords, and an array of entries that have matched some portion of the specific keywords. Table 4 displays an example of the data members in a Keyword Object.
### Table 4 - Keyword Object Structure

<table>
<thead>
<tr>
<th>Title</th>
<th>Windows Live</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keywords</td>
<td>login.live.com</td>
</tr>
<tr>
<td></td>
<td>explore.live.com</td>
</tr>
<tr>
<td>Results</td>
<td>Contains any entries that matched the keywords</td>
</tr>
</tbody>
</table>

Each of the entries found in an index.dat file are searched to determine if any of the keywords from the user generated keyword.txt file are present. If any keyword matches, then the entry is added to the corresponding Keyword Object’s Results as a result for the corresponding keyword group. Once the parsing of an entire index.dat file is finished, the entries that contained keywords are then written out to a Keyword Group in a specific HTML table within a specific HTML webpage. Figures 5, 6, and 7 display the final report format. The exact same steps are repeated for each index.dat file found and Cloud Signature writes a separate HTML report for each index.dat file.

---

**URL Results Report for Username: Jeremy Koppen**
**2/29/2012 1:14:58 AM UTC**

File Parsed: C:\Users\Jeremy Koppen\AppData\Local\Microsoft\Windows\History\Low\History.IE5\index.dat

- Dropbox Web Application Occurrences = 5
- Facebook Occurrences = 2
- Google Mail Occurrences = 9
- MSN Occurrences = 5
- Windows Live Occurrences = 8
- Windows SkyDrive Occurrences = 2
- Google Occurrences = 14

---

**Figure 5 - Web Based Report Title Page**
3.2 Client Based Cloud-Computing Applications

As noted before, a client-based cloud-computing application requires the user to install a piece of client software on a physical computing device. For this thesis, the client-based cloud-computing applications Dropbox and Windows Live Mesh were researched. Dropbox was chosen due to its immense popularity. In April 2011, Dropbox announced they had over 25 million users (Arrington, 2011). Windows Live Mesh was chosen due to the fact that it is supported by Microsoft, and is expected to be included in future operating systems (“BUILD, 2011”). Both of these allow for a
user to upload files and folders via a client application. Then these files and folders can be downloaded on any other computer (“Dropbox”, “Windows Live Mesh 2011”).

3.2.1 Dropbox Client Application

The research performed in this thesis corresponds to Dropbox version 1.1.3.5. The first step of this research was to determine if Dropbox stored any information physically on the device where the application was installed. After monitoring the resources on the physical computer that the Dropbox Application accessed, it was determined the application created a directory within the following paths:

Windows 7 Dropbox Directory Location 32 and 64-Bit

{Drive Letter}\Users\{User Name}\AppData\Roaming\Dropbox

Windows XP Dropbox Directory Location 32 and 64-Bit

{Drive Letter}\Documents and Settings\{User Name}\Application Data\Dropbox

After analyzing the Dropbox directory location, several SQLite database files were found within the directory. After further analysis, it was determined that two of the database files contained data that would be of evidentiary value. These two database files were named config.db and filecache.db. The config.db database file contained the following information regarding the user’s Dropbox account: (See Figure 8)

1. Dropbox Version - The version of Dropbox that is being used on the computational device
2. Unique Dropbox Host ID - A unique 128 bit key pertaining to a user account
3. Dropbox Path - The path on the computational device where Dropbox has mounted its virtual folder
4. Dropbox Username - A string username that is specified by the user
5. Recently Changed Files - List of files that were most recently changed on the Dropbox account

<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>config_schema_version</td>
<td>1</td>
</tr>
<tr>
<td>show_bubbles</td>
<td>1</td>
</tr>
<tr>
<td>fixed_dropbox_perms</td>
<td>1</td>
</tr>
<tr>
<td>recently_changed3</td>
<td>(lp1)</td>
</tr>
<tr>
<td>host_id</td>
<td>f1e8a2e7b13b0ba60d8474e55a2d6c3f</td>
</tr>
<tr>
<td>dropbox_path</td>
<td>C:\Documents and Settings\Jeremy\My Documents\Dropbox</td>
</tr>
<tr>
<td>root_ns</td>
<td>4627460</td>
</tr>
<tr>
<td>email</td>
<td><a href="mailto:jeremy.koppen1@gmail.com">jeremy.koppen1@gmail.com</a></td>
</tr>
<tr>
<td>stats_dont_send_until_upgrade</td>
<td></td>
</tr>
<tr>
<td>stats_next_report_time</td>
<td>1332213286.076</td>
</tr>
<tr>
<td>stats_next_report_id</td>
<td>33916433</td>
</tr>
<tr>
<td>ns_p2p_key_map</td>
<td>(dp1)</td>
</tr>
<tr>
<td>sandboxes</td>
<td>(lp1)</td>
</tr>
<tr>
<td>last_update</td>
<td>(1331003687.640001...</td>
</tr>
</tbody>
</table>

**Figure 8 - Config.db**

The filecache.db database file contains a table called file_journal, which contains the following information for any files stored on the Dropbox account: (See Figure 9)

1. Server Path - Stores the path of the file with the server identification
2. Local Filename - Stores the local filename of the file that was added to the Dropbox account
3. SHA-256 Hash - Stores the SHA-256 Hash in a Base-64 encoded string
4. Local Size (MB) - Stores the Local Size of the file in MB
5. Modified Time (UTC) - Stores the time that the file was modified

6. Created Time (UTC) - Stores the time that the file was created

<table>
<thead>
<tr>
<th>id</th>
<th>server path</th>
<th>local filename</th>
<th>local blocklist</th>
<th>local size</th>
<th>local intime</th>
<th>local time</th>
</tr>
</thead>
<tbody>
<tr>
<td>61</td>
<td>4827460/packetfilt...</td>
<td>Test Code Output.txt</td>
<td>CaE0D4hFnFPG...</td>
<td>1625</td>
<td>1259497754</td>
<td>1229590553</td>
</tr>
<tr>
<td>62</td>
<td>4827460/packetfilt...</td>
<td>Sample output.txt</td>
<td>1BD9DmY0C68R5u...</td>
<td>3632</td>
<td>1240947768</td>
<td>1229590553</td>
</tr>
<tr>
<td>63</td>
<td>4827460/packetfilt...</td>
<td>PacketFilter1.vcpov</td>
<td>QJUN6VmQCGiqQ...</td>
<td>4540</td>
<td>1262413634</td>
<td>1229590553</td>
</tr>
<tr>
<td>64</td>
<td>4827460/packetfilt...</td>
<td>filecache.obj</td>
<td>Km5DqoFbPZN7q...</td>
<td>13051</td>
<td>1254130505</td>
<td>1229590553</td>
</tr>
<tr>
<td>65</td>
<td>4827460/packetfilt...</td>
<td>PacketFilter1.pch</td>
<td>JA06pe4qJb/0x3...</td>
<td>3211254</td>
<td>1258413050</td>
<td>1229590553</td>
</tr>
<tr>
<td>66</td>
<td>4827460/packetfilt...</td>
<td>int.dsp</td>
<td>3b06o067NuW/af...</td>
<td>60</td>
<td>1258419839</td>
<td>1229590553</td>
</tr>
<tr>
<td>67</td>
<td>4827460/packetfilt...</td>
<td>PacketFilter1.suo</td>
<td>qfK8hF/8DpPn...</td>
<td>8724</td>
<td>1258497836</td>
<td>1229590553</td>
</tr>
<tr>
<td>68</td>
<td>4827460/packetfilt...</td>
<td>Build.log</td>
<td>lqgfYtFrJo/m1Qg...</td>
<td>46318</td>
<td>1258497836</td>
<td>1229590553</td>
</tr>
<tr>
<td>69</td>
<td>4827460/packetfilt...</td>
<td>PacketFilter1.cpp</td>
<td>Sfco2Ch6fV2cmi...</td>
<td>110062</td>
<td>1258497826</td>
<td>1229590564</td>
</tr>
<tr>
<td>70</td>
<td>4827460/packetfilt...</td>
<td>PacketFilter1.exe</td>
<td>1L1H5D5jLQ/vAn...</td>
<td>155744</td>
<td>1258497836</td>
<td>1229590564</td>
</tr>
<tr>
<td>71</td>
<td>4827460/packetfilt...</td>
<td>vs00.db</td>
<td>1M4Mpa45DlNUP...</td>
<td>135924</td>
<td>1258497836</td>
<td>1229590564</td>
</tr>
<tr>
<td>72</td>
<td>4827460/packetfilt...</td>
<td>vs00.cdb</td>
<td>15bc7hN90MOQJl...</td>
<td>230262</td>
<td>1258497834</td>
<td>1229590564</td>
</tr>
</tbody>
</table>

Figure 9 - File_Journal Table From File_Cache.db

Cloud Signature scans every applicable user account to determine the presence of a Dropbox directory. In Windows 7 the tool scans the user’s AppData directory, and in Windows XP the tool scans the user’s Application Data directory. If the Dropbox directory is found, then Cloud Signature begins to parse the config.db database file by using an instance of the SQLReader class. After the applicable data has been parsed, it is written out to a Dropbox Results HTML webpage report. Figure 10 displays the output created from the parsing of the config.db file. Next, Cloud Signature parses the filecache.db database file and the tool sends all of the parsed data to the same Dropbox Results HTML webpage report. Figures 11 and 12 display the output created from the parsing of the filecache.db file. In Figure 11, the SHA-256 Hash is blank because directories do not contain a hash value. After, the Dropbox parsing is complete; the Dropbox Results Webpage contains a listing of all relevant user information along with a listing of all files present in the Dropbox account with all corresponding file information.

23
This information has the capability of providing law enforcement unique user identification information, along with all of the files that the user had placed on the Dropbox account for storage. Cloud Signature is capable of searching for and displaying all of this Dropbox user information, which ultimately helps law enforcement obtain a warrant and find potential evidence for an investigation.

**DropBox Report Report for Username: koppen**

2/22/2012 4:46:59 PM UTC

**Dropbox Version:** 5Dropbox-win-1.1.35.5p1.

**Host ID:** a2587c62b78a5aa881ca597fd2be6d34

**Dropbox Path:** C:/Users/koppen/Dropbox

**Username:** jeremy.koppen1@gmail.com

**Recently Changed Files:**

ProcmonConfiguration.pmc Ntp2 a(V4827460)/SQLite-1.0.66.0-setup.exe

myapplication.zip

/Release to test.zip Ntp5 a(V4827460)/Release.7z

Figure 10 - Dropbox Config.db Report
### Dropbox File List

<table>
<thead>
<tr>
<th>ID</th>
<th>Server Path</th>
<th>Local Filename</th>
<th>SHA-256 Hash</th>
<th>Local Size (MB)</th>
<th>Modified Time (UTC)</th>
<th>Created Time (UTC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4827460/photos/sample album</td>
<td>Sample Album</td>
<td>46e2da64646a4912a149d4f4ea51c2193e38121f13b79faea528b74b</td>
<td>0</td>
<td>8/14/2011 11:45 AM</td>
<td>8/10/2011 3:07 AM</td>
</tr>
<tr>
<td>2</td>
<td>4827460/packetlist/packetlist/Debug</td>
<td>Debug</td>
<td>94e502157e27b46fd35887c498f67c6c5a5848b2a13b6d8511d208da</td>
<td>0</td>
<td>4/5/2011 2:32 PM</td>
<td>4/5/2011 2:32 PM</td>
</tr>
<tr>
<td>3</td>
<td>4827460/packetlist/packetlist/Packetlist</td>
<td>Packetlist</td>
<td>45e502157e27b46fd35887c498f67c6c5a5848b2a13b6d8511d208da</td>
<td>0</td>
<td>4/5/2011 2:32 PM</td>
<td>4/5/2011 2:32 PM</td>
</tr>
<tr>
<td>4</td>
<td>4827460/packetlist/packetlist/Debug</td>
<td>Debug</td>
<td>94e502157e27b46fd35887c498f67c6c5a5848b2a13b6d8511d208da</td>
<td>0</td>
<td>4/5/2011 2:32 PM</td>
<td>4/5/2011 2:32 PM</td>
</tr>
<tr>
<td>5</td>
<td>4827460/packetlist/packetlist/Debug</td>
<td>Debug</td>
<td>94e502157e27b46fd35887c498f67c6c5a5848b2a13b6d8511d208da</td>
<td>0</td>
<td>4/5/2011 2:32 PM</td>
<td>4/5/2011 2:32 PM</td>
</tr>
</tbody>
</table>

**Figure 11 - Dropbox Report For Directories**

- Figure 11 shows the file list of directories in Dropbox.
- Each row represents a directory, with columns for directory path, local filename, SHA-256 hash, local size, modified time, and created time.

**Figure 12 - Dropbox Report For Files**

- Figure 12 highlights the file list of Dropbox, similar to Figure 11 but focusing on files.
- The same structure is used for listing files with columns for file path, local filename, SHA-256 hash, local size, modified time, and created time.
3.2.2 Windows Live Mesh Client Application

The research performed in this thesis corresponds to Windows Live Mesh Version 2011 Build 15.4.3538.0513. Windows Live Mesh is an application that allows a user to sync multiple folders, and subsequent files, within a supported Windows operating system to a cloud storage device, which is maintained by Microsoft. Windows Live Mesh is not supported on the Windows XP Operating System.

The first step of this research was to identify what computing resources the Windows Live Mesh client was accessing on the physical device. After monitoring the way in which Live Mesh operates, it was detected that the application created a directory in the following directory path:

Windows 7 Windows Live Mesh Directory Location

{Drive Letter}\Users\{User Name}\AppData\Local\Microsoft\Windows Live Mesh

After researching this directory it became apparent that there were several .edb database files. These files are Extensible Storage Engine, JET Blue, database files. JET Blue was created by Microsoft and implements an ISAM, Indexed Sequential Access Method, data storage approach. In order to parse this database file Cloud Signature uses the ManagedEsent .NET library. This library provided the ability to load the database and extract data.

Inside of this Windows Live Mesh directory there exists a directory titled “DB”. In Figure 13 a file tree of this directory is shown. The “DB” directory contains one
subdirectory called “Device” which contains Device.edb database file. In this file there is a User table that corresponds to the account information. The User table contains all of the user specific information such as: the email address of the account used, a unique ID, the last time the account was updated, the published date of the account, and the Name that corresponds to the user’s account. This tabled provided a wealth of information about the user’s Windows Live Mesh account. Once all of this table’s information is parsed it is printed out to a table inside of an .HTML webpage.

The “DB” directory also contains subdirectories for each user account that was accessed, which are named from a unique user GUID. This GUID can be found in the User table from the Device.edb file. In Figure 14 the User table is shown. In this Figure, the User table lists one GUID: “ec6048b2-9d02-c264-8851-d18177d9e889”. It is important to note that this is why the DB directory contains a subdirectory titled “ec6048b2-9d02-c264-8851-d18177d9e889”, as seen in Figure 13. This directory contains an .edb database file that is also named based off of the unique user GUID. This file contains information corresponding to the user’s files.
Inside of this user GUID database file there are several tables. Cloud Signature first accesses the MeshObject table, which contains fields called “Id”. These unique Ids correspond to directories that were synced with Windows Live Mesh. Windows Live Mesh creates tables for each directory that the user synced. The tool parses each Id found in the Mesh Object in order to detect the names of the tables that contain information regarding the files that were synced. Figure 15 shows the MeshObject Table along with the Ids. Once this Id has been parsed the tool opens up another table that is titled “{MeshObject Id}_DataEntity_Enclosure”.

Figure 13 - Windows Live Mesh Directory File Tree
Inside of this table there exists the following information for each file or directory added:

- Filename
- Parent History
- Creation Time (UTC)
- File Last Write Time (UTC)

In Figure 16 this DataEntity_Enclosure is shown. For every file that was added to the user’s Windows Live Mesh account the fields above are parsed, and the information is written out to the same HTML webpage that has the user’s information. When Cloud Signature is done parsing both .edb database files, Device.edb, {USER GUID}.edb, the tool outputs a complete webpage with all the data in a readable format for law enforcement. Figure 17 shows the output report that is created.

![Figure 14 - Windows Live Mesh User Table](image)
Figure 15 - Windows Live Mesh MeshObject Table

Figure 16 - Windows Live Mesh DataEntity_Enclosure
Figure 17 - Windows Live Mesh Report

<table>
<thead>
<tr>
<th>Filename</th>
<th>Parent History</th>
<th>Creation Time</th>
<th>File Last Write Time (UTC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>main.c</td>
<td>1_jerms@<a href="mailto:36@hotmail.com">36@hotmail.com</a>,WIEGBACTVSMFCCCLGAXQPWFIRE_YAMYMBDAQKW50EU3TQYPCPSE</td>
<td>12/20/2011 8:59:55 AM</td>
<td>12/20/2011 11:59:54 AM</td>
</tr>
<tr>
<td>sub.c</td>
<td>1_jerms@<a href="mailto:36@hotmail.com">36@hotmail.com</a>,WIEGBACTVSMFCCCLGAXQPWFIRE_YAMYMBDAQKW50EU3TQYPCPSE</td>
<td>12/20/2011 8:59:49 AM</td>
<td>12/20/2011 11:59:32 AM</td>
</tr>
<tr>
<td>attach.c</td>
<td>1_jerms@<a href="mailto:36@hotmail.com">36@hotmail.com</a>,WIEGBACTVSMFCCCLGAXQPWFIRE_YAMYMBDAQKW50EU3TQYPCPSE</td>
<td>12/20/2011 9:00:14 AM</td>
<td>12/20/2011 11:50:13 AM</td>
</tr>
<tr>
<td>math.c</td>
<td>1_jerms@<a href="mailto:36@hotmail.com">36@hotmail.com</a>,WIEGBACTVSMFCCCLGAXQPWFIRE_YAMYMBDAQKW50EU3TQYPCPSE</td>
<td>12/20/2011 9:00:19 AM</td>
<td>12/20/2011 11:50:18 AM</td>
</tr>
<tr>
<td>backup.c</td>
<td>1_jerms@<a href="mailto:36@hotmail.com">36@hotmail.com</a>,WIEGBACTVSMFCCCLGAXQPWFIRE_YAMYMBDAQKW50EU3TQYPCPSE</td>
<td>12/20/2011 9:00:26 AM</td>
<td>12/20/2011 11:50:26 AM</td>
</tr>
</tbody>
</table>
CHAPTER 4: FINDINGS

To determine the findings of Cloud Signature, it was necessary to test the tool to ensure that all requirements were met. To test Cloud Signature, several virtual machines were implemented. VMWare defines a virtual machine as “a tightly isolated software container that can run its own operating systems and applications as if it were a physical computer. A virtual machine behaves exactly like a physical computer and contains its own virtual (i.e., software-based) CPU, RAM hard disk and network interface card (NIC)” (“Virtualization Basics”). By using virtual machines, it allows for a system to be created, and restored, back to a clean environment before beginning the testing procedure.

For the testing procedure, a generic virtual machine was created that had the proper version of Windows operating system installed. Next, several web-based cloud-computing applications were accessed via multiple versions of Microsoft’s Internet Explorer. Then, depending on the operating system, the corresponding Dropbox and Windows Live Mesh applications were installed. The applications were then accessed and files were added, modified, and deleted. By following these exact steps each time, it could be ensured that the testing environment contained the data it was trying to locate, and that the environment was a clean testing environment.

Along with verifying that all of the evidence was detected and presented to the user, the completion of all requirements listed in Section 1.2.3 must to be verified. In order to test the performance aspects of Cloud Signature, a program called Process Monitor was used. Microsoft defines Process Monitor as “an advanced monitoring
tool for Windows that shows real-time file system, Registry, and process/thread activity” (Russinovich, Mark, & Bryce, 2011).

Process Monitor analyzes a process and returns the CPU Utilization, File I/O Bytes, File I/O Operations, Registry Operations, Network Bytes, Network Operations, Private Memory Bytes, and Memory Working Set. The Private Memory Bytes reports the amount of memory that the process requests, but this number excludes any memory being used by mapped files or DLL’s. The Memory Working Set reports the actual amount of RAM that is being used by the process (“Process Monitor”, 2011).

To evaluate Cloud Signature, the number reported from the Memory Working Set is used. By using Process Monitor it was possible to see the exact running time and memory usage of Cloud Signature. These are very important to law enforcement due to the rapid pace of an investigation, and the various situations they may encounter.

In order to be useful in these various situations Cloud Signature has been created to be both a triage forensics tool and a disk level forensics tool. A triage forensic tool refers to a tool is a tool that runs on the computer that is being investigated while the computer is in a live running state. A disk level forensic tool refers to a tool that analyzes an image of the evidentiary drive using an investigation workstation computer. It is important Cloud Signature supports both of these types of forensic situations in order to be as useful to law enforcement as possible. Regardless of whether Cloud Signature is ran as a triage forensics tool or a disk level forensics tool it outputs identical reports. This HTML report begins with a main title page. This title page breaks down the results based by the computer account username. All of the results are hyperlinked together so that law enforcement can view all of the reported
data within a web browser. Figure 18 shows the structure of the main title page and the results that it determined.

![Cloud Signature Main Index Page]

**4.1 Triage Forensic Test Findings**

In a triage scenario, Cloud Signature would run from a form of external storage. The risk of using Cloud Signature in a triage manner is that several files may be modified when the files with evidentiary value are opened and parsed. It was necessary to run a triage test on virtual machines that were running the following operating systems: Windows XP 32-Bit, Windows XP 64-Bit, Windows 7 32-Bit, and Windows 7 64-Bit.
4.1.1 Windows XP 32-Bit Triage

As shown in Figure 19, Cloud Signature took approximately 10 seconds to complete its analysis. Once again, it is important to note that Windows XP 32-Bit does not support Windows Live Mesh. Upon further analysis of Process Monitor results, shown in Figure 20, it demonstrates that Cloud Signature used approximately 31.6 MB of memory. The tool successfully detected all file additions, removals, and changes in the Dropbox application. It was also able to detect all web sites accessed by the user that matched. These results were reported to the user in an easy to use HTML format.

![Figure 19 - Windows XP 32-Bit Triage Running Time](image)

![Figure 20 - Windows XP 32-Bit Triage Resource Result](image)
4.1.2 Windows XP 64-Bit Triage

As shown in Figure 21, Cloud Signature took approximately 10 seconds to complete its analysis. Once again, it is important to note that Windows XP 64-Bit does not support Windows Live Mesh. Upon further analysis of Process Monitor results, shown in Figure 22, it demonstrates that Cloud Signature used approximately 24.1 MB of memory. The tool successfully detected all file additions, removals, and changes in the Dropbox application. It was also able to detect all web sites accessed by the user that matched the specific keywords. These results were reported to the user in an easy to use HTML format.

| Started: 2/28/2012 7:05:57 | Total User CPU: 00:00:00.3281250 |
| Ended: 2/28/2012 7:06:07  | Total Kernel CPU: 00:00:00.2656250 |

Figure 21 - Windows XP 64-Bit Triage Running Time

Figure 22 - Windows XP 64-Bit Triage Resource Result
4.1.3 Windows 7 32-Bit Triage

In Figure 23 it can be seen that Cloud Signature completed its process in approximately 13 seconds. Upon further analysis of Process Monitor results, shown in Figure 24, it demonstrates that Cloud Signature used approximately 34.6 MB of memory. The tool successfully detected all file additions, removals, and changes in both the Dropbox and Windows Live Mesh applications. It was also able to detect all web sites accessed by the user that matched the specific keywords. These results were reported to the user in an easy to use HTML format.
4.1.4 Windows 7 64-Bit Triage

In Figure 25 it can be seen that Cloud Signature completed its process in approximately 26 seconds. Upon further analysis of Process Monitor results, shown in Figure 26, it demonstrates that Cloud Signature used approximately 35.9 MB of memory. The tool successfully detected all file additions, removals, and changes in both the Dropbox and Windows Live Mesh applications. It was also able to detect all web sites accessed by the user that matched the specific keywords. These results were reported to the user in an easy to use HTML format.

![Figure 25 - Windows 7 64-Bit Triage Running Time](image)

![Figure 26 - Windows 7 64-Bit Triage Resource Result](image)
4.2 Disk Level Forensics Testing Findings

Generally, an investigator makes a bit-by-bit copy of the original drive to ensure that the original evidence is never modified. In order to create an image of an operating system disk, the Helix Live CD and the DD command were used. The Helix Live CD is a forensically sound Linux Live Distribution CD. The DD command can be accessed within the Linux Live CD, and it copies a source file to a destination location bit-by-bit. To begin the process, the external drive was first securely wiped by writing 0’s to every bit. Then the DD process was used to create an exact copy of the original evidence drive. After the copying was complete, the image was confirmed that it was indeed an exact copy by comparing MD5 hashes of both devices.

When using a disk level forensic tool, it is common to implement a software write blocker. A software write blocker is used to eliminate the possibility of any writing being done to the drive. For this research, the investigator’s forensic machine consisted of a Windows 7 32-Bit PC with 4 GB of RAM. ForensicSoft’s SAFEBlock was the software write blocker used in combination with Cloud Signature to ensure a safe environment for testing.
4.2.1 Windows XP 32-Bit Disk Image

As shown in Figure 27, Cloud Signature took approximately 16 seconds to complete its analysis of the Windows XP - 32-Bit drive. Once again, it is important to note that Windows XP - 32-Bit does not support Windows Live Mesh. Upon further analysis of Process Monitor results, shown in Figure 28, it demonstrates that Cloud Signature used approximately 30.0 MB of memory. The tool successfully detected all file additions, removals, and changes in the Dropbox application. It was also able to detect all web sites accessed by the user that matched the specific keywords. These results were reported to the user in an easy to use HTML format.

![Figure 27 - Windows XP 32-Bit Disk Image Running Time](image)

![Figure 28 - Windows XP 32-Bit Disk Image Resource Result](image)
4.2.2 Windows XP 64-Bit Disk Image

As shown in Figure 29, Cloud Signature took approximately 10 seconds to complete its analysis of the Windows XP - 64-Bit drive. Once again, it is important to note that Windows XP 64-Bit does not support Windows Live Mesh. Upon further analysis of Process Monitor results, shown in Figure 30, it demonstrates that Cloud Signature used approximately 26.3 MB of memory. The tool successfully detected all file additions, removals, and changes in the Dropbox application. It was also able to detect all web sites accessed by the user that matched the specific keywords. These results were reported to the user in an easy to use HTML format.

![Figure 29 - Windows XP 64-Bit Disk Image Running Time](image)

![Figure 30 - Windows XP 64-Bit Disk Image Resource Result](image)
4.2.3 Windows 7 32-Bit Disk Image

As shown in Figure 31, Cloud Signature completed its process in approximately 16 seconds. Upon further analysis of Process Monitor results, shown in Figure 32, it demonstrates that Cloud Signature used approximately 35.2 MB of memory. The tool successfully detected all file additions, removals, and changes in both the Dropbox and Windows Live Mesh applications. It was also able to detect all web sites accessed by the user that matched the specific keywords. These results were reported to the user in an easy to use HTML format.

![Windows 7 32-Bit Disk Image Running Time](image)

![Windows 7 32-Bit Disk Image Resource Result](image)
4.2.4 Windows 7 64-Bit Disk Image

As shown in Figure 33, Cloud Signature completed its process in approximately 12 seconds. Upon further analysis of Process Monitor results, shown in Figure 34, it demonstrates that Cloud Signature used approximately 31.0 MB of memory. The tool successfully detected all file additions, removals, and changes in both the Dropbox and Windows Live Mesh applications. It was also able to detect all web sites accessed by the user that matched the specific keyword. These results were reported to the user in an easy to use HTML format.

<table>
<thead>
<tr>
<th>Started: 2/28/2012 8:29:31</th>
<th>Total User CPU: 00:00:00.1560010</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ended: 2/28/2012 8:29:43</td>
<td>Total Kernel CPU: 00:00:01.5756101</td>
</tr>
</tbody>
</table>

Figure 33 - Windows 7 64-Bit Disk Image Running Time

Figure 34 - Windows 7 64-Bit Disk Image Resource Result
4.3 Requirements Findings

Now that Cloud Signature’s performance has been documented, it is possible to evaluate the tool based on the requirements that were stated in section 1.2.1.

**The tool should support Windows XP 32-Bit, Windows XP 64-Bit, Windows 7 32-Bit, and Windows 7 64-Bit**

Cloud Signature was tested as both a triage based forensic tool and a disk level forensic tool on Windows XP 32-Bit, Windows XP 64-Bit, Windows 7 32-Bit, and Windows 7 64-Bit. The tests proved that Cloud Signature was successful with all four of the operating systems.

**The tool should support Microsoft’s Internet Explorer Web Browser Version 6, 7, 8, and 9**

Cloud Signature was designed to scan all of the directory paths that store temporary Internet files implemented by Internet Explorer Web Browser Version 6, 7, 8, and 9. These web browsers were tested on Windows XP 32-Bit, Windows XP 64-Bit, Windows 7 32-Bit, and Windows 7 64-Bit. It is important to note that Windows XP does not support Internet Explorer version 9. Also, Windows 7 does not support Internet Explorer version 6.

**The tool should have a low memory footprint**

After completing the analysis phase of Cloud Signature, it was determined that the average amount of memory used by tool was 31.0875 MB of memory. This is well below the requirement value of 50 MB.
The tool should be reasonably fast at returning results

After completing the analysis phase of Cloud Signature, it was calculated that the average running time of the tool was 14.125 seconds. This running time is under the requirement value of 1 minute for a standard drive.

The tool should accurately find all supported cloud-computing applications

After completing the analysis phase of Cloud Signature, it was found that the tool detected every file addition, deletion, or modification that occurred in the Dropbox application. When the tool was analyzing a Windows 7 Operating System it was also able to find every file addition, deletion, or modification that occurred in the Windows Live Mesh application. Finally, Cloud Signature was able to parse all corresponding Internet Explorer index.dat files, locate URL entries that contained any instance of a web-based cloud-computing application keyword.

The tool should be easily updated

Cloud Signature has the capability to be easily updated by any software developer. The developer would simply have to add an additional class for the new cloud-computing application. Any user would be able to modify the Keyword file in order to update the specific keywords that Cloud Signature is analyzing for.

The tool should report as much of the information required by a search warrant as possible

Cloud Signature parses all relevant information that assists in the creation of a search warrant. This information includes usernames, email addresses, IP addresses, dates of use, and any applicable filenames. This information is presented to the user in
an easy to read report. The tool also provides the user with the proper contact information for the supported cloud-computing application companies.

**The tool should contain a built-in reporting option**

Cloud Signature writes the parsed information to an HTML web page report. In this report the data is presented in easy to read table formats. For the web-based cloud-computing search, the results are displayed by the Keyword that was detected. The reports also include several HTML features to improve the ability to read the specific portions of the report that are of interest to the user.

**The tool should be easy to use and require minimal training**

Cloud Signature was designed to be very intuitive. The user simply has to check any searches that he/she wishes to perform along with the drive letter to perform them on. The user is then required to choose where the resulting report should be saved. No special training is required to learn how to use this tool.

### 4.4 Next Steps

In order to ensure that Cloud Signature remains a productive tool for law enforcement there are several steps that need to be completed in the future. Since the number of cloud-computing applications is growing at a rapid pace, it is important that Cloud Signature is able to detect the most popular cloud-computing applications being used at that moment. In the future, it would be beneficial if Cloud Signature had monthly updates in order to ensure that the most popular applications were being detected.
Cloud Signature will also need to have program infrastructural changes. These changes correspond to the process of updating the applications that Cloud Signature reports. Currently, Cloud Signature requires a new class to be created for each new application that is added. It would be beneficial to structure Cloud Signature so that the classes are much broader, and then have the classes created dynamically. This would create a situation in which an update could be released that simply informed Cloud Signature as to the locations where to search for the cloud-computing application, the database format that the cloud-computing application is implementing, and any applicable queries to parse data corresponding to the cloud-computing application. This would also give the user the ability to create their own custom version of classes, in which they could determine the location to look, the database format being used by the application, and the queries that the user wants to perform. By implementing this type of structure it would greatly increase the modularity and extensibility of Cloud Signature.

Finally, it will be necessary to check that Cloud Signature is still able to correctly detect and parse currently supported applications. Since the cloud-computing applications are updated at a very rapid and unsystematic pace, it is necessary to ensure that the application is detected and parsed successfully. One particular instance of this deals with the updates performed on the application Dropbox.

When this thesis research was conducted the most recent released version of Dropbox was 1.1.3.5. As of March 16, 2012 the most recent released version of Dropbox is 1.2.5.2. In this upgrade the Dropbox database files that are parsed with Cloud Signature are now encrypted. For this thesis the encryption was not a part of
the original research, but for the future of Cloud Signature this is a vital process. This is an example as to why Cloud Signature would require a strict maintenance and upgrade schedule.
CHAPTER 5: CONCLUSION

The results of this research show that it was possible to create an application that is capable of detecting the presence of cloud-computing based applications on a computational device. Cloud Signature is able to detect the presence of web-based cloud-computing applications, Dropbox applications, and Windows Live Mesh applications. The tool is then able to display any data that may be of evidentiary value to the user in an easy to read format. While Cloud Signature is effective for its intended purpose, there are numerous areas that need to be further analyzed in order to create a commercial grade cloud-computing forensic analysis tool.

The lack of unity between cloud-computing application developers is one of the main issues to be dealt with when creating a tool to detect cloud-computing applications. This lack of unity creates a situation in which every individual application uses its own method of storing data. This results in numerous types of databases and program structure. This requires a developer to research each individual application storage methodologies, and then develop the appropriate code to locate and parse the data. Considering the amount of cloud-computing applications that are present and the rate at which they are being developed, it is not feasible to create a tool that is capable of detecting every cloud-computing application.

Currently, the main computational forensic tools do not address the issue of cloud-computing applications at all. Some tools have the ability to parse temporary internet files, but that only takes care of a portion of the applications that may contain data of evidentiary value. This issue is vital to the future of digital forensics because
many software companies are beginning to implement the cloud as a way to run applications and soon users will begin to store most of their data on cloud devices. The absence of having a tool to detect this information is generating a gap in the digital forensic technology.

In conclusion, this research provides a positive outlook on the fact that it is feasible to create a tool to detect cloud-computing based applications as long as the tool is restricted to certain boundaries. I feel that a tool of this kind is vital to the future of digital forensics. The lack of this tool could be hampering a law enforcement officer’s ability to do his/her job to the fullest of their ability
BIBLIOGRAPHY


