Cloud Storage Client Application Analysis

Rakesh Malik *rxm058@shsu.edu*
Department of Computer Science
Sam Houston State University
Huntsville, TX 77341, USA

Narasimha Shashidhar *karpoor@shsu.edu*
Department of Computer Science
Sam Houston State University
Huntsville, TX 77341, USA

Lei Chen *lchen@georgiasouthern.edu*
Department of Information Technology
Georgia Southern University
Statesboro, GA 30458, USA

Abstract

The research proposed in this paper focuses on gathering evidence from devices with UNIX/Linux systems (in particular on Ubuntu 14.04 and Android OS), and Windows 8.1, in order to find artifacts left by cloud storage applications that suggests their use even after the deletion of the applications. The work performed aims to expand upon the prior work done by other researches in the field of cloud forensics and to show an example of analysis. We show where and what type of data remnants can be found using our analysis and how this information can be used as evidence in a digital forensic investigation.

Keywords: Cloud Storage Forensics, Cloud Application Artifacts, Data Remnants, Data Carving, Digital Forensic Investigations.

1. INTRODUCTION

As we all noticed, in the last decade there was an exponential increase in the use of cloud computing [1]. Unsurprisingly, this also led to an increase of cybercrime that involves the cloud infrastructure, and therefore arose the need of cloud forensics [2]. This increase brought to light many issues and challenges for digital forensics experts. In fact, the National Institute of Standards and Technology (NIST) identified 65 of these challenges that need to be addressed [3]. There are two means to connect to one's cloud storage account. The first is through the use of a web browser and the second through a client application installed on the user's device. This research will analyze some of these client applications and provide some examples of how to gather data. Since the previous work in this field analyzed applications such as Microsoft OneDrive, Google Drive and Dropbox on a Windows 7 [4] [5] [6] [7], this research will show an example of client application analysis on a Windows 8.1 operating system virtual machine created with VMware Workstation 10. In addition, to diversify and provide different examples, the cloud storage services that we selected are different from the one chosen by the authors in previous work. Nevertheless, after testing these applications, we found that they represent valid alternatives to the most known cloud storage services. These alternatives are Copy [8] and ownCloud [9]. Dropbox was also analyzed, since unlike Copy and ownCloud it encrypts the files that contain evidence, and it was interesting to compare the two different scenarios. To decrypt some of the files, Dropbox Decryptor by Magnet Forensics was used [10]. No work so far has been done on UNIX/Linux, even though UNIX/Linux-based operating systems are commonly used as well, especially when providing applications on a server. The research proposed in this paper expands upon the work done by prior research. The chosen operating system is one of the most known debian-based Linux distributions, Ubuntu. It is very likely that in the most of the

cases, a user downloads the client application on multiple devices. For example, the client application can be downloaded on both a PC and a smartphone. Therefore, a second objective of this research was to perform the gathering of evidence on a smartphone, in particular on Android OS. There are multiple cloud storage services options on UNIX/Linux, however, not very surprisingly, Microsoft OneDrive does not provide a client, and very surprisingly enough, at the time of this writing, an official client of Google Drive has not been released for UNIX/Linux yet. The work in this research has been therefore divided in two main parts. The first on Windows 8.1 and the second on UNIX/Linux. Finally, in the Conclusions and Future Work section, we will see and compare the result between the two different operating systems.

2. PRIOR WORK

The following literature review, explores the procedures and approaches used by other researchers in this particular field. Three main prior and related researches were analyzed to discover the approach taken in order to collect artifacts. Artifacts collected can be files either accessed or modified by the cloud storage applications on the client devices, or artifacts related to web-based cloud storage services (which are accessed through a web browser). Two main approaches were identified. The first approach represents a presumption of where artifacts should be located on a device, and then perform a search in those specifics locations, based on the examiner's knowledge. Meanwhile, the second approach is based on the use of programs and tools, such as Process Monitor from the Sysinternals Suite [11] to determine the location, in a dynamic manner, of the artifacts and data remnants. All the prior work done in this field was performed on a Windows 7 system using virtual machines. The following is a brief discussion of the prior work.

The paper by H. Chung, J. Park, S. Lee and C. Kang [4] provides a procedure to investigate devices such as PCs and smartphones. According to this procedure, the investigator collects and analyzes data from all devices that a user has used to access a cloud storage service. Based on the type of the device that is being analyzed, the procedure can take a different approach. Simply put, if the device is a PC then it is very important to collect volatile data from physical memory (if live forensic analysis is possible) and nonvolatile data such as files, directories, internet history, and log files. Physical memory contains useful information about users and their activities. For example, physical memory can contain login attempts and login credentials used to access cloud storage accounts through a web browser. If the device is instead a smartphone, and if the system is running Android OS, after rooting, it is possible to collect data from the main system folders. In the case of an iPhone, after connecting the device to a PC, important data of user activity related to the cloud can be found in backup files or in data synchronized with iTunes. Once all data is collected an analysis in order to find useful artifacts is performed. According to the paper, cloudstorage services can be web-based services accessed through a web browser or client applications installed on the device. In the first case, it is essential for an expert to analyze data such as web browser log and database files (cache, history, cookies, and downloaded files) that are stored in the user profile directory on a Windows system. Cache files include downloaded image files, text files, icons, HTML files, XML files, download times, and data sizes. History files contain visited URLs, web pages titles, visit times, and the number of visits. Cookie files store information about hosts, paths, cookies modification times, cookies expiration times, names, and values. Download lists include local paths of downloaded files, URLs, file size, download times, and whether downloaded files were successful. Through such web browser files an expert can identify a user's activity, including access or logins to a cloud storage service. However, when a client application is installed on a Windows system, traces of it are left in the registry, log files and database files. Mac systems have similar traces except registry files. These files are essential during a digital forensic analysis, since they provide proof of the use of a cloud storage service. These log files contain information such as logins attempts, if and when services were used, and times of synchronized files. Database files contain information about synchronized folders and files (creation times, last modified times, and whether files were deleted) on a PC. All this information can be used to create a timeline of the user activity. In a smartphone device traces are left in database files, XML files, and plist files (which contain information about a user account). Finally, the rest of the paper provides examples of forensic analysis and shows where

data is found on a PC or a smartphone. The cloud services that were used in this work, are Amazon S3, Dropbox, Google Docs, and Evernote.

In the research work done by M. Katz and R. Montelbano [5], to obtain the locations of artifacts, Process Monitor is used. The result is filtered to show the file system activity, and changes to the registry and files. The cloud storage applications used in this research are SkyDrive (now OneDrive), Dropbox, and Google Drive. When SkyDrive was installed 4959 artifacts were either created or modified. Presence of the files modified using the client, was found in unallocated space, $Recycle.Bin CSV files, pagefile.sys, and inside the AppData folder. In the case of Dropbox installation, 4163 artifacts were either created or modified. Evidence of deleted files was found in unallocated space and in pagefile.sys. During Google Drive installation, 9438 artifacts where either created or modified. Evidence of files modified or deleted was found in unallocated space, $Recycle.Bin CSV files, pagefile.sys, and configuration files. The result of this research proved that a large number of files are affected during the installation of the application, and a large number of files are left behind, once the uninstallation process is completed. Evidence of file manipulation was mainly found in the unallocated space, $Recycle.Bin CSV files, and pagefile.sys. The type and number of artifacts varied depending on the application, but evidence of the use of the cloud application was still present after uninstallation of the client in all the cases.

The following research, performed by D. Quick, B. Martini, and R. Choo [6], provides a well formatted methodology and a very exhaustive analysis of data remnants left by cloud storage applications. This research is performed on a Windows 7 machine and the cloud storage services analyzed are Microsoft SkyDrive (again, now OneDrive), Dropbox, and Google Drive. The objective of the research was to solve questions, such as, which data remains on the hard disk after a user used the client software? Which data is left once the user has had access to the cloud storage through a web browser? What is the location of the data remnants on the operating system and in the memory? Other questions that were attempted to answer relate to network traffic data and smartphones. Based on the work of this research, artifacts of files either access or modified, and data remnants left behind by the applications are found inside prefetch files (which are used to analyze the software activity, such as the number of times the software has run or the associated files used by the application), registry files (they can contain references, activities, settings or other information), link files (files' shortcuts), thumbnails pictures within the thumbcache, event logs (which contain information relating to system, software, and other events recorded by the operating system), and finally, directory lists file ($MFT files). Forensic analysis has been also performed on the memory, $Recycle.Bin (in order to find deleted files), client applications (analysis of installation path, sample files, synchronized files and folders), on the account accessed through a web browser (can contain information about the number and the type of devices used to access the storage space), and finally, on the files related to the browser. Network traffic was also captured and analyzed to find activity related to login sessions. To conclude, data carving was performed through allocated and unallocated data. Thumbnails icons and large size pictures were recovered. This research, used a dynamic approach: tools to dynamically find evidence that were used were Process Monitor, Wireshark, among others. Another research was performed by M. Epifani et al [7], on Microsoft SkyDrive (OneDrive), Google Drive, Dropbox, and iCloud. Again in this case, the collection of artifacts left behind by the applications was performed on a Windows 7 system. To track the disk usage DiskPulse was used (to determine information related to created, modified and deleted files), Regshot, and RegFromApp were used to track registry changes. By monitoring the registry changes, researchers were able to obtain installation locations and installed client applications versions. Other useful data was collected from configuration files present in the installation folder (inside the user profile), from online accounts (information about deleted files, devices connected to the account, version history for every file, and last browser sessions), from the memory (it can contain user email, display name, filecache.dbx path, server time, file list, deleted files, username and passwords in the case of a web-based storage access), from Hiberfil.sys and pagefile.sys, link files, browser history and cache, registry point, and volume shadow copies. As we can see, this research collected evidence from the same locations as the previous researches.

To conclude this literature review, we can assert that the procedures and approaches taken in these prior research works are in concordance with each other. Even if the approaches were of two different types, the locations analyzed and the data remnants found were similar.

3. METHODOLOGY FOR WINDOWS 8.1

To perform this research and to gather data, both a static analysis and a dynamic analysis are used. For example, the directory where all the information related to Google Chrome can be found (databases, history files, cookies and so on) are analyzed in a static way. The database files were accessed through SQLite Browser [12]. Then, the client application is installed. During the installation, Process Monitor and Process Explorer are executed to find the locations on the system that stores the main files and directories. It is possible to use the Process ID (PID) of the process during the installation in Process Explorer, and use it as a filter parameter in Process Monitor. Once the useful data is gathered during the installation, the application is removed and the folders that were created during the installation process are controlled. To see if there is any data remnant left, the registry is scanned with RegScanner [13] searching for strings that may indicate the presence of evidence. In the section called "Recovering Deleted Files", we will see that data carving from the client applications is possible. To confirm this possibility, the TheSleuth Kit [14] will be used.

4. MAIN RESEARCH FOR WINDOWS 8.1

As pointed out in the previous work section, there are two ways to connect to the cloud storage account. The first way is through a Web browser, while the second one is through a client application. It is important, therefore, to gather information in both cases. This sections contains many different subsections that explain where evidence can be found on both the web browser (Google Chrome), and in the main directories of client applications. Finally, the physical memory contains a great source of useful data. Therefore, a subsection is dedicated to the volatile information found in the memory.

4.1 Web Browser Analysis

C:\Users\<User_Name>\AppData\Local\Google\Chrome\User Data\Default is the main directory that contains Google Chrome files where evidence of accessing the cloud storage space through Chrome can be found and contains the following described files. The table "*cookies*" inside the database file *Cookies*, stores the creation times of the cookies, the host names, the names of the cookies, the expiration times, the times of last access, and the encrypted value. The host name field of the table contains the name of the website accessed, and here it is possible to find evidence: in fact, some values are *.dropbox.com*, *.google.com*, *.copy.com*, *onedrive.live.com*, and *.owncloud.com*. The creation times shows when the website was first accessed by the user, and the time of last visit. The database file *Favicons* stores icons associated to websites, along with URLs of the *favicons*. Among these URLs it is possible to find the cloud storage server address plus the *favicon* path. One of the most interesting files in the directory above mentioned, is the *history* database file. The table *urls* in this file stores the URL of the web pages visited, the titles of the pages, the number of visits and the last visit times. The table *downloads* contains the names of the downloaded files, the target paths on the local system, the start and end times of download, the received bytes and total bytes counts, the servers addresses, the last modified times, among many other useful information. The *keyword_search_terms* table contains the keywords typed inside the browser search bar. The *Login Data* database file stores the logins attempts and the server logins file paths, the types of the usernames (for example, a string username or a *login_email*), the passwords types, the encrypted passwords value, the timestamps and other information that can be useful during a digital forensic analysis.

The *Network Action Predictor* database file contains the URLs visited by the user. In order to load web pages faster, based on the input text in the search bar typed by the user, the browser tries to predict which webpage should be opened [15]. The *QuotaManager* database file can contain reference to the same URLs. The *Web Data* database file stores the *autofill* table, which logs usernames and other credential values and personal information, such as the location, address or phone number of the user. The files *Current Session*, *Current Tabs*, *Last Session*, and *Last Tabs*

could store a great source of valuable information since they collect all the sessions started and the tabs opened by the user. This is in order to restore the *sessions* and *tabs* in case of an unexpected crash of the browser. Other files that may contain evidence are *History Provider Cache*, which appears to contain random strings and references to the user activity.

Once useful artifacts have been looked through the web browser files, it is important to analyze the memory to see if it contains references to the user activities on the cloud storage accounts by using the web browser. It was possible to find evidence inside the memory by simply taking a snapshot of the state of the virtual machine, and then performing a keyword search (after importing the *.vmem* file in a Linux machine, Ubuntu 14.04 64-bit) using the commands *strings* and *grep* to filter the output. Inside the captured memory there is many useful information. In fact, it is possible to locate in plain text user's credential such as email addresses, usernames, user IDs, first names and last names of the user, logins attempts, timestamps of logins, paths on the server, server names or server addresses, lists of files accessed, creations times, times of last synchronization, uploaded files and sizes, modification times, deletion times, messages and actions taken by the server. For example using the keyword *login*, it is possible to see that a login attempt has been made for both Dropbox and OneDrive.

4.2 Client Application Analysis: Copy

Launching *Process Explorer* when installing the Copy client application and when running it, we can easily find where Copy stores the main files. In our case, the files are located in *C:\Users\<User_Name>\AppData\Roaming\Copy*. The same result can be confirmed by using the *PID* found in *Process Explorer* and using it as a filter parameter in *Process Monitor*.

The following are the important files found during the analysis:

- *config.db*: this file stores settings such as the user email, first name, last name, and user ID. It also contains the path of Copy's root directory and the root cache, among other settings.
- *trace.txt*: this log file's entries contain information regarding the hosting machine (operating system, host information, etc.), the client application, and the server.
- *synclog.txt*: this is another log file and it stores the operations executed by the application along with timestamps and other information relative to the operations types. Some operations are authentication attempts and file manipulation (such as upload, download, modification, and deletion).
- *copy <User_Email>.db* (in our case the name of the file is *cloudstorage.test.mail@gmail.com.db*): this file is very interesting from a forensic point of view since it contains the list of files and metadata stored in the root directory of Copy.

After uninstalling Copy, the main root folder is still present on the hard disk, as well as all the files contained in the folder. The folder *C:/Users/<User_Name>/AppData/Roaming/Copy* is still present, however *config.db*, *trace.txt*, *synclog.txt* and *copy cloudstorage.test.mail@gmail.com.db* have been deleted. This means that after the uninstallation of Copy, evidence of use is still present on the system and some user activity can be determined. Finally, by using RegScanner we performed a search based on the string *Copy*, and it was possible to find some registry keys left once is the application is removed and uninstalled. This strengthens the possibility for a forensics expert to find evidence related to the use of the client application.

ownCloud

After installing and executing ownCloud for the first time, the output of *Process Explorer* is checked. The main directory of ownCloud is stored in *C:\Users\<User_Name>\ownCloud*. In this directory the most interesting files are hidden, and are:

- *.csync_journal.db*: this database file contains the metadata for the synchronized files. It also stores in the table *downloadinfo* and *uploadinfo*, information regarding downloaded and uploaded files.
- *.owncloudsync*: this log file keeps track of the user and the application activity. Some fields of the log entries are timestamps, duration of the actions, files involved, types of instruction, working directories, size of files, file IDs, modification times, and so on.

During the uninstallation of the ownCloud client application, the uninstaller asks the user whether the ownCloud root folder should be deleted or not. If it is not deleted, then the files, database files, and log files will still be available for analysis. Otherwise, they will be deleted. However, if the hard disk has not been wiped, or the unallocated space overwritten, recovering these files is still possible. This will be shown in one of the subsections of the Main Research section. Inside the registry, after the client application is removed, there are not many references to ownCloud. However, some traces are still left. In fact, the string ownCloud is still present in the *AUTORUN* key.

4.3 Client Application Analysis: Dropbox

During Dropbox installation and execution, we started *Process Explorer*, which shows in the lower pane, the files that Dropbox opens, reads and writes to. Unlike Copy, Dropbox encrypts the configuration and database files, and does not release the decryption keys to the users. Due to this reason, it is difficult to open and analyze these files. However, during a digital investigation, and with the use of a proper warrant, investigators might be able to obtain the encryption keys from the cloud service providers. Nevertheless, as the time of this writing, there is one particular tool that allows to decrypt some of the files of Dropbox. The tool name is *Magnet Forensics Dropbox Decryptor* (see the references for more information). To decrypt .dbx files (the encrypted files) with *Magnet Forensics Dropbox Decryptor* the Dropbox's .dbx file, the output folder where the decrypted files will be stored, the Windows protection folder, the value of the registry key *HKEY_CURRENT_USER/NTUSER.DAT/SOFTWARE/DROPBOX/KS/CLIENT*, and the user Windows's account password should be specified within the fields of the tool.

The most important files that can be analyzed during a forensic investigations are stored in the folder *C:\Users\<User_Name>\AppData\Roaming\Dropbox* and are contained in sub-directory *instance1* and are:

- *config.dbx*: this file is one of the encrypted files, and after it has been decrypted with *Magnet Forensics Dropbox Decryptor* it is possible to see that it stores the host IDs, the user's email addresses, the Dropbox root folder paths, among other settings.
- *filecache.dbx*: the table *file_journal* stored in this file contains the server paths, the files lists and the files names, the sizes of the files, the modification and creation times.

When Dropbox is uninstalled, the Dropbox root folder is still present on the disk, as well as all the files contained in it. The folder, *C:\Users\<User_Name>\AppData\Roaming\Dropbox* is still present, however, the encrypted files have been deleted. There are also many registry keys and traces left once the uninstallation of Dropbox is completed, that can be used as evidence of the use of client application.

4.4 Deleted File Analysis

One important challenge that arises during a digital investigation, is the recovery of deleted files. The unallocated space can store valuable information for an investigator and cannot be ignored during an analysis. There are two methods to recover files deleted from the cloud storage account: the first consists of recovering a deleted file from the server-side, while the second from the client-side. In fact, both Copy and ownCloud client applications have a feature known as *undelete*. This feature restores a deleted file both locally and on the server. The Dropbox client does not have a similar feature, however, it allows to restore a deleted file or a modified file, once the account has been accessed with a web browser and not through the application like Copy and ownCloud. Dropbox also allows to permanently delete a file, in a way that it cannot be restored.

In order to recover deleted files on the client, some basic understanding of the *NTFS* structure is necessary. The *NTFS* file system maintains a table known as *MFT* (*Master File Table*), in which each folder and file stored on the file system have an entry. The entries in the *MFT* table describe the files metadata information and contain pointers to the clusters that contain the file's data content. When the files are saved onto the hard drive, both the entries inside the *MFT* and the clusters that store the data are allocated [16].

To attempt a recovery of deleted files, we decided to use TheSleuth Kit, which is installed by default on a Kali Linux operating system. To attempt the deleted files recovery, we can either image the Windows 8.1 operating system, or use VMWare Workstation to mount the hard drive as read-only on the Kali Linux virtual machine. We chose to use the second option. Once the hard drive is mounted, the command *ls –ai*, allows to list all files on the terminal, even the hidden ones, along with their *inode* address. The *inode* address is an EXT file system concept, but it basically has the same function as the *MFT* entry. Using *ls –ai* we were able find the entry number for a file inside the *MFT*. The *istat* tool from the TheSleuth Kit prints in output the metadata information, described in the *MFT*. In this output, the the clusters numbers are specified. This clusters numbers are the pointers contained in *MFT* that point to the data content. If we open the logical hard drive with a *hex editor* tool, such as *HxD* [17], we can prove that those clusters really contains the file's data. The hard drive is organized in sectors, so a conversion from cluster to sectors is needed. Since in this version of Windows 8.1, there are 8 sectors per cluster, we can use the following equation to find the sector address:

$$\text{Sector} = \text{Cluster} * \text{Sectors\_per\_cluster} \quad (1)$$

Inserting the cluster number and the number 8 (sectors per cluster) in (1) we obtain the sector number, or the sector address. When opening the logical hard drive in *HxD*, this editor allows to jump directly to the specified sector. This will confirm that the sector do in fact contain the data content and it is allocated.

Going back to Kali Linux, now it is possible to use the *blkstat* tool from TheSleuth Kit, which allows to see the allocation status of a specified data unit (in this case a cluster). As expected, the clusters are allocated. The next step is to un-mount the hard disk, delete the file that was contained in those clusters, and then mounting the drive again in Kali Linux, as read-only. We run again *istat* on the same entry address and this time the output is different, since the entry header says that the file is not allocated. Even if the file is deleted, the entry in the table is not deleted and it still contains the metadata information of the deleted files. When a file is deleted, a Boolean value is changed so the file system knows that the file is deleted and it is now unallocated. However, the *MFT* entry still points to the clusters that contain the file's data content. By running the *blkstat* tool from the TheSleuth Kit toolkit, the output is also in this case different since now the clusters are not allocated. Therefore, it is possible to deduce that the file is now in the unallocated space. In addition, one can also deduce that when a file is deleted, both the entry and the clusters still contain data (the file system can thought to be "lazy", since it does not wipe the content of the cluster). Unless the clusters that contain the file's data content are wiped by a user, or the clusters are over-written by the operating system, when a different file is stored on the drive, the data content of the deleted file is still recoverable. A simple tool used to recover the content inside the clusters is *icat* from the TheSleuth Kit. By using *icat*, the content of the clusters is printed on output on the terminal. It is possible to redirect the output to a file and analyze the file. The *icat* tool from the TheSleuth Kit is not the only tool an expert can use. There are plenty of recovery tools, such as Foremost and Scalpel and they both represent excellent tools. Finally, to prove that our deductions were correct, we open the drive again with a *hex editor* and then access the first sector of the file, and confirm that the content is still there. This deleted file recovery demonstration was executed on a file deleted from the Copy root folder, however, this method works for every cloud storage client application installed and tested in this research.

4.5 Physical Memory Analysis

To analyze the memory, a snapshot of the Windows 8.1 virtual machine was taken and a keyword search on the *.vmem* file was performed. Inside the memory there is a great deal of useful information and evidence. However, performing a live analysis is not always possible, so there is a possibility that memory cannot be analyzed when performing a digital forensic investigation. Nevertheless, if a live acquisition of the memory is possible, then memory should be acquired. Evidence that can be found inside the memory can include emails and user credentials, personal details, passwords, file lists and file's information, accessed files and folders, processes information, processes instructions, host names, loaded libraries and modules, libraries imported

through the server, temporary files, accessed database files and log files, log files entries, authentication attempts, and so on.

5. METHODOLOGY FOR UNIX/LINUX

To perform our research, different programs and tools were used. *VMWare Workstation 10* was used to create a virtual machine of *Ubuntu 14.04 ("Trusty Tahr")* 64-bit. Once the installation was complete, a snapshot was taken in order to revert to a clean state once the analysis of the client application was concluded. After the client application was installed, a keyword search on the system was used to find the locations of the files and other artifacts. Specifically, the command "*find / -print | grep –i 'keyword'*" was executed on the terminal, and the output was filtered to gather paths, files, and other artifacts. Unfortunately, there is not a version of *Process Monitor* (from the *Sysinternals Suite*) for Linux. Nevertheless, the command *lsof* (list open file) can be executed to list files opened by the client process, by using its PID. The tool used to acquire memory is *LiME* [18]. In addition, a keyword search (using for example login credentials) on the memory was performed in order to find artifacts.

Ubuntu 14.04 64-bit uses *Ext4* as a default file system and to find where the locations of files on the file system, *TheSleuth Kit* was used. In particular, the *istat* tool is useful to find *inode* information, status of directories, and status about files inside the main directory that are created by the cloud storage application in order to synchronize files with web-based the account. The *istat* tool was also used to determine the group to which the *inode* belongs, and to find the block address where the content of the data is stored. Using the *dd* command it is possible to create an image of the partition or, as explained later, to image ranges of blocks and groups. The images can be viewed with a *hex editor*. *Foremost* [19] is an example of a data carving tool that makes it possible to recover deleted files. Finally, *SQLite Browser* was used to open database files created by a web browser, such as *Mozilla Firefox,* or by cloud storage applications. The *Android OS* analysis was performed on an *Android OS x86 RC1* virtual machine.

6. MAIN RESEARCH FOR UNIX/LINUX

As pointed out in the previous work section, there are two means to connect to the cloud storage account. The first is through a web browser, while the second is through a client application. It is important, therefore, to gather information in both cases. This section contains many different subsections that explain where evidence can be found on both Mozilla Firefox (which is installed by default on Ubuntu operating systems), and in the main directories of the client applications. As previously seen, another source of useful information is found on a running system inside the physical memory. Therefore, a subsection is dedicated to the volatile information found in the physical memory. One subsection will also briefly describe where other artifacts can be found on the file system for all the applications and, finally, the last two subsections are dedicated to the possibility of recovering deleted files, and the gathering of data on Android OS.

6.1 Web Browser Analysis

All the web-based accounts for all the applications were accessed through Mozilla Firefox and, after operations such as uploading, deleting, and modifying files, the memory was captured and analyzed. Inside the memory it is possible to find in plain text users' credentials such as email addresses, usernames and passwords in plain text, user IDs, first names, last names, logins attempts, timestamps of logins, paths on the server, server names or addresses, references to files accessed, timestamps of creation times, data of last synchronization, files uploaded, files sizes, files modification times, files deletion times, messages and actions taken by the server.

Accessing the hidden folder inside the user's home directory, by default named *.mozilla*, it is possible to find database files that store useful information. The database file *content-prefs.sqlite* contains preferred websites, such as the cloud storage websites accessed. The file *cookies.sqlite* stores cookies that are created when the cloud storage website was accessed. The *formhistory.sqlite* database file contains the history and tracks the websites visited by the user, the number of visits, along with the last visited time. The database file *places.sqlite* contains multiple tables where evidence of the use of the cloud storage services websites can be found.

For example, the table *moz_places* stores the name of the server, paths, and files accessed on the server, while the table *moz_hosts* and *moz_favicon* store hosts names and icons paths. Finally, the file *login.json* stores logins attempts, hostnames, usernames, URLs of the forms in which credentials were inserted, encrypted passwords, and MAC times. Inside the user's home directory, the hidden directory *.cache* contains thumbnails that show pictures of websites and accounts accessed.

6.2 Client Application Analysis: Copy

When the Copy client application is executed it launches a process is called *CopyAgent*, which functions as a background process and enables the application to properly run and execute its operations. Once the location of the data remnants was found, by sorting through the artifacts, it is possible to determine that the main evidence resides in the Copy main folder and the hidden folder *.copy*, both present in the user's home directory. The Copy main folder contains the files that are synchronized with the account, a hidden folder called *.copy-cache*, which contains temporary files (references to these files were found during the memory analysis), and a hidden text file called *.user_info*, which does not contain any useful information. The hidden *.copy* folder contains two directories: *cache* and *resources*, however they are both empty. The most interesting information is contained in the following files: *config.ini* contains the host *UUID*, which is an alphanumeric string of 32 characters; the file *config.db* contains settings, such as the root cache path, the database version, the cloud server address, the root folder path, the authentication token, the client ID, the user ID, the user first name, last name, email address, the push token and push URL, among other settings. The file "*copy <user_email>.db*" (in this research the email address used is *cloudstorage.test.mail@gmail.com.db*) contains a list of files that are uploaded on the cloud storage space or the files that are synchronized with the local folder. The *file* table in this database, contains the files metadata such as the file path, name, parentID, volumeID, inode address, attributes, mtime, rstate, ctime, size, and child count. Other tables in this database contain more useful information such as the owner ID and the file fingerprint. The *synclog.txt* file, logs the application operations such as logins attempts, uploaded files, deleted files, and modified files, along with other timestamps. Finally, the file *trace.txt* stores information regarding the user, the process (*CopyAgent*), the operating system, and the server. Another Copy configuration file is contained inside */home/rakesh/.config/Barracuda Networks, Inc\Copy.conf*. Once the client application is removed, the *find* command revealed folders and files left on system. Since *CopyAgent* was never installed but simply run from the Copy client downloaded folder, once it is removed, it will not delete the Copy main folder; however, a user can simply delete the Copy main folder found in user's home profile. If the user is not aware of the hidden *.copy* folder, this will remain on the system, along with all the information contained, which represents a great deal of useful information during a forensic analysis. Even if the Copy main folder is deleted, both the inode entries and the data content are still present on the hard disk, and unless it is wiped or the data is overwritten, recovery of deleted files is possible (see *Recovering Deleted Files subsection*).

6.3 Client Application Analysis: ownCloud

Once the client application of ownCloud is installed, the main folder is by default present in the user's home directory. Unlike Copy, there is no hidden folder, but hidden files are present in the client main folder inside the user's home directory. This folder contains the files that are synchronized with the account, along with the hidden database file *.csync_journal.db*. The table *metadata* of the database file, stores valuable information regarding files, such as paths, inode addresses, UIDs, GIDs, MAC times, md5 hashes, among other information,. The table *version* contains the version of the client application. The hidden file *.owncloudsync.log* is a log file that stores timestamps, operations executed by the application (instructions), the length of the operations, the names of the files involved, MAC times, sizes, the file IDs, among other useful information. The configuration file */home/<user name>/.local/share/data/ownCloud/owncloud.cfg* contains the URL of the ownCloud server, the username, authentication type (http), and other settings. After removing the application, the files on the local system will still be present. Therefore a user has to manually delete the ownCloud main directory. Unfortunately, from a forensics point of view, the database containing the metadata and the log files, even if are hidden, are removed with the directory, that includes also the stored files. However, inode entries and

data content are still present on the hard disk, so as for Copy, unless it is wiped or overwritten, recovery is still possible.

6.4 Client Application Analysis: Dropbox

Dropbox, unlike Copy and ownCloud, encrypts the database files and log files for security purposes. The encryption keys are not released even to the user. However, a tool exists for Windows versions that decrypts these files. Unfortunately, this tool does not have a counterpart for UNIX/Linux, nevertheless, according to the Magnet Forensic team, a version of the decryptor should be available for UNIX/Linux in the future [20]. Once the application is installed and the account is set, the Dropbox main folder is created in the same location as Copy and ownCloud, which is the user's home directory. The main Dropbox folder maintains a hidden cache directory that stores the files synchronized with the account. In addition to the main folder, there is a hidden folder named *.dropbox* inside user's home directory, which contains multiple interesting files and sub-directories: a file named *info.json* stores the path of the dropbox root directory and the file *dropbox_pid* that contains the Process ID, the sub-directory *instance1* stores all the encrypted files so it is not possible to collect evidence unless the files are first decrypted. One of these file, is *aggregation.dbx,* contains timestamp values, server paths, and a blocklist value. The encrypted files are *config.dbx* (contains configuration settings, user, host machine and server information), *deleted.dbx*, *filecache.dbx*, (contains files metadata), *notifications.dbx*, *sigstore.dbx*, *PENDING_cHDqrT*, *TO_HASH_3WP99a*, and *UPDATE_XyrFxy*. The folder *instance_db* stores another encrypted file named *instance.dbx*. When uninstalled, Dropbox does not delete the main Dropbox directory in the /home/rakesh folder. Even if the folder is deleted, a user might be unaware of the presence of the hidden folder *.dropbox*, and this will be helpful if the database files and logs can be decrypted. Like Copy and ownCloud deleted file are possible to recover.

6.5 Physical Memory Analysis

As mentioned in the Methodology section, physical memory was acquired using LiME and it was analyzed with keyword searches. Volatile evidence found includes names of files present in the application's main folder, recently accessed folders, name and client application main process information, process instructions (for example, in the case of Copy, *copy-sync, copy-update, copy-paused, or owncloud_init, owncloud_commit, owncloud_opendir* in the case of owncloud, etc.) icons paths, hosts names, loaded libraries and modules used by the process, libraries imported through the server, temporary files created by the client application main process in the respective application cache folders, database files accessed, log files accessed, log files entries, logins attempts and credentials (for example, during the memory analysis after the use of Copy, the string *–Login-success U: 'cloudstorage.test.mail@gmail.com'* was found.

6.6 File System Analysis

Several artifacts are found in various locations of the file system. However, most of these files are not useful during a digital forensic investigation, but to only demonstrate the installation and use of the client applications by the user. The directories */usr/share/man* and */usr/share/doc* on Ubuntu contain the man pages and documentation of the applications, while the directories */bin/* and */usr/bin* store executables or daemons files used to launch the applications. A great number of icons related to the cloud storage applications are found inside */home/<user_name>/.icons*. Information can also be found inside systems logs: for example, the log auth.log inside */var/log* stores command issued on the terminal, along with the working directory and it can be used to analyze the user recent activity. The log file */home/<user_name>/.bash_history* contains a list of the recently commands issued by the user. The *dpkg.log* contains references to packages installed on the system, and it can reveal if any cloud storage client application was installed from the commandline. Other possible locations that may contain useful evidence are the folders */home/<user_name>/.local/share/Trash,* which stores deleted files, and */home/rakesh/.cache/thumbnails*, which contains thumbnails associated with the client applications. If an application during its execution should crash, the *syslog* and *kern.log* files found inside */var/log* store references of the crash. The log file *history.log* contains the recently installed packages and the command issued to install them, while *term.log* contains libraries used to install the applications.

6.7 Deleted File Analysis

It was well clear and established in the Prior Work section that it is possible to find artifacts of cloud applications inside the unallocated space. Therefore, it is possible to recover the data stored inside this space. This subsection will be dedicated to the recovery of deleted files, which is a process known as data carving. Recovering deleted files can be done using two different methods. The first is to recover the deleted files from the server, but it requires the application to be logged in, or the account to be accessed through a web browser. Copy and ownCloud applications both allow the user to recover deleted files with a feature known as '*undelete*'. Dropbox does not have a similar feature, but it is possible to recover deleted files or previous version of files on an account accessed through a web browser. However, unlike Copy and ownCloud, Dropbox allows a permanent deletion option, that will permanently delete a file from an account. The second way to recover deleted files is through the use of a data carving tool on the local device. All the client applications analyzed in this research, when they first synchronized with the server, downloaded a copy of each file and stored them locally. As mentioned earlier, even if the files are deleted, the *Ext4* file systems does not delete the *inode entry*, so both the *inode entries* and the files' *data content* are still present on the hard drive. To see how the recovery is done, first we have to analyze how the file are allocated. Using the *istat* tool from the *TheSleuth Kit* an examiner is able to find the *inode addresses* of the files, the *group* to which they belong, and the *block address* of where the *data content* of the files is stored. The files are usually (but not necesseraly) stored in contigous *inode addresses*. Using the *blkstat* tool on the *block addresses* that contain the files' *data content*, the output shows that the *blocks* are in fact allocated. By executing the *fsstat* tool it is possible to find the *inode range address* of the *group* and the range of the *blocks* where the *data content* is present (usually stored in a different group). Using *dd* on these ranges it will allow a forensic expert to create a small image that can be further analyzed with a hex editor. Now, we can examine if the files are still present once they are deleted. After deleting the files, *blkstat* will return that the *blocks* that store the *data content* are not allocated anymore. However, by opening the images with a hex editor, it is possible to determine that the data is still present inside the *block*. In addition, the inode entries are still present (a character has been added before the file name to indicate that the files pointed by the inode have been deleted). At this point, using a simple data carving tool such as *Foremost* or *icat* from the *TheSleuth Kit*, recovering the deleted files is fairly easy.

6.8 Android OS

A common feature of cloud storage services is that they allow multiple devices to be synchronized. Therefore, it is very likely that one user will synchronize the same cloud storage account on multiple devices. Thus, in order to gather as much data as possible, it is important to analyze not only workstations, desktops, and laptops but also devices such as smartphones and tablets. An operating systems that comes as default on many smartphone is Android OS. Client applications were both available for Copy and Dropbox, however, an official client at the time of the writing of this paper is not available for ownCloud. Nevertheless, an unofficial but very efficient client application has been published by the BezKloboukuNos team [21]. In the case of Copy the main files are found inside the folder */data/media/0/Android/data/com.copy/files*. Among these files, the *configuration.ini* file contains different configuration and account settings, while the file *copy.db* stores names of the synchronized files along with the metadata of the files. The log file *trace.log* stores information regarding logins, synchronized files, timestamps, account information, and the client application's process operations such as uploading, deleting and/or modifying files. The *download* directory inside the *temp* subdirectory contains the files that were locally saved, while the directory *thumbnails-cache* contains thumbnails. The file *com.copy_preferences.xml* in the directory */data/data/com.copy/shared_prefs* stores other useful data such as the last opened file along with its timestamp. When the Copy application is deleted from Android OS, all the files are also deleted. However, information about the use of the application can be found in logs, such as *logcat*, which contains the system recent activity. Operations and actions executed by the Copy application, files and directory accessed were found inside *logcat*. ownCloud does not have an official client, but as mentioned above, an unofficial client is available. Once the application is installed, the directory */data/media/0/owncloudApp* stores the sub-directory *owncloud_user@server_address*, which stores files that were downloaded locally on the device. The database file named *filelist* in the

folder */data/data/com.owncloud.androidApp/databases/* stores the list of the files synchronized along with the type of the file, the username, server name and address. The file *com.owncloud.android/App_preferences.*xml stores as well, the username and server address. When the application is removed, the directory */data/media/0/owncloud* still contains the files that were locally stored, along with other temporary files. The rest of the files created by the application are deleted. Also in this case, recent activity can be found in logs such as *logcat.* The directory */data/data/com.dropbox.android/* stores local files, and the subdirectory *databases* contains database files. Unfortunately, also in the case of Android OS most of these files are encrypted. A file named *361753330-db.db* inside the directory, stores the names of the synchronized files, and the file *prefs-shared.db* stores account preferences. Finally, recent activity of the user, including the use of Dropbox, can be found using *logcat*. A lack of analysis tools for Android OS, did not let us perform a dynamic search to gather the useful data remnants. In fact, the files above mentioned and listed were found through a manual search.

7. CONCLUSIONS AND FUTURE WORK

The first part of the research showed that it is possible to find plenty of evidence that relates to the use of a cloud storage client application on Windows 8.1, and evidence that relates to the activity of the user. These evidence is found mainly in databases and log files created by the application, but it also can be found inside the browser if a desktop client application version is not used, inside the memory, and inside the registry. If the application is then uninstalled, traces of the use and evidence can be found inside the memory, system log files, and registry keys and so on. If files are removed by the user there are two approaches that can help a forensic analyst to recover them. Possible future work can consist in a forensic analysis on the server side of the cloud, however, this is complicate issue since many problems arise, such as geography impossibility (servers can be spanned all over the world) and the servers can be outside jurisdiction of the investigators as well as a not transparent collaboration from the cloud service providers. The second part of the research proposed in this paper, highlighted the main locations and files on a client device with an Ubuntu 14.04, and more generally, a UNIX/Linux operating system. Although the operating system is different, as well as the file system, the results obtained by previous researches and the first part of this work, the results of this research are in accordance and similarities arise. Differences arise due to the different feature of the Windows 7 and 8.1 operating systems and the UNIX/Linux OS, therefore locations and file types may vary. In fact, the registry, which is a classic feature of Windows is not present on UNIX/Linux systems so there will be no evidence related to it. Unfortunately, the amount of tools used to analyze dynamically a process, such as Process Monitor does not have a valid counterpart for UNIX/Linux, therefore a dynamical analysis to gather evidence is a bit harder. Nevertheless, it was still possible to find evidence in the hidden directories or hidden files created by the application, as well as in database or log files, inside web browser files, in the memory, and in both allocated and unallocated space. Many other locations and files on the file system, such as the cache, system logs, and configuration files stored useful data. Files stored on Android OS are similar to the files stored on laptops or desktops. However, one big difference is that, in the case of Ubuntu, the cloud application downloads and stores a copy of each file locally, while in the case of Android OS, a copy of a file is stored locally only if the file is accessed by the user. The fact that the files are stored locally, is really helpful during digital investigations since deleted files and their metadata are still present on the hard disk, unless the unallocated space is wiped or overwritten. Recently, most of the main cloud storage services, such as iCloud or Dropbox, started to encrypt their files, which makes it harder for digital experts to gather useful evidence. With a proper search warrant, the cloud service provider may be able to decrypt the files and allow forensics experts to gather evidence. Another key factor that can be seen in this analysis, is that even if the applications are different, the configuration files, database files, and log files (and use of hidden files and directories, cache directories and temporary files, database and log files) are similar. Finally, this research highlighted that a great deal of evidence can still be found on the Ubuntu system once the application is uninstalled or simply removed.

8. REFERENCES

[1] Columbus, L. (2013, September 4). Predicting Enterprise Cloud Computing Growth. Retrieved December 5, 2014, from http://www.forbes.com/sites/louiscolumbus/2013/09/04/predictingenterprise-cloud-computing-growth/

[2] Millis, E. (2012, June 30). Cybercrime moves to the cloud. Retrieved December 5, 2014, from http://www.cnet.com/news/cybercrime-moves-to-the-cloud/

[3] NIST Cloud Computing Forensic Science Challenges (2014, June). Draft NISTIR 8006, NIST Cloud Computing Forensic Science Working Group Information Technology Laboratory, NIST. Retrieved December 5, 2014, from http://csrc.nist.gov/publications/drafts/nistir8006/draft_nistir_8006.pdf

[4] Chung, H., Park, J., Lee, S., & Kang, C. (2012). Digital forensic investigation of cloud storage services. *Elsevier*.

[5] Katz M., Montelbano R. (2013). Cloud Forensics, the Senator Patrick Leahy Center for Digital Investigation, Champlain College.

[6] Quick, D., Martini, B., & Choo, R. (2013). Cloud Storage Forensics (1st ed., p. 208). Syngress.

[7] Epifani, M. (2013). Cloud Storage Forensics. Retrieved from SANS European Digital Forensics Summit, Prague website: https://digital-forensics.sans.org/summitarchives/Prague_Summit/Cloud_Storage_Forensics_Mattia_Eppifani.pdf

[8] Copy: Store, protect, and share amazing things. (n.d.). Retrieved December 5, 2014, from https://www.copy.com/home/

[9] OwnCloud 7 Community Edition is here! (n.d.). Retrieved December 5, 2014, from http://owncloud.org/

[10] Dropbox Decryptor: A Free Digital Forensics Tool. (2014, June). Retrieved December 5, 2014, from http://www.magnetforensics.com/dropbox-decryptor-a-free-digital-forensics-tool/

[11] Windows Sysinternals: Documentation, downloads and additional resources. (n.d.). Retrieved December 5, 2014, from http://technet.microsoft.com/enus/sysinternals/bb545021.aspx

[12] DB Browser for SQLite. (n.d.). Retrieved December 5, 2014, from http://sqlitebrowser.org/

[13] RegScanner: Alternative to RegEdit find/search/scan of Windows. (n.d.). Retrieved December 5, 2014, from http://www.nirsoft.net/utils/regscanner.html

[14] Carrier, B. (n.d.). The Sleuth Kit: Download. Retrieved December 5, 2014, from http://www.sleuthkit.org/sleuthkit/download.php

[15] Make webpages load faster. (n.d.). Retrieved December 5, 2014, from https://support.google.com/chrome/answer/1385029?hl=en

[16] Carrier, B. (2005). NTFS Analysis. In *File system forensic analysis*. Boston, Mass.: AddisonWesley.

[17] HxD - Freeware Hex Editor and Disk Editor. (n.d.). Retrieved December 5, 2014, from http://mh-nexus.de/en/hxd/

[18] LiME, Linux Memory Extractor, Avaialable: https://github.com/504ensicsLabs/LiME

[19] Foremost, SourceForge.net, Avaialable: http://foremost.sourceforge.net

[20] Dropbox Decryptor: A Free Digital Forensics Tool. Available: http://www.magnetforensics.com/dropbox-decryptor-a-free-digital-forensics-tool/

[21] [21] Client for ownCloud, BezKloboukuNos, 2014, June 29 Available: https://play.google.com/store/apps/details?id=com.owncloud.androidApp&hl=en