MONOCLE – Extensible open-source forensic tool applied to cloud storage cases

J. Rodriguez-Canseco, J. M. de Fuentes, L. González-Manzano, A. Ribagorda

Abstract— **Current forensic tools highly depend on the analyst awareness of evidences in order to retrieve them by means of a proactive search methodology. This paper presents MONOCLE, an open-source extensible framework for automated forensic analysis. MONOCLE provides automation over the forensic procedure by means of user-created plugins, reducing the complexity of evidence retrieval in target's machine hard disk and memory. The software makes use of external tools such as the Volatility Framework in order to provide extended functionality to the executed plugins. To show the applicability of the proposal, in this paper MONOCLE is applied to two user-side cloud storage scenarios -- iCloud and Box. Results show that MONOCLE is able to retrieve relevant information regarding end-users systems and cloud services interaction in the client machine.**

Keywords— **computer forensics, analysis, memory forensics, software tools.**

I. Introduction

COMPUTER forensics has proved to be a key factor for criminal investigations and law enforcement, as IT devices are increasingly more present in our society [1]. Such procedures are of high importance not only in the elucidation of traditional criminal cases where new technologies are present as a helper tool for criminals to operate, but also in modern crimes where IT devices are the main platform (e.g. illegal distribution of copyrighted material or online fraud) [2].

In order to conduct such investigations law enforcers make use of a huge variety of forensic tools for the collection of evidences. This is of particular interest because most best forensic practices are standardized, so they can be done in an algorithmic way (e.g. evidence integrity maintenance, evidence classification or data carving). For example, Spanish UNE 71506 describes a forensic analysis methodology [3]. It is thus interesting to automate the collection of such elements in order to speed up the whole process.

Most of the current forensic analysis tools (especially privative ones, such as EnCase [4]) provide an interactive interface to analyse the target system. By doing so the analyst is able to perform a proactive analysis to find where relevant information regarding the conducted case can be found. At the same time, an overview of the overall system state is provided. This can help the analyst to find new sources of information that he might not have appreciated at a first glance.

There is, however, a disadvantage within this paradigm, namely the fact that the user has to manually select which elements to categorize as evidences. Whether this can be useful when conducting a non-deterministic case in which the procedure to retrieve the evidences or their nature are not clear, it is a tedious process when facing some cases in which the traces to be found are quite standardized (e.g. traces regarding the installation of a well-known program).

Although some tools provide a way to extend and automatize user-defined procedures by means of scripts (e.g. [5], [6]), this is not straightforward. Furthermore, it usually implies buying a license for specialized programs. These issues together with the fact that technologies are becoming more complex as time goes by has also revealed some new challenges for forensic analysis tools, being cloud forensics a relevant one.

Cloud services are becoming popular among companies and end users. The main reason is that a cloud environment allows the use of hybrid hardware and systems in order to set up virtual structures, which can be created, modified and destroyed on users demand. Cloud systems are traditionally structured into three main categories depending on the service model they provide, namely Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) [7].

Such structures include a brand new set of problems for law enforcers in order to conduct the analysis [8], which range from the geographical dispersion found within cloud service infrastructure (i.e. resulting in different jurisdictional issues) to the fact that physical machines might hold data from different individuals, (i.e. resulting into potential privacy and confidentiality problems). Several approaches have been tested as to overcome such issues [9]. There are differences however in the complexity involved within the analysis for each of the aforementioned services (SaaS, PaaS and SaaS).

More specifically, cloud storage services based on a SaaS model present one of the most interesting targets for forensic analysis as the number of users and the amount of data in such platforms is increasing [10]. They also do provide a promising window for forensic analysts due to the fact that they tend to leave high numbers of data remnants in end users machines [11]. Such end users access these services by means of personal computers or mobile devices, which do not present the issues existing in cloud forensics.

This context would make useful the existence of an open source tool able to conduct forensic procedures applying automated scripts. The existence of such software would avoid organizations and individuals affording the expensive licenses

J. Rodríguez-Canseco, J. M. de Fuentes, L. González-Manzano and A. Ribagorda are with the Computer Security Lab (COSEC) of University Carlos III of Madrid (Spain).

E-mail: {jorrodri, jfuentes, lgmanzan, arturo}@inf.uc3m.es

of commercial forensic tools.

This paper presents MONOCLE, an open source framework for forensic analysis implementing an extensible script-driven system. The software targets relevant sources of information on IT systems, such as volatile memory dumps and disk images. This framework is used afterwards to study a pair of user-side cloud scenarios in different platforms, namely iCloud [12] and Box [13]. This implies that several modules will be created both to test the effectiveness of the framework when targeting those scenarios and to study the scenarios as such.

The structure of this paper is as follows: Section II present a summary of related work in the scope of forensic analysis tools and user-side cloud forensics. Section III introduces the necessary background knowledge to understand user-side cloud scenarios. Section IV presents the forensic analysis framework and its workflow at a high level. Section V depicts and describes the detailed design of the framework. In Section VI the application of the proposed framework to a user-side cloud scenario is presented. Section VII compares MONOCLE against existing tools. Finally, conclusions and future work is outlined in Section VIII.

II. Related Work

Previous research contributions have addressed cloud forensics and their peculiarities regarding traditional forensics problems, such as jurisdictional issues and privacy concerns. Most of this literature focuses on the study of the open problems and possible improvements in cloud structures rather than in the actual forensic analysis of cloud systems within the current architectures. Zawoad et al. [9], and Shah et al. [14] discuss the different problems inherent to cloud systems regarding digital forensics. Almulla et al. propose different approaches to provide forensics friendly cloud services [15], but such solutions require cloud service providers to adopt them and do not solve the problem in the short term scope.

The increasing number of users of the SaaS cloud architectures in which data storage is offered as a service has motivated the research on this specific area. Quick et al. [16] provide an extensive listing of traces left on client machines by different cloud storage platforms, such as Dropbox [11], Google Drive [17] Microsoft SkyDrive [18] or ownCloud [19], found by means of different forensic tools. Such studies reveal the utility of client-side forensics in which they call Storage as a Service (StaaS) platforms.

A. On software tools

Most commercial tools provide a wide set of procedures in order to recover data evidences, such as data carving [20] and memory data analysis [21]. These tools are implemented following an approach which requires the user interaction to retrieve items considered as evidences. This is done in order to allow the required flexibility when conducting forensic investigations (i.e. as there are a broad number of interpretations an analyst might give to an evidence element depending on the context of the conducted case).

Such tools can be classified in several ways depending on their nature, e.g. being commercial software or freeware. Commercial software tools are usually more complete than their free counterparts, and thus include a broader set of features.

Among commercial tools, some of the most relevant ones due to the amount of provided features are EnCase Forensics [4] by Guidance Software, FTK (Forensic ToolKit) [22], from AccessData, or Pro Discover [23], from the ARC group of NY. Another interesting software due its relationship with the context discussed in the present paper is IEF (Internet Evidence Finder) [24], from Magnetic Forensics, as it performs forensic analysis of internet services.

On the other hand, open source tools have a more limited scope in terms of analysis targets and capabilities. There exists however several powerful tools to perform forensic analysis, such as The Sleuth Kit [25], by Brian Carrier (disk image analysis), the Volatility Framework [26], by Volatility Foundation (memory analysis) or Simsong's Bulk Extractor [27] (disk data carving).

III. Cloud Forensics

Cloud computing can be divided in two different parts, namely the front end, that is based on a client's computer and the back end, which consists of one or multiple computers, servers and data storages [28]. Moreover, cloud servers may be used by huge amount of users or entities and the appropriate management has to be provided. Likewise, servers or data storages can be spread all over the world and their management has to be carried out according to laws established in their particular location as well as guaranteeing availability even when failures occur.

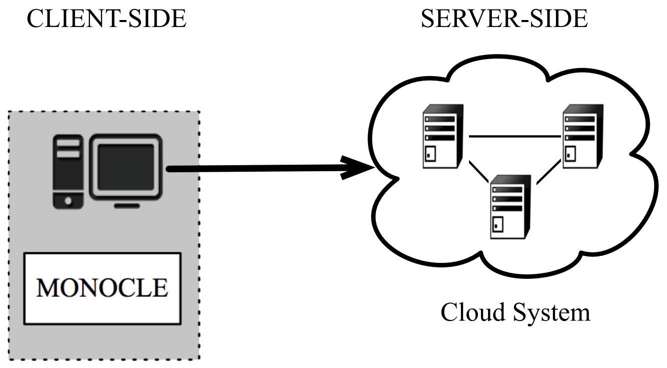

Figure 1. MONOCLE in the cloud infrastructure

Cloud systems are a popular choice due to their fast resource allocation and elasticity capabilities. They use hybrid hardware and virtualization to reduce overall costs [29]. Although this versatility is quite useful for companies, in terms of computer forensics the nature of the cloud environment makes difficult forensic analysis using traditional procedures. Besides, forensic analysis in cloud servers may become a burden due to their location and the amount of possible clients together with applicable laws.

As aforementioned, given that cloud forensics is hard to analyze from the back end (server-side) point of view, MONOCLE addresses computer forensics from the front end (client-side) perspective (see Figure 1). Data stored and managed in the cloud is accessed by clients, e.g. a browser or desktop application.

A. *User-side cloud forensics*

Cloud storage platforms usually provide two different ways to access their services. The first one is the access by means of a web application using a web browser. Such application allows the user to remotely interact with stored data within the server. In addition to the web browser, most of cloud storage service providers have a synchronization client program which updates data on the server side based on changes performed at client-side.

This interaction from the client-side exposes a series of evidences, which can be potentially meaningful for investigators. Three different levels of relationship between the user and the cloud service can be defined as to quantify the amount of potential evidences to be found during the analysis:

- **Access the cloud service through a web browser**. Data remnants are found mainly within the web browser history and RAM memory.
- **Access the service through a cloud sync program**. Data remnants are related to the installation and operations of the synchronization program.

Different sets of artifacts can be elicited for each of the levels described. Such artifacts analysis provides a source of potential evidences. The relation amongst different artifacts will likely reveal meaningful information to the conducted investigation.

Regarding what can be found in memory, one of the most relevant evidences to carve are the URLs associated to the web interface of the cloud services. Memory-mapped structures such as registry keys are also valuable as they might contain data regarding to the web browser (e.g. URLs recently typed and temporal files downloaded or visualized within the browser).

The analysis of the disk image provides more information than memory analysis. Temporal files are stored in each web browser folders. All registry hives are usually present. The most interesting refers to traces left by a newly installed program which are easy to identify. In this context, the installation folder of the program, the different registry keys, which have been modified by the software, and all data the tool is syncing with the cloud service are potential sources of information. System logs are also valuable as they provide a way to track changes within the system.

Once searched evidences to look for are stated, tools provided by MONOCLE can be applied as to retrieve the specific elements for each case.

IV. SYSTEM OVERVIEW

A. *Goals*

In this section the main goals pursued in MONOCLE's design are described.

The main objective of the present paper is the creation of a utility framework (MONOCLE) for the development of automated analysis on third party target machines. Many of the current tools (recall Section II) in the forensic analysis field are either commercial software, or present a limited scope in terms of evidence sources. In this respect, the proposed tool would **target main evidence sources** present on IT devices, e.g. namely volatile memory dumps and disk images.

In addition to that, most of tools do provide a proactive (interactive) search paradigm. This stands for the fact of the tool carving data in different areas, and presenting it to the analyst so as for him to select elements to categorize as evidence. This is a time-consuming process when facing data retrieval of well-known elements, such as program installation data or web access history. By contrast, MONOCLE provides a **script-based analysis** procedure. This implies that the analyst, targeting specific well-known elements, can code several modules. Such plugins can be reused in further investigations with minimum effort, speeding up the analysis process and reducing the analyst manual workload. In this respect, it is necessary for the tool to be **extensible** as to adapt new needs of the analyst. Besides, MONOCLE detects and loads seamessly new scan plugins (i.e. scripts) without incuring into extra efforts to the user.

Due to this need of extensibility, the proposed software includes several utility tools which can be used in order to speed up the creation of plugins. Such utilities include automatic evidence management, RAM memory reconstruction, registry hives parsing, and visual timeline representation of results.

Finally, the implementation of a **GUI** for the easy of use and the release of the tool under an **open source license** are also current objectives.

B. *High level functionality*

The framework is intended to run user-created plugins, which will perform different analysis over the evidence sources. Although the user can create its own plugins, there is the possibility of download plugins created by other users.

Monocle loads evidence sources from different data dumps. In its curret version MONOCLE process raw format RAM memory dumps and disk images. Once the user has selected an evidence source, it is necessary to state wether it is a Disk image or a memory dump. Plugins differ depending on the type of the source. Available plugins can be choosen by the user in order to execute them over the selected evidence source.

A different process occurs in order for the framework to access each type of evidence source. Disk images are mounted using system-specific commands. This process is performed in read-only mode so as to preserve data integrity. Memory dump images are treated as regular files, being the user plugin the one choosing how to interact with them.

Apart from functionalities provided by each user plugin, the framework performs, without the requirement of user interaction, some of the standardized tasks common to any digital crime scene investigation phase (e.g. integrity checking and secure storage of evidences) [30]. To do so an evidence manager has been developed and integrated in MONOCLE. This manager provides digital-feasible documentation of the evidence (e.g. retrieving data such as evidence location within the digital container or metadata) and evidence data recovery. Such recovery includes a copy of the evidence element (if possible) on a local directory within the analyst machine, as well as optional comments made by the analyst. Besides, evidence integrity is achieved by means of automatic computing of both MD5 and SHA1 checksums over the evidence element.

MONOCLE provides additional tools which can be used during the scripts. Such tools are integrated within a simple interface as to simplify their usage. For this first version of the software, only one auxiliary tool for each target type will be implemented.

Regarding memory-targeted modules, the Volatility Framework is applied to analyse such evidence dumps. The integration of this system within the present framework allows plugins to use the whole Volatility functionality over a simplified interface. Results coming from Volatility scripts are returned to the user script as a data array set. Regarding disk-targeted scripts, Windows registry analyzer is one of the most useful tools. This allows the user plugin to be able to extract information from the Windows registry, which is one of the most interesting elements to analyze in Windows systems [31]. Python Registry library by Will Ballenthin [32] is applied in MONOCLE to provide such functionality.

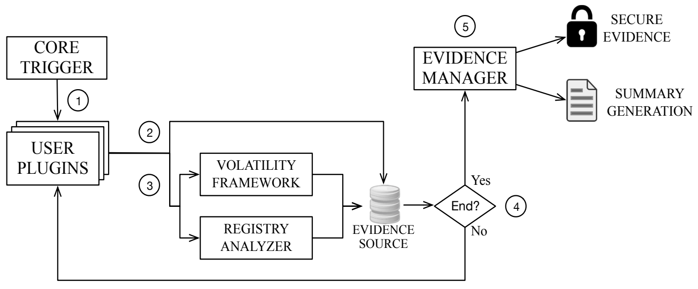

Figure 2. MONOCLE workflow

Regarding the execution of the framework, plugins are executed once the user has selected both the plugin to run and the evidence digital container to analyze (e.g. either a disk image or a memory dump). Subsequently, the framework starts working in order to execute the chosen plugin. Figure 2 depicts the general workflow of the framework when executing a user module.

The framework prepares the analysis environment in order to start the plugin (1) (e.g. evidence secure mounting and parameter parsing). Once the user plugin is executing, there are two possible ways interact with the evidence source, namely to accessing directly the evidence source (2), or making use of one of the built-in tools (3) through the framework interface (e.g. the Volatility Framework). This process can be repeated as many times as needed (4) until the analysis finishes. Finally (5) the evidence manager acts over each single evidence, providing integrity checking, local allocation and indexing such evidence to present a summary of the analysis.

V. Software Design

Applied technology has to be carefully considered because it affects the internals of the system as well as framework scripts by applying a set of constraints over the plugin development procedure. The Python programming language has been chosen due to its versatility and cross-platform attributes. Python provides an effective fast-prototyping approach useful for further plugin development. MONOCLE has been coded using JetBrains' PyCharm [33] development environment as to help in the organizing of the overall project.

Regarding the functionality defined in the previous section, Figure 3 depicts MONOCLE components. The tool is divided into four components with well-defined functionality each.

The main framework functionality is located in the Core component. This part of the system does not depend on the user modules employed. The *Core* component is in charge of loading the user modules to be executed within the framework, as well as the utility classes the user module can make use of. The *SetupHandler* component registers different analysis parameters, which can be parsed either by means of the *GUI* or through command line execution. It is also responsible for setting up MONOCLE's required starting environment. The *wrapper* component within the core creates an abstraction layer for plugins and manages evidence data extracted by the modules. The loading procedure differs regarding target type of the module to load (i.e. either a memory dump or a disk image). Report management and evidence management data are parsed to the *XMLParser* component to generate summaries.

The *MemoryModule* and *HDModule* are two different interfaces, used to load the user plugin. The main difference between them stands for the auxiliary tools to load. Whether a memory-targeted plugin (i.e. making use of the *MemoryModule* interface) is able to use the Volatility Framework by means of the *VolActor* component, a disk-targeted plugin is only able to load the registry module by means of the *RegistryActor* component. Both module interfaces will create an *EvidenceManager* to process found evidences along plugins execution.

Volatility Framework features are accessed through the *VolActor* component, which is in charge of loading and automatically configuring the Volatility Framework into the Monocle Framework. This component can be loaded on demand by an plugin during its execution, and it provides auxiliary functions with methods to call the different Volatility

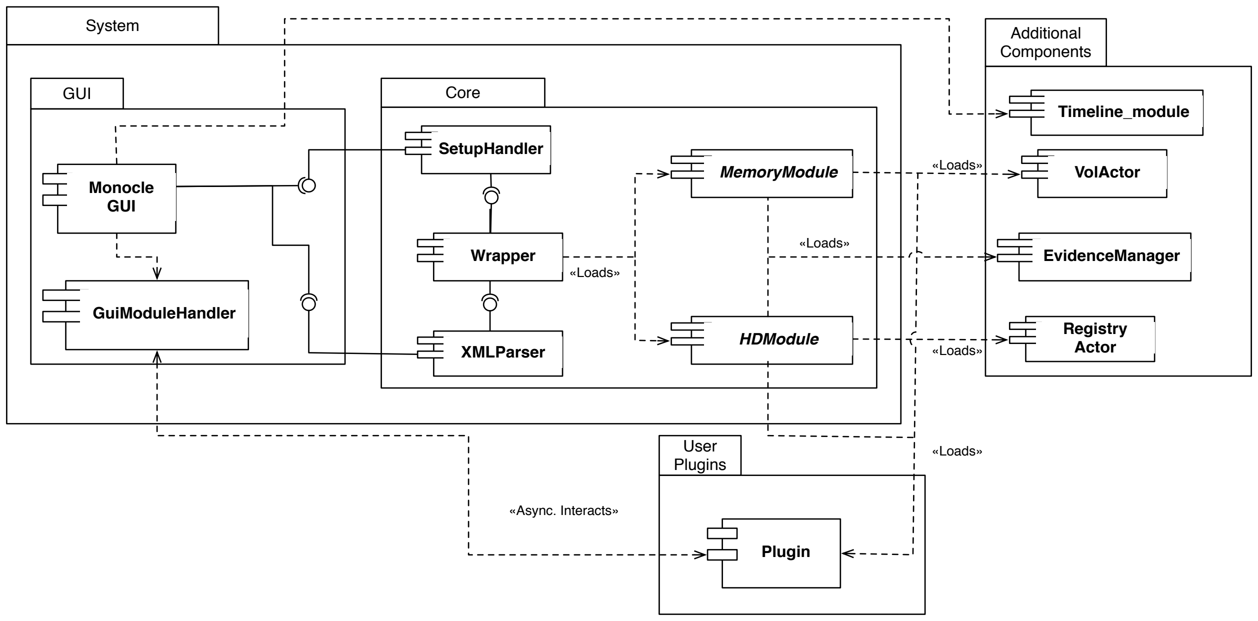

Figure 3. MONOCLE High level decomposition diagram

Plugins included in the Volatility installation framework. Memory type identification is performed if unknown. The *RegistryActor* works similarly with the Python Registry plugin, providing commodity methods as to retrieve data from memory hives.

Users have freedom to code the plugin as desired. The only requirements is to include MONOCLE's libraries in the plugin code and inherit from either a *HDModule* or a *MemoryModule* to use specific module tools (e.g. *VolActor, RegistryActor, EvidenceManager*).

VI. EVALUATION

This section first compares the features provided by existing well-known tools with the ones provided by MONOCLE (Section A). Secondly, Secondly, MONOCLE is applied in a pair of cloud scenarios, namely, iCloud and Box. Lastly, a performance evaluation in regard to both previous scenarios is presented (Section B). Lastly, a performance evaluation in regard to both previous scenarios (Section C).

A. Goals analysis comparison

This comparison is performed according to the different features stated in the pursued objectives (Section IV.a):

- **Privative Software**. Whether the software is a free-to-use tool or a commercial one. MONOCLE is an open-source (free of charge) software.
- **GUI Interface Available**. Whether the interaction with the tool can be performed by means of a graphical user interface or not. Our framework provides a GUI, namely by *MonocleGUI* component (see Section V).
- **Target**. Whether the tool targets memory dumps or disk images as evidences sources. MONOCLE targets both memory dumps and disk images in RAW format.
- **Interactivity / Proactive analysis**. Whether the tool allows the analyst to operate over the results and to perform different analysis over the image in a non-deterministic way. This is the opposite of a scripting deterministic analysis where the tool has a predefined set of operations. MONOCLE focuses in script-based analysis.
- **Scripting Capability**. Whether the tool has the capability of execute user-defined scripts, which perform different analysis or procedures sequentially without user interaction. MONOCLE is script driven.
- **Extensibility**. Whether the original functionality of the system can be enhanced by means of tools provided by the system itself. MONOCLE's scripting-based nature provides extensibility to the framework.
- **Extensibility type**. How extensibility is performed within the given application if the application allows for an extensible behavior. MONOCLE provides extensibility as it allows to include new plugins on the fly.

There are hybrid combinations of such features depending on the tool. Table 3 depicts the analyses.

EnCase Forensic, developed by Guidance Software [4], provides a broad number of utilities, such as data carving, and automatic generation of reports. EnCase also counts with its own scripting language in order to automate different tasks and extend its functionality depending on the analyst needs. EnCase is however a commercial software. Another interesting yet similar tool is FTK [22] by AccessData. This tool provides similar functionality as EnCase and it allows visualization of data in real time, multiple source image detection and automatic password recovery from a big set of applications, among others. Like EnCase, FTK is a non-free application.

Internet Evidence Finder (IEF), by Magnetic Forensics, is a specialized tool focused on the discoverage of data remnants of Internet services, such as email, web navigation and cloud-based storage. Alhough quite complete, this is yet another commercial tool.

Regarding open-source forensic software, The Volatility Framework is another interesting tool. It provides memory forensics services off the shelf, such as open connections enumeration and running processes carving. Although Volatility was initially a framework to perform analysis over memory dumps on Windows machines, contributions from the open source community allows now Volatility to target also some versions of Linux and Mac OSX. Volatility also offers a plugin-oriented execution model, allowing its extensibility with new functionalities by means of user scripts written in Python. However, Volatility Framework scope does not involve disk images analysis and a different tool should be used in this regard, e.g. Sleuth Kit [25] is one of the most popular ones.

The Sleuth Kit (TSK) consists of a set of tools for forensic investigations over disk images. TSK is an open-source project whose functionality can be used by means of a library or executed as a standalone application. A GUI for TSK is provided by Autopsy [34] project, by Basis Technology. It is however focused only on disk images, not being able to relate memory artifacts to the conducted investigation.

When coming to analyze each of these tools, differences between commercial software and open-source projects are quite noticeable in terms of scope. Whether commercial software usually targets more than one evidence source (e.g. RAM memory and disk artifacts), open source projects usually target a single source because resources are more limited.

Additionally, MONOCLE provides better low level analysis of the results, as found evidences can be automatically classified and processed. If the applied plugin already exists, the analyst can execute it without dealing with low level details. This saves time and effort in the investigation process. Besides, MONOCLE provides users the ability to easily create new plugins if needed.

B. *Application to cloud scenarios*

MONOCLE is flexible enough to perform a wide variety of types of analysis. It depends on the user module being executed. In the present context, it is used in a user-side cloud scenario namely in iCloud [12] and in Box [13] cloud storage services. The objective is to analyze the interaction with the storage services by analizing disk and memory dumps of client's machine.

1) *Definition of the scenarios*

Two different cloud platforms are evaluated to prove the suitability of the framework for the cloud-based scenario, namely iCloud, by Apple Inc. and the cloud service of Box. Two different modules re developed for each scenario, one focused on RAM memory and another one focus of disk remnants. The objective of the investigation is to prove the interaction between the user and the cloud service, and to retrieve as much information as possible (i.e. downloads, visits, hosted files).

The preparation of these analyses follows a similar approach to the one proposed by Quick et al. [11], [17], [18]. Both analyses are conducted under a Windows 7 x64 system emulated through a VMWare Virtual Machine. No additional software is installed except for the web browser (i.e. namely Internet Explorer 11) which is used to test the web clients of the different platforms and the installation of the platform third party client on each case. Accounts for the different platforms are created outside the virtual machines, limiting the interaction of the user to a single connection to the service in order to either view the files or download and install the client. From each virtual machine, the RAM memory and the virtual hard disk are analyzed as raw dumps.

2) *Identification of the traces*

Traces to be found in both scenarios involve several different elements, ranging from URLs with specific formats (e.g. all Box cloud services URLs have a fixed starting pattern of the form *app.box.com*) to SQLite databases and configuration files containing information for each specific platform.

Cloud-related traces are particularly noticeable when using

Software	**Privative**	**GUI**	**Target**	**Interactivity**	**Scripting**	**Extensibility**	**Extensibility Type**
Encase	Yes	Yes	Memory, Disk	Yes	Yes	Yes	EnScript. A scripting language automating EnCase tools
FTK	Yes	Yes	Memory, Disk	Yes	No	No	–
IEF	Yes	Yes	Memory, Disk	Yes	No	No	–
Volatility	No	No	Memory	Yes	Yes	Yes	Volatility Framework plugin system. Allows to create new analysis within the Volatility Framework structures.
TSK	No	Yes	Disk	Yes	Yes	Yes	Sleuth Kit plugin framework. Allows the automation of the different stages of TSK
ProDiscover v9	Yes	Yes	Memory, Disk	No	No	No	–
Bulk Extractor	No	No	Disk	No	No	No	–

• **FTKv4**: Forensic Toolkit, version 4 • **TSK**: The Sleuth Kit, version 4.1.3 • **IEF**: Internet Evidence Finder Bundle (IEF Standard + IEF Triage)

Table 1. Comparision of different tools regarding their usability and extensibility features.

a synchronization program. Installation fingerprint can be found in the Registry, Windows Logs and other control elements present on a Windows system. Such installation also involves the creation of the synchronization folder, which potentially contains elements residing in the cloud storage platform.

The implementation of MONOCLE's scripts to recover evidences requires, it is necessary to take into account all traces and to specify them in the plugin in order for MONOCLE to find them. The complete set of traces found for both scenarios is described in Appendix I.

3) *Implementation of the plugins*

Memory analysis can be performed either by means of the Volatility Framework or by manually carving files contained in the memory dump. The chosen approach depends on the existence or not of a running web browser process. If a web process exists, it is possible to acquire the reconstructed allocated memory of such process applying Volatility's *MemDump* plugin. It allows the reconstruction of the fragmented memory belonging to the process in a continuously allocated dump. Search of URL's can be then performed over this reconstructed memory space. In case the process cannot be found, the whole memory dump has to be carved as to find those URLs.

Volatility also provides a way to retrieve Internet Explorer history if the process was running when the memory was acquired (i.e. namely *IEHistory* plugin). Finally, registry keys mapped into memory are extracted using the *hivescan, hivelist* and *printkey* modules.

Regarding the analysis of the disk drive, Monocle automatically tries to mount the targeted disk. Once it is mounted, the plugin starts running and marks as evidences chosen files, folders and elements through the *EvidenceManager*. This module translates paths to the local equivalent within the disk. All evidences retrieved in this way can either be manually labeled by the user, or automatically be classified by the *EvidenceManager*. Moreover, the *EvidenceManager* automatically calculates integrity checksums for all found evidence elements, and stores them in a secure folder within the host system.

Registry keys can be easily parsed and analyzed by means of the Python-Registry library integrated within the framework. MONOCLE implements an interface to such library to provide the user with straightforward functions able to retrieve the Registry keys by just specifying the path to the registry hive and the desired key (or sub-keys) to extract.

C. *Performance evaluation*

Concerning previous scenarios (iCloud and Box), the overhead caused by MONOCLE is measured. Specifically, time measurements from states of the execution are studied. MONOCLE execution workflow is divided in three main phases, (recall Figure 2). The first phase (*wrapper pre-setup*) consists of setting up MONOCLE's environment and mounting the evidence digital containers. The second phase (*module execution*) is the execution of the plugin. In the last phase (*wrapper post-setup*) the *EvidenceManager* processes evidences found in the second phase and presents the results to the user.

Table 2 shows times for plugins described in Section VI.B. to analyse disk and memory for Box and iCloud. Modules use MONOCLE tools to retrieve evidences. In addition, a Box memory plugin not using Volatility was created to show Volatility's overhead during execution.

Memory Modules	**Wrapper pre-setup**	**Module execution**	**Wrapper post-setup**
Box (w/o Volatility)	0,498s	25,940s	2,40E-05s
Box	0,737s	206,806s	3,20E-05s
iCloud	0,622s	49,048s	1,00E-05s
Disk Modules			
Box	1,227s	0,913s	3,20E-05s
iCloud	0,695s	3,893s	3,80E-05s

Table 2. iCloud metadata remnants

Results show that executed plugins leverage overall running time, which implies MONOCLE's overhead is almost neglectable regarding execution times. This overhead is more noticeable for disk modules, 57,3% and 15,6% of the overall time for Box and iCloud respectively. Memory-targeted modules reconstruct the memory address space of the machine (if using Volatility) and perform a complete scan of the memory dump. Such operational charge results in higher module execution time than for disk modules. MONOCLE's overhead is thus small in memory modules, being 1,8% for Box module running Volatility and 0,6% for the one not using Volatility. MONOCLE's overhead on iCloud memory module is 1,25%. Disk-targeted plugins overhead seems higher than memory ones. This fact is because disk plugins neither perform data carving nor have to search the entire digital container.

Significant differences exist between times of the module execution phase, as this stage depends directly on executed the plugin. It is interesting to notice the overhead of using Volatility on a memory targeted plugin, e.g. Box plugin not using Volatility is 8 times quicker than the version using Volatility to reconstruct the memory space.

VII. CONCLUSIONS AND FUTURE WORK

In this paper MONOCLE, a framework for forensic analysis on a plugin-based execution model is presented. This framework is intended to provide fast prototyping of efficient forensic analysis over well-known structures by the use of plugins. Its functionality is apply to evaluate user-side cloud storage scenarios in two particular cases, namely iCloud and Box. This evaluation has additionally proved the importance of such remnants regarding cloud-based storage, as several traces relating the user and the platform are to be found in the client machine.

Our proposed tool is on its very first development stage. There are thus several improvements to be apply. The inclusion of the TSK Framework within MONOCLE, so as to provide all the functionality available in such open source tool is a future enhancement. The disk carving capabilities of TSK and the automated evidence mounting of corrupted images are features of interest concerning the improvement of disk images plugins. The processing of proprietary and compressed evidence sources is currently out of the scope of this tool regarding its first version. Support for different formats is a future objective for MONOCLE. The inclusion of plugin-sharing capabilities of the framework by creating a centralized knowledge base where plugins can be uploaded and downloaded by different users is another matter to address in the future. This would reduce time and effort and facilitate collaboration among different analysts.

VIII. References

[1] L. Ericsson, "More than 50 Billion Connected Devices," *www.ericsson.com/res/docs/whitepapers/wp-50-billions.pdf*, no. February, 2011.

[2] A. Guinchard, "Cybercrime: The Transformation of Crime in the Information Age," *Information, Communication & Society*, vol. 11, no. 7. pp. 1030–1032, 2008.

[3] "AENOR. UNE 71506. Accessed 06/09/2015 at http://www.aenor.es/aenor/normas/normas/fichanorma.asp?tipo=N&codigo=N0051414#.VXcbfmCkaQw." .

[4] G. Software, "Encase Forensic v7. Accessed 03/17/2015, at https://www.guidancesoftware.com/products/Pages/encase-forensic/overview.aspx," *https://www.guidancesoftware.com/products/Pages/encase-forensic/overview.aspx*. .

[5] "EnCase EnScript programming. Accessed 06/09/2015 at https://www.guidancesoftware.com/training/Pages/courses/classroom/EnCase®-EnScript®-Programming.aspx."

[6] "The Sleuth Kit pipeline system. Accessed 06/09/2015 at http://www.sleuthkit.org/sleuthkit/framework.php."

[7] W.-T. Tsai, X. Sun, and J. Balasooriya, "Service-Oriented Cloud Computing Architecture," *2010 Seventh Int. Conf. Inf. Technol. New Gener.*, pp. 684–689, 2010.

[8] D. Birk and C. Wegener, "Technical issues of forensic investigatinos in cloud computing environments," *Syst. Approaches to Digit. Forensic Eng.*, 2011.

[9] S. Zawoad and R. Hasan, "Cloud Forensics: A Meta-Study of Challenges, Approaches, and Open Problems," *arXiv Prepr. arXiv1302.6312*, pp. 1–15, 2013.

[10] D. J. Abadi, "Data Management in the Cloud : Limitations and Opportunities," *Bull. IEEE Comput. Soc. Tech. Commitee Data Eng.*, pp. 1–10, 2009.

[11] D. Quick and K. K. R. Choo, "Dropbox analysis: Data remnants on user machines," *Digit. Investig.*, vol. 10, no. 1, pp. 3–18, 2013.

[12] Apple Inc., "iCloud. Accessed 04/20/2015 at https://www.icloud.com.," *iCloud. Accessed 04/20/2015 at https://www.icloud.com.* .

[13] Box, "Box Cloud. Accessed 20/04/2015 at https://app.box.com."

[14] J. J. Shah and L. G. Malik, "Cloud Forensics: Issues and Challenges," *2013 6th Int. Conf. Emerg. Trends Eng. Technol.*, pp. 138–139, 2013.

[15] S. Almulla, Y. Iraqi, and A. Jones, "Cloud forensics: A research perspective," in *2013 9th International Conference on Innovations in Information Technology, IIT 2013*, 2013, pp. 66–71.

[16] D. Quick, B. Martini, and K.-K. R. Choo, *Cloud Storage Forensics*. 2014.

[17] D. Quick and K. K. R. Choo, "Google drive: Forensic analysis of data remnants," *J. Netw. Comput. Appl.*, vol. 40, pp. 179–193, 2014.

[18] D. Quick and K. K. R. Choo, "Digital droplets: Microsoft SkyDrive forensic data remnants," *Futur. Gener. Comput. Syst.*, vol. 29, no. 6, pp. 1378–1394, 2013.

[19] B. Martini and K. K. R. Choo, "Cloud storage forensics: OwnCloud as a case study," *Digit. Investig.*, vol. 10, no. 4, pp. 287–299, 2013.

[20] A. Pal and N. Memon, "The evolution of file carving," *IEEE Signal Process. Mag.*, vol. 26, no. 2, pp. 59–71, 2009.

[21] R. B. van Baar, W. Alink, and A. R. van Ballegooij, "Forensic memory analysis: Files mapped in memory," *Digit. Investig.*, vol. 5, no. SUPPL., 2008.

[22] Access Data, "FTK. Accessed 12/05/2015 at

http://accessdata.com/solutions/digital-forensics/forensic-toolkit-ftk/capabilities."

[23] The ARC group of NY, "Pro Discover. Accessed 21/05/2015 at http://www.arcgroupny.com/products/prodiscover-forensic-edition/."

[24] Magnet Forensics, "Internet Evidence Finder. Accessed 21/05/2015 at http://www.magnetforensics.com/mfsoftware/internet-evidence-finder/."

[25] B. Carrier, "The Sleuth kit. Accessed 05/04/2015 at http://www.sleuthkit.org/sleuthkit/."

[26] Volatility Foundation, "Volatility Framework. Accessed 12/05/2015 at http://www.volatilityfoundation.org/#!releases/component_71401."

[27] "Bulk Extractor. Accessed 21/05/2015 at https://github.com/simsong/bulk_extractor."

[28] Y. Jadeja and K. Modi, "Cloud computing - Concepts, architecture and challenges," *2012 Int. Conf. Comput. Electron. Electr. Technol. ICCEET 2012*, pp. 877–880, 2012.

[29] P. Mell and T. Grance, "The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology," *Natl. Inst. Stand. Technol. Inf. Technol. Lab.*, vol. 145, p. 7, 2011.

[30] B. Carrier and E. Spafford, "Getting physical with the digital investigation process," *Int. J. Digit. Evid.*, vol. 2, no. 2, pp. 1–20, 2003.

[31] K. Alghafli, A. Jones, and T. Martin, "Forensic Analysis of the Windows 7 Registry.," *J. Digit. …*, p. 17, 2010.

[32] W. Ballenthin, "Python-Registry. Accessed 04/10/2015 at http://www.williballenthin.com/registry/," *http://www.williballenthin.com/registry/*, 2014. [Online]. Available: http://www.williballenthin.com/registry/.

[33] JetBrains, "PyCharm. Accessed 16/05/2015 at https://www.jetbrains.com/pycharm/."

[34] Basis Technology, "Autopsy Project, accessed 12/05/2015 at http://www.basistech.com/digital-forensics/autopsy/."

APPENDIX I

This appendix presents evidences discovered by MONOCLE's plugins described in Section VI.C.

A. Data remnants in Box cloud storage

Box cloud services provide storage capabilities within the cloud. Users can use the service by registering with an email account. This section presents traces left on client's machines memory and disk when accessing Box services.

Box cloud services URLs have a fixed starting pattern of the form *app.box.com*. The existence of this sole URL in the web browser memory address is not enough to state that there is a relationship between the user and the platform. This URL can also be found in the registry hives storing the recent visited URL. The extraction of them can be done by means of the Volatility Framework too, as it includes a plugin to retrieve registry hives mapped into memory. Registry keys can be accessed more easily from the disk as their location is well known..

Box URLs are however meaningful by their sole composition. The structure of Box web cloud service organizes all files previewed with a URL of the form app.box.com/files/0/f/0/1/f_[ID], where ID is a numerical value assigned by Box to the stored files. This identifier allows referencing such files within the memory dump.

Further analysis shows that the file metadata stored within the platform is sent to the browser in the form of a JSON structure. This structure has several meaningful fields such as the file ID and the file owner ID. Most relevant fields are depicted in Table 2.

The analysis of disk artifacts comprehend hosted disk files (i.e. images, text files...) as well as registry hives and temporal cached elements.

Study of memory artifacts already unveiled some traces present in the disk too. As part of files metadata, it is interesting to look at the *image* tag, which represents the name of the temporal file cached in the web browser temporal folder. By looking for such name in the aforementioned location, files metadata and the files themselves are found.

The registry hives are also a potential source of information here. Some of them are of particular interest, such as SOFTWARE\Box\BoxSync\InstallID, which specifies the identification number assigned by the system to the sync program when it was installed in the machine, SOFTWARE\Box\BoxSync\InstallID\InstallPath, which specifies the path of the Box Sync installation files within the system. NTUSER\Software\Microsoft Internet Explorer\TypedURLs, containing URLs typed by the user in Internet Explorer and NTUSER\Software\Microsoft\WindowsNT\CurrentVersion\AppCompatFlags\CompatibilityAssistantPersisted, which contains a list of files names downloaded from the browser.

In case the sync program is installed within the browser, the installation path (i.e. by default ProgramFiles\Box\Box

Sync) contains installation information. Specific elements of the user configuration are located under \Users\[user] \AppData\Local\BoxSync folder. This folder contains Box's own logs and SQLite databases with the synced files. Besides, Box Sync folder is available too (i.e. by default under \Users\[user]\Box Sync path.), containing all the synced files.

Metadata Tag	Description
user_id	Numerical value assigned by the platform to each registered user. The value represents the user viewing the file at the moment this data is requested.
owner	The owner of the file referenced. This is the user hosting the file within the cloud service.
name	Name of the linked file.
raw_size	Size in bytes of the linked file.
raw_date	The EPOCH date where the file was uploaded or created in the cloud platform
content_type	The kind of data the file represents. Experiments show it is not reliable as it guesses the file by its liked extension and not by means of checking the content
sha1	The SHA1 message digest of the file. Used with integrity purposes.
last_updated_by	Registered name for the last user who edited the file in the cloud platform. It is linked with parameter last_updated_by_user_id.
last_updated_by_user_id	Similar to the *user* field, this field represents the numerical value assigned by the platform to the user who last updated the file.
created_by	Registered name for the user who uploaded/created the file in the cloud platform. It is linked with parameter created_by_user_id.
created_by_user_id	Similar to the *user* field, this field represents the numerical value assigned by the platform to the user who created/uploaded the file.
pic_l	Link of the picture within the cloud platform (i.e. suffix to the root URL).
image	Name of the image within the local temporal folder of the web browser.

Table 3. Box metadata remnants

B. Data Remnants in iCloud cloud storage

Data can be extracted from the web browser process memory address, this finding mapped files or URLs of the iCloud service. Following the same approach as in Box, though being more complex, an authorization token is requested for each single file. These tokens are invalidated after a certain amount of time, and are given to iCloud web API through HTTP. The requests to iCloud web also contain a set of attributes as parameters. Most important attributes are defined on Table 3.

Similar to Box, the registry is checked to find relevant keys of the disk image. Most meaningful ones are SOFTWARE/ Classes/AppID/iCloudServices.EXE which specifies the identification number assigned by the system to iCloud client program when it was installed in the machine, SOFTWARE/ Classes/ iCloudServices.AccountInfo., providing additional mapping of other iCloud-related registry keys within the registry, SOFTWARE/Classes/Wow6432Node/AppID/iCloud Services.EXE, being an alternative way to find the AppID of the iCloud sync program., NTUSER/Software/Microsoft/ InternetExplorer/ TypedURLs containing URLs typed by the user in Internet Explorer, and NTUSER/ Software/ Microsoft/ WindowsNT/CurrentVersion/AppCompatFlags/Compatibility Assistant/ Persisted. which contains a list of files names downloaded from the browse.

Metadata Tag	Description
hs	Value identifying the user within the cloud service. It is constant for the files belonging to the same iCloud user
k	ID of the file within the iCloud drive service. This ID remains the same for the same file and is used in order to identify it within the cloud service.
e	Time the authorization to see the file on the web link is valid. It lasts for one hour since the request of the file.

Table 4. iCloud metadata remnants

In case the sync program is installed within the browser, the installation path (i.e. by default being \Program Files (x86)\Common Files\Apple\Internet Services) contains installation information. The synchronization program also makes use of the AppData (under [*user*]*/AppData/Local/Apple Inc/iCloudDrive/*) folder in order to store configuration parameters and other useful information. It is of particular interest the existence of SQLite databases. They are used to provide different parameters regarding the user account, such as account details or the synchronzed files within the cloud server, e.g. images or text files. Finally, the acquisition of the synchronization folder (by default located at Users\[user]\iCloudDrive) presents synced files with the cloud platform at the moment of the disk dump acquisition.