Digital Investigation 16 (2016) S104–S113

Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin

DFRWS 2016 Europe — Proceedings of the Third Annual DFRWS Europe

Forensic analysis of cloud-native artifacts

Vassil Roussev*, Shane McCulley

Greater New Orleans Center for Information Assurance (GNOCIA), University of New Orleans, New Orleans, LA, 70148, USA

Keywords:
Cloud forensics
Google docs format
Reverse engineering
Cloud-native artifacts
kumodocs
kumodd

A B S T R A C T

Forensic analysis of cloud artifacts is still in its infancy; current approaches overwhelming follow the traditional method of collecting artifacts on a client device. In this work, we introduce the concept of analyzing *cloud-native digital artifacts*—data objects that maintain the persistent state of web/SaaS applications. Unlike traditional applications, in which the persistent state takes the form of files in the local file system, web apps download the necessary state on the fly and leave no trace in local storage.

Using *Google Docs* as a case study, we demonstrate that such artifacts can have a completely different structure—their state is often maintained in the form of a complete (or partial) log of user editing actions. Thus, the traditional approach of obtaining a snapshot in time of the state of the artifacts is *inherently* forensically deficient in that it ignores potentially critical information on the evolution of a document over time. Further, cloud-native artifacts have no standardized external representation, which raises questions with respect to their long-term preservation and interpretation.

Introduction

The traditional business model of the software industry has been *software as a product* (SaaP); that is, software is acquired like any physical product and, once the sale is complete, the owner can use it as they see for an unlimited period of time. The alternative—*software as a service* (SaaS)—is a subscription-based model, which did not start becoming practical until the emergence of widespead Internet access some two decades ago. Conceptually, the move from SaaP to SaaS shifts the responsibility for operating the software and its environment from the customer to the provider. Technologically, such a shift was enabled by the growth of the Internet as a universal means of communications, and was facilitated by the emergence of the web browser as a standardized client user interface (UI) platform.

The traditional analytical model of digital forensics has been client-centric—the investigator works with physical evidence carriers, such as storage media or integrated compute devices (e.g., smartphones). On the client (or standalone) device it is easy to identify where the computations are performed and where the results/traces are stored. Therefore, research has focused on discovering and acquiring every little piece of log and timestamp information, and extracting every last bit of discarded data that applications and the OS may have left behind.

The introduction of *Gmail* in 2004—the first web 2.0 application in widespread use—demonstrated that all the essential technological prerequisites for mass, web-based SaaS deployments have been met. The introduction of the first public cloud services by Amazon in 2006 enabled *any* vendor to rent scalable, server-side infrastructure and become a SaaS provider. A decade later, the transition to SaaS is moving at full speed, and the need to understand it forensically is becoming ever more critical.

This massive technological shift presents a *qualitatively* new challenge for forensics; one that cannot be addressed by minor adjustments to tools and practices. Specifically,

* Corresponding author.
E-mail addresses: vassil@roussev.net (V. Roussev), smcculle@my.uno.edu (S. McCulley).

http://dx.doi.org/10.1016/j.diin.2016.01.013

the SaaS model disrupts the familiar client-centric world—both code and data are delivered over the network on demand, and thus become moving forensic targets. For example, a *Google Docs* document shows up as nothing more than a hyperlink on the local disk; the actual content is downloaded and made available for editing only in the browser.

In this work, we approach the problem by going directly to the data source—the service provider—using both public and private APIs and data structures. This leads to a new approach that, we believe, is a preview of how cloud forensic tools will be built.

Related work

The primary focus of previous work on cloud storage forensics has been on adapting the traditional application forensics approach to finding client-side artifacts. This involves blackbox differential analysis, where before and after images are created and compared to deduce the essential functions of the application. Section (Client-based data acquisition & analysis) summarizes representative work in this area.

Section (API-based data acquisition & analysis) presents a more recent alternative, which seeks to avoid the limitations of client acquisition by working with the provider's API.

Client-based data acquisition & analysis

Chung et al. (2012) analyzed four cloud storage services (*Amazon S3*, *Google Docs*, *Dropbox*, and *Evernote*) in search of traces left by them on the client system that can be used in criminal cases. They reported that the analyzed services may create different artifacts depending on specific features of the services, and proposed a process model for forensic investigation of cloud storage services based on the collection and analysis of artifacts of the target cloud storage services from client systems. The procedure includes gathering volatile data from a Mac/Windows system (if available), and then retrieving data from the Internet history, log files, and directories. On mobile devices they rooted an Android phone to gather data and for iPhone they used iTunes information like backup iTunes files. The objective was to check for traces of a cloud storage service exist in the collected data.

In Hale (2013), Hale analyzes the *Amazon Cloud Drive* and discusses the digital artifacts left behind after an Amazon Cloud Drive account has been accessed or manipulated from a computer. There are two possibilities to manipulate an Amazon Cloud Drive Account: one is via the web application accessible using a web browser and the other is a client application provided by Amazon and can be installed on the system. After analyzing the two methods, he found artifacts of the interface in the web browser history, and among cached files. He also found application artifacts in the Windows registry, application installation files on default location, and an SQLite database used to keep track of pending upload/download tasks.

Quick and Choo (2013) analyzed *Dropbox* and discuss the artifacts left behind after a *Dropbox* account has been accessed, or manipulated. Using hash analysis and keyword searches they try to determine if the client software provided by *Dropbox* has been used. This involves extracting the account username from browser history (Mozilla Firefox, Google Chrome, and Microsoft Internet Explorer), and the use of the *Dropbox* through several avenues such as directory listings, prefetch files, link files, thumbnails, registry, browser history, and memory captures. In follow-up work, Quick and Choo (2014) use a similar conceptual approach to analyze the client-side operation and artifacts of *Google Drive,* and provide a starting point for investigators.

Martini and Choo (2013) have researched the operation of *ownCloud*, which is a self-hosted file synchronization and sharing solution. As such, it occupies a slightly different niche as it is much more likely for the client and server sides to be under the control of the same person/organization. They were able to recover artifacts including sync and file management metadata (logging, database and configuration data), cached files describing the files the user has stored on the client device and uploaded to the cloud environment or vise versa, and browser artifacts.

API-based data acquisition & analysis

The client-side acquisition approaches discussed so far have one big assumption in common; namely, that *all* the data artifacts of interest *can* be acquired from the client. The problem is that this is *not* true in the general case, and is likely to be not true in the common case. As illustrated on Fig. 1, the client can no longer be considered the original source of the data. Rather, it maintains a *cached* version that is *likely* incomplete (in more ways than one) and potentially out of date.

Considering the above functional architecture, there are three major lapses in client-based acquisitions of cloud-hosted data:

Partial replication. The most obvious problem is that none of the clients working with a cloud storage account may have a complete copy of the data. Currently, cloud storage providers offer selective replication so that devices with less local storage (smartphones) are not overwhelmed. Going forward, as data accumulates online, it would become increasingly impractical (and unnecessary) to maintain a complete local copy. Amazon already offers unlimited storage at $60/year, and that is a lot of data to clone locally. From a forensic standpoint, a client-based acquisition is blind to the overall picture, and has no means to guarantee completeness.

Artifact revisions. Most storage services provide automatic revision tracking that keeps copies of previous

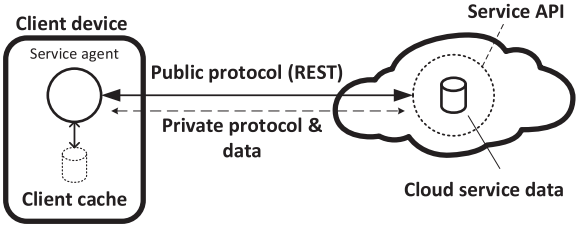

Fig. 1. SaaS application architecture.