Summer 8-11-2015

# API-Based Acquisition of Evidence from Cloud Storage Providers

Andres E. Barreto
*University of New Orleans*, aebarret@uno.edu

# API-Based Acquisition of Evidence from Cloud Storage Providers

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
Computer Science

by

Andres E. Barreto

B.Eng. Escuela Superior Politécnica del Litoral (ESPOL), 2006

August 2015

# Abstract

Cloud computing and cloud storage services, in particular, pose a new challenge to digital forensic investigations. Currently, evidence acquisition for such services still follows the traditional approach of collecting artifacts on a client device. In this work, we show that such an approach not only requires upfront substantial investment in reverse engineering each service, but is also inherently incomplete as it misses prior versions of the artifacts, as well as cloud-only artifacts that do not have standard serialized representations on the client.

In this work, we introduce the concept of *API-based evidence acquisition* for cloud services, which addresses these concerns by utilizing the officially supported API of the service. To demonstrate the utility of this approach, we present a proof-of-concept acquisition tool, *kumodd*, which can acquire evidence from four major cloud storage providers: *Google Drive*, *Microsoft One*, *Dropbox*, and *Box*. The implementation provides both command-line and web user interfaces, and can be readily incorporated into established forensic processes.

**Keywords:** Cloud forensics, cloud storage, kumodd, API-based evidence acquisition, Google Drive, Dropbox, OneDrive, Box.

# Contents

**1 Introduction** **1**

**2 Background** **3**

2.1 Cloud computing . . . 3

2.1.1 Infrastructure as a Services (IaaS) . . . 3

2.1.2 Platform as a Service (PaaS) . . . 3

2.1.3 Software as a Service (SaaS) . . . 4

2.1.4 Issues at each stage of a forensic investigation . . . 4

2.2 Cloud forensics . . . 5

2.2.1 Saas Forensics . . . 5

2.3 Summary . . . 6

**3 Design and Implementation** **8**

3.1 Architecture . . . 8

3.2 Tool Use . . . 9

3.2.1 Command-line arguments . . . 9

3.2.2 User authentication . . . 11

3.2.3 Content discovery and acquisition . . . 12

3.2.4 Web GUI . . . 15

3.3 Implementation . . . 18

3.3.1 Development environment . . . 18

3.3.2 Python . . . 18

3.3.3 Packages . . . 18

3.4 Installation . . . 19

3.4.1 Requirements . . . 19

3.4.2 Package installation . . . 19

3.5 Configuration . . . 20

**4 Cloud Storage Providers APIs** **21**

4.1 Google Drive . . . 21

4.1.1 API authorization . . . 21

4.1.2 `Files` resource . . . 23

4.2 Dropbox . . . 26

4.2.1 API authorization . . . 26

4.2.2 Account information . . . 27

4.2.3 Files and metadata . . . 28

4.3 OneDrive . . . 29

4.3.1 API authorization . . . 29

4.3.2 Drive resource . . . 30

4.3.3 Item resource . . . 31

4.4 Box . . . 33

4.4.1 API authorization . . . 33

4.4.2 File object . . . 33

4.4.3 Folder object . . . 35

**5 Evaluation** **37**

5.1 Functionality . . . 37

5.1.1 Filtering . . . 37

5.1.2 Acquisition . . . 37

5.2 Download times . . . 37

5.3 Completeness . . . 38

5.3.1 Mirrored data . . . 38

5.3.2 Cloud-only data . . . 38

5.4 Discussion . . . 38

5.5 Future work . . . 39

**6 Conclusion** **40**

**Appendix A Google Drive API** **43**

A.1 Google Drive resources . . . 43

A.1.1 About Resource . . . 43

A.1.2 Children resource . . . 44

A.1.3 Revisions resource . . . 46

A.2 Google Drive resource representations . . . 48

A.2.1 About resource . . . 48

A.2.2 Files resource . . . 49

A.2.3 Children resource . . . 51

A.2.4 Revisions resource . . . 52

**Vita** **53**

# List of Figures

2.1 Cloud service models and ownership of layers . . . . 4

3.1 *Kumodd* architectural diagram . . . . 9
3.2 Authentication step 2: Provide account credentials . . . . 11
3.3 Authentication step 3: Authorize access to the account by `kumodd` . . . . 12
3.4 Steps of evidence acquisition . . . . 13
3.5 Contents of the generated CSV file . . . . 13
3.6 Drive contents downloaded to a local directory . . . . 14
3.7 Metadata of every downloaded file saved in a local directory . . . . 15
3.8 Web GUI: Service selection . . . . 16
3.9 Web GUI: File selection . . . . 17
3.10 Web GUI: Results . . . . 17
3.11 Service credentials in the developer console (Google Drive) . . . . 20

4.1 Google Developers Console . . . . 22
4.2 Files:list method response example (abbreviated) . . . . 24
4.3 Files:get method response example (abbreviated) . . . . 25
4.4 Dropbox App Console . . . . 26
4.5 OAuth 2 flow diagram. [1] . . . . 27
4.6 OAuth 2 flow diagram. [2] . . . . 30
4.7 Box Developers Console . . . . 33
4.8 Get file response example (abbreviated) . . . . 34

A.1 About:get method response example (abbreviated) . . . . 44
A.2 Children:get method response example . . . . 45
A.3 Revisions:get method response example . . . . 47

# List of Tables

4.1 Files resource methods; all URIs relative to https://www.googleapis.com/drive/v2. A complete list of methods is available in in the Drive REST API reference provided by Google [26] . . . 23
4.2 OneDrive API authorization scopes . . . 30

5.1 Kumodd download times . . . 38

A.1 Children resource methods; URIs relative to https://www.googleapis.com/drive/v2. A complete list of methods is available in in the Drive REST API reference provided by Google . . 45
A.2 Revisions resource methods; URIs relative to https://www.googleapis.com/drive/v2. A complete list of methods is available in in the Drive REST API reference provided by Google [26] . . . 46

# Listings

3.1 Listing all files stored in a Dropbox account . . . . . . . . . . . . . . . . . . . . . . . . . 10
3.2 Listing only images stored in a Box account . . . . . . . . . . . . . . . . . . . . . . . . . 10
3.3 Download just PDF files stored in OneDrive and save the in the Desktop folder . . . . . . . . 10
3.4 Download files from Google Drive using files listed in a CSV file stored in /home/user/Desktop/ 10
3.5 Authentication step 1: Connect to Google Drive for the first time . . . . . . . . . . . . . . . . 11
3.6 Authentication step 4: Input access code in the terminal (if necessary) and allow the application to retrieve results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
3.7 Output of listing all files in a Google Drive account (partial results are being shown) . . . . . 13
3.8 Starting the web GUI . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
3.9 Kumodd download . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
3.10 Obtaining the Python version . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
3.11 Installation of Python 2.7.6 and its dependencies on Debian/Ubuntu . . . . . . . . . . . . . . 19
3.12 python-gflags installation command . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
3.13 google-api-python-client installation command . . . . . . . . . . . . . . . . . . . . . . . 20
3.14 dropbox installation command . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
3.15 boxsdk installation command . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
3.16 Flask and Flask-Classy installation command . . . . . . . . . . . . . . . . . . . . . . . . 20
4.1 Google Drive Authorization scopes [28] . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
4.2 Files:list response tructure . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
4.3 Dropbox user information example . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
4.4 get_file request using the Dropbox Python library . . . . . . . . . . . . . . . . . . . . . . 28
4.5 Metadata of a file in Dropbox . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28
4.6 revision request using the Dropbox Python library . . . . . . . . . . . . . . . . . . . . . . 28
4.7 Revisions method response body example . . . . . . . . . . . . . . . . . . . . . . . . . . 29
4.8 Drive resource response example . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
4.9 Item resource response example . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
4.10 OneDrive list children example . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32
4.11 Successful download file response . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32
4.12 Successful Content API authorization response . . . . . . . . . . . . . . . . . . . . . . . 33
4.13 Array of versions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
4.14 Items inside a folder . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36
A.1 Children:list response body structure . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46
A.2 Revisions:list response body structure . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48
A.3 About resource representation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48
A.4 Files resource representation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49
A.5 Children resource representation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
A.6 Revision resource representation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 52

# Chapter 1

# Introduction

Cloud computing (or, *the cloud*) is the emerging primary model for delivering information technology (IT) services to Internet-connected devices. It abstracts away the physical compute and communication infrastructure, and allows customers to effectively rent, instead of own and maintain, as much compute capacity as needed. As per NIST's definition [3], there are five characteristics–*on-demand self service*, *broad network access*, *resource pooling*, *rapid elasticity*, and *measured service*–that distinguish the cloud service model from previous ones.

The underpinning technology development that has made the cloud possible is the massive adoption of virtualization on commodity hardware systems. Ultimately, it allows for a large pool of resources, such as a data center, to be provisioned and load-balanced at a fine granularity, and for the computations of different users (or uses) to be strongly isolated.

The first public cloud services–*Amazon Web Services* (AWS)–were introduced by Amazon in 2006. As of 2015, according to RightScale's *State of the Cloud Report* [4], cloud adoption has become ubiquitous: 93% of businesses are at least experimenting with cloud deployments, with 82% adopting a hybrid strategy, which combines the use of multiple providers (usually in a public-private configuration). Nonetheless, much of the technology transition is still ahead as 68% of enterprises have less than 20% of their application portfolio running in a cloud setup. Similarly, *Gartner* [5] predicts another 2-5 years will be needed before cloud computing reaches the "plateau of productivity" marking the period of mass mainstream adoption and widespread productivity gains.

Unsurprisingly, cloud forensics is still in its infancy; there are few practical solutions for the acquisition and analysis of cloud evidence and most of them are minor adaptations of existing methods and tools. Indeed, NIST–the main standardization body in the US–is still working to build consensus on what the *challenges* are with respect to performing forensics of cloud data; in [3] they have enumerated 65 separate ones.

In this work, we are concerned with one particular problem–the acquisition of data from cloud storage services. These have emerged as one of the most popular services with the average consumer as many providers, such as *Dropbox*, *Box*, *Google Drive*, and Microsoft's *OneDrive*, allow between 2 and 15GB of cloud storage for free. Cloud storage is also widely used on mobile devices to share data across applications (which are otherwise isolated from each other). Therefore, having a robust evidence acquisition method is a necessity *today*; further, due to the wide variety of these services, and the rapid introduction of new ones, the tool and methodology should be adaptable and extensible.

In traditional forensic models, the investigator works with physical evidence carriers, such as storage media or integrated compute devices. Thus, it is possible to identify the computer performing the computations and the media that store (traces of) the processing, and to physically collect, preserve and analyze information content. Because of this, 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 have left behind.

Conceptually, cloud computing breaks this model in two major ways. First, resources–CPU cycles, RAM, storage, etc.–are first pooled (e.g., RAID storage) and then allocated at a fine granularity. This results in physical media potentially containing data owned by many users, and to data relevant to a single case being spread among numerous providers. Applying the conventional model creates a long list of procedural, legal, and technical problems that are unlikely to have an efficient solution in the general case. Second, both computations and storage contain a much more ephemeral record as virtual machine (VM) instances are created and destroyed with regularity and working storage gets sanitized.

As we discuss in more detail in the next chapter, current work on cloud storage forensics has treated the problem as just another instance of application forensics. It applies basic differential analysis techniques to gain a basic understanding of the artifacts left on client devices by taking before and after snapshots of the target compute system, and deducing relevant cause and effect relationships. During an actual investigation, the analyst would be interpreting the state of the system based on these known relationships.

Unfortunately, there are several serious problems with this extension of existing client-side methods:

- *Completeness.* The reliance in client-side data can leave out critical case data. One example is older versions of the files, which most services provide; another one is cloud-only data, such as a *Google Docs* document, which literally has no serialized local representation other than a link. Some services, such as a variety of personal information management applications, live only in the browser so a flush of the cache would make them go away.

- *Reproducibility.* As cloud storage applications get updated on a regular basis and versions have a relatively short live span, it becomes harder to maintain the reproducibility of the analysis and make require frequent repetition of the original procedure.

- *Scalability.* As a continuation of the prior point, manual client-side analysis is burdensome and simply does not scale with the rapid growth of the variety of services and their versions.

In this work, we propose an alternative approach for the acquisition of evidence data from cloud storage providers that uses the official APIs provided by the services. Such an approach has the major advantage of taking all the reverse-engineering work out of the picture:

- APIs are well-documented, official interfaces through which cloud applications on the client communicate with the service; they tend to change slowly and changes are clearly marked–only new features need to be incrementally incorporated into the acquisition tool.

- It is easy to demonstrate completeness (based on the API specification) and reproducibility becomes straightforward.

- Web APIs tend to follow patterns, which makes it possible to adapt existing code to a new (similar) service with modest effort. It is often feasible to write an acquisition tool for a completely new service from scratch in a short time.

To demonstrate the feasibility of our approach, and to gain first-hand experience with the process, we have developed a proof-of-concept prototype called *kumodd*[1], which can perform complete (or partial) acquisition of a cloud storage account's data. It works with four popular services–*Dropbox*, *Box*, *Google Drive*, and Microsoft's *OneDrive*–and supports the acquisition of revisions and cloud-only documents. The prototype is written in *Python* and offers both a command line and web-based user interfaces.

[1]The tool name is derived from the Japanese word for cloud *kumo*.

# Chapter 2

# Background

## 2.1 Cloud computing

The reference definition of cloud computing provided by the National Institute of Standards and Technology (NIST) states that “cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” [3]. With respect to *public* cloud services–the most common case–this means that the physical hardware on which the computation takes place is owned and maintained the provider, as is at least part of the deployed software stack. Generally, customers have the option to pay per unit of CPU/storage/network use, although other business arrangements are also possible.

Depending on which part of the software stack is managed by the provider, there are three baseline deployment options:

### 2.1.1 Infrastructure as a Services (IaaS)

In IaaS, cloud service providers create virtual machines and assign their full control to customers. Customers then install operating system, and applications within the machine without any interference from the service providers. Amazon Web Service (AWS), Microsoft Azure, Google Compute Engine (GCE) are popular examples of IaaS. IaaS provides capabilities to take snapshots of the disk and physical memory of virtual machines, which has significant forensic value for quick acquisition of disk and memory. Since virtual machines have closely represent physical machines, the traditional forensic tools for data acquisition and analysis can also be used inside the virtual machines as remote investigation of a physical machine is performed. Furthermore, virtual machine introspection provided by hypervisor enables cloud service providers to examine live memory and disk data, perform instant data acquisition and analysis. However, since the functionality is supported at hypervisor level, customers cannot take advantage of this functionality [6]. In summary IaaS consists of the rental and use of hardware and the user is in charge of operating system and applications deployment.

### 2.1.2 Platform as a Service (PaaS)

PaaS essentially consists of the use operating system supplied by the provider and the user deploying the software to run on them, customers develop their applications using software components built into middleware. Google App Engine is a example of PaaS, offering quick and cost-effective solution for developing, testing, and deploying customer applications. In other words, the cloud infrastructure hosts customer-developed applications. PaaS provides full control to customers on the application layer including interaction of applications with dependencies (such as databases, storage etc.) and enabling customers to perform extensive logging for forensics and security purposes [6].

### 2.1.3 Software as a Service (SaaS)

In SaaS, the provider is responsible for everything and customers access the functionality as either a web application, or a platform-specific mobile client. Cloud service manage all the layers including application layer that runs the software offered as a service to customers. In other words, customer has no control on the underlying operating infrastructure and application. However, since cloud service provider manages the infrastructure (including application), the maintenance cost on customer side is almost zero.. Google Gmail, Microsoft 365, Salesforce, Citrix GoToMeeting, Cisco WebEx are popular examples of SaaS, which run directly from web browser without downloading and installing any software. Their desktop and smartphone versions are also available to run on client machine. The applications whether running on web browser or desktop have limited presence on the client machine making investigators rely on server-side logs. SaaS applications log extensively, which has significant forensic value. For instance, Google Docs records every insert, update, and delete operation of characters performed by user along with the timestamps, which makes it possible to identify specific changes made by different user in a document [6].

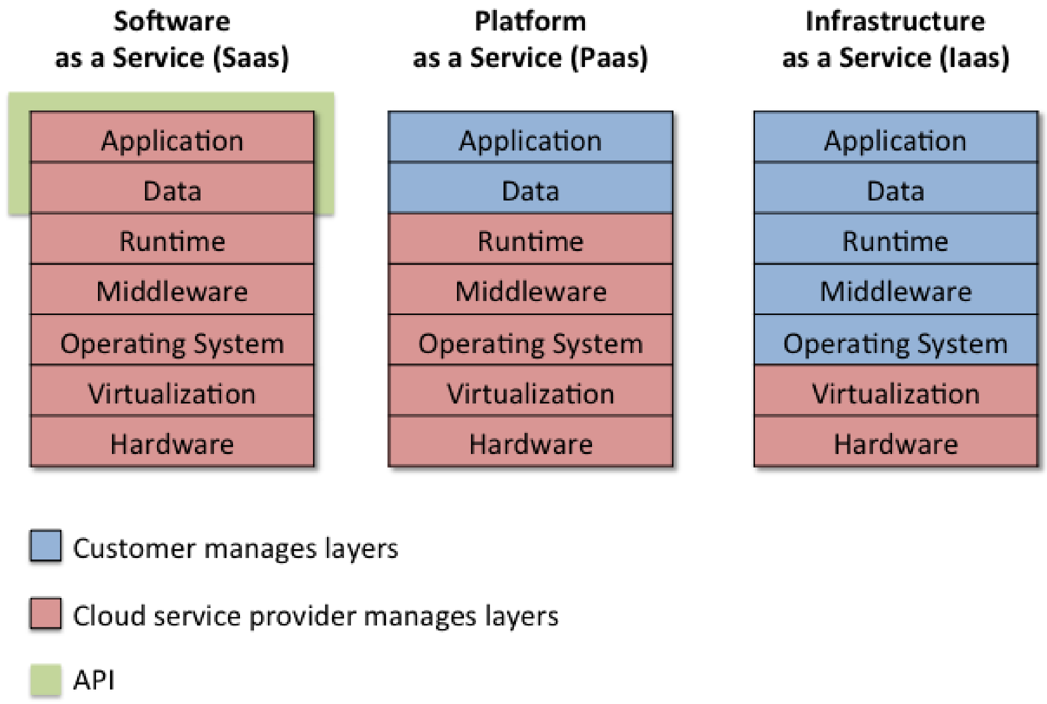

Software as a Service (Saas)
Platform as a Service (Paas)
Infrastructure as a Service (Iaas)
Application
Data
Runtime
Middleware
Operating System
Virtualization
Hardware
Customer manages layers
Cloud service provider manages layers
API

Figure 2.1: Cloud service models and ownership of layers

### 2.1.4 Issues at each stage of a forensic investigation

Using the process proposed by McKemmish [7], we briefly outline the main issues that can be encountered in each stage of forensic analysis in a cloud environment.

#### Data identification

One of the main issues in analyzing evidence that could lead to information stored in the cloud is to identify where the data resides and what services where being used. In these cases the source of evidence on the client system will be web browser information such as history caches. In the case of cloud storage services it will be remnants of client application installations and also files downloaded for local use and edition using them.

#### Data preservation

Data stored in the cloud can appear and disappear or change from location rapidly, which makes it difficult to collect. Volatility of data have to be considered when undertaking collection from cloud resources. Tools that collect data from a cloud storage service with a focus on forensic investigations are not currently available [8]. As present, there are only two ways of getting information from a cloud drive account; via web browser login or using a client application provided by the service. It is important to have access to a tool that does not manipulate evidence from its source in any way.

#### Data analysis

One fast growing problem is the large amount of data may need to be obtained. Therefore, an important feature of the next generation of acquisition and analysis tools will be the ability to pre-screen and selectively obtain relevant data. In traditional, client-centric forensics, much of the filtering is based on file crypto hashes and timestamp data. However, these will need to be adapted to a new environment where neither filesystem metadata (such as the *MS Windows* registry), nor standard crypto hashes (e.g. *Dropbox*) are available [8]. At the same time, individual services may offer *search* capabilities that can potentially save substantial amounts of processing time.

## 2.2 Cloud forensics

Cloud forensics is defined as "the application of digital forensics in cloud computing as a subset of network forensics. It is a cross reference between cloud computing and digital forensics" [9]. This paper focuses in SaaS forensics (cloud storage forensics specifically) as the proposed application was developed to work with cloud storage providers which fall in the Software as a Service classification.

Cloud storage related crimes fall in the electronic crime category which involves the use of a computer as a tool, target or storage device. Data stored in cloud services can be a target for criminals, while cloud storage can be used to save illicit data or crime related data. Computer related crime has three factors that attract people to commit them; motivation, opportunity and the absence of capable guardianship [10]. As online services adoption grow, criminal are tempted to exploit these opportunities.

### 2.2.1 Saas Forensics

Overall, there have been very few publications and practical tools to address identification, preservation and analysis of cloud-stored data. The main approach has been to analyze trace data and remnant artifacts left behind by the client applications. Since the applications are accessed from a client machine, remnants of digital artifacts pertaining to user activities on the applications are present and can be forensically retrieved and analyzed from the hard disk of the machine.

#### Cloud application forensics

Somers [11] has developed *DraftBack*–a browser extension that can replay the activities in a document created in *Google Docs*. As it turns out, the latter records and timestamps *all* user inputs since the beginning the creation of the document. *DraftBack* makes use of timestamps associated with every change made to a document since it was created to identify sequence of changes, and derives the state of the document at any instance of time.

Somers' work on Google Docs, although not motivated by forensics, is probably the single best example of SaaS analysis that does not rely on trace data resident on the client–all results are produced solely by querying the web application's interface. Assuming that an investigator is in possession of valid user credentials, the examination can be performed on the spot. This is likely to be sufficient for most informal scenarios; however, in order to make it to the court room, some additional tooling and procedure need to be developed. Clearly, the acquisition and long-term preservation of the evidence is yet to be addressed; yet another concern is the ability to replay the acquired log in the application. Unlike client-side applications, a web app resides entirely in the server, and there is no practical way to acquire an archival copy of the execution environment as of a particular date. One possible solution is to record a screencast video of the entire user editing session.

Microsoft Office 365 also keeps detailed revisions based on edits as it supports real-time collaboration, so–in principle–it should be possible to develop a *Draftback*-like tool. Other cloud drive services offer generic file revision history, however, the available data is more limited as it is stored as simple snapshots.

#### Cloud drive forensics

Chung et al. [12] analyzed four cloud storage services (Amazon S3, Google Docs, Dropbox, and Evernote) in search for traces left by them in the client system that can be helpful to investigate criminal cases. In their research, Chung et al. reported that the analyzed services may create different artifacts depending on specific features of the services. Chung et al. proposed a process model for forensic investigation of cloud storage services which is based in the collection and analysis of artifacts of the analyzed cloud storage services from client systems. The procedure includes gathering volatile data from a Mac or Windows system (if available), and then retrieving data from the Internet history, log files, and directories. In 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.

Hale [13] analyzed 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 in the system. After analyzing the two methods he found artifacts of the interface on web browser history, and cache files. He also found that the client application left artifacts on Windows registry, application installation files on default location, and an SQLite database used by the application to hold upload and download tasks while the status of the task is pending.

Quick et al. [14] analyzed Dropbox and discusses the digital 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. They get the Dropbox 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.

Martini and Choo [15] discuss the digital artifacts of ownCloud (server and client side). 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.

Clark [16] performed metadata analysis of pictures in SkyDrive (now called OneDrive), Windows Azure and Flickr discovering that information like positioning can be obtained from publicly shared pictures.

## 2.3 Summary

Considering previous work on cloud storage forensic analysis, it is evident that research and development efforts have been focused on the traditional approach of finding local artifacts on the client. This is a limited and inefficient approach requiring considerable manual reverse engineering effort, which ultimately is not guaranteed to be complete. In our view, the future of SaaS forensics lies in working *with* the supporting web infrastructure the way web applications do: through APIs.

Currently, the amount of data in cloud storage is relatively small (compared to the size of local storage) as free services only offer up to 15GB. However, as adoption of these services increases, a fast and substantial increase in volume is expected. As an example, GoogleDrive currently offers up to 30TB at $10/month per terabyte. In other words, cloud storage acquisition and analysis is a problem that we can expect to grow very quickly and current approaches offer little hope of being able to cope with it.

# Chapter 3

# Design and Implementation

The traditional local storage forensic processing pipeline–physical acquisition, file carving, analysis of unallocated space, etc–which is historically based on the properties of file systems for magnetic disks, is quickly becoming obsolete. For example, the mass adoption of solid-state drives (SSD), which automatically sanitize unused space for performance reasons has eliminated unallocated space as a source of evidence. Large HDD drives are increasingly encrypted as the performance penalty is gone; also, sanitizing a slow 6/8/10TB drive is only practical by encrypting it and disposing of the key.

More generally, *all* storage devices are becoming “smart” by incorporating ever more powerful processors and ever more complex data layout algorithms. The concept of “physical” acquisition is on its way out as the CPU does not have direct access to the media and all data access becomes *de facto* logical, even for local storage. To be sure, storage devices still support legacy protocols that nominally contain addresses, however, there is no promise that this maps to the actual physical layout. In cloud deployments, all storage including HDD-backed is pooled (RAID-ed), virtualized, shared, and automatically sanitized to avoid data leaks. Therefore, attempting a traditional “physical” acquisition creates a long list of legal, procedural, and technical problems.

Considering cloud storage services from a user perspective, they offer a similar functionality to a physical drive–persistent file storage and retrieval. As files are not stored locally, an Internet connection is clearly necessary, as is a client application (or web browser) to provide a convenient UI to retrieve, edit or share them. This setup, as explained in Section 2.1.4, is a new challenge for forensics as none of the traditional approaches for retrieving evidence can be considered valid.

Fortunately, cloud storage providers offer a well-defined mechanism to work their service via a public API. Indeed, the applications installed on client devices work through the same mechanism (with the possible addition of undocumented API calls). An API opens up the opportunity to build a forensic-specific client, which can enumerate and acquire cloud-hosted artifacts in a controlled fashion. Further, an API has well-defined semantics, which can be modeled and tested to ensure sound acquisition. This is in stark contrast to the traditional approach of collecting leftover client artifacts, which treats the application as a black box whose functionality is to be reverse engineered. One clear limitation here is that an application can only work with the functionality exposed by the service; however, this is not likely to be a major restriction for basic forensic processing, such as data acquisition.

For the remainder of this chapter, we describe the design and implementation of a proof-of-concept prototype called `kumodd`; it employs the public API of four popular cloud storage services to obtain a usable forensic image of the content of a user account.

## 3.1 Architecture

The overall architecture of `kumodd`, as shown on Figure 3.1, consists of three separate layers–user interface, dispatcher, and acquisition drivers. The user interface is responsible for interaction with the forensic analyst and provides both a web-based GUI (suitable for interactive exploration), and a command-line interface (suitable for automation). The dispatcher collects acquisition parameters and schedules the acquisition tasks against the forensic targets. The tasks are executed by service-specific driver modules that work via the public API supplied by the cloud provider and return the data in different, user-specified formats, such as JSON and CSV.

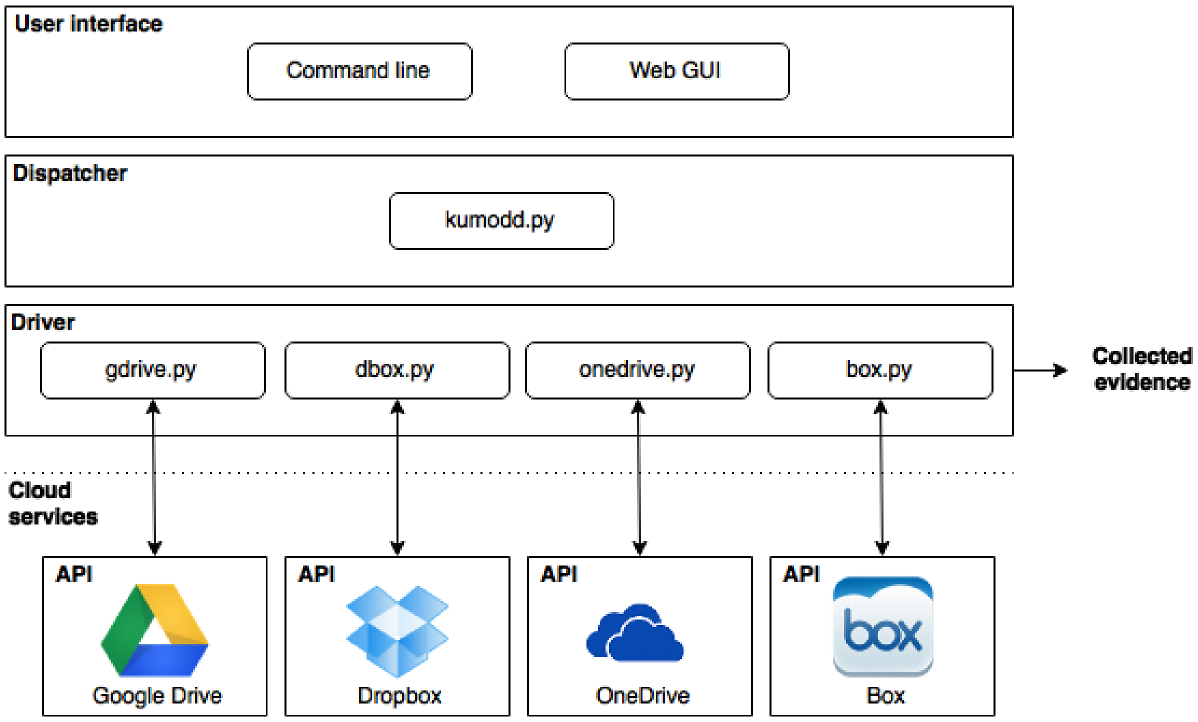


Figure 3.1: *Kumodd* architectural diagram

Overall, the service APIs are similar in the type of capabilities they provide. However, as our further discussion will show, they vary substantially in their implementation and provide varying levels of detail in terms of artifact metadata. Eventually, we expect the different drivers to provide a common, high-level interface that can abstract away most of the implementation details.

## 3.2 Tool Use

After `kumodd` is properly configured–a process described later in this chapter–it is ready to be used from the terminal.

### 3.2.1 Command-line arguments

The general format of the commands is:

```
python kumodd.py -s [service] [action] [filter]
```

The following are the currently supported argument values:

#### [service]

The type of cloud service being accessed:

| | |
|---|---|
| `gdrive` | Google Drive |
| `dropbox` | Dropbox |

`onedrive` Microsoft OneDrive

`box` Box

**[action]**

The action to be perform using the selected service:

`-l` List files stored in an account drive as a plain text table.

`-d` Download files stored in an account drive in an specified location.

`-csv` Use a CSV file to specify which files to download.

**[filter]**

The **[filter]** parameter specifies the subset of files to be listed/downloaded based on file type:

`all` All files stored in an account drive;

`doc` Only .doc/.docx/.odf files;

`xls` Only spreadsheet files;

`ppt` Only presentation files;

`text` Only text/source code files;

`pdf` Only PDF files;

`officedocs` All document, spreadsheet and presentation files;

`image` Only images;

`audio` Only audio files;

`video` Only video files;

`<file>` Only with `-csv` action: CSV file containing the files to be downloaded.

`-p <path>` Used with `-d` and `-csv` action: Used to specify the path in which the files will be downloaded.

**Command line examples**

```
python kumodd.py -s dbox -l all
```

Listing 3.1: Listing all files stored in a Dropbox account

```
python kumodd.py -s box -l image
```

Listing 3.2: Listing only images stored in a Box account

```
python kumodd.py -s onedrive -d all -l /home/user/Desktop/
```

Listing 3.3: Download just PDF files stored in OneDrive and save the in the Desktop folder

```
python kumodd.py -s gdrive -csv /home/user/Desktop/gdrive_list.csv
```

Listing 3.4: Download files from Google Drive using files listed in a CSV file stored in /home/user/Desktop/

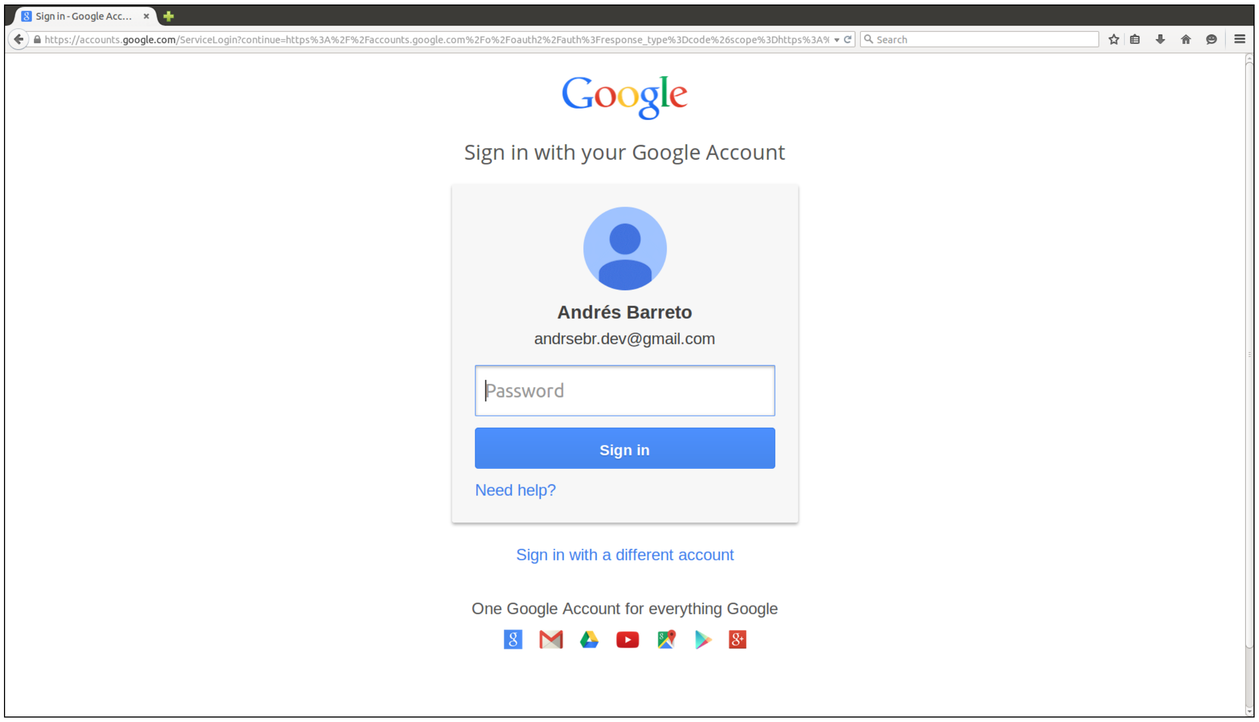
### 3.2.2 User authentication

When `kumodd` is used for the first time to connect to a cloud service, the respective driver will initiate the authorization process which requires the user to provide access credentials (user/password) for the account. The tool provides the user with a URL that needs to be opened in a web browser, where the standard authentication interface for the service will request the relevant username and password:

```
andres@ubuntu:~/kumodd$ python kumodd.py -s gdrive -d all
Your browser has been opened to visit:

https://accounts.google.com/o/oauth2/auth?scope=https%3A%2F%2Fwww.googleapis.com...

--noauth_local_webserver
```

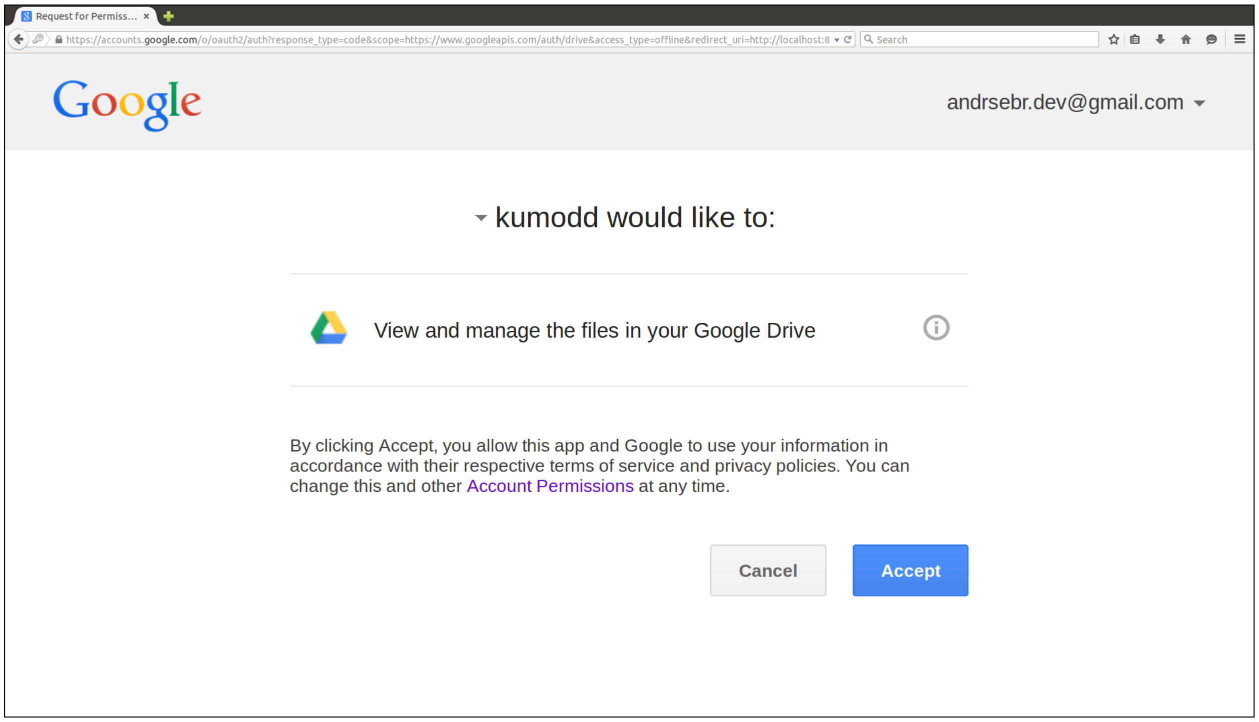
Listing 3.5: Authentication step 1: Connect to Google Drive for the first time

After supplying the correct credentials, the service returns an access code which the user has to input in the command line to finish the authentication process and authorize `kumodd` access to the account (Figures 3.2, 3.3). If the authentication is successful, the provided access token is cached persistently in a `.dat` file is saved under `/config` folder with the name of the service. Future requests will find the token and will not prompt the user for credentials.



Sign in - Google Acc...
https://accounts.google.com/ServiceLogin?continue=https%3A%2F%2Faccounts.google.com%2Fo%2Foauth2%2Fauth%3Fresponse_type%3Dcode%26scope%3Dhttps%3A%
Search
Google
Sign in with your Google Account
Andrés Barreto
andrsebr.dev@gmail.com
Password
Sign in
Need help?
Sign in with a different account
One Google Account for everything Google

Figure 3.2: Authentication step 2: Provide account credentials


Request for Permiss...
https://accounts.google.com/o/oauth2/auth?response_type=code&scope=https://www.googleapis.com/auth/drive&access_type=offline&redirect_uri=http://localhost:8
Search
Google
andrsebr.dev@gmail.com
Cancel
Accept

kumodd would like to:

View and manage the files in your Google Drive

By clicking Accept, you allow this app and Google to use your information in accordance with their respective terms of service and privacy policies. You can change this and other Account Permissions at any time.

Figure 3.3: Authentication step 3: Authorize access to the account by `kumodd`

```
andres@ubuntu:~/kumodd$ python kumodd.py -s gdrive -d all
Working...
TIME(UTC) APPLICATION  USER FILE-ID REMOTE PATH REVISION LOCAL PATH HASH(MD5)
2015-06-25 03:48:43.600028 kumodd-1.0 andrsebr.dev@gmail.com 1L-7
    o0rgPT2f6oX600PtF4ZUF0m0JW1Crktr3DPril8o My Drive/ppt test   v.2 downloaded/andrsebr.
    dev@gmail.com/My Drive/ppt test -
2015-06-25 03:48:44.951131 kumodd-1.0 andrsebr.dev@gmail.com 1
    huaRT0udVnLe4SPMXhMRnNQ9Y_DUr69m4TEeD5dIWuA My Drive/revision doc test   v.3 downloaded/
    andrsebr.dev@gmail.com/My Drive/revision doc test -

...

2015-06-25 03:48:54.254104 kumodd-1.0 andrsebr.dev@gmail.com 0B4wSliHoVUbhUHdhZlF4NlR5c3M My
     Drive/test folder/stuff/more stuff/tree.py   v.1 downloaded/andrsebr.dev@gmail.com/My
    Drive/test folder/stuff/more stuff/tree.py 61366435095ca0ca55e7192df66a0fe8

9 files downloaded and 0 updated from andrsebr.dev@gmail.com drive
Duration: 0:00:13.671442
```

Listing 3.6: Authentication step 4: Input access code in the terminal (if necessary) and allow the application to retrieve results

### 3.2.3 Content discovery and acquisition

Conceptually, the acquisition process goes through three phases–discovery, filtering, and acquisition (Figure 3.4). During content discovery, the tool queries the target service and obtains a list of files of interest to the analyst. Commonly, this would be a complete listing of all available file metadata (name, size, timestamps, revisions, etc.); however, `kumodd` offers basic filtering options based on data type. Some services offer a search-based selection, which could (in the future) be deployed to facilitate e-discovery, especially at scale.

The result of the discovery is a list of available data in the form of a CSV file, one file per line, which is suitable for further filtering using standard command-line tools. Alternatively, it could be imported into a separate program (text editor/spreadsheet/etc.) for manual or custom processing. In all cases, the end result is a file of the same format that specifies an *acquisition plan* that is passed on to `kumodd` for execution.

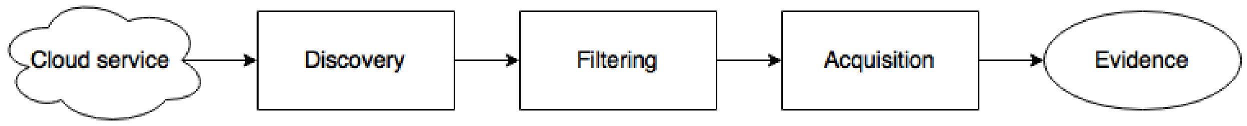


Figure 3.4: Steps of evidence acquisition

### Discovery

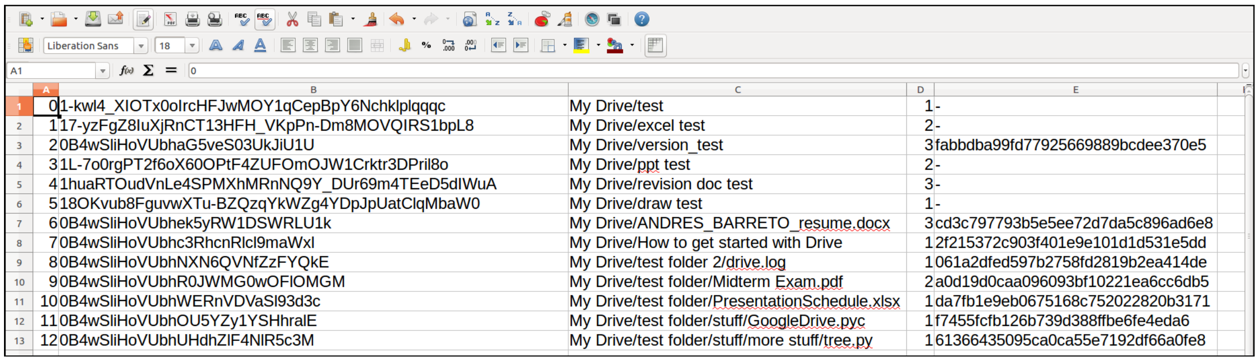
In the current implementation, the discovery is implemented by the *list* (`-l`) command, followed by a [**filter**] following the format shown earlier.

**Output:** A list of files that meet the provided criteria will be displayed with general information such as the file ID, remote path, number of revisions and (cryptographic) hashes.

```
andres@ubuntu:~/kumodd$ python kumodd.py -s gdrive -l all
Working...
FILE-ID REMOTE PATH REVISION HASH(MD5)
...1qCepBpY6Nchklplqqqc My Drive/test 1 -
...oVUbhaG5veS03UkJiU1U My Drive/version_test 3 ...bcdee370e5

...

...oVUbhUHdhZlF4NlR5c3M My Drive/test folder/stuff/more stuff/tree.py ...2df66a0fe8
```

Listing 3.7: Output of listing all files in a Google Drive account (partial results are being shown)

A CSV file with a list of the displayed files is saved under `/localdata` folder with the name of the service and the user. The generated file can be edited or used to download files listed in it. Figure 3.5 shows the file opened in a spreadsheet application.



| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | 0 | 1-kwl4_XIOTx0oIrcHFJwMOY1qCepBpY6Nchklplqqqc | My Drive/test | 1 | - |
| 2 | 1 | 17-yzFgZ8IuXjRnCT13HFH_VKpPn-Dm8MOVQIRS1bpL8 | My Drive/excel test | 2 | - |
| 3 | 2 | 0B4wSliHoVUbhaG5veS03UkJiU1U | My Drive/version_test | 3 | fabbdba99fd77925669889bcdee370e5 |
| 4 | 3 | 1L-7o0rgPT2f6oX60OPtF4ZUFOmOJW1Crktr3DPril8o | My Drive/ppt test | 2 | - |
| 5 | 4 | 1huaRTOudVnLe4SPMXhMRnNQ9Y_DUr69m4TEeD5dIWuA | My Drive/revision doc test | 3 | - |
| 6 | 5 | 18OKvub8FguvwXTu-BZQzqYkWZg4YDpJpUatClqMbaW0 | My Drive/draw test | 1 | - |
| 7 | 6 | 0B4wSliHoVUbhek5yRW1DSWRLU1k | My Drive/ANDRES_BARRETO_resume.docx | 3 | cd3c797793b5e5ee72d7da5c896ad6e8 |
| 8 | 7 | 0B4wSliHoVUbhc3RhcnRlcl9maWxl | My Drive/How to get started with Drive | 1 | 2f215372c903f401e9e101d1d531e5dd |
| 9 | 8 | 0B4wSliHoVUbhNXN6QVNfZzFYQkE | My Drive/test folder 2/drive.log | 1 | 061a2dfed597b2758fd2819b2ea414de |
| 10 | 9 | 0B4wSliHoVUbhR0JWMG0wOFlOMGM | My Drive/test folder/Midterm Exam.pdf | 2 | a0d19d0caa096093bf10221ea6cc6db5 |
| 11 | 10 | 0B4wSliHoVUbhWERnVDVaSl93d3c | My Drive/test folder/PresentationSchedule.xlsx | 1 | da7fb1e9eb0675168c752022820b3171 |
| 12 | 11 | 0B4wSliHoVUbhOU5YZy1YSHhralE | My Drive/test folder/stuff/GoogleDrive.pyc | 1 | f7455fcfb126b739d388ffbe6fe4eda6 |
| 13 | 12 | 0B4wSliHoVUbhUHdhZlF4NlR5c3M | My Drive/test folder/stuff/more stuff/tree.py | 1 | 61366435095ca0ca55e7192df66a0fe8 |

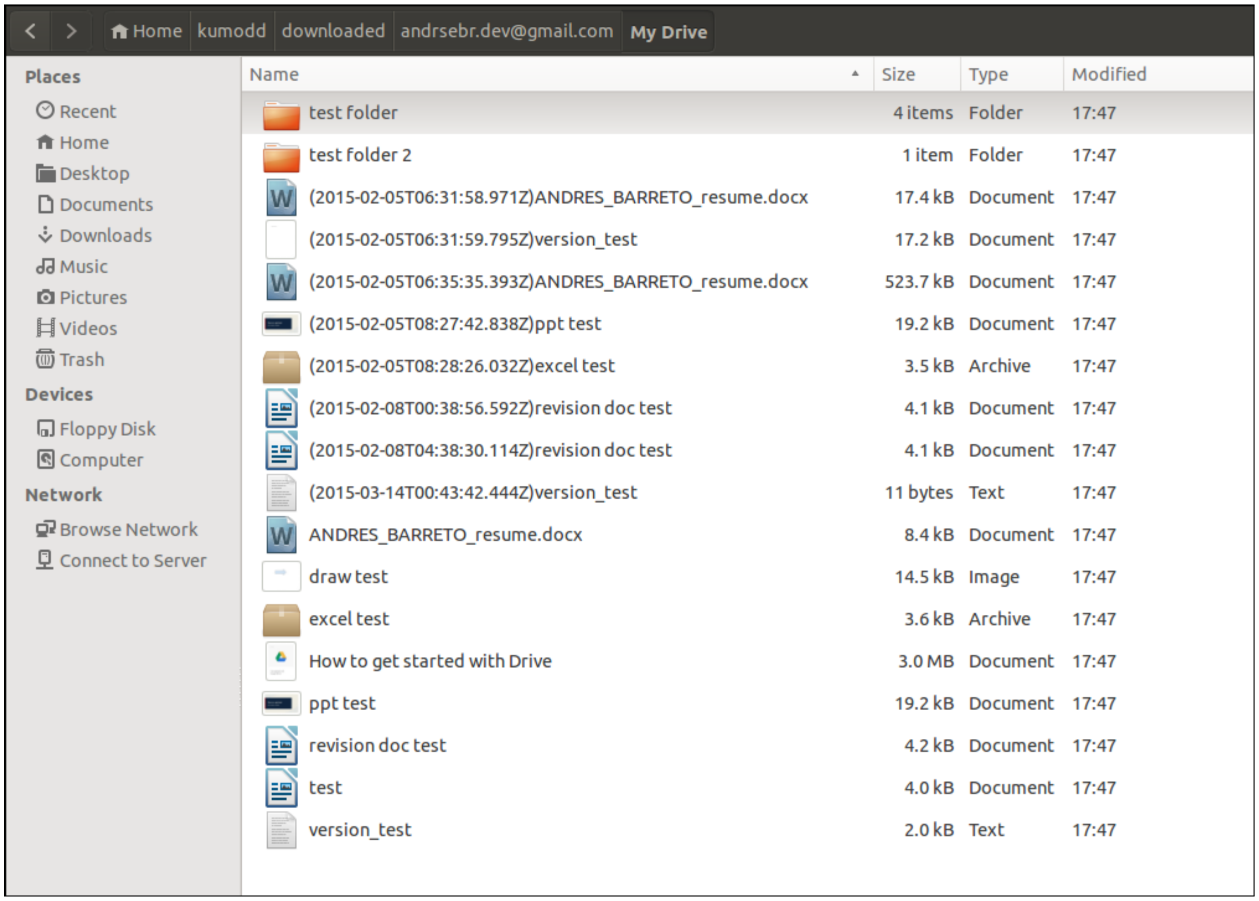
Figure 3.5: Contents of the generated CSV file

### Acquisition

The acquisition is performed by the *download* (`-d`), followed by a [**filter**] specification. In this case, the download is performed as a single discovery-and-acquisition step. The intended use is either one-command full acquisition, or interactive exploration, followed by one, or more, download commands.

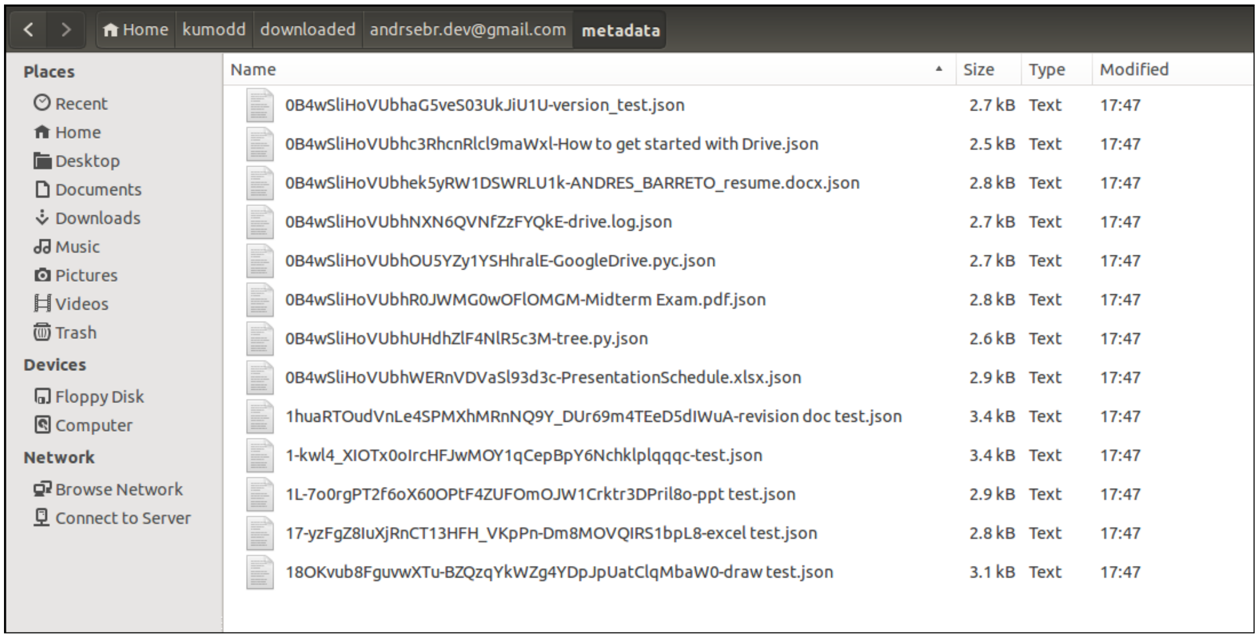
**Output:** A list of successfully downloaded files be displayed with information such as download date, application version, username, file ID, remote path, download path, revisions, and cryptographic hashes. This information is also logged in the log file `/downloaded/<username>/<service-name>.log` Downloaded files will be located in the `/downloaded/<username>/` directory. Figure 3.6. Metadata files with detailed information of downloaded files is stored in `/downloaded/< username>/metadata/` in JSON format.

Another acquisition option is the `-csv` command, which takes as an input a CSV file, which contains the acquisition plan and must be in the correct format. For this the directory in which the CSV file is located must be specified.



| Name | Size | Type | Modified |
|---|---|---|---|
| test folder | 4 items | Folder | 17:47 |
| test folder 2 | 1 item | Folder | 17:47 |
| (2015-02-05T06:31:58.971Z)ANDRES_BARRETO_resume.docx | 17.4 kB | Document | 17:47 |
| (2015-02-05T06:31:59.795Z)version_test | 17.2 kB | Document | 17:47 |
| (2015-02-05T06:35:35.393Z)ANDRES_BARRETO_resume.docx | 523.7 kB | Document | 17:47 |
| (2015-02-05T08:27:42.838Z)ppt test | 19.2 kB | Document | 17:47 |
| (2015-02-05T08:28:26.032Z)excel test | 3.5 kB | Archive | 17:47 |
| (2015-02-08T00:38:56.592Z)revision doc test | 4.1 kB | Document | 17:47 |
| (2015-02-08T04:38:30.114Z)revision doc test | 4.1 kB | Document | 17:47 |
| (2015-03-14T00:43:42.444Z)version_test | 11 bytes | Text | 17:47 |
| ANDRES_BARRETO_resume.docx | 8.4 kB | Document | 17:47 |
| draw test | 14.5 kB | Image | 17:47 |
| excel test | 3.6 kB | Archive | 17:47 |
| How to get started with Drive | 3.0 MB | Document | 17:47 |
| ppt test | 19.2 kB | Document | 17:47 |
| revision doc test | 4.2 kB | Document | 17:47 |
| test | 4.0 kB | Document | 17:47 |
| version_test | 2.0 kB | Text | 17:47 |

Figure 3.6: Drive contents downloaded to a local directory



| Name | Size | Type | Modified |
|---|---|---|---|
| 0B4wSliHoVUbhaG5veS03UkJiU1U-version_test.json | 2.7 kB | Text | 17:47 |
| 0B4wSliHoVUbhc3RhcnRlcl9maWxl-How to get started with Drive.json | 2.5 kB | Text | 17:47 |
| 0B4wSliHoVUbhek5yRW1DSWRLU1k-ANDRES_BARRETO_resume.docx.json | 2.8 kB | Text | 17:47 |
| 0B4wSliHoVUbhNXN6QVNfZzFYQkE-drive.log.json | 2.7 kB | Text | 17:47 |
| 0B4wSliHoVUbhOU5YZy1YSHhralE-GoogleDrive.pyc.json | 2.7 kB | Text | 17:47 |
| 0B4wSliHoVUbhR0JWMG0wOFlOMGM-Midterm Exam.pdf.json | 2.8 kB | Text | 17:47 |
| 0B4wSliHoVUbhUHdhZlF4NlR5c3M-tree.py.json | 2.6 kB | Text | 17:47 |
| 0B4wSliHoVUbhWERnVDVaSl93d3c-PresentationSchedule.xlsx.json | 2.9 kB | Text | 17:47 |
| 1huaRTOudVnLe4SPMXhMRnNQ9Y_DUr69m4TEeD5dIWuA-revision doc test.json | 3.4 kB | Text | 17:47 |
| 1-kwl4_XIOTx0oIrcHFJwMOY1qCepBpY6Nchklplqqqc-test.json | 3.4 kB | Text | 17:47 |
| 1L-7o0rgPT2f6oX60OPtF4ZUFOmOJW1Crktr3DPril8o-ppt test.json | 2.9 kB | Text | 17:47 |
| 17-yzFgZ8IuXjRnCT13HFH_VKpPn-Dm8MOVQIRS1bpL8-excel test.json | 2.8 kB | Text | 17:47 |
| 18OKvub8FguvwXTu-BZQzqYkWZg4YDpJpUatClqMbaW0-draw test.json | 3.1 kB | Text | 17:47 |

Figure 3.7: Metadata of every downloaded file saved in a local directory

### 3.2.4 Web GUI

Another alternative to use `kumodd` is using the provided Web GUI, which is run from a local web server invoked with the `kumodd-gui.py` module:

```
python kumodd-gui.py
```

The application GUI will be available at `http://localhost:5000` and is accessible via any web browser. (Note: At this point, the server is not secure, so it should not be run on a remote system, unless security is assured by other mechanisms.)

```
andres@ubuntu:~/kumodd$ python kumodd-gui.py
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
```

Listing 3.8: Starting the web GUI

The web GUI uses the same modules used with the command line application for authenticate, discovery and acquisition.

#### Authentication

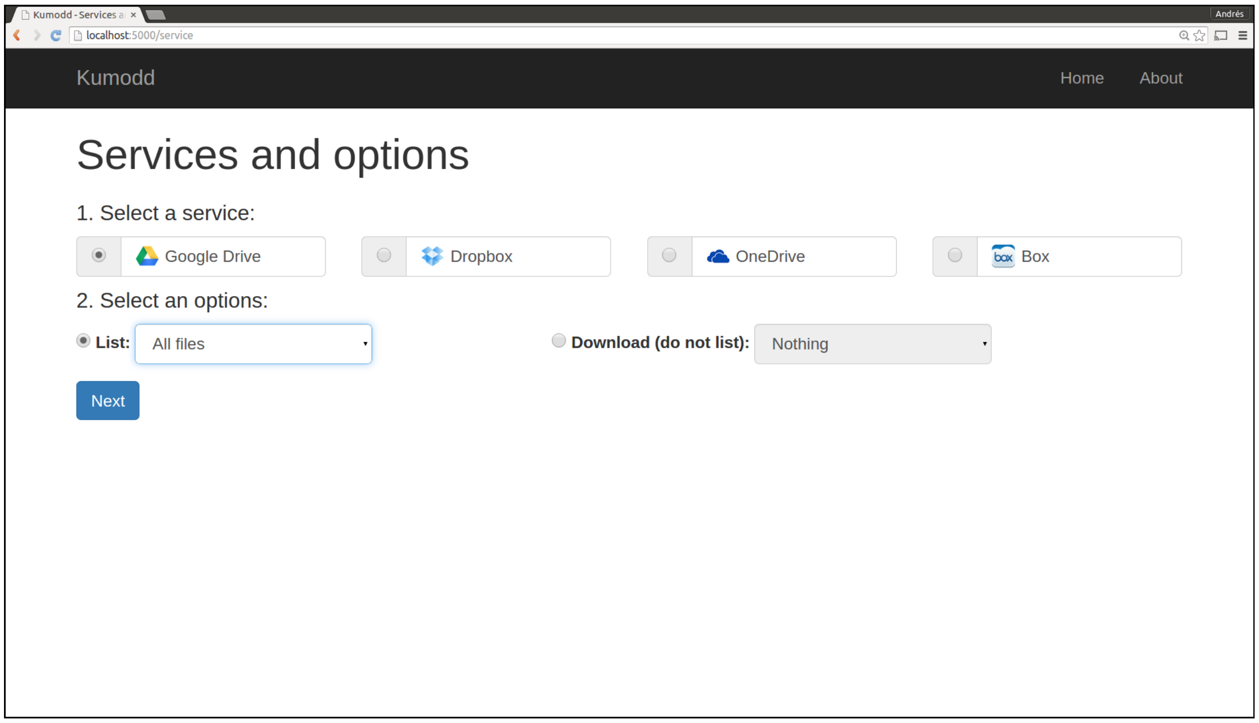
As `kumodd` was originally planed to be a command line application; there is a step that involves inputting data in the command line while the web server is running. This step is the authentication, once the access code is retrieved from a service it has to be put in the command line while the application waits to retrieve this value (this step is not necessary while using Google Drive).

Just like using the command line interface; after the authentication is granted all the authorization information is saved locally so the user will not have to go through the authentication step again unless a new user account needs to be accessed or the access token provided by the service has expired and need to be retrieved again.

#### Discovery and acquisition

The web interface simplifies the user interactions and, for the simple case of wholesale data acquisition, the process can be accomplished in three button clicks.

After pressing the *Get started!* button, the user is presented with the choice of the target service and the action to perform (Figure 3.8). After this step a detailed window with a list of files and the option of choosing which to download is presented (Figure 3.10). Once the files to acquire are selected, a result screen is presented with paths to files and other relevant information. (Note: Every step of the process can be seen in the terminal as well.)



Services and options

1. Select a service:

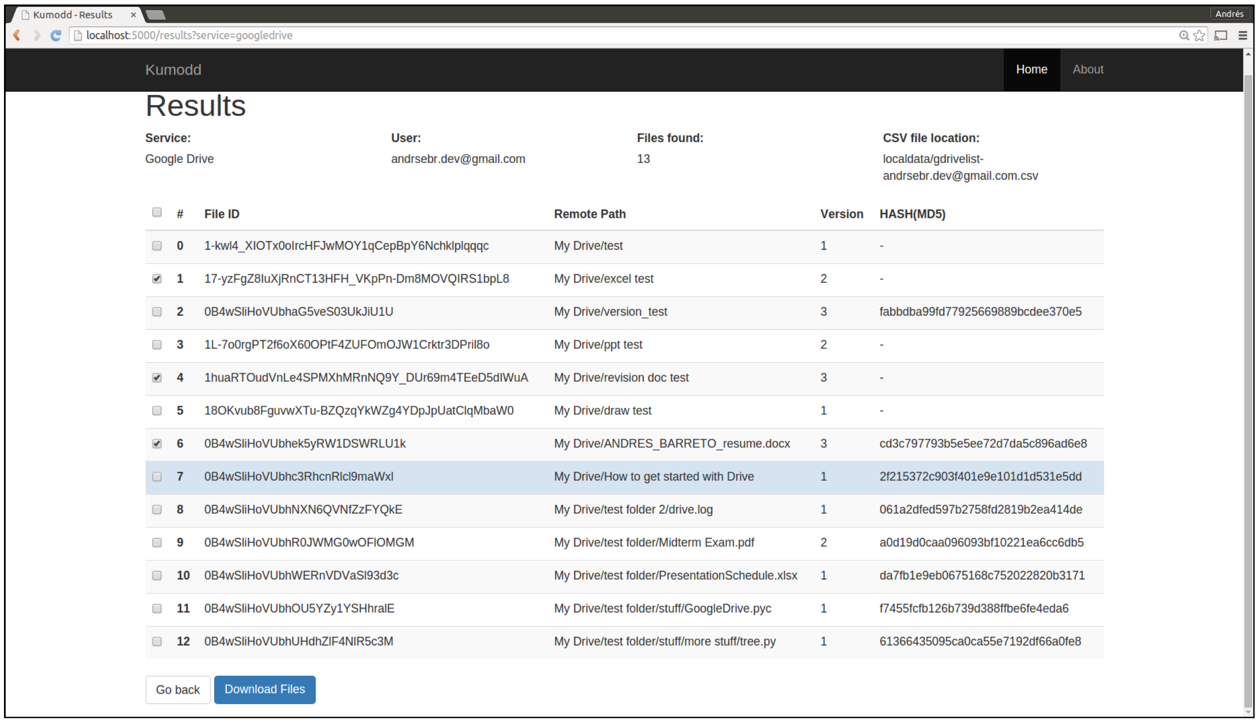
[✓ Google Drive] Dropbox OneDrive Box

2. Select an options:

[✓ List:] All files

Download (do not list): Nothing

Figure 3.8: Web GUI: Service selection

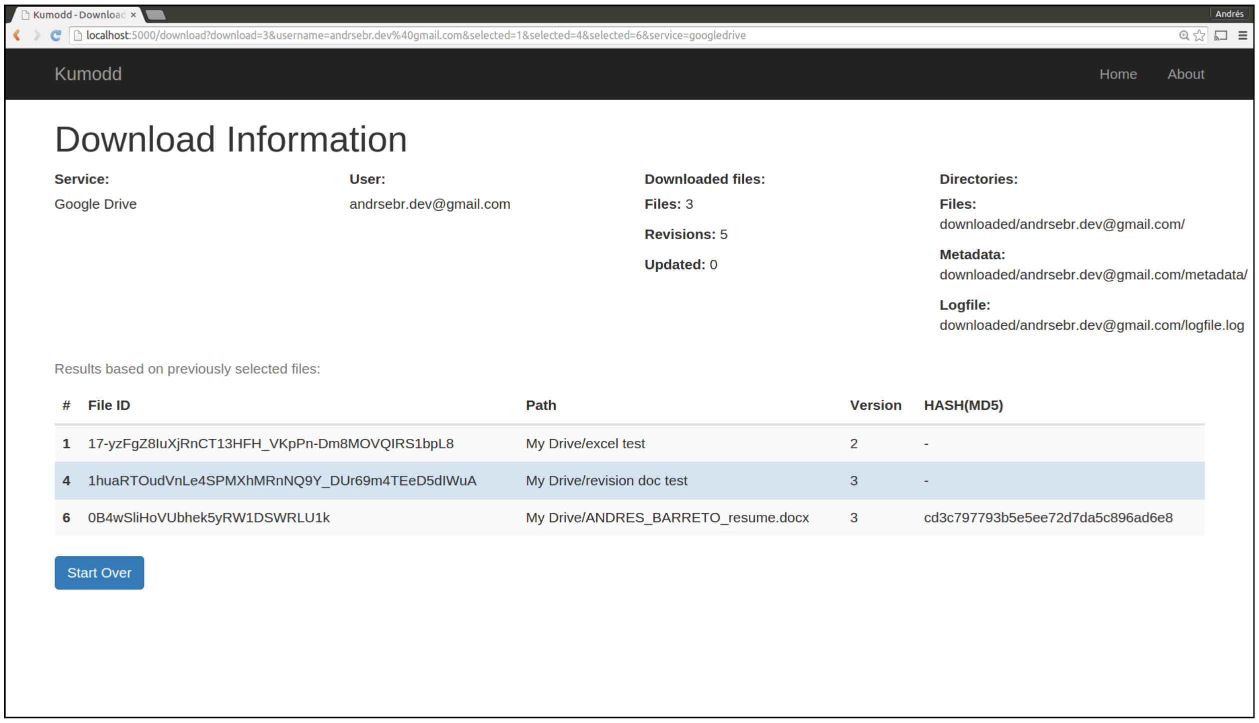


Results

| Service: | User: | Files found: | CSV file location: |
|---|---|---|---|
| Google Drive | andrsebr.dev@gmail.com | 13 | localdata/gdrivelist-andrsebr.dev@gmail.com.csv |

| | # | File ID | Remote Path | Version | HASH(MD5) |
|---|---|---|---|---|---|
| | 0 | 1-kwl4_XIOTx0olrcHFJwMOY1qCepBpY6Nchklplqqqc | My Drive/test | 1 | - |
| ✓ | 1 | 17-yzFgZ8luXjRnCT13HFH_VKpPn-Dm8MOVQIRS1bpL8 | My Drive/excel test | 2 | - |
| | 2 | 0B4wSliHoVUbhaG5veS03UkJiU1U | My Drive/version_test | 3 | fabbdba99fd77925669889bcdee370e5 |
| | 3 | 1L-7o0rgPT2f6oX60OPtF4ZUFOmOJW1Crktr3DPril8o | My Drive/ppt test | 2 | - |
| ✓ | 4 | 1huaRTOudVnLe4SPMXhMRnNQ9Y_DUr69m4TEeD5dIWuA | My Drive/revision doc test | 3 | - |
| | 5 | 18OKvub8FguvwXTu-BZQzqYkWZg4YDpJpUatClqMbaW0 | My Drive/draw test | 1 | - |
| ✓ | 6 | 0B4wSliHoVUbhek5yRW1DSWRLU1k | My Drive/ANDRES_BARRETO_resume.docx | 3 | cd3c797793b5e5ee72d7da5c896ad6e8 |
| | 7 | 0B4wSliHoVUbhc3RhcnRlcl9maWxl | My Drive/How to get started with Drive | 1 | 2f215372c903f401e9e101d1d531e5dd |
| | 8 | 0B4wSliHoVUbhNXN6QVNfZzFYQkE | My Drive/test folder 2/drive.log | 1 | 061a2dfed597b2758fd2819b2ea414de |
| | 9 | 0B4wSliHoVUbhR0JWMG0wOFlOMGM | My Drive/test folder/Midterm Exam.pdf | 2 | a0d19d0caa096093bf10221ea6cc6db5 |
| | 10 | 0B4wSliHoVUbhWERnVDVaSl93d3c | My Drive/test folder/PresentationSchedule.xlsx | 1 | da7fb1e9eb0675168c752022820b3171 |
| | 11 | 0B4wSliHoVUbhOU5YZy1YSHhralE | My Drive/test folder/stuff/GoogleDrive.pyc | 1 | f7455fcfb126b739d388ffbe6fe4eda6 |
| | 12 | 0B4wSliHoVUbhUHdhZlF4NlR5c3M | My Drive/test folder/stuff/more stuff/tree.py | 1 | 61366435095ca0ca55e7192df66a0fe8 |

Figure 3.9: Web GUI: File selection


Kumodd - Download
localhost:5000/download?download=3&username=andrsebr.dev%40gmail.com&selected=1&selected=4&selected=6&service=googledrive
Kumodd
Home
About

Download Information

**Service:** Google Drive

**User:** andrsebr.dev@gmail.com

**Downloaded files:**
**Files:** 3
**Revisions:** 5
**Updated:** 0

**Directories:**
**Files:** downloaded/andrsebr.dev@gmail.com/
**Metadata:** downloaded/andrsebr.dev@gmail.com/metadata/
**Logfile:** downloaded/andrsebr.dev@gmail.com/logfile.log

Results based on previously selected files:

| # | File ID | Path | Version | HASH(MD5) |
|---|---|---|---|---|
| 1 | 17-yzFgZ8luXjRnCT13HFH_VKpPn-Dm8MOVQIRS1bpL8 | My Drive/excel test | 2 | - |
| 4 | 1huaRTOudVnLe4SPMXhMRnNQ9Y_DUr69m4TEeD5dIWuA | My Drive/revision doc test | 3 | - |
| 6 | 0B4wSliHoVUbhek5yRW1DSWRLU1k | My Drive/ANDRES_BARRETO_resume.docx | 3 | cd3c797793b5e5ee72d7da5c896ad6e8 |

Figure 3.10: Web GUI: Results

## 3.3 Implementation

### 3.3.1 Development environment

Kumodd was developed and tested using the following environment:

- **Operating System:** Ubuntu Linux 14.04 LTS (64-bit)
- Python 2.7.6
- **Packages:**
  - python-gflags 2.0
  - google-api-python-client 1.3.1
  - dropbox 2.2.0
  - boxsdk 1.1.3
  - Flask 0.10.1
  - Flask-Classy 0.6.10
- Google Chrome 43.0.2357.81 (64-bit)

### 3.3.2 Python

Python is a multi-platform, interpreted, object-oriented, high-level programming language with dynamic semantics. It supports modules and packages for program modularity and code reuse. [17]

### 3.3.3 Packages

#### python-gflags

This package is a an equivalent of the C++ command line flag implementation developed by Google (*gflags*). This package contains a library that implements command line flags processing. It includes built-ins support for Python types and allows to define flags in the source file which they are used. [18] [19]

#### google-api-python-client

This package contains the Python client library provided by Google for discovery-based APIs. [20]

#### dropbox

This package contains Python specific libraries that wrap the raw HTTP calls to the Dropbox API. [21]

#### boxsdk

This package contains Python specific libraries that wrap the raw HTTP calls to the Box API. [22]

#### Flask

Flask is a web application framework written in Python. It is based on the Jinja2 template engine and Wekzeug toolkit. This framework is used to give back-end applications a web based interface. [23]

#### Flask-Classy

This is an extension of Flask that adds class based views functionality. [24]

## 3.4 Installation

### 3.4.1 Requirements

1. Unix-based operating system
2. Python 2.7.x or above
3. **Required packages:**
   - python-gflags
   - google-api-python-client
   - dropbox
   - boxsdk
   - Flask
   - Flask-Classy
4. Google Chrome or Firefox web browser (for web interface)
5. User credentials for account access

### 3.4.2 Package installation

#### Kumodd

Kumodd can be downloaded from its Gitlab repository using the following command:

```
git clone https://github.com/andresebr/kumodd.git
```

Listing 3.9: Kumodd download

#### Python

Python comes preloaded in almost every Linux distribution and OSX by default. It is possible to know if the system has Python installed and its version by running the command. (For package installations, either the `easy_install`, or the `pip` package managers can be used.)

```
python --version
```

Listing 3.10: Obtaining the Python version

If Python is not installed in the system:

```
sudo apt-get install build-essential
sudo apt-get install libreadline-gplv2-dev libncursesw5-dev libssl-dev libsqlite3-dev tk-dev
     libgdbm-dev libc6-dev libbz2-dev
cd ~/Downloads/
wget http://python.org/ftp/python/2.7.6/Python-2.7.6.tgz
tar -xvf Python-2.7.6.tgz
cd Python-2.7.6
./configure
make
sudo make install
```

Listing 3.11: Installation of Python 2.7.6 and its dependencies on Debian/Ubuntu

### python-gflags

```
sudo easy_install python-gflags
```

Listing 3.12: python-gflags installation command

### google-api-python-client

```
sudo easy_install --upgrade google-api-python-client
```

Listing 3.13: google-api-python-client installation command

### dropbox

```
sudo easy_install dropbox
```

Listing 3.14: dropbox installation command

### boxsdk

```
sudo easy_install boxsdk
```

Listing 3.15: boxsdk installation command

### Flask and Flask-Classy

```
sudo easy_install flask flask-classy
```

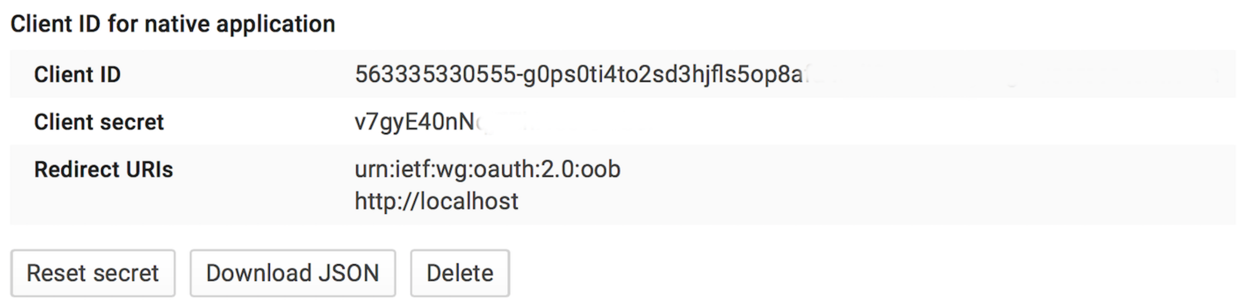
Listing 3.16: Flask and Flask-Classy installation command

## 3.5 Configuration

Once `kumodd` has been downloaded from the repository, it is necessary to configure the services that are be going to be used. The first step is registering the application in its respective service consoles. See sections 4.1.1, 4.2.1, 4.3.1, and 4.4.1. For this step a developer account must be created in every service intended to be used.

Once the application is registered, two unique keys will be provided; an application identifier and a secret key. The keys needed will be available in the credentials section. In the developer consoles it is also possible to configure application redirections and permission settings (Figure 3.11).

The keys provided by the services must be used in the `kumodd` configuration file `/configuration/config.cfg` before first use. For Google Drive the credentials must be specified in `/configuration/gdrive_config.json`.



**Client ID for native application**

| | |
|---|---|
| **Client ID** | 563335330555-g0ps0ti4to2sd3hjfls5op8a |
| **Client secret** | v7gyE40nN |
| **Redirect URIs** | urn:ietf:wg:oauth:2.0:oob<br>http://localhost |

Figure 3.11: Service credentials in the developer console (Google Drive)

# Chapter 4

# Cloud Storage Providers APIs

This chapter provides additional technical details in the form of descriptions of the service APIs that `kumodd` utilized. Namely, each of the four providers in our study–Google, Dropbox, Microsoft, and Box–provide a REST API and a Python SDK that simplifies its use druting development. As our discussion will show, the various services carry both substantial similarities and non-trivial differences.

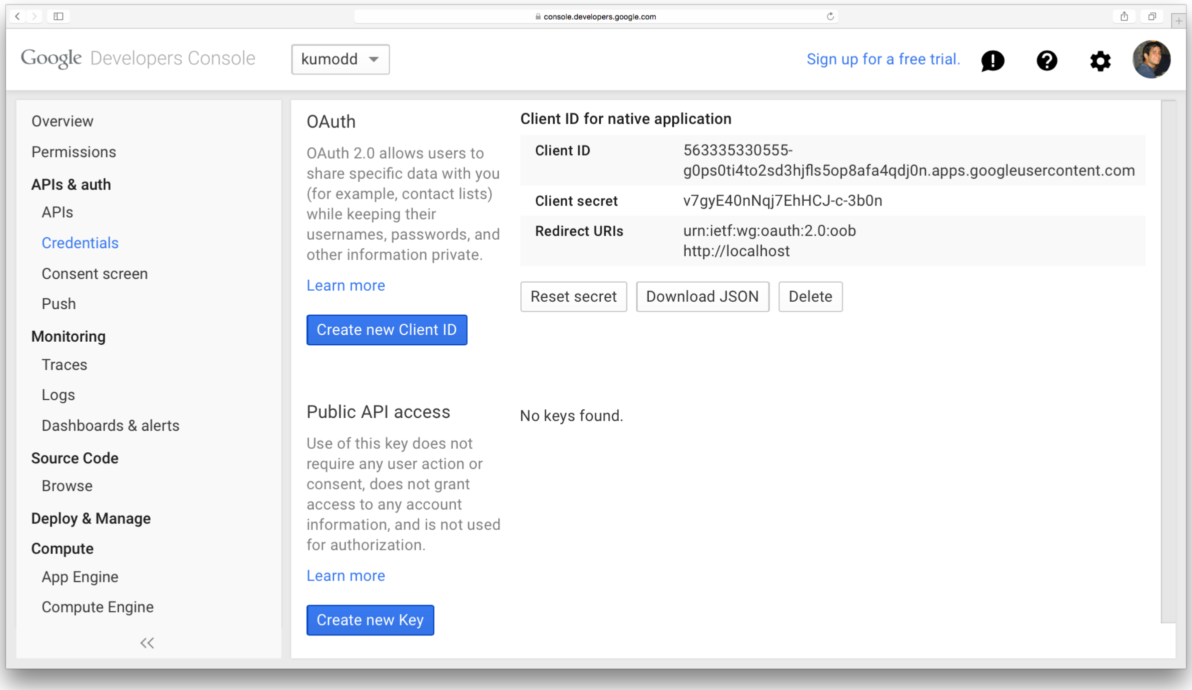
## 4.1 Google Drive

The main functionality of the Drive API provided by Google is to upload and download in Google Drive. It also supports other features, such as file searching, file sharing and collaboration, and conversion and export of Google Docs [25], [26].

### 4.1.1 API authorization

All requests to the Google Drive API need to be authorized by an authenticated user; this process is performed using the OAuth 2.0 protocol [27]. Applications using Google Drive (or any of the Google services) must follow the same basic authorization pattern:

1. Register the application in the Google Developers Console (Figure 4.1).
2. Request that the user grant access to data in their account.
3. If the user allows grants access, the application request and receives access token credentials to access the Drive API.
4. When the token expires, the authentication process must be repeated.



OAuth

OAuth 2.0 allows users to share specific data with you (for example, contact lists) while keeping their usernames, passwords, and other information private.

Client ID for native application

| | |
|---|---|
| Client ID | 563335330555-g0ps0ti4to2sd3hjfls5op8afa4qdj0n.apps.googleusercontent.com |
| Client secret | v7gyE40nNqj7EhHCJ-c-3b0n |
| Redirect URIs | urn:ietf:wg:oauth:2.0:oob<br>http://localhost |

Public API access

Use of this key does not require any user action or consent, does not grant access to any account information, and is not used for authorization.

No keys found.

Figure 4.1: Google Developers Console

*Authorization scopes* are used to express the permissions requested to users when they use authenticate using an application that request access to their account. In Listing 4.1 all the authorization scopes supported by Google Drive are explained.

```
https://www.googleapis.com/auth/drive.file  "Per-file access to files created or opened by
    the app"

https://www.googleapis.com/auth/drive "Full, permissive scope to access all of a user's
    files. Request this scope only when it is strictly necessary. Tokens with scope https://
    docs.google.com/feeds are accepted and treated the same as tokens with scope https://www
    .googleapis.com/auth/drive."

https://www.googleapis.com/auth/drive.apps.readonly "Allows apps read-only access to the
    list of Drive apps a user has installed."

https://www.googleapis.com/auth/drive.readonly  "Allows read-only access to file metadata
    and file content"

https://www.googleapis.com/auth/drive.metadata.readonly "Allows read-only access to file
    metadata, but does not allow any access to read or download file content"

https://www.googleapis.com/auth/drive.metadata  "Allows read-write access to file metadata,
    but does not allow any access to read, download, write or upload file content. Does not
    support file creation, trashing or deletion. Also does not allow changing folders or
    sharing in order to prevent access escalation."

https://www.googleapis.com/auth/drive.install "Special scope used to let users approve
    installation of an app"

https://www.googleapis.com/auth/drive.appfolder "Allows access to the Application Data
    folder"

https://www.googleapis.com/auth/drive.scripts "Allows access to Apps Script files"
```

Listing 4.1: Google Drive Authorization scopes [28]

### 4.1.2 Files resource

The *files* resource has several methods that allow external access to the files stored in a Google Drive account. These are summarized in Table 4.1 with the two methods used for `kumodd` development shown in bold.

Table 4.1: Files resource methods; all URIs relative to https://www.googleapis.com/drive/v2. A complete list of methods is available in in the Drive REST API reference provided by Google [26]

| Method | HTTP request | Description |
|---|---|---|
| **list** | `GET /files` | Lists the user's files. |
| **get** | `GET /files/fileId` | Gets a file's metadata by ID. |

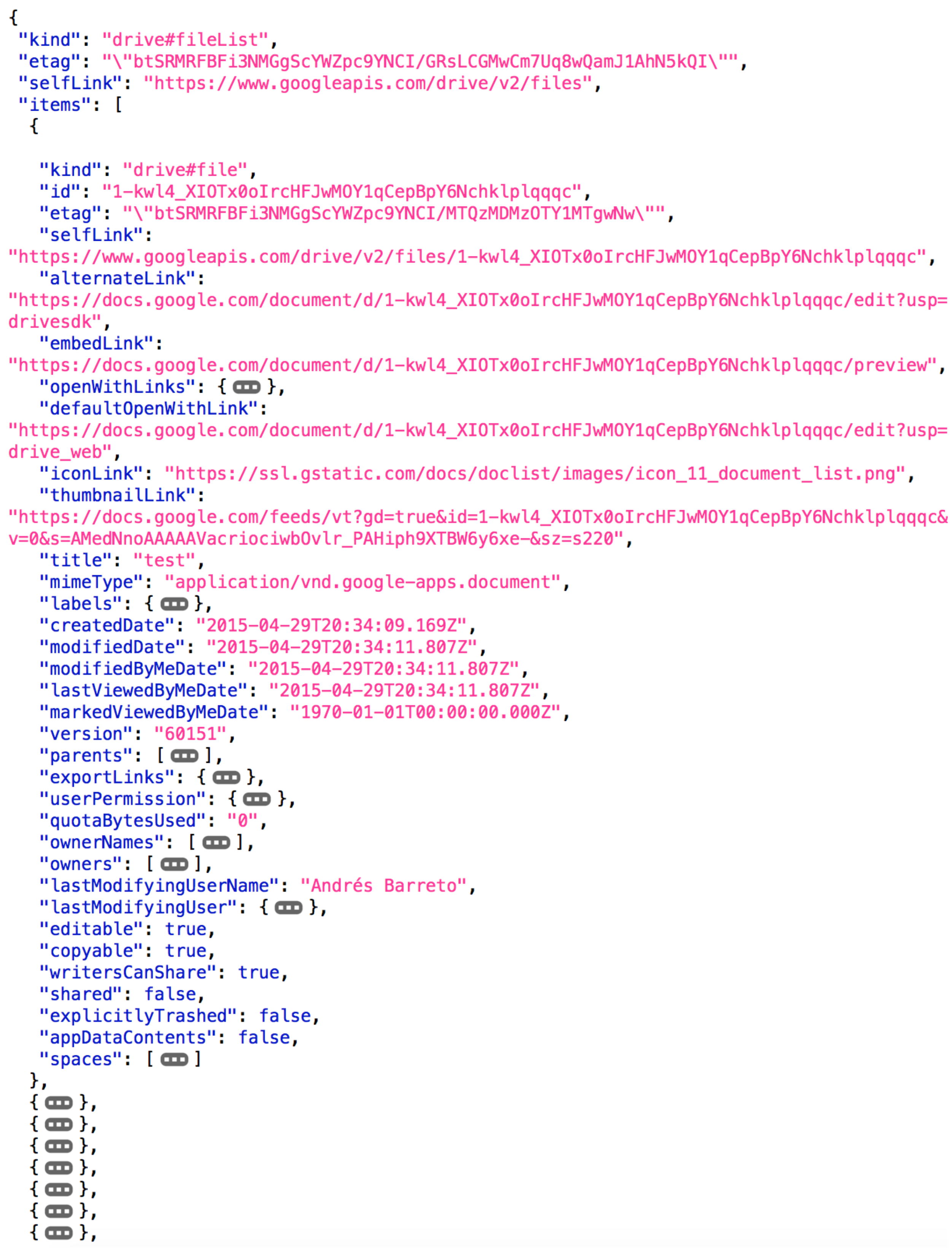
#### Files:list

This method lists the files stored in an user account; this includes *trashed* files as long as they have not been entirely deleted from the server.

- **HTTP Request:** `GET https://www.googleapis.com/drive/v2/files`
- **Parameters:** No parameter is required.
- **Authorization:** For this request an authorization with at least one of he following scopes is required:
  - `https://www.googleapis.com/auth/drive`
  - `https://www.googleapis.com/auth/drive.file`
  - `https://www.googleapis.com/auth/drive.readonly`
  - `https://www.googleapis.com/auth/drive.metadata.readonly`
  - `https://www.googleapis.com/auth/drive.appdata`
  - `https://www.googleapis.com/auth/drive.apps.readonly`
  - `https://www.googleapis.com/auth/drive.metadata`
- **Response:** If successful, this method returns a response body with the following structure:

```
{
  "kind": "drive#changeList",
  "etag": etag,
  "selfLink": string,
  "nextPageToken": string,
  "nextLink": string,
  "largestChangeId": long,
  "items": [
    changes Resource
  ]
}
```

Listing 4.2: Files:list response tructure



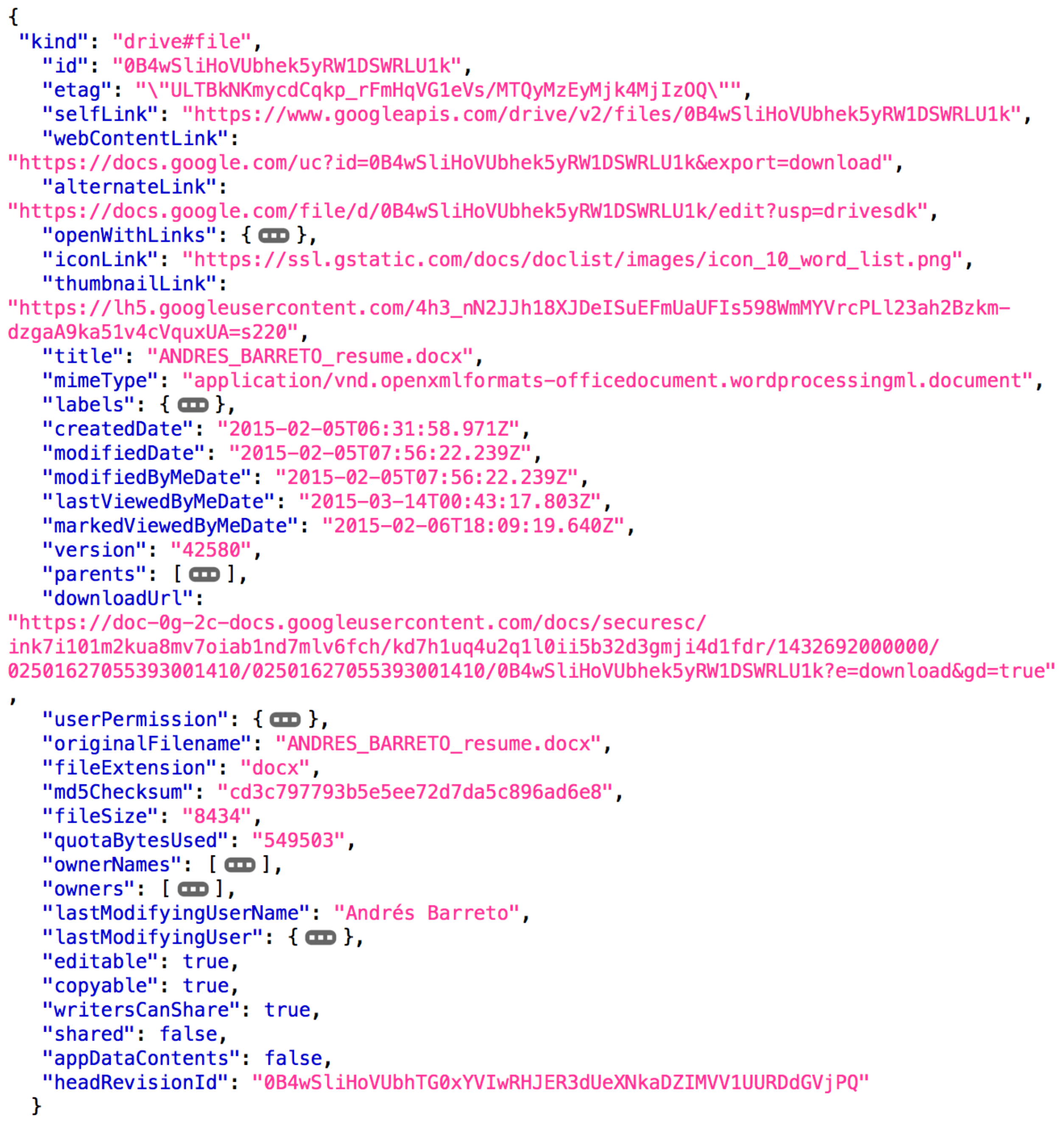
```
{
 "kind": "drive#fileList",
 "etag": "\"btSRMRFBFi3NMGgScYWZpc9YNCI/GRsLCGMwCm7Uq8wQamJ1AhN5kQI\"",
 "selfLink": "https://www.googleapis.com/drive/v2/files",
 "items": [
  {

   "kind": "drive#file",
   "id": "1-kwl4_XIOTx0oIrcHFJwMOY1qCepBpY6Nchklplqqqc",
   "etag": "\"btSRMRFBFi3NMGgScYWZpc9YNCI/MTQzMDMzOTY1MTgwNw\"",
   "selfLink":
"https://www.googleapis.com/drive/v2/files/1-kwl4_XIOTx0oIrcHFJwMOY1qCepBpY6Nchklplqqqc",
   "alternateLink":
"https://docs.google.com/document/d/1-kwl4_XIOTx0oIrcHFJwMOY1qCepBpY6Nchklplqqqc/edit?usp=
drivesdk",
   "embedLink":
"https://docs.google.com/document/d/1-kwl4_XIOTx0oIrcHFJwMOY1qCepBpY6Nchklplqqqc/preview",
   "openWithLinks": {⋯},
   "defaultOpenWithLink":
"https://docs.google.com/document/d/1-kwl4_XIOTx0oIrcHFJwMOY1qCepBpY6Nchklplqqqc/edit?usp=
drive_web",
   "iconLink": "https://ssl.gstatic.com/docs/doclist/images/icon_11_document_list.png",
   "thumbnailLink":
"https://docs.google.com/feeds/vt?gd=true&id=1-kwl4_XIOTx0oIrcHFJwMOY1qCepBpY6Nchklplqqqc&
v=0&s=AMedNnoAAAAAVacriociwbOvlr_PAHiph9XTBW6y6xe-&sz=s220",
   "title": "test",
   "mimeType": "application/vnd.google-apps.document",
   "labels": {⋯},
   "createdDate": "2015-04-29T20:34:09.169Z",
   "modifiedDate": "2015-04-29T20:34:11.807Z",
   "modifiedByMeDate": "2015-04-29T20:34:11.807Z",
   "lastViewedByMeDate": "2015-04-29T20:34:11.807Z",
   "markedViewedByMeDate": "1970-01-01T00:00:00.000Z",
   "version": "60151",
   "parents": [⋯],
   "exportLinks": {⋯},
   "userPermission": {⋯},
   "quotaBytesUsed": "0",
   "ownerNames": [⋯],
   "owners": [⋯],
   "lastModifyingUserName": "Andrés Barreto",
   "lastModifyingUser": {⋯},
   "editable": true,
   "copyable": true,
   "writersCanShare": true,
   "shared": false,
   "explicitlyTrashed": false,
   "appDataContents": false,
   "spaces": [⋯]
  },
  {⋯},
  {⋯},
  {⋯},
  {⋯},
  {⋯},
  {⋯},
  {⋯},
```

Figure 4.2: Files:list method response example (abbreviated)

**Files:get**

Obtains the metadata of a file by its (service-provided) ID.

- **HTTP Request:** GET `https://www.googleapis.com/drive/v2/files/`*`fileId`*
- **Parameters:** *`fileId`*
- **Response:** JSON with a File resource in the response body. Figure 4.3



```
{
 "kind": "drive#file",
   "id": "0B4wSliHoVUbhek5yRW1DSWRLU1k",
   "etag": "\"ULTBkNKmycdCqkp_rFmHqVG1eVs/MTQyMzEyMjk4MjIzOQ\"",
   "selfLink": "https://www.googleapis.com/drive/v2/files/0B4wSliHoVUbhek5yRW1DSWRLU1k",
   "webContentLink":
"https://docs.google.com/uc?id=0B4wSliHoVUbhek5yRW1DSWRLU1k&export=download",
   "alternateLink":
"https://docs.google.com/file/d/0B4wSliHoVUbhek5yRW1DSWRLU1k/edit?usp=drivesdk",
   "openWithLinks": {⋯},
   "iconLink": "https://ssl.gstatic.com/docs/doclist/images/icon_10_word_list.png",
   "thumbnailLink":
"https://lh5.googleusercontent.com/4h3_nN2JJh18XJDeISuEFmUaUFIs598WmMYVrcPLl23ah2Bzkm-
dzgaA9ka51v4cVquxUA=s220",
   "title": "ANDRES_BARRETO_resume.docx",
   "mimeType": "application/vnd.openxmlformats-officedocument.wordprocessingml.document",
   "labels": {⋯},
   "createdDate": "2015-02-05T06:31:58.971Z",
   "modifiedDate": "2015-02-05T07:56:22.239Z",
   "modifiedByMeDate": "2015-02-05T07:56:22.239Z",
   "lastViewedByMeDate": "2015-03-14T00:43:17.803Z",
   "markedViewedByMeDate": "2015-02-06T18:09:19.640Z",
   "version": "42580",
   "parents": [⋯],
   "downloadUrl":
"https://doc-0g-2c-docs.googleusercontent.com/docs/securesc/
ink7i101m2kua8mv7oiab1nd7mlv6fch/kd7h1uq4u2q1l0ii5b32d3gmji4d1fdr/1432692000000/
02501627055393001410/02501627055393001410/0B4wSliHoVUbhek5yRW1DSWRLU1k?e=download&gd=true"
,
   "userPermission": {⋯},
   "originalFilename": "ANDRES_BARRETO_resume.docx",
   "fileExtension": "docx",
   "md5Checksum": "cd3c797793b5e5ee72d7da5c896ad6e8",
   "fileSize": "8434",
   "quotaBytesUsed": "549503",
   "ownerNames": [⋯],
   "owners": [⋯],
   "lastModifyingUserName": "Andrés Barreto",
   "lastModifyingUser": {⋯},
   "editable": true,
   "copyable": true,
   "writersCanShare": true,
   "shared": false,
   "appDataContents": false,
   "headRevisionId": "0B4wSliHoVUbhTG0xYVIwRHJER3dUeXNkaDZIMVV1UURDdGVjPQ"
  }
```

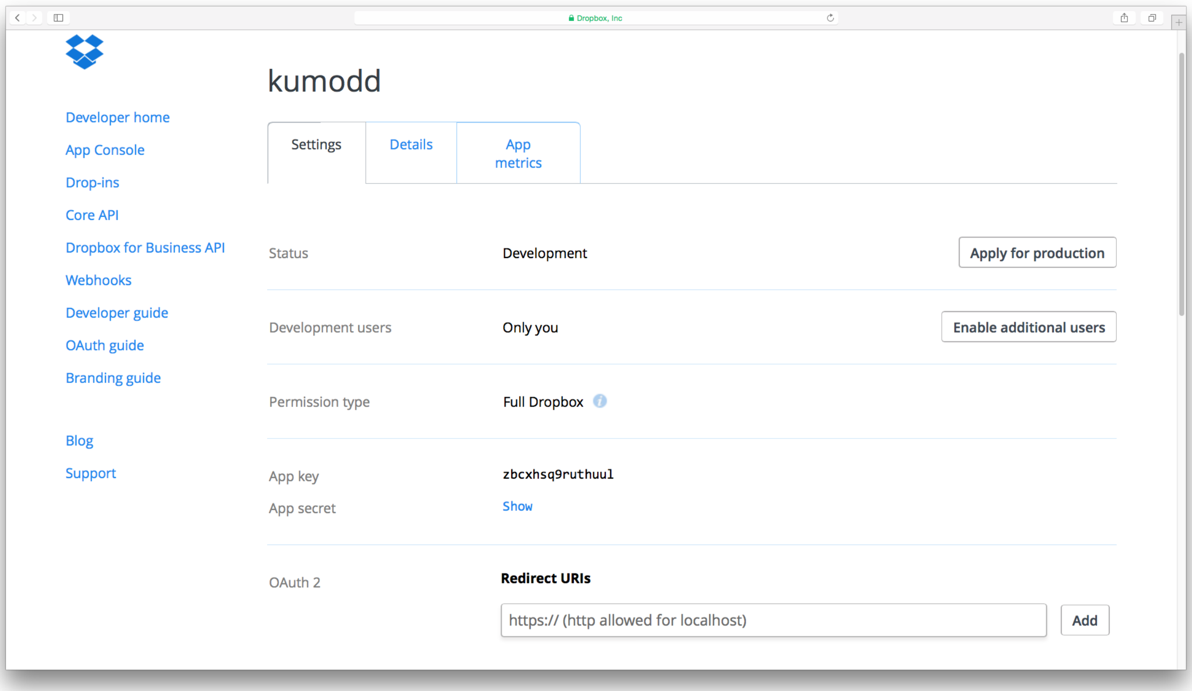
Figure 4.3: Files:get method response example (abbreviated)

Information about additional resources is available in Appendix A.

## 4.2 Dropbox

Like other cloud storage platforms, Dropbox allows developer access to user documents through its Core API which is based on HTTP and OAuth. There are SDK libraries that handle low-level calls to access and manipulate Dropbox accounts for a wide range of programming languages such as Java, Python, PHP, and popular mobile platforms [29].

To start using the Dropbox API it is necessary to register the application in the Dropbox App Console as shown in Figure 4.4. The App Console is also used to manage general application settings and permissions that are required to access user information.



| | | |
|---|---|---|
| Status | Development | |
| Development users | Only you | |
| Permission type | Full Dropbox | |
| App key | zbcxhsq9ruthuul | |
| App secret | Show | |
| OAuth 2 | Redirect URIs | |
| | https:// (http allowed for localhost) | |

Figure 4.4: Dropbox App Console

### 4.2.1 API authorization

For an application to connect to a Dropbox account it is necessary that a user authenticate through Dropbox for identity verification and to give the application permission to access their data. Dropbox API uses OAuth v2 for the authentication process [1]. At the end of the authorization process the application receives an access token–a string generated by Dropbox which uniquely identify both the app and the end user. The access token is provided with all requests made to Dropbox through the application (Figure 4.5).

The use of OAuth means that the application does not store or transmit user passwords, and that user can control the permissions at all times.

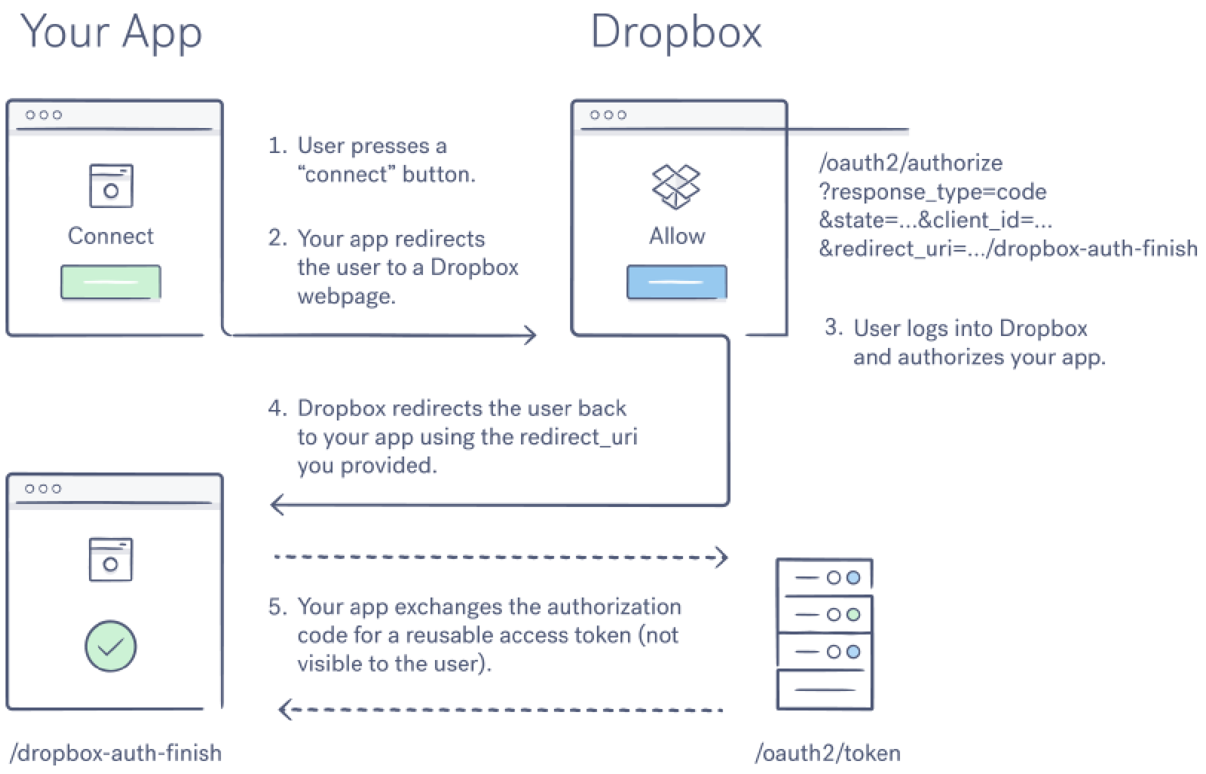

Your App
Dropbox
Connect
1. User presses a "connect" button.
2. Your app redirects the user to a Dropbox webpage.
Allow
/oauth2/authorize ?response_type=code &state=...&client_id=... &redirect_uri=.../dropbox-auth-finish
3. User logs into Dropbox and authorizes your app.
4. Dropbox redirects the user back to your app using the redirect_uri you provided.
5. Your app exchanges the authorization code for a reusable access token (not visible to the user).
/dropbox-auth-finish
/oauth2/token

Figure 4.5: OAuth 2 flow diagram. [1]

### 4.2.2 Account information

The following sections explains the method used to obtain general information form the analyzed Dropbox account.

#### /account/info

Retrieves account information of the user.

- **HTTP Request:** GET `https://api.dropbox.com/1/account/info`
- **Parameters:** No parameter required.
- **Response:** User account information:

```
{
"uid": 12345678,
"display_name": "John User",
"name_details": {
"familiar_name": "John",
"given_name": "John",
"surname": "User"
},
"referral_link": "https://www.dropbox.com/referrals/r1a2n3d4m5s6t7",
"country": "US",
"locale": "en",
"is_paired": false,
"team": {
"name": "Acme Inc.",
"team_id": "dbtid:1234abcd"
```

```
  },
  "quota_info": {
  "shared": 253738410565,
  "quota": 107374182400000,
  "normal": 680031877871
  }
  }
```

Listing 4.3: Dropbox user information example

### 4.2.3 Files and metadata

In the following sections the methods used in the development of `kumodd` to handle files an metadata that belongs to an authenticated user are explained.

#### /files(GET)

This method is used to download a file from a Dropbox account using the remote file path as identifier.

```
f,metadata = client.get_file_and_metadata('/magnum-opus.txt')
out = open('magnum-opus.txt', 'wb')
out.write(f.read())
out.close()
```

Listing 4.4: get_file request using the Dropbox Python library

- **HTTP Request:** `GET https://api-content.dropbox.com/1/files/auto/`*`path`*
- **Parameters:** *`path`*
- **Response:** The specified file content.

```
{
  'bytes': 77,
  'icon': 'page_white_text',
  'is_dir': False,
  'mime_type': 'text/plain',
  'modified': 'Wed, 20 Jul 2011 22:04:50 +0000',
  'path': '/magnum-opus.txt',
  'rev': '362e2029684fe',
  'revision': 221922,
  'root': 'dropbox',
  'size': '77 bytes',
  'thumb_exists': False
}
```

Listing 4.5: Metadata of a file in Dropbox

#### /revisions

Obtains metadata for all available revisions (max. 1,000) of a specific file by providing its remote path as an identifier. Revision are only available for thirty days unless the user has an extended version history which is a paid feature.

```
revisions(path)
```

Listing 4.6: revision request using the Dropbox Python library

- **HTTP Request:** `GET https://api.dropbox.com/1/files/auto/`*`path`*
- **Parameters:** *`path`*

- **Response:** A list of revisions:

```
[
{
  "is_deleted": true,
  "revision": 4,
  "rev": "40000000d",
  "thumb_exists": false,
  "bytes": 0,
  "modified": "Wed, 20 Jul 2011 22:41:09 +0000",
  "path": "/hi2",
  "is_dir": false,
  "icon": "page_white",
  "root": "app_folder",
  "mime_type": "application/octet-stream",
  "size": "0 bytes"
},
{
  "revision": 1,
  "rev": "10000000d",
  "thumb_exists": false,
  "bytes": 3,
  "modified": "Wed, 20 Jul 2011 22:40:43 +0000",
  "path": "/hi2",
  "is_dir": false,
  "icon": "page_white",
  "root": "app_folder",
  "mime_type": "application/octet-stream",
  "size": "3 bytes"
}
]
```

Listing 4.7: Revisions method response body example

## 4.3 OneDrive

Microsoft OneDrive API provides access to data stored in an authenticated OneDrive user account. Interaction with OneDrive API is done using RESTful patterns. [30]

OneDrive API support OData V4; an open protocol used to standardize the way RESTful APIs are consumed. It defines an abstract data model and a protocol that let any client access information stored in any data source. [31]

OneDrive has classified its API functions by resource type. The two major resources are:

- **Drive** (top-level object)
- **Item** (files, folders, etc)

### 4.3.1 API authorization

To have access to OneDrive API and start sending requests; an application must be registered in the Microsoft Developer Center before starting using the OneDrive API. [2]

Like Google Drive and Dropbox; Ondrive uses OAuth 2.0 for authentication and the procedure (flow) is almost the same. In this case it is a three-step process with separate calls. The first step of the process is to get an authorization code which will allow the app to retrieve an access token to start sending requests.

After a successful authentication a token is received which contains a set of permissions for a user; this access token is used for every API request. Figure 4.6 [2]

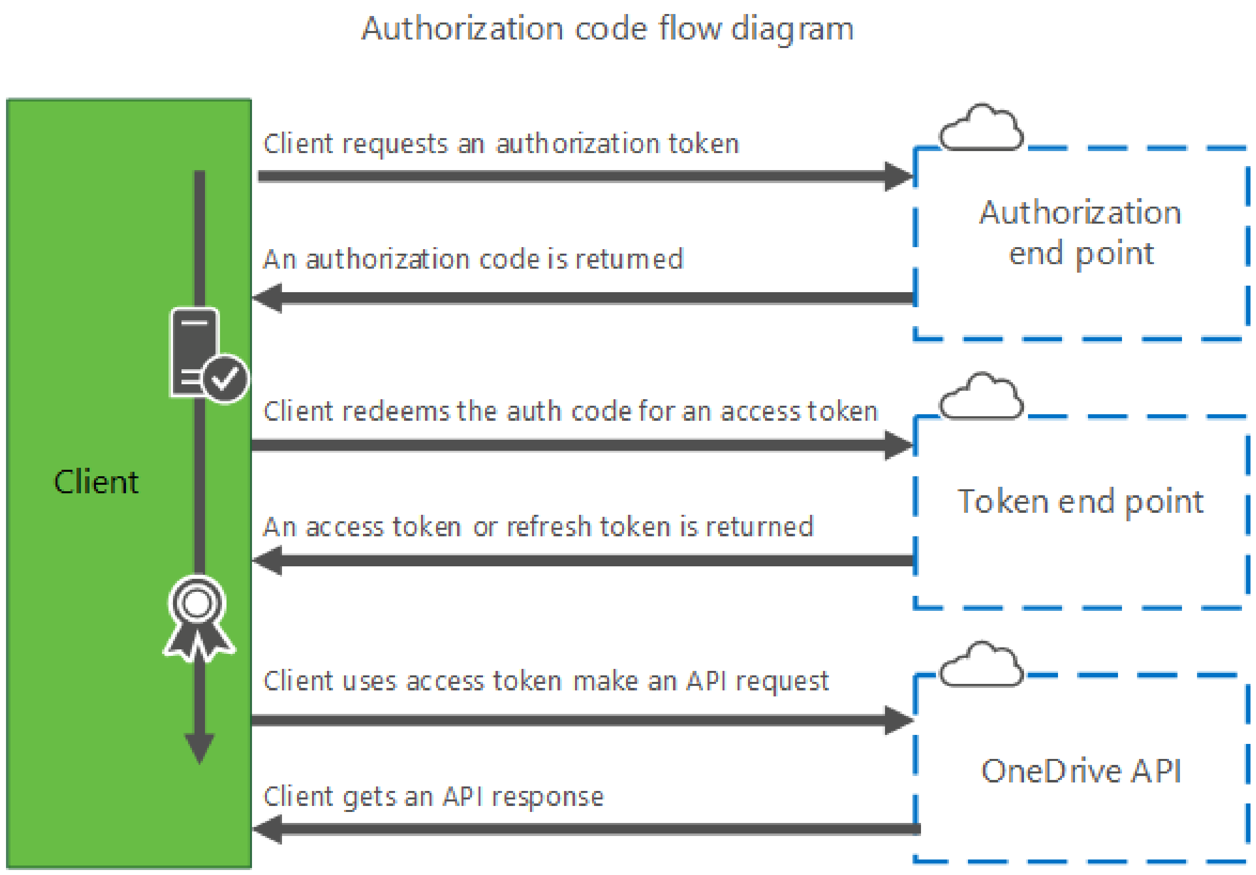

Authorization code flow diagram
Client
Client requests an authorization token
An authorization code is returned
Authorization end point
Client redeems the auth code for an access token
An access token or refresh token is returned
Token end point
Client uses access token make an API request
Client gets an API response
OneDrive API

Figure 4.6: OAuth 2 flow diagram. [2]

For every request the access token can be specified in the HTTP header (`Authorization: bearer` {*`token`*}) or as a query parameter to the end of the request URL (`?access_token=`*`token`*).

### Authentication scopes

Authentication scopes are used to determine the access will be granted to an application when an user authenticates trough it. Table 4.2

| Scope name | Description | Required |
|---|---|---|
| wl.signin | Allows sign-on capabilities. | No |
| wl.offline_access | Receive refresh token. | No |
| onedrive.readonly | Grants read-only permission | Yes |
| onedrive.readwrite | Read and write permission. | Yes |
| onedrive.appfolde | Read and write permission to a folder | Yes |

Table 4.2: OneDrive API authorization scopes

## 4.3.2 Drive resource

This is the top level resource of a user OneDrive account. It contains lower level resources such as files and folder. The Drive resource method used to develop Kumodd is explained.

#### /drive

This request is used to get metadata with general information about the default drive of an user account.

- **HTTP Request:** GET `https://api.onedrive.com/v1.0/drive`
- **Parameters:** No parameter required.
- **Request body:** No request body is necessary.
- **Response:** If successful, a Drive resource is returned in the response body:

```
{
  "id": "0123456789abc",
  "driveType": "consumer",
  "owner": {
    "user": {
      "id": "12391913bac",
      "displayName": "Ryan Gregg"
    }
  },
  "quota": {
    "total": 1024000,
    "used": 514000,
    "remaining": 1010112,
    "deleted": 0,
    "state": "normal"
  }
}
```

Listing 4.8: Drive resource response example

### 4.3.3 Item resource

Items are the folders or files that reside in OneDrive file system. The main item methods used to develop Kumodd are explained.

#### /drive/items/{id}

Retrieve metadata of an item on OneDrive by giving its remote path or id.

- **HTTP Request:** Any of the following requests are valid:
  - GET `https://api.onedrive.com/v1.0/drive/items/`*`item-id`*
  - GET `https://api.onedrive.com/v1.0/drive/root:/`*`item-path`*
- **Authorization scope:** Read access.
- **Parameters:** *`item-id`* or *`item-path`*.
- **Request body:** No request body is necessary.
- **Response:** If successful, an Item resource is returned in the response body:

```
{
  "id": "0123456789abc",
  "name": "example.xlsx",
  "eTag": "etag",
  "cTag": "etag",
  "createdBy": { "user": { "id": "1234", "displayName": "Ryan Gregg" } },
  "createdDateTime": "datetime",
  "lastModifiedBy": { "user": { "id": "1234", "displayName": "Ryan Gregg" } },
  "lastModifiedDateTime": "datetime",
  "size": 1234,
  "webUrl": "http://onedrive.com/...",
  "parentReference": { "driveId": "12345", "id": "root", "path": "/drive/root:" },
  "folder": { "childCount": 4 }
}
```

Listing 4.9: Item resource response example

### /drive/items/{id}/children

This request is used to get a list of items inside a folder by specifying it id or remote path.

- **HTTP Request:** Any of the following requests are valid:
  - GET `https://api.onedrive.com/v1.0/drive/items/`*`item-id`*`/children`
  - GET `https://api.onedrive.com/v1.0/drive/root:/`*`item-path`*`/children`
- **Authorization scope:** Read access.
- **Parameters:** *`item-id`* or *`item-path`*.
- **Request body:** No request body is necessary.
- **Response:** If successful, an list of items that are children of the target item resource is returned in the response body:

```
{
  "value": [
  {"name": "myfile.jpg", "size": 2048, "file": {} },
  {"name": "Documents", "folder": { "childCount": 4} },
  {"name": "Photos", "folder": { "childCount": 203} },
  {"name": "my sheet(1).xlsx", "size": 197 }
  ]
```

Listing 4.10: OneDrive list children example

### /drive/items/{id}/content

Method to download an item from OneDrive by providing its id or remote path.

- **HTTP Request:** Any of the following requests are valid:
  - GET `https://api.onedrive.com/v1.0/drive/items/`*`item-id`*`/content`
  - GET `https://api.onedrive.com/v1.0/drive/root:/`*`item-path`*`/content`
- **Authorization scope:** Read access.
- **Parameters:** *`item-id`* or *`item-path`*.
- **Request body:** No request body is necessary.
- **Response:** If successful, the method returns a 302 Found response directing to the URL of the item to proceed to download it.

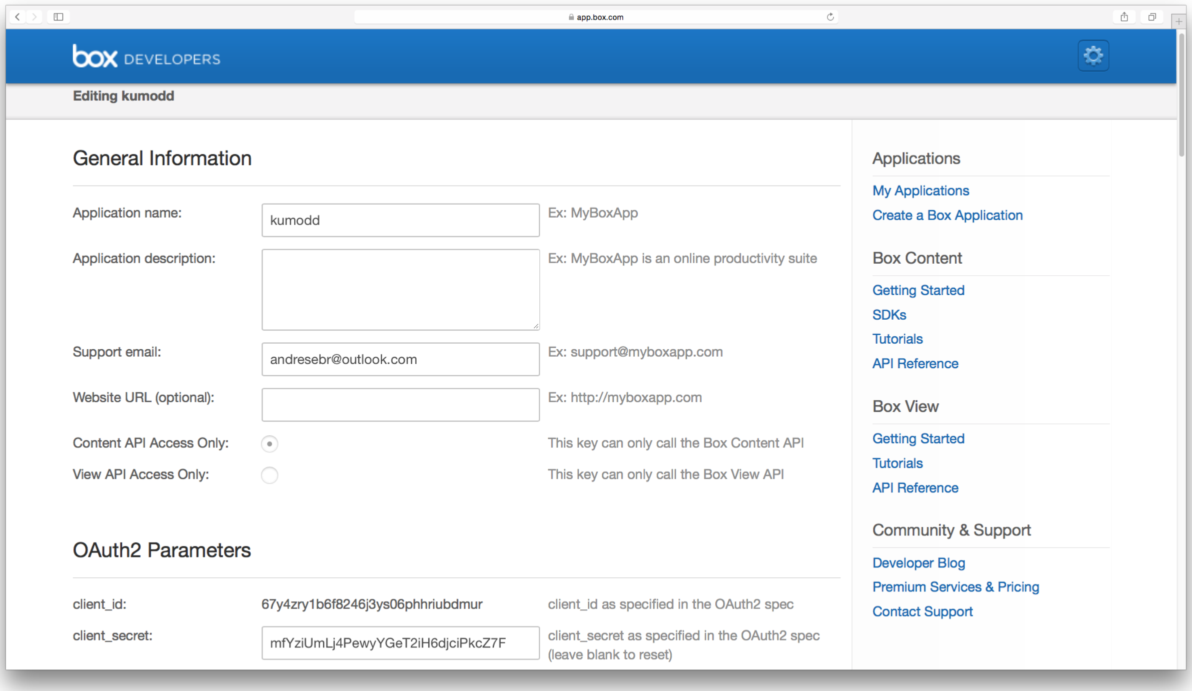
```
HTTP/1.1 302 Found
Location: https://b0mpua-by3301.files.1drv.com/y23vmagahszhxzlcvhasdhasghasodfi
```

Listing 4.11: Successful download file response

## 4.4 Box

Box allows developers to access user documents through its RESTful Content API. Libraries that simplify development are available for different programming languages such as Java, Python, Objective-C, etc. [32]

To start using the Dropbox API it is necessary to register the application in the Box Developers Console. Figure 4.7. The console is also used to manage general settings and authorization scopes. Box has classified its API in objects; the objects used for Kumodd development are Files and Folders.



Editing kumodd

General Information

| | | |
|---|---|---|
| Application name: | kumodd | Ex: MyBoxApp |
| Application description: | | Ex: MyBoxApp is an online productivity suite |
| Support email: | andresebr@outlook.com | Ex: support@myboxapp.com |
| Website URL (optional): | | Ex: http://myboxapp.com |
| Content API Access Only: | [✓] | This key can only call the Box Content API |
| View API Access Only: | [ ] | This key can only call the Box View API |

OAuth2 Parameters

| | | |
|---|---|---|
| client_id: | 67y4zry1b6f8246j3ys06phhriubdmur | client_id as specified in the OAuth2 spec |
| client_secret: | mfYziUmLj4PewyYGeT2iH6djciPkcZ7F | client_secret as specified in the OAuth2 spec (leave blank to reset) |

Figure 4.7: Box Developers Console

### 4.4.1 API authorization

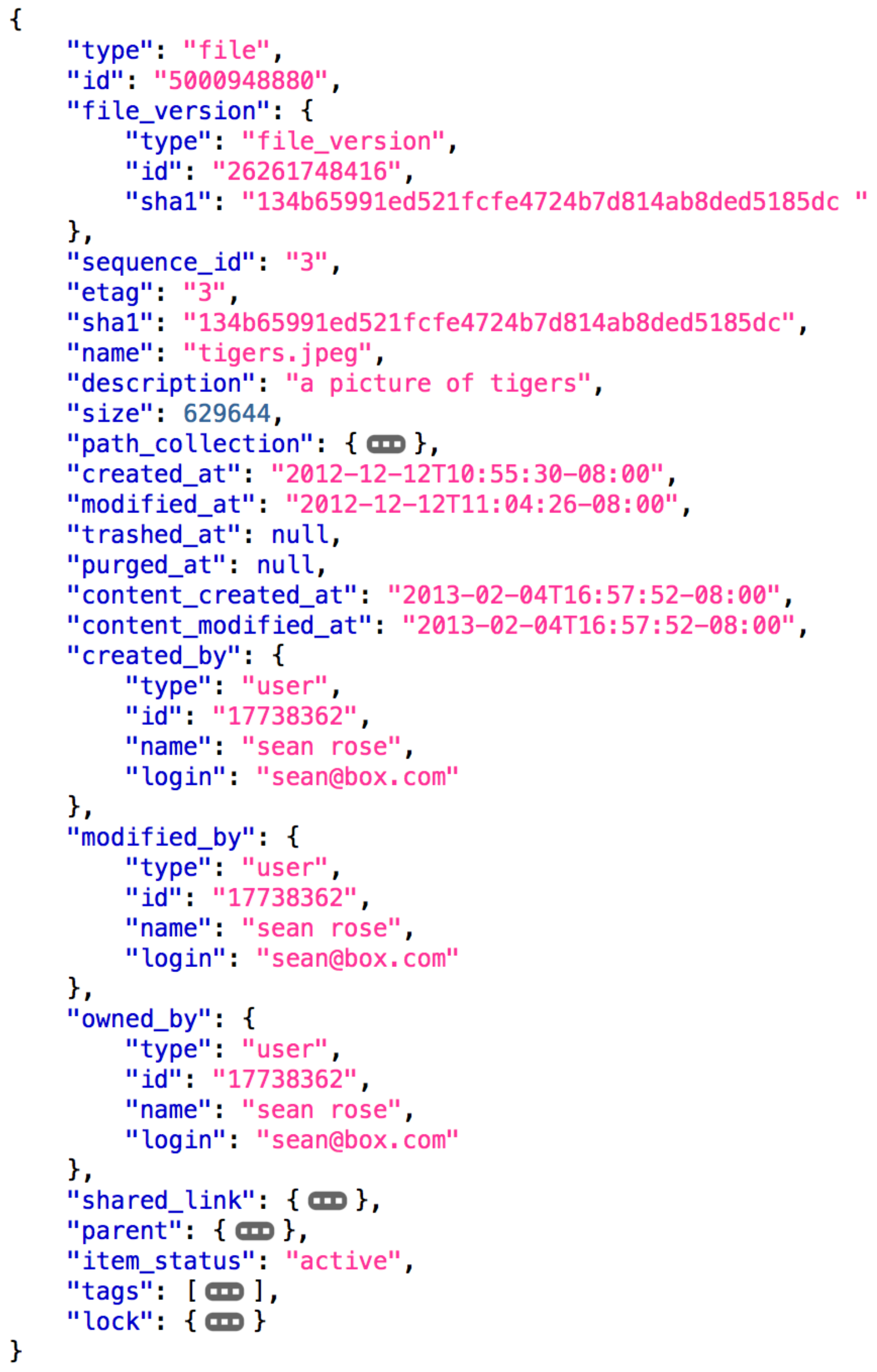
Box uses OAuth 2.0 for authentication. The process it uses to authenticate an is similar to the other services. After a user authenticates and gives the application the permissions necessary to interact with its Box account; it receives an access token that will be used in every request header. [33]

```
{
  "access_token": "T9cE5asGnuyYCCqIZFoWjFHvNbvVqHjl",
  "expires_in": 3600,
  "restricted_to": [],
  "token_type": "bearer",
  "refresh_token": "J7rxTiWOHMoSC1isKZKBZWizoRXjkQzig5C6jFgCVJ9
  bUnsUfGMinKBDLZWP9BgR"
}
```

Listing 4.12: Successful Content API authorization response

### 4.4.2 File object

A file object is every item stored in a Box drive. Except folders. Methods used to get information about file objects are explained.



```
{
    "type": "file",
    "id": "5000948880",
    "file_version": {
        "type": "file_version",
        "id": "26261748416",
        "sha1": "134b65991ed521fcfe4724b7d814ab8ded5185dc "
    },
    "sequence_id": "3",
    "etag": "3",
    "sha1": "134b65991ed521fcfe4724b7d814ab8ded5185dc",
    "name": "tigers.jpeg",
    "description": "a picture of tigers",
    "size": 629644,
    "path_collection": { ... },
    "created_at": "2012-12-12T10:55:30-08:00",
    "modified_at": "2012-12-12T11:04:26-08:00",
    "trashed_at": null,
    "purged_at": null,
    "content_created_at": "2013-02-04T16:57:52-08:00",
    "content_modified_at": "2013-02-04T16:57:52-08:00",
    "created_by": {
        "type": "user",
        "id": "17738362",
        "name": "sean rose",
        "login": "sean@box.com"
    },
    "modified_by": {
        "type": "user",
        "id": "17738362",
        "name": "sean rose",
        "login": "sean@box.com"
    },
    "owned_by": {
        "type": "user",
        "id": "17738362",
        "name": "sean rose",
        "login": "sean@box.com"
    },
    "shared_link": { ... },
    "parent": { ... },
    "item_status": "active",
    "tags": [ ... ],
    "lock": { ... }
}
```

Figure 4.8: Get file response example (abbreviated)

### /files/{id}

Method to get information of a file.

- **HTTP Request:** `GET https://api.box.com/2.0/files/`*`file-id`*

- **Parameters:** *file-id*.
- **Response:** If successful, the method returns a File object in the response body. Figure 4.8

### /files/{id}/content

Method to get the content and download a file.

- **HTTP Request:** GET `https://api.box.com/2.0/files/`*file-id*`/content`
- **Parameters:** *file-id*.
- **Response:** If successful, the method returns 302 Found response along with the download URL of the target file.

### /files/{id}/versions

Method to retrieve older versions of a file.

- **HTTP Request:** GET `https://api.box.com/2.0/files/`*file-id*`/versions`
- **Parameters:** *file-id*.
- **Response:** If the target file has older versions an array of version objects is returned:

```
{
 "total_count": 1,
 "entries": [
 {
   "type": "file_version",
   "id": "672259576",
   "sha1": "359c6c1ed98081b9a69eb3513b9deced59c957f9",
   "name": "Dragons.js",
   "size": 92556,
   "created_at": "2012-08-20T10:20:30-07:00",
   "modified_at": "2012-11-28T13:14:58-08:00",
   "modified_by": {
     "type": "user",
     "id": "183732129",
     "name": "sean rose",
     "login": "sean+apitest@box.com"
   }
 }
 ]
}
```

Listing 4.13: Array of versions

## 4.4.3 Folder object

### /folders/{FOLDER-ID}/items

Retrieves all files and folders contained in the target folder.

- **HTTP Request:** GET `https://api.box.com/2.0/folders/`*FOLDER-ID*`/items`
- **Parameters:** *FOLDER-ID*.
- **Response:** If successful, a collection of items contained in the folder is returned:

```
{
  "total_count": 24,
  "entries": [
  {
    "type": "folder",
    "id": "192429928",
    "sequence_id": "1",
    "etag": "1",
    "name": "Stephen Curry Three Pointers"
  },
  {
    "type": "file",
    "id": "818853862",
    "sequence_id": "0",
    "etag": "0",
    "name": "Warriors.jpg"
  }
  ],
  "offset": 0,
  "limit": 2,
  "order": [
  {
    "by": "type",
    "direction": "ASC"
  },
  {
    "by": "name",
    "direction": "ASC"
  }
  ]
}
```

Listing 4.14: Items inside a folder

# Chapter 5

# Evaluation

## 5.1 Functionality

As explained in Chapter 3. The current version of Kumodd has three main functions–content discovery, filtering, and acquisition (Figure 3.4).

#### Discovery

Discovery is implemented by means of listing the files in the target account–by default, all files and revisions are exhaustively listed. An abbreviated version of the results is sent to the standard output device, with the complete data returned placed in a log file. The fields listed include the ID of the file which is important for request the actual file content and metadata at the time of the data acquisition step. Also other important information is included such as the remote path where the file is contained, the number of revisions and cryptographic hash information.

### 5.1.1 Filtering

The tool has built-in filters for a number of common (groups of) file types–documents, text, images, audio, video, etc.–which can be used to quickly filter both the list of discovered files and the list of files to acquire. For more complex filtering criteria, the analyst can build on-the-fly text filters from Unix-style text processing commands.

### 5.1.2 Acquisition

When the investigator has discovered and filtered relevant files. Then the last step would be to download the local files.

It is important to note that the data downloaded is preserved as it is in the cloud, no information is manipulated in the process. Kumodd also recreates the directory structure of the cloud drive and download every change made to a file as a separate files.

Metadata information of each file will also be downloaded which contain more detailed information about the file. Like creation dates, owners, etc.

If a file already exists locally the application will perform an update to that if necessary, or will skip it if no changes were made.

Every download or update operation is logged; this way the investigator would be aware of errors and what files where downloaded including directory information.

## 5.2 Download times

Using an Amazon EC2 (US-West, N. California) instance for each one of the four supported services and running a cron job in Linux everyday for seven days at 10:00 AM (PDT) To download 1024 MB of data

(1023 files). Download times for each service supported by Kumodd were obtained:

Table 5.1: Kumodd download times

| Service | Average time (mm:ss.ms) |
|---|---|
| Google Drive | 17:21.57 |
| Dropbox | 16:59.71 |
| OneDrive | 18:21.29 |
| Box | 18:21.00 |

The instances used to get these results were general purpose instances; variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only, low to moderate network performance.

## 5.3 Completeness

In this section the completeness of Kumodd is evaluated in terms of what it can do with mirrored data and cloud-only data compared to traditional acquisition approaches.

### 5.3.1 Mirrored data

We define mirrored data as files that can reside in a client device and also in a cloud storage service. When a user has installed a client application from one (or more) cloud storage provider. It is possible to use traditional methods to obtain this files as are downloaded locally, provided the user has not modified default setting and the files sored in the services are being downloaded partially (files from a specific remote directory) or completely (all the cloud drive content). Traditional acquisition methods are the option to take in cases that it is known the suspect had made use of cloud storage services and access credentials are not available.

In case credentials are available it is possible to acquire all the files stored in a cloud drive using Kumodd. This will obtain not only the files but its metadata with detailed information and older version if available.

### 5.3.2 Cloud-only data

There are some files that only reside in the cloud. For example, files created using Google web applications such as Google Docs, Sheets or Spreadsheets can be only obtained using an API approach and exporting them to a compatible format. This can be done using Kumodd as a traditional approach would only get a placeholder file which is basically a link to the resource.

Kumodd ca also obtain a complete history of changes of this files, downloading them as revisions.

## 5.4 Discussion

As mentioned in earlier sections, some differences can be found between the analyzed cloud storage providers. For example; it is possible to retrieve cryptographic hashes from the files stored in each services, except for Dropbox.

Dropbox provides with very basic information when the metadata of a file is requested. The only identifier that it offers in place of a cryptographic hash is a nine characters long string.

Other difference between these services is that Google Drive has its Google Docs service. This is a web-only suite for creation and edition of documents. Google Docs is embedded in Google Drive, in fact; when a file is created using a Google Docs application its content is stored in the cloud and this information is not mirrored in the system of the user. The client application downloads these files as an URL which redirects the user to the application location in the cloud for viewing and edition purposes. The only way to obtain the content of the files for local analysis is by exporting it to a local document viewer extension such as .pdf, .doc, etc. As the files created by Google Docs are cloud artifacts, cryptographic information about these files cannot be obtained.

Other services suffers from lack of important features such as the retrieval of revisions. OneDrive for example does not offer any way to get older revision of its files trough its API, although it is known that the service store this information in its servers.

There are other limitations that are different even in the same service. For example, Box has a lot of limitations of what the API can do when connected to a non-premium account. Revisions are just stored for 30 days in free tiers and these could represent an important loss of evidence in an investigation.

It is important to take into account that as of today, cloud storage services offers a large amount of data for premium users. As of today, Google Drive offers a max amount of 30TB for a rate of $299.99 per month [34]. Using an API approaches in these case could be very ineffective and time consuming.

## 5.5 Future work

Kumodd was structured in a way that adding more functions to it would not represent too much work or a need for the developer to understand code of a current version.

In a future it could be possible to support more services. Not only cloud drive services but note taking services like Evernote and social network support for services such as Facebook or Google+ which could be useful to get chat logs and other forensically relevant information of users that have accounts on those services. To do this is it is important to perform an API evaluation of the services and analyze if what they provide could be useful for evidence acquisition in an investigation.

It is important to implement a way of organizing investigation cases as there is the possibility that the investigator is involved in more than one investigation. Also implement more forensic aimed functionalities such as time-lining the cloud account from the time its user logged in from the first time would be helpful in the future. As of now this is only possible by using file time stamps but some services offer a way of obtaining information about file deletion or relocation.

In terms of usability it would be very useful to optimize the application to make better use of the web GUI and also develop an authentication system so it can be used by various users.

# Chapter 6

# Conclusion

Some of previous investigations done was based on traditional approaches; thats is finding ways to get forensically relevant information from remnants left by client applications or user web navigation. While this approach could lead to some advances without having user credentials to authenticate in every cloud drive account the user has logged in, it is almost impossible to get a complete information of he user interaction with those services.

Using the API acquisition approach, it is possible to obtain the content of each file stored in an account, metadata information and also a detailed history of file changes. The downside of this is that user credentials are necessary to be able to access user accounts and perform API request to the service. But this is a only issue that this paper does not cover.

Kumodd was developed thinking in the API approach. This Python application connects to four cloud storage services; Google Drive, Dropbox, OneDrive and Box. It retrieves every possible information from the files (metadata) stored in each service and also gives a way of trace the changes made too each one of them.

From the cloud storage providers we analyzed it is possible to be aware of the differences each one of them has compared to the others. One ef the problems with trying to develop tools for cloud forensic investigation is that each service provider is different; each one of them has its own standards about what information can be obtained and the format it is presented. This aspect complicates performing forensic investigations on the cloud and every attempt to develop any tool for this cause will reach a point where is not possible to cover every cloud service as long as the different guidelines, rules and requirement each company enforces on their respective services is not standardized.

An example of this issue is that from the four cloud storage services covered in this paper. It is possible to obtain cryptographic hashes of the files that reside in them except for files stored in Dropbox. Also Google Drive offers cryptographic hash information for each file that was not created using a Google Web (Google Docs, Google Sheets, Google Slides, etc.). It is possible to download these files by exporting them to a format that a client application like one of the Microsoft Office Suite or other document editors can process and open, but it is impossible to obtain cryptographic hashes of them as these documents reside in the cloud.

Another problem that should be addressed in the future is that the amount of data the cloud services are offering is increasing and new approaches should be implemented. Cloud service providers should offer a way to process a large amount of data to improve efficiency in an investigation.

# Bibliography

[1] (2015) Dropbox - OAuth guide. Dropbox Inc. [Online]. Available: https://www.dropbox.com/developers/reference/oauthguide

[2] (2015) OneDrive Dev Center - OneDrive authentication and sign-in. Microsoft. [Online]. Available: https://dev.onedrive.com/auth/msa_oauth.htm

[3] P. Mell and T. Grance, "The nist definition of cloud computing." [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[4] (2015) Rightscale releases 2015 state of the cloud report. [Online]. Available: http://www.rightscale.com/press-releases/rightscale-releases-2015-state-of-the-cloud-report

[5] (2014) title. Gartner. [Online]. Available: http://www.gartner.com/technology/research/methodologies/hype-cycle.jsp

[6] I. Ahmed and V. Roussev, "Analysis of cloud digital evidence."

[7] R. McKemmish, "What is forensic computing," no. 118. [Online]. Available: http://www.aic.gov.au/media_library/publications/tandi_pdf/tandi118.pdf

[8] D. Quick, "Cloud storage forensic analysis." [Online]. Available: https://wiki.cis.unisa.edu.au/wki/images/1/18/QUICK_Cloud_Storage_Forensic_Analysis.pdf

[9] The basics of cloud forensics. [Online]. Available: http://cloudtimes.org/2012/11/05/the-basics-of-cloud-forensics/

[10] P. Grabosky. Computer crime: A criminological overview. Australian Institute of Criminology. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/

[11] J. Somers. How i reverse engineered google docs to play back any document's keystrokes. [Online]. Available: http://features.jsomers.net/how-i-reverse-engineered-google-docs/

[12] H. Chung, J. Park, S. Lee, and C. Kang, "Digital forensic investigation of cloud storage services," vol. 9. [Online]. Available: http://dx.doi.org/10.1016/j.diin.2012.05.015

[13] J. Hale, "Amazon cloud drive forensic analysis," vol. 10, pp. 295–265, October 2013. [Online]. Available: http://dx.doi.org/10.1016/j.diin.2013.04.006

[14] D. Quick and K. R. Choo, "Dropbox analysis: Data remnants on user machines," vol. 10, pp. 3–18, June 2013. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-24212-0_3

[15] B. Martini and K. R. Choo, "An integrated conceptual digital forensic framework for cloud computing," vol. 9, pp. 71–80, November 2012. [Online]. Available: http://dx.doi.org/10.1016/j.diin.2012.07.001

[16] P. Clark, "Digital forensics tool testing – image metadata in the cloud." [Online]. Available: http://brage.bibsys.no/xmlui/handle/11250/143978

[17] What is python? executive summary. Python.org. [Online]. Available: https://www.python.org/doc/essays/blurb/

[18] python-gflags. [Online]. Available: https://code.google.com/p/python-gflags/

[19] python-gflags documentation. [Online]. Available: http://python-gflags.googlecode.com/svn/trunk/gflags.py

[20] google-api-python-client. Google Inc. [Online]. Available: https://github.com/google/google-api-python-client

[21] Install Core API SDKS. Dropbox Inc. [Online]. Available: https://www.dropbox.com/developers/core/sdks/python

[22] box-python-sdk. [Online]. Available: https://github.com/box/box-python-sdk

[23] A. Ronacher. (2014) Welcome — Flask (A Python Microframework). [Online]. Available: http://flask.pocoo.org/

[24] Flask-Classy — Flask Classy 0.6.7 documentation. [Online]. Available: https://pythonhosted.org/Flask-Classy/

[25] Google Drive REST API Overview - Drive REST API — Google Developers. Google Inc. [Online]. Available: https://developers.google.com/drive/web/about-sdk

[26] API Reference - Drive REST API — Google Developers. Google Inc. [Online]. Available: https://developers.google.com/drive/v2/reference/

[27] Authorizing Your App with Google Drive - Drive REST API — Google Developers. Google Inc. [Online]. Available: https://developers.google.com/drive/web/about-auth

[28] Choose Auth Scopes — Drive REST API — Google Developers. Google Inc. [Online]. Available: https://developers.google.com/drive/web/scopes

[29] (2015) Dropbox - Developer guide. Dropbox Inc. [Online]. Available: https://www.dropbox.com/developers/reference/devguide

[30] (2015) OneDrive Dev Center - Develop with the Onedrive API. [Online]. Available: https://dev.onedrive.com/README.htm

[31] (2015) Introducing OData. Microsoft. [Online]. Available: https://msdn.microsoft.com/en-us/data/hh237663.aspx

[32] (2015) Box Platform Developer Documentation. Box Inc. [Online]. Available: https://developers.box.com/sdks/

[33] (2015) Content API. Box Inc. [Online]. Available: https://box-content.readme.io/

[34] Buy and manage storage plans. Google Inc. [Online]. Available: https://support.google.com/drive/answer/2375123?hl=en

# Appendix A

# Google Drive API

## A.1 Google Drive resources

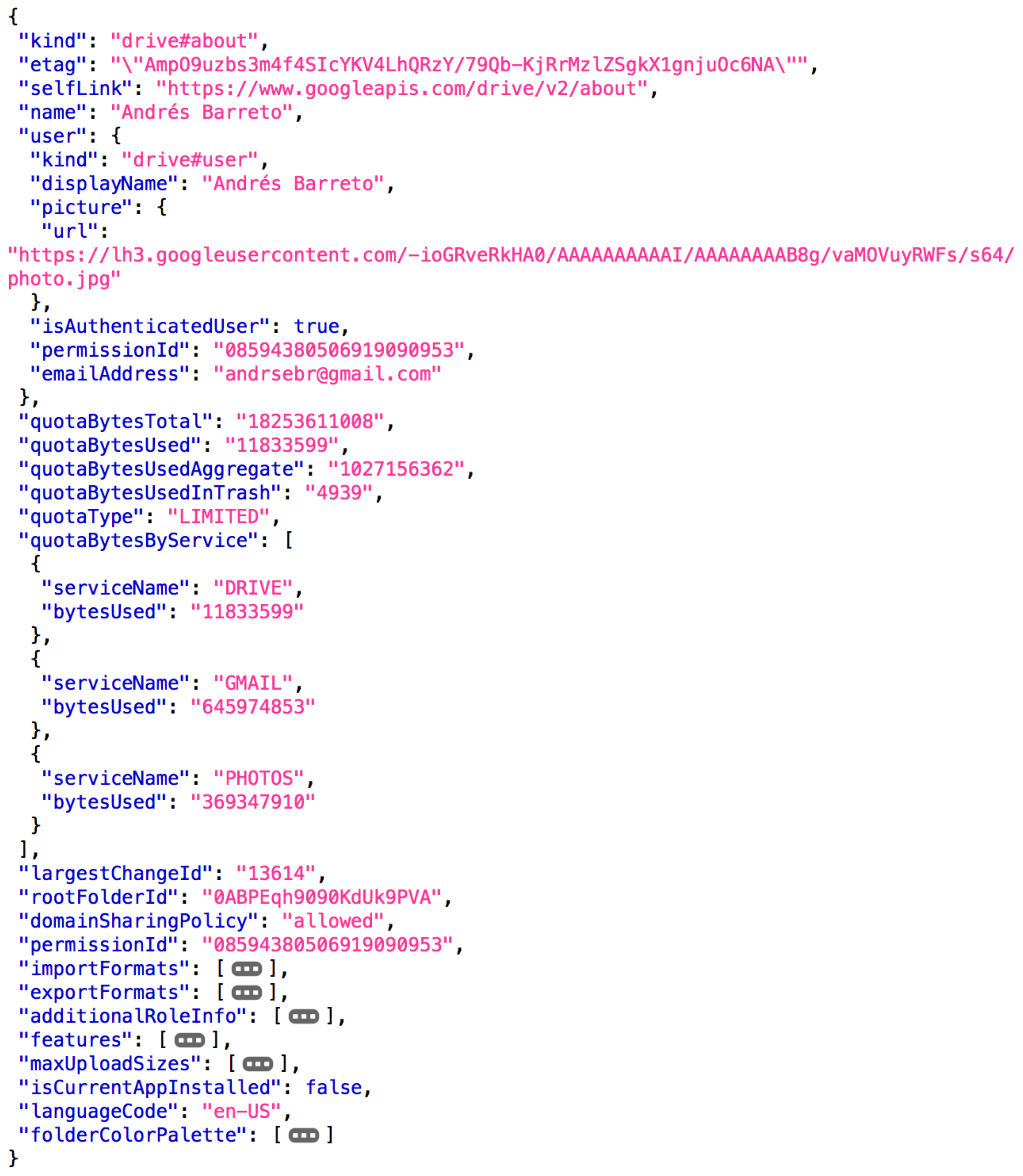
### A.1.1 About Resource

The about resource has a general method to obtain information about the authenticated user Google Drive Account.

#### About:get

Gets information about the user and Drive API settings.

- **HTTP Request:** `GET https://www.googleapis.com/drive/v2/about`
- **Parameters:** No parameter is required.
- **Authorization:** For this request an authorization with at least one of he following scopes is required:
  - `https://www.googleapis.com/auth/drive`
  - `https://www.googleapis.com/auth/drive.file`
  - `https://www.googleapis.com/auth/drive.readonly`
  - `https://www.googleapis.com/auth/drive.metadata.readonly`
  - `https://www.googleapis.com/auth/drive.appdata`
  - `https://www.googleapis.com/auth/drive.metadata`
- **Response:** If successful, this method returns an about resource in the response body. Figure A.1



```
{
 "kind": "drive#about",
 "etag": "\"Amp09uzbs3m4f4SIcYKV4LhQRzY/79Qb-KjRrMzlZSgkX1gnjuOc6NA\"",
 "selfLink": "https://www.googleapis.com/drive/v2/about",
 "name": "Andrés Barreto",
 "user": {
  "kind": "drive#user",
  "displayName": "Andrés Barreto",
  "picture": {
   "url":
"https://lh3.googleusercontent.com/-ioGRveRkHA0/AAAAAAAAAAI/AAAAAAAAB8g/vaMOVuyRWFs/s64/
photo.jpg"
  },
  "isAuthenticatedUser": true,
  "permissionId": "08594380506919090953",
  "emailAddress": "andrsebr@gmail.com"
 },
 "quotaBytesTotal": "18253611008",
 "quotaBytesUsed": "11833599",
 "quotaBytesUsedAggregate": "1027156362",
 "quotaBytesUsedInTrash": "4939",
 "quotaType": "LIMITED",
 "quotaBytesByService": [
  {
   "serviceName": "DRIVE",
   "bytesUsed": "11833599"
  },
  {
   "serviceName": "GMAIL",
   "bytesUsed": "645974853"
  },
  {
   "serviceName": "PHOTOS",
   "bytesUsed": "369347910"
  }
 ],
 "largestChangeId": "13614",
 "rootFolderId": "0ABPEqh9090KdUk9PVA",
 "domainSharingPolicy": "allowed",
 "permissionId": "08594380506919090953",
 "importFormats": [ ⋯ ],
 "exportFormats": [ ⋯ ],
 "additionalRoleInfo": [ ⋯ ],
 "features": [ ⋯ ],
 "maxUploadSizes": [ ⋯ ],
 "isCurrentAppInstalled": false,
 "languageCode": "en-US",
 "folderColorPalette": [ ⋯ ]
}
```

Figure A.1: About:get method response example (abbreviated)

### A.1.2 Children resource

The children resource has several methods that allow external interaction with children resources that are stored in an specific resource. Like files that are contained in a folder. Children methods used in the development of Kumodd are summarized in Table A.1.

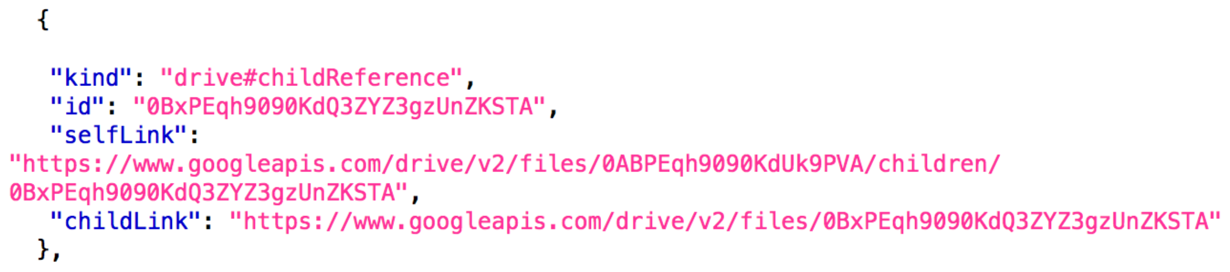
Table A.1: Children resource methods; URIs relative to https://www.googleapis.com/drive/v2. A complete list of methods is available in in the Drive REST API reference provided by Google

| Method | HTTP request | Description |
|---|---|---|
| **get** | `GET /files/folderId/children/childId` | Gets a specific child reference. |
| **list** | `GET /files/folderId/children` | Lists a folder's children. |

**Children:get**

This methods is used to get a specific child reference.

- **HTTP Request:** `GET https://www.googleapis.com/drive/v2/files/`*`folderId`*`/children/`*`childId`*
- **Parameters:** *`childId`*, *`folderId`*.
- **Authorization:** For this request an authorization with at least one of he following scopes is required:
  - `https://www.googleapis.com/auth/drive`
  - `https://www.googleapis.com/auth/drive.file`
  - `https://www.googleapis.com/auth/drive.readonly`
  - `https://www.googleapis.com/auth/drive.metadata.readonly`
  - `https://www.googleapis.com/auth/drive.appdata`
  - `https://www.googleapis.com/auth/drive.metadata`
- **Response:** If successful, this method returns a children resource in the response body. Figure A.2



```
{
 "kind": "drive#childReference",
 "id": "0BxPEqh9090KdQ3ZYZ3gzUnZKSTA",
 "selfLink":
"https://www.googleapis.com/drive/v2/files/0ABPEqh9090KdUk9PVA/children/
0BxPEqh9090KdQ3ZYZ3gzUnZKSTA",
 "childLink": "https://www.googleapis.com/drive/v2/files/0BxPEqh9090KdQ3ZYZ3gzUnZKSTA"
},
```

Figure A.2: Children:get method response example

**Children:list**

Method used to list the children of a folder. The folderId of the root folder is **root**.

- **HTTP Request:** `GET https://www.googleapis.com/drive/v2/files/`*`folderId`*`/children`
- **Parameters:** *`folderId`*.
- **Authorization:** For this request an authorization with at least one of he following scopes is required:
  - `https://www.googleapis.com/auth/drive`
  - `https://www.googleapis.com/auth/drive.file`
  - `https://www.googleapis.com/auth/drive.readonly`
  - `https://www.googleapis.com/auth/drive.metadata.readonly`
  - `https://www.googleapis.com/auth/drive.appdata`
  - `https://www.googleapis.com/auth/drive.metadata`

- **Response:** If successful, this method returns a response body with the following structure:

```
{
"kind": "drive#childList",
"etag": etag,
"selfLink": string,
"nextPageToken": string,
"nextLink": string,
"items": [
children Resource
]
}
```

Listing A.1: Children:list response body structure

### A.1.3 Revisions resource

The revisions resource has several methods that allow interaction with versions of an specific file resource. For Google Docs a revision is created every time the document is edited. For other files, revisions are created if the user uploads manually a new file version or the file is edited in the Google Drive application (mobile or desktop). Revisions methods used in the development of Kumodd are summarized in Table A.2.

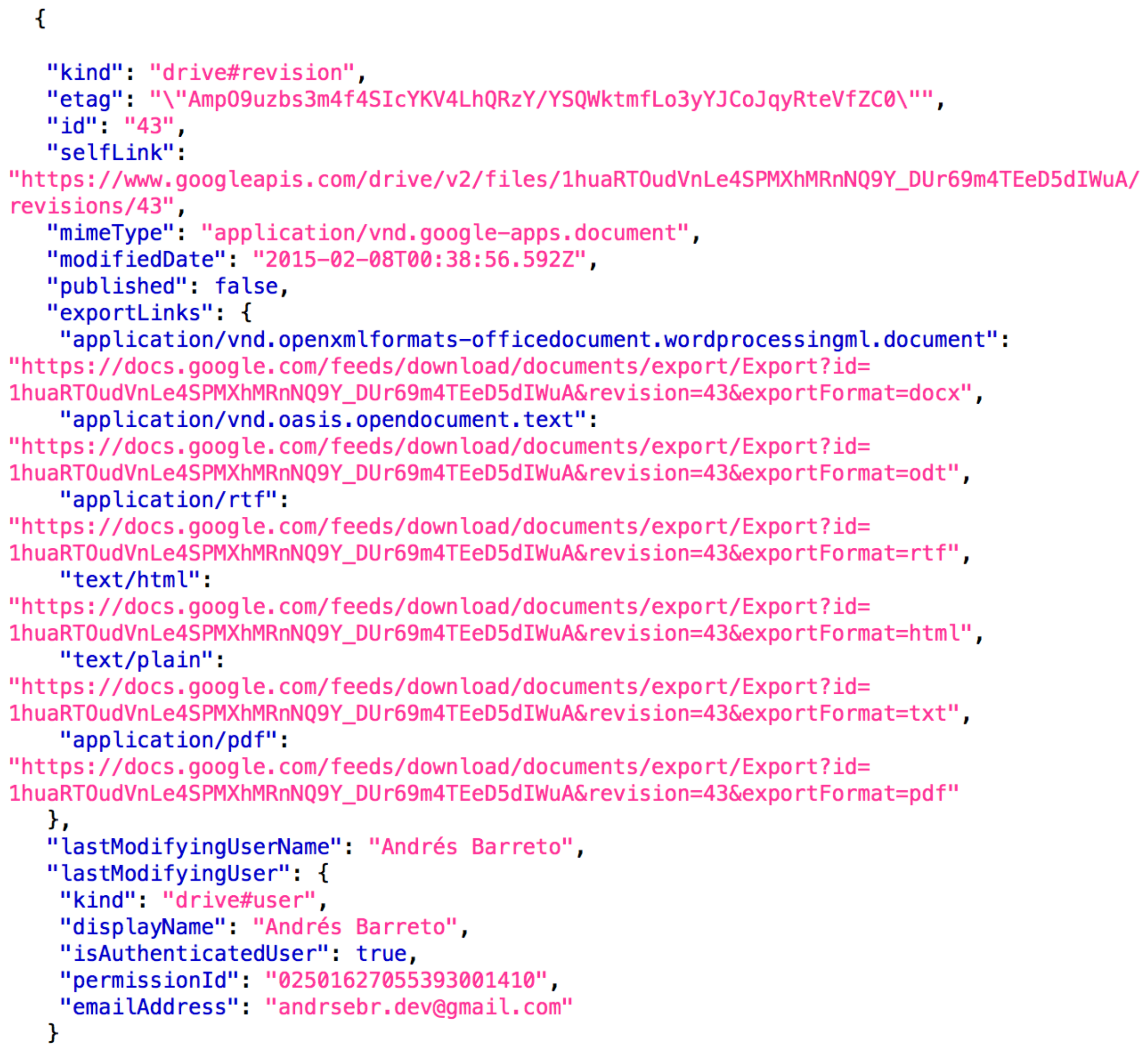
Table A.2: Revisions resource methods; URIs relative to https://www.googleapis.com/drive/v2. A complete list of methods is available in in the Drive REST API reference provided by Google [26]

| Method | HTTP request | Description |
|---|---|---|
| **get** | `GET /files/fileId/revisions/revisionId` | Gets a specific revision. |
| **list** | `GET /files/fileId/revisions` | Lists a file's revisions. |

#### Revisions:get

Gets metadata of a revision by specifying its ID and the file ID of the original file.

- **HTTP Request:** `GET` `https://www.googleapis.com/drive/v2/files/`*`fileId`*`/revisions/`*`revisionId`*
- **Parameters:** *`fileId`*, *`revisionId`*.
- **Authorization:** For this request an authorization with at least one of he following scopes is required:
  - `https://www.googleapis.com/auth/drive`
  - `https://www.googleapis.com/auth/drive.file`
  - `https://www.googleapis.com/auth/drive.readonly`
  - `https://www.googleapis.com/auth/drive.metadata.readonly`
  - `https://www.googleapis.com/auth/drive.appdata`
  - `https://www.googleapis.com/auth/drive.metadata`
- **Response:** JSON with a File resource in the response body.Figure A.3



```
 {

  "kind": "drive#revision",
  "etag": "\"AmpO9uzbs3m4f4SIcYKV4LhQRzY/YSQWktmfLo3yYJCoJqyRteVfZC0\"",
  "id": "43",
  "selfLink":
"https://www.googleapis.com/drive/v2/files/1huaRTOudVnLe4SPMXhMRnNQ9Y_DUr69m4TEeD5dIWuA/
revisions/43",
  "mimeType": "application/vnd.google-apps.document",
  "modifiedDate": "2015-02-08T00:38:56.592Z",
  "published": false,
  "exportLinks": {
   "application/vnd.openxmlformats-officedocument.wordprocessingml.document":
"https://docs.google.com/feeds/download/documents/export/Export?id=
1huaRTOudVnLe4SPMXhMRnNQ9Y_DUr69m4TEeD5dIWuA&revision=43&exportFormat=docx",
   "application/vnd.oasis.opendocument.text":
"https://docs.google.com/feeds/download/documents/export/Export?id=
1huaRTOudVnLe4SPMXhMRnNQ9Y_DUr69m4TEeD5dIWuA&revision=43&exportFormat=odt",
   "application/rtf":
"https://docs.google.com/feeds/download/documents/export/Export?id=
1huaRTOudVnLe4SPMXhMRnNQ9Y_DUr69m4TEeD5dIWuA&revision=43&exportFormat=rtf",
   "text/html":
"https://docs.google.com/feeds/download/documents/export/Export?id=
1huaRTOudVnLe4SPMXhMRnNQ9Y_DUr69m4TEeD5dIWuA&revision=43&exportFormat=html",
   "text/plain":
"https://docs.google.com/feeds/download/documents/export/Export?id=
1huaRTOudVnLe4SPMXhMRnNQ9Y_DUr69m4TEeD5dIWuA&revision=43&exportFormat=txt",
   "application/pdf":
"https://docs.google.com/feeds/download/documents/export/Export?id=
1huaRTOudVnLe4SPMXhMRnNQ9Y_DUr69m4TEeD5dIWuA&revision=43&exportFormat=pdf"
  },
  "lastModifyingUserName": "Andrés Barreto",
  "lastModifyingUser": {
   "kind": "drive#user",
   "displayName": "Andrés Barreto",
   "isAuthenticatedUser": true,
   "permissionId": "02501627055393001410",
   "emailAddress": "andrsebr.dev@gmail.com"
  }
```

Figure A.3: Revisions:get method response example

### Revision:list

This method lists the revisions of a file.

- **HTTP Request:** GET `https://www.googleapis.com/drive/v2/files/`*`fileId`*`/revisions`
- **Parameters:** *`fileId`*.
- **Authorization:** For this request an authorization with at least one of he following scopes is required:
  - `https://www.googleapis.com/auth/drive`
  - `https://www.googleapis.com/auth/drive.file`
  - `https://www.googleapis.com/auth/drive.readonly`
  - `https://www.googleapis.com/auth/drive.metadata.readonly`
  - `https://www.googleapis.com/auth/drive.appdata`

    – `https://www.googleapis.com/auth/drive.metadata`

- **Response:** If successful, this method returns a response body with the following structure:

```
{
"kind": "drive#revisionList",
"etag": etag,
"selfLink": string,
"items": [
revisions Resource
]
}
```

Listing A.2: Revisions:list response body structure

Information about additional resources, methods, parameters and a complete response overview are available in the Drive REST API reference provided by Google [26].

## A.2 Google Drive resource representations

### A.2.1 About resource

```
{
  "kind": "drive#about",
  "etag": etag,
  "selfLink": string,
  "name": string,
  "user": {
    "kind": "drive#user",
    "displayName": string,
    "picture": {
      "url": string
    },
    "isAuthenticatedUser": boolean,
    "permissionId": string,
    "emailAddress": string
  },
  "quotaBytesTotal": long,
  "quotaBytesUsed": long,
  "quotaBytesUsedAggregate": long,
  "quotaBytesUsedInTrash": long,
  "quotaType": string,
  "quotaBytesByService": [
  {
    "serviceName": string,
    "bytesUsed": long
  }
  ],
  "largestChangeId": long,
  "remainingChangeIds": long,
  "rootFolderId": string,
  "domainSharingPolicy": string,
  "permissionId": string,
  "importFormats": [
  {
    "source": string,
    "targets": [
    string
    ]
  }
  ],
  "exportFormats": [
  {
    "source": string,
    "targets": [
    string
    ]
  }
  ],
  "additionalRoleInfo": [
  {
    "type": string,
    "roleSets": [
    {
      "primaryRole": string,
      "additionalRoles": [
      string
      ]
    }
    ]
  }
  ],
  "features": [
  {
    "featureName": string,
    "featureRate": double
  }
  ],
  "maxUploadSizes": [
  {
    "type": string,
    "size": long
  }
  ],
  "isCurrentAppInstalled": boolean,
  "languageCode": string,
  "folderColorPalette": [
  string
  ]
}
```

Listing A.3: About resource representation

### A.2.2 Files resource

```
{
  "kind": "drive#file",
  "id": string,
  "etag": etag,
  "selfLink": string,
  "webContentLink": string,
  "webViewLink": string,
  "alternateLink": string,
  "embedLink": string,
  "openWithLinks": {
    (key): string
  },
  "defaultOpenWithLink": string,
  "iconLink": string,
  "thumbnailLink": string,
  "thumbnail": {
    "image": bytes,
    "mimeType": string
  },
  "title": string,
  "mimeType": string,
  "description": string,
  "labels": {
    "starred": boolean,
    "hidden": boolean,
    "trashed": boolean,
    "restricted": boolean,
    "viewed": boolean
  },
  "createdDate": datetime,
  "modifiedDate": datetime,
  "modifiedByMeDate": datetime,
  "lastViewedByMeDate": datetime,
  "markedViewedByMeDate": datetime,
  "sharedWithMeDate": datetime,
  "version": long,
  "sharingUser": {
    "kind": "drive#user",
    "displayName": string,
    "picture": {
      "url": string
    },
    "isAuthenticatedUser": boolean,
    "permissionId": string,
    "emailAddress": string
  },
  "parents": [
  parents Resource
  ],
  "downloadUrl": string,
  "exportLinks": {
    (key): string
  },
  "indexableText": {
    "text": string
  },
  "userPermission": permissions Resource,
  "permissions": [
  permissions Resource
  ],
  "originalFilename": string,
  "fileExtension": string,
  "md5Checksum": string,
  "fileSize": long,
  "quotaBytesUsed": long,
  "ownerNames": [
  string
  ],
  "owners": [
  {
    "kind": "drive#user",
    "displayName": string,
    "picture": {
      "url": string
    },
    "isAuthenticatedUser": boolean,
    "permissionId": string,
    "emailAddress": string
  }
  ],
  "lastModifyingUserName": string,
  "lastModifyingUser": {
    "kind": "drive#user",
    "displayName": string,
    "picture": {
      "url": string
    },
    "isAuthenticatedUser": boolean,
    "permissionId": string,
    "emailAddress": string
  },
  "editable": boolean,
  "copyable": boolean,
  "writersCanShare": boolean,
  "shared": boolean,
  "explicitlyTrashed": boolean,
  "appDataContents": boolean,
  "headRevisionId": string,
  "properties": [
  properties Resource
  ],
  "folderColorRgb": string,
  "imageMediaMetadata": {
    "width": integer,
    "height": integer,
    "rotation": integer,
    "location": {
      "latitude": double,
      "longitude": double,
      "altitude": double
    },
    "date": string,
    "cameraMake": string,
    "cameraModel": string,
    "exposureTime": float,
    "aperture": float,
    "flashUsed": boolean,
    "focalLength": float,
    "isoSpeed": integer,
    "meteringMode": string,
    "sensor": string,
    "exposureMode": string,
    "colorSpace": string,
    "whiteBalance": string,
    "exposureBias": float,
    "maxApertureValue": float,
    "subjectDistance": integer,
    "lens": string
  },
  "videoMediaMetadata": {
    "width": integer,
    "height": integer,
    "durationMillis": long
  }
}
```

Listing A.4: Files resource representation

### A.2.3 Children resource

```
{
  "kind": "drive#childReference",
  "id": string,
  "selfLink": string,
  "childLink": string
}
```

Listing A.5: Children resource representation

### A.2.4 Revisions resource

```
{
  "kind": "drive#revision",
  "etag": etag,
  "id": string,
  "selfLink": string,
  "mimeType": string,
  "modifiedDate": datetime,
  "pinned": boolean,
  "published": boolean,
  "publishedLink": string,
  "publishAuto": boolean,
  "publishedOutsideDomain": boolean,
  "downloadUrl": string,
  "exportLinks": {
    (key): string
  },
  "lastModifyingUserName": string,
  "lastModifyingUser": {
    "kind": "drive#user",
    "displayName": string,
    "picture": {
      "url": string
    },
    "isAuthenticatedUser": boolean,
    "permissionId": string,
    "emailAddress": string
  },
  "originalFilename": string,
  "md5Checksum": string,
  "fileSize": long
}
```

Listing A.6: Revision resource representation

# Vita

Andres Barreto was born in Portoviejo, Ecuador on November 29th 1988. He received his Bachelor degree in Computer Science from Escuela Superior Politécnica del Litoral (ESPOL) in 2006. He worked as a Webmaster for three years after obtaining his bachelor degree. Currently he is pursuing his master's in the field of Information Assurance, under the Computer Science Department, at University of New Orleans, Louisiana, United States of America.