CATCH: Cloud Data Acquisition through Comprehensive and Hybrid Approaches

By:
Jihyeok Yang, Jieon Kim, Jewan Bang, Sangjin Lee and Jungheum Park

From the proceedings of
The Digital Forensic Research Conference
DFRWS APAC 2022
Sept 28-30, 2022

Forensic Science International: Digital Investigation 43 (2022) 301442

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

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

DFRWS 2022 APAC - Proceedings of the Second Annual DFRWS APAC

CATCH: Cloud Data Acquisition through Comprehensive and Hybrid Approaches

Jihyeok Yang [a, 1], Jieon Kim [a, 1], Jewan Bang [b], Sangjin Lee [a], Jungheum Park [a, *]

[a] School of Cybersecurity, Korea University, 145 Anam-Ro, Seongbuk-Gu, Seoul, South Korea
[b] Cyber Investigation Bureau, National Office of Investigation, Korean National Police Agency, Seoul, South Korea

ARTICLE INFO

Article history:

Keywords:
Digital forensics
Cloud forensics
Online data
Selective data collection
Web APIs

ABSTRACT

With the development of Internet technology, cloud-based services have improved the availability and usability of resources. Among them, cloud storage services enable users to remotely store, access, or share data over a network. Therefore, digital forensic investigators need to collect data stored in remote servers to comprehensively understand a suspect's activities. Although several well-known commercial digital forensic tools provide features for cloud data acquisition in order to support this requirement, fewer studies have addressed whether they have full access to cloud resources and collect all the data as expected. In this regard, our findings from this work show that those commercial tools do not completely identify and collect data that are obviously available through dedicated clients (e.g., web-browsers and desktop/mobile apps). In this paper, we propose an investigative framework, CATCH (Cloud Data Acquisition through Comprehensive and Hybrid Approaches), which is composed of four steps (Authentication, Exploration, Filtering, and Collection). CATCH collects authentication data to access cloud resources and then, explores, filters, and collects all accessible metadata as well as contents from remote cloud servers by using *Open* and *Internal* APIs. To demonstrate our proposal, the CATCH framework is applied to collect a user's Microsoft OneDrive storage from digital forensics perspectives. We then evaluate data collection results generated from a self-developed tool based on the proposed framework, by comparing them to results from commercial digital forensic tools.

1. Introduction

Cloud storage has recently become a popular alternative to local storage because of its large capacity and ease of synchronization across multiple devices. Market Insights Reports (U. $3900, 2021) predicted the global cloud storage market would reach approximately $33,730 million in 2020 to $122,490 million by 2027. Cloud forensics is becoming increasingly important in digital forensics as cloud storage services gradually replace local storage. To collect data stored in cloud storage, forensic investigators can ask for the cooperation of relevant Cloud Service Providers (CSPs) (Farina et al., 2015). Without their cooperation, the investigators need to collect those data in local side. Quick and Choo confirmed that downloading files from cloud storage to local computers did not compromise their integrity (Quick and Choo, 2013a). Therefore, when CSP does not cooperate, it is necessary to perform online data collection at local side if obtaining appropriate user credentials for target cloud services is possible (Martini and Choo, 2012).

1.1. Motivation

Cloud storage generally provides features similar to local storage, and offers several special features for various purposes, including collaboration, productivity, and security. For example, Google Drive allows multiple users to share files and collaborate through a 'Shared Drive.' Microsoft OneDrive's 'Personal Vault,' which requires secondary authentication, allows users to manage data securely. Dropbox provides the ability to manage files separately by locating some of them in the 'Starred.' BOX also supports a collaboration feature between users through 'Box Notes.' However, existing cloud forensic tools that heavily rely on open APIs officially provided by service providers cannot acquire all the stored data, especially those managed by the above-mentioned special features.

* Corresponding author.
E-mail addresses: piki@korea.ac.kr (J. Yang), kijie@korea.ac.kr (J. Kim), jwbang@police.go.kr (J. Bang), sangjin@korea.ac.kr (S. Lee), jungheumpark@korea.ac.kr (J. Park).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.fsidi.2022.301442

Unintended omissions or inaccuracies during data collection can prevent a thorough investigation from a digital forensics perspective (Zawoad et al., 2015). Therefore, identifying and collecting all available data is essential.

Several existing forensic tools with cloud data acquisition feature rely mainly on open APIs provided by cloud-based services. As of April 2022, Oxygen Forensics Detective (OFD - version 4.0.0.1367) supports data collection on approximately 90 cloud-based services, including seven cloud storage services. Our tests show that OFD attempts to access data using open APIs with OAuth 2.0 authentication for cloud storage services. Magnet AXIOM (version 5.10.0.30634) supports the analysis of 16 cloud-based services including, seven cloud storage services. Similar to OFD, AXIOM also uses open APIs with OAuth 2.0. Both OFD and AXIOM analyze data in the cloud storage service after downloading all files present in the cloud storage service and storing them as compressed files. Finally, Cellebrite UFED Cloud (CUC - version 7.49.0.28) supports cloud forensics for 55 cloud services, including six cloud storage services. Testing with CUC shows that it employs additional methods different from the previously mentioned tools, which only use open APIs.

Existing cloud forensic tools cannot identify and collect all the data stored in cloud storage. AXIOM and OFD, relying on open APIs, could not collect data stored in special features. CUC could collect cloud resources more than AXIOM and OFD, but it still does not collect data in particular areas properly. Additionally, the metadata of files identified from commercial tools is insufficient than those manually identified through dedicated clients. If the tools cannot guarantee the accuracy and completeness of data collection, this reduces the reliability of digital forensic activities.

1.2. Contribution

This study makes the following contributions:

- Identifying available methods to collect cloud resources through understanding operations of dedicated clients officially provided by service providers and several commercial forensic tools
- Proposing an investigative framework using *open* and *internal* APIs to collect cloud resources as thoroughly as possible
- Demonstrating the usefulness of the proposed framework by applying it to Microsoft OneDrive as a case study

The remainder of this paper is structured as follows: Section 2 introduces related works on collecting metadata and contents from cloud-based services. Section 3 introduces a new framework to collect cloud resources using open APIs and internal APIs. Section 4 describes cloud resources that can be collected or cannot be collected using APIs for Microsoft OneDrive. Then, we collect data stored on Microsoft OneDrive by applying CATCH framework. Section 5 compares the usefulness of CATCH framework with those of other cloud forensic tools. Finally, Section 6 outlines expectations and limitations of the proposed framework.

2. Background and related works

2.1. Background

We propose a new framework that combines open and internal APIs to collect data stored in cloud storage. This section provides a brief description of those APIs.

2.1.1. Open API

An open API, which is also known as a public API, is a programming interface that a service provider has released for users. Therefore, a developer or forensic investigator can use open APIs to implement various functions to upload, download, or modify cloud-based data. Cloud storage services such as Google Drive, Microsoft OneDrive, Box, and MEGA, provide open APIs. The functions of open APIs depend on what each cloud-based service defines. In general, open APIs require OAuth 2.0 authentication (Hardt, 2012).

Many cloud forensic tools use open APIs that allow forensic investigators to collect data easier, however, they can only access limited areas defined by service providers. Therefore, investigators can miss significant digital evidence when they collect data using open APIs.

2.1.2. Internal API

Compared to the open API, an internal API is not open to users in public. It is also called an unofficial API or a private API, that can provide more functions than an open API. In other studies, analyzing internal APIs has been already applied to collect data stored on cloud storage generated from IoT devices (Kayode and Tosun, 2019; Wu et al., 2021). Requests for cloud resources (e.g., file list, file download, and thumbnail) are sent to the servers using GET and POST methods. Analyzing responses received in multiple formats (e.g., JSON, GZIP, HTML, and Javascript) helps investigators access and collect data where open API cannot. Furthermore, forensic investigators can collect cloud data using internal APIs even though service providers do not support open APIs.

This paper examines how cloud servers and clients communicate over a network using web debugging proxy tools and analyzes internal APIs for a thorough forensic analysis. We also explore whether internal APIs can access the data where open APIs cannot.

2.2. Related works

Chung et al. proposed an integrated digital forensic process to investigate cloud storage services (Chung et al., 2012). They analyzed local forensic artifacts on PC and mobile devices. The authors emphasized obtaining a user ID and password to collect cloud resources from cloud servers. This paper assumes that a user ID and password are collected before acquiring cloud resources.

Quick and Choo analyzed digital forensic artifacts left behind in multiple locations (e.g., log files, memory, and networks) using Microsoft Skydrive (Quick and Choo, 2013b). The authors identified a parameter named ‘cid’ and an URL (https://skydrive.live.com/?cid={UserId}) by conducting a memory forensic inspection. The URL can be obtained when a user logs in to Skydrive. To obtain ‘cid’ from the URL is meaningful because it specifies a user who is not identifiable. Similarly, another study analyzed a method to acquire cloud data from Google Drive (Quick and Choo, 2014). They also found a particular URL format to access files in Google Drive as follows: https://docs.google.com/file/d/{ResourceID}/. They figured out Resource-identifier (Resource ID) for each file from the client-side artifacts. The resource ID can be used to access each file when using open and internal APIs and to acquire metadata and contents in the cloud server.

Roussev et al. proposed kumodd, kumodocs, and kumofs, open-source cloud forensic tools to acquire cloud resources (Roussev et al., 2016). The authors used open API to acquire cloud resources from Google Drive, Dropbox, MS OneDrive, and Box and internal API to acquire the editing history of Google Docs. However, open API does not always support collecting all the cloud resources. Their study did not address the details on how to collect all available and accessible cloud resources thoroughly, which generally requires to use different sets of parameters and/or to use additional authentication information. The kumodd tool also provides a

filtering option using open API, but they did not give the metadata range supported by the API. The study does not conduct a comparative study on collected results between their proposed tools and the existing commercial forensic tools. From their perspective of using APIs, we propose a systematic framework using APIs to acquire cloud resources thoroughly providing detailed information regarding requests and responses. We also identify filtering ranges from calling APIs and develop a tool to allow filtering based on both APIs and collected metadata.

Han et al. researched a selective acquisition using OneDrive and Dropbox as a case study (Han et al., 2020). The study used open APIs that cloud-based services provide. The authors analyzed to get authentication and acquired file lists from the cloud-based services. Then, they performed searching and collecting for a target file using metadata. However, cloud resources that an investigator can access through open APIs are limited to the scope set by a service provider. Thus, the authors will have difficulties acquiring all the cloud resources if a service provider gives limited permissions.

Chung et al. analyzed unofficial APIs and collected data for Amazon Alexa, where open APIs are not supported (Chung et al., 2017). They performed an intensive traffic analysis using web proxy tools. They also found that most network traffic was sent over an encrypted connection and returned in JSON-formatted response. Youn et al. developed an existing method to analyze unofficial APIs used by Amazon Alexa and applied it to Echo Show 2nd (Youn et al., 2021). We also examine internal APIs, then propose a new methodology to collect data from cloud-based services.

Jo et al. conducted a study on AI speaker ecosystems through the case study of Naver Clova (Jo et al., 2019). The authors analyzed how the AI speaker and cloud server communicate with each other. This study captured a network packet from a laptop as an access point between an Android mobile application and the AI speaker. The authors found that packet data contains meaningful data, including user accounts and access tokens, which are useful for digital forensic investigation. As many cloud-based services provide their services in mobile applications, it is meaningful to analyze HTTP/HTTPS data between the companion mobile app and the IoT device.

Hilgert et al. collected micro-mobility data from a user's phone and a cloud server (Hilgert et al., 2021). This study collected cloud data using APIs for mobility applications, Lime and TIER. are obtained from cloud servers using APIs. Because user authentication information is essential to call the APIs, they acquired a 'bearer token' stored on the user's phone. From the mobile device, they could recover various information such as 'payment data' and 'ride history.' The authors also visualized riding paths using GPS data from the cloud server. If user credentials are not available, they suggested obtaining the access token remained on mobile devices. They then send requests for cloud data using APIs.

Cloud forensic research on local artifacts for several cloud storage services is actively being studied. Daryabar et al. analyzed forensic artifacts by performing several activities (e.g., installation, login, download, and sharing) on Android and iOS devices for a cloud storage service, MEGA (Daryabar et al., 2017). They analyzed network packets using the iOS device and acquired forensic artifacts of interest. Chen et al. analyzed Windows and Android artifacts utilizing three popular cloud storage providers in China (BaiduNetDisk, WeiYun, and 115yun) (Chen et al., 2020). However, these studies focused on forensic artifacts remaining on local devices. Cloud forensics using open and internal APIs has not been studied in depth.

3. CATCH: cloud data acquisition through Comprehensive and Hybrid Approaches

We suggest a new framework called CATCH to collect cloud resources more thoroughly using open and internal APIs. Fig. 1 illustrates the CATCH framework, structured in four steps: (1) Authentication, (2) Exploration, (3) Filtering, and (4) Collection.

3.1. Open API

We explain how to collect cloud resources through open APIs. Table 1 summarizes whether cloud storage services provide open APIs or not. As described in Section 2.1.2, open APIs have different functions for each cloud storage service. Therefore, it is necessary to identify the data categories that can be accessed using open APIs for cloud storage services.

3.1.1. Authentication

Cloud storage services providing open APIs mostly use OAuth 2.0 for authentication. The OAuth client can have a connection with the cloud server with minimal permission, such as 'ReadOnly.' Authentication begins when trying to log in with user credentials. There are two different methods to obtain an authorization code. One approach is to obtain the code from an authenticated webpage. The second method is to obtain the code from the 'Redirect URI' assigned by the OAuth Client. Then, the client uses the code to exchange an access token, allowing access to cloud resources, as illustrated in Fig. 2. If authentication is successful, the process proceeds to the exploration step.

3.1.2. Exploration

Open APIs can be called for exploring cloud resources with a valid access token. These APIs are used with special request parameters supported by each service provider, in order to perform getting a list of files, getting metadata of a file, downloading thumbnails of a file, and so on. When cloud-based services receive those API requests, they send responses in a standardized form, such as JavaScript Object Notation (JSON), and sometimes return additional data in other web-related file formats such as HTML and JavaScript.

Open APIs help users to access cloud resources more efficiently. However, cloud storage service providers do not offer full functionality through open APIs. Using open APIs could not explore the data completely, which may require additional investigation. On the other hand, the acquired file lists and metadata have various formats for each cloud-based service and data category. The acquired metadata may be required normalization.

3.1.3. Filtering

Filtering step scans explored metadata to search for files and folders that match a specific condition.

Several cloud-based services provide a function for searching files using metadata, such as file names, dates, and extensions. The filtering step sends requests of the URLs and search queries, including keywords or a specific time to the server. The server then returns the search results. It is possible to search for files through the normalization of explored metadata in the exploration step for the cloud-based services that do not provide search-related APIs. When searching metadata is completed, it moves to the collection step.

3.1.4. Collection

We can download each file at the collection step using open APIs. Cloud storage services provide a function for downloading files or folders through open APIs. Most of the services support file downloads using file ID as one of the required parameters. The collection step is complete when downloading files through open APIs is accomplished.

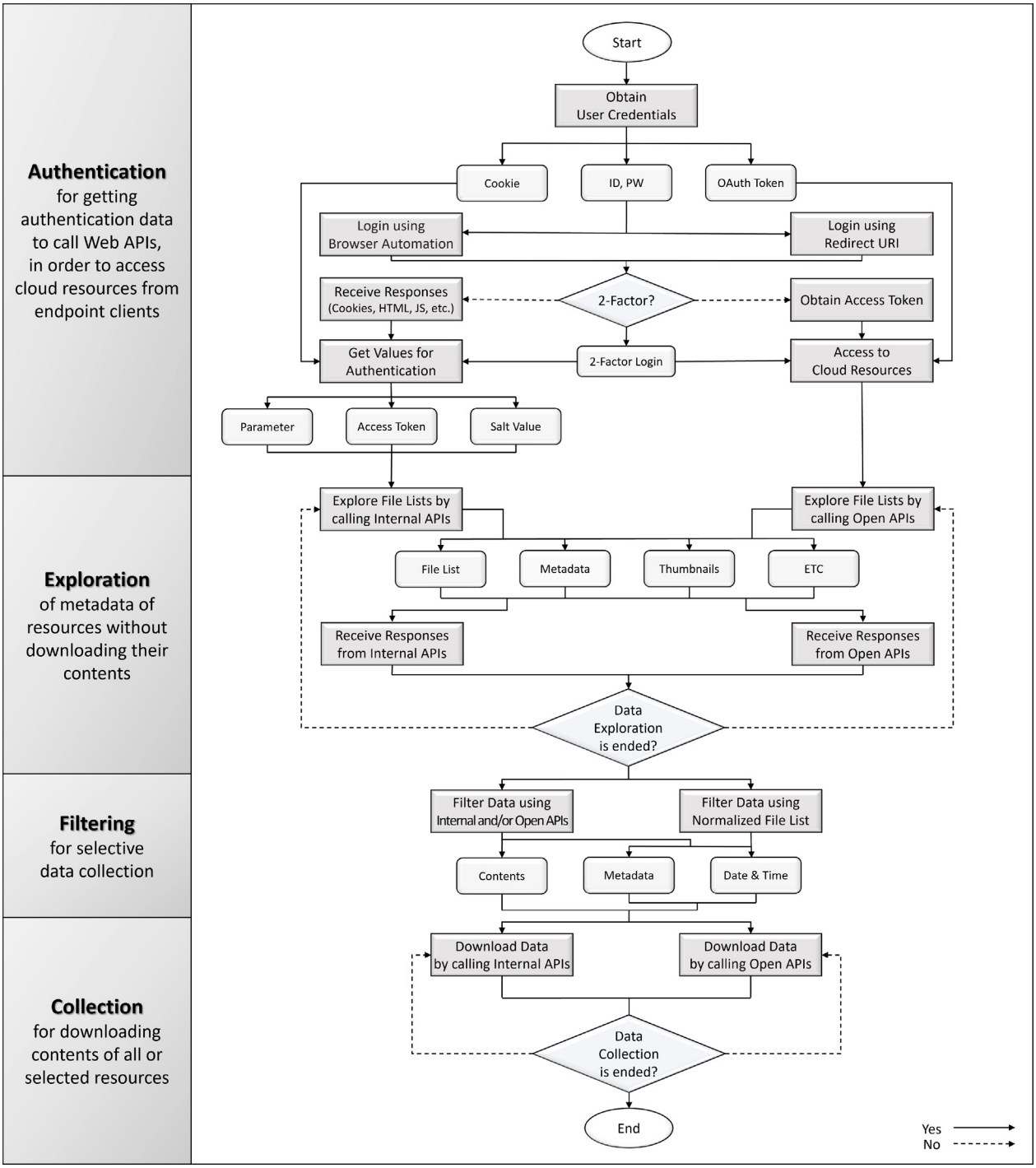

Fig. 1. CATCH: A four-step framework for selectively collecting cloud data through both *open* and *internal* APIs.

3.2. Internal API

We explain how to collect data stored on cloud storage using internal APIs. Cloud storage services use an internal API to provide services through a dedicated client (website or application). Additionally, they should use private APIs to operate their services internally. Thus, identifying and sending requests through internal APIs allows access to cloud resources that open APIs and existing cloud forensic tools are unable to access.

3.2.1. Authentication

The cloud server verifies the user ID and password entered by the user. If the user credentials are verified, authentication information is generated and sent to the client, permitting access to cloud resources. Therefore, it is necessary to collect authentication data to obtain cloud resources. Web-based services store authentication information, such as cookie values on the local side and enable communication until the authentication data expires.

To collect authentication information, we can attempt to log in to the website using the user credentials obtained in advance. We

Table 1
Summary regarding open APIs for nine cloud storage services: O means *supported*, X means *not supported*.

Service Name	Service Provider	Country	Open APIs
Amazon Drive	Amazon	USA	X
BOX	BOX	USA	O
Dropbox	Dropbox	USA	O
Google Drive	Google	USA	O
iCloud	Apple	USA	O
MEGA	MEGA	New Zealand	O
My Box	NAVER	South Korea	X
OneDrive	Microsoft	USA	O
pCloud	pCloud	Switzerland	O

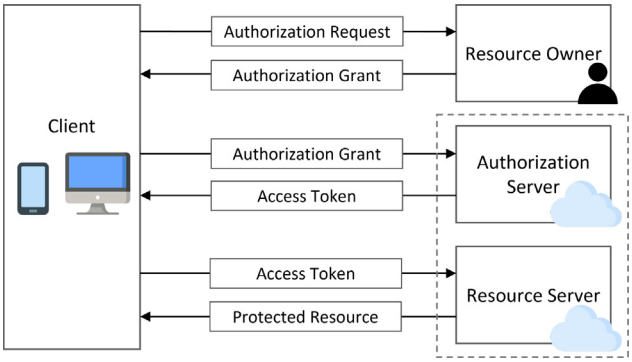

Fig. 2. Workflow of the OAuth 2.0 authentication.

automate the login process using a browser automation open framework (e.g., Selenium and Playwright). This is achieved by analyzing the HTML login pages for each cloud storage service to classify tags related to user ID and passwords. Additional HTML tag analysis is required if a 2-factor authentication is turned on. When login is successful, clients obtain authentication information, including cookie values.

3.2.2. Exploration

The exploration step observes cloud resources after obtaining authentication data including cookies from the server. We can access and explore all data categories that cloud-based services offer using internal APIs.

All files and their metadata accessible by dedicated clients (e.g., web-browser) can be obtained through special request URLs internally used by the clients (= internal APIs). To utilize these internal APIs properly, it is necessary to deeply analyze the request URLs (including detailed parameters) and cookies required to access cloud resources. Each request URL along with required parameters and cookies can access the cloud data. Similar to a situation where an open API is used, cloud services generally send responses to an internal API request in standardized formats such as JSON, HTML, or JavaScript.

We collect all information displayed on the web-browser using internal APIs during the exploration step. We can also collect metadata that cannot be collected by using open APIs. Furthermore, data not displayed on the web-browsers can be discovered.

As described in Section 3.1.2, a normalization process is necessary because the types of metadata obtained for each data category, are not always identical.

3.2.3. Filtering

Some cloud storage services provide a search function on web-browsers or applications. The search function queries the server using the requested URLs and receives the searched result as a response. Even if cloud-based services do not provide a search function from an internal API, it is possible to search data using normalized metadata. The normalized metadata enables to search for files that match a keyword on the file name, metadata, and time. Additionally, files can be selected using the thumbnail of the file.

3.2.4. Collection

The primary purpose of this step is to collect data (all available resources) using metadata (e.g., file's properties, thumbnails, etc.) acquired from the previous steps. Cloud storage services generally manage URLs for accessing individual resources on remote servers, so that users can directly download their resources through the URLs returned as an element of metadata in the exploration step. In some cases, explicit download URLs are not identified from the metadata. In this situation, there may be a unique ID for individual resource, and it can be used as a required parameter for calling an internal API predefined for the download operation.

4. MS OneDrive as a case study

This section comprehends how a user can manage files and folders in OneDrive and how contents and associated metadata can be collected. The implemented system was in a Windows 10 × 64 environment. We developed a tool[2] using Python 3.7 to automate data collection. Fig. 3 shows the process of collecting data stored in OneDrive following the CATCH framework.

The process of the case study is as follows:

- Investigate how Microsoft OneDrive manages files and folders in the cloud server
- Identify authentication information and request URLs to collect files using open and internal APIs
- Compare collected files from using open and internal APIs in the following steps: authentication, exploration, filtering, and data collection

4.1. How OneDrive organizes files and folders

To collect the most complete data stored in a drive, we must understand how OneDrive manages files and folders. This helps a practitioner to determine which data category should be collected using open and internal APIs. The categories where OneDrive constructs files and folders are listed in Table 2.

OneDrive has six categories including 'My Files,' 'Recent,' 'Photos,' 'Shared,' 'Recycle Bin' and 'Personal Vault' to store files and folders. To access 'Personal Vault' requires verifying additional 2-Factor authentication (e.g., SMS and E-mail). Investigators need to collect all data stored in the categories, so that they will not miss digital evidence.

4.2. Authentication

The authentication step describes how to get authentication to access OneDrive. As mentioned in Section 2.2, processes using open and internal APIs begin with the assumption that investigators have already collected user credentials.

4.2.1. Open API

To obtain authorization to cloud storage, OneDrive's open APIs use OAuth 2.0 for authentication. To obtain authentication, we created a client application to use open APIs. We also acquired

[2] https://github.com/dfrc-korea/CATCH_OneDrive.