국제학석사 학위논문

API-Based Cloud Data Acquisition and Analysis from Smart Home IoT Environments

스마트홈 IoT 환경에서의 API 기반 클라우드 데이터 획득 및 분석

Brhan, Yohannes Yemane (브라한 요하네스 예멘)

International Studies (국제학과)

Legal Informatics and Forensic Science (정보법과학전공)

한림대학교 대학원

Graduate School, Hallym University

국제학석사학위논문

API-Based Cloud Data Acquisition and Analysis for Smart Home IoT Environment

2019

Brhan, Yohannes Yemane

국제학석사 학위논문

API-Based Cloud Data Acquisition and Analysis from Smart Home IoT Environments

스마트홈 IoT 환경에서의 API 기반 클라우드 데이터 획득 및 분석

Brhan, Yohannes Yemane (브라한 요하네스 예멘)

International Studies (국제학과)

Legal Informatics and Forensic Science (정보법과학전공)

한림대학교 대학원

Graduate School, Hallym University

장윤식, Joshua I. James 교수지도

국제학석사 학위논문

브라한 요하네스 예멘의 석사학위논문을 합격으로

판정함

2019 년 6 월 28 일

심사위원장 박노섭

심사위원 안정민

심사위원 장윤식

심사위원 Joshua I. James

Table of Contents

List of Figures

List of Tables

List of Listings

CHAPTER 1. INTRODUCTION

1.1. Background

IoT analytics [1] defines the Internet of things as "sensors and actuators embedded in physical objects are linked through wired and wireless networks, often using the same Internet Protocol (IP) that connects the Internet." The word "Internet of Things" has coined by Kevin Ashton in 1999 to describe the network connecting objects in the physical world to the Internet. Nowadays, these IoT technologies cover all aspects of our life from managing our home temperature to smart cars and smart management of the cities and power grids. The statistics portal said that the Internet of Things (IoT) is a promising concept of the technology industry for the coming years [2]. By 2019, the global IoT market is forecast to be valued at more than 1.7 trillion U.S. dollars. By 2020, there will be 30B connected devices. Connected cities, hubs that use information and communication technology, and connectivity to address urban problems, remain the most important IoT subsystems to be used.

The already existing application domains are turning into IoT technologies. Objects and application domain that are not of themselves smart are becoming "Smart" with the introduction communication capabilities through the use of technologies such as RFID, sensors, and the Internet as shown in Figure 1. In the context of smart homes, *Oriwoh* defined Smart Home (SH) as an intelligent, networked, autonomous abodes or dwelling places and they are set to become some of the most ubiquitous and prevalent smart spaces of the near future [3]. When compared to non-SH, SH has an increased number of (exposed) communication, control, and computing interfaces. Smart Home devices could be an ecosystem which consists of sensors, hub, speakers, mic, companion client applications such as Mobile Apps, web-apps, and so on. Some examples of Smart Home devices are Samsung SmartThings [4], Amazon Echo [5], Philips Hue [6], Nest

Thermostat [7], and Google Home [8]. For most of IoT ecosystems, including SH environment, they depend on their cloud service to perform tasks such as processing, commanding, analytics. In this thesis, we will analyze and investigate Samsung SmartThings, Amazon Alexa and Google Home as a case study to demonstrate the approach in acquiring cloud service data using their respective APIs. We will discuss the detail of each device and the results in the case study section.

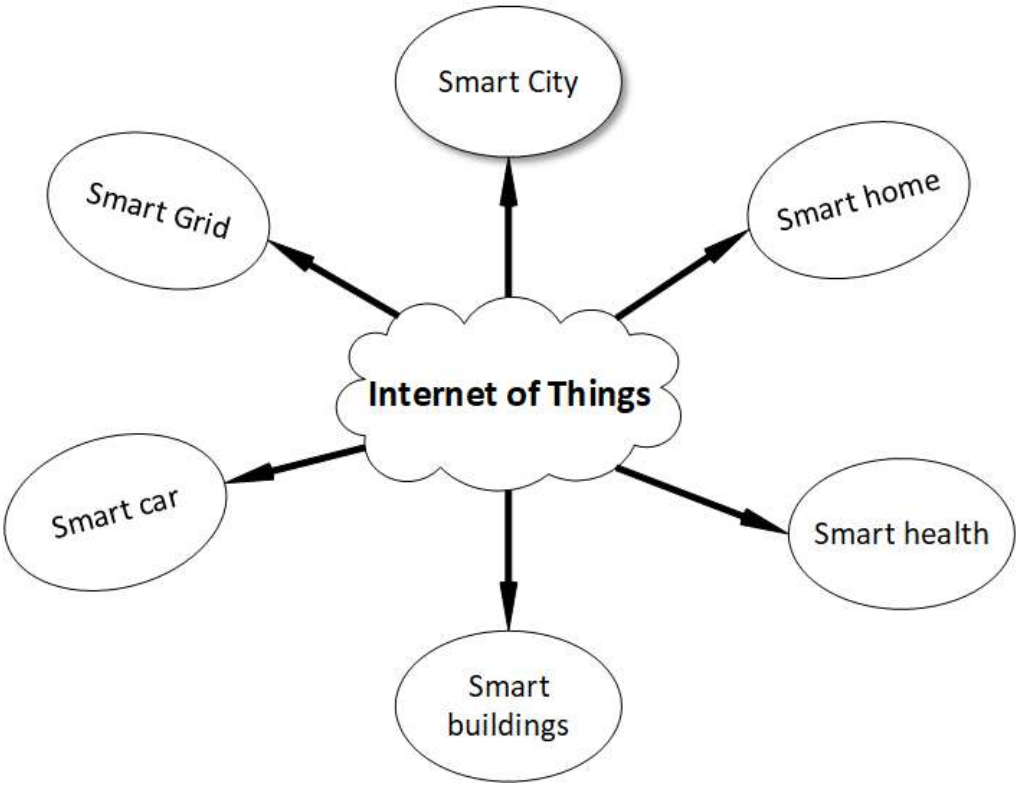

Figure 1: Some of IoT application domains.

Cloud computing enables accessing data and programs from a centralized pool of computing resources that can be ordered and consumed on demand [9]. IoT devices generate massive amounts of data in gigabytes, and cloud computing platforms provide storage and computation capability for this data. Which means, IoT environments, including its sensors and devices, collect data and perform an action, then, the processing, commanding, and analytics happens in the cloud. For instance, in perspective of smart home IoTs, the sensors in home collect and send data to the cloud such as the status of the door, temperature, and motion, and then the cloud compute what

the data received mean and send data/notification to the app/user or a command to another device in the house. In digital forensics perspective, these cloud data from smart home devices play a significant role for investigators and law enforcement agencies.

IoT devices produce huge data (in petabytes or gigabytes) which, in turn, could be a potential data source for crime investigation. Like the traditional digital forensics, we can acquire data in the IoT ecosystem from the client-side, network or cloud. However, unlike the computer forensics, IoT forensics does not contain much data in client-side. The device's storage is limited, and the companion applications do not save much data except some cache and configuration files. Because of the limitation on client-side data, cloud-centric data is becoming the main source of IoT-related environments.

Furthermore, Cloud storage is the head of IoT-related environments where the smart stuff is happening such as computation, analytic or commands. This cloud data can be accessed through interfaces such as browser or client applications using user authentication methods, or through collaboration with the Cloud Service Provider to collect the data. Law enforcement is normally following the latter option, which is requesting cloud service providers to give user data [10]. However, additional methods to acquire of cloud-centric data are crucial because of the challenge in international collaboration [10].

1.2. Recent IoT Device Court Cases

Law enforcement agencies and investigators are using data from IoT devices when available on the crime scene [11]. Also, in the future, the increase in the IoT devices in combination with 5G and AI will change the digital forensics landscape [12]. Below are some of the past criminal cases where evidence obtained from smart home devices used in court.

An individual's Fitbit – a wearable activity tracker that measures things like steps, sleep, and calories – used in sexual assault allegations [13]. Specifically, the Fitbit's activity logs were able

to show that the individual was, in fact, awake and also walking around, as the device contained an accelerometer with continuous logging when they had originally claimed that she was asleep. The court ruled that the individual had fabricated the incident based on this finding. In another case, police use Fitbit data to charge a 90-year-old man, Anthony Aiello, with killing his step-daughter killing [14]. Mr. Aiello visited his stepdaughter with homemade pizza and biscotti to her house in San Jose, Calif. He told investigators that she then walked him to the door and handed him two roses in gratitude. However, the Fitbit fitness tracker she wears told otherwise. The data from the Fitbit showed that her heart rate had spiked significantly around 3:20 p.m. on Sept. 8, when Mr. Aiello was there. Then it recorded her heart rate slowing rapidly, and stopping at 3:28 p.m., about five minutes before Mr. Aiello left the house. In 2017, a man, James Bates, was accused of killing a former police officer [15]. In support of the investigation, the prosecutor requested Amazon to release voice recordings from the suspect's Amazon Echo device. Since the device stores all of its recordings on Amazon's cloud, the investigators made a formal request to the company for the data. However, Amazon agreed to give the records to the prosecutor until after the owner permitted to release his device's recordings.

1.3. Thesis Contribution

The use of IoT devices in an investigation, shown in the above case, could indicate the importance of acquiring evidence from smart home IoT devices in a criminal investigation and the subsequent legal proceedings. This thesis will focus on the acquisition of cloud-native artifacts from smart home IoT devices.

The main contribution of this thesis is the procedure for cloud acquisition from Smart Home IoT device's cloud services using APIs — official or/and unofficial. IoT ecosystems use pre-defined communication APIs (Application Programming Interface) to send and receive data. Then, using these cloud APIs [16], cloud-native data will be acquired. We will investigate three

popular smart home devices and its cloud services (Amazon Echo/Alexa, Google Home, SmartThings) as a case study, and acquire their cloud data. We also develop an open source tool which can acquire cloud-centric data from each the selected case study IoT-related cloud service providers using the APIs uncovered during the research. Overall, the following are the contribution of this thesis:

- We will uncover unofficial APIs which the smart home IoT environment use to communicate with their respective cloud.
- We will use these unofficial APIs, in combination with the official APIs if provided, to obtain cloud-native evidence.
- Based on our research, we also present an open source tool that can acquire cloud artifacts from three (3) smart home device's corresponding cloud services, namely Amazon Alexa, Google Assistant, and SmartThings cloud.
- Finally, we evaluate the completeness of the acquired data with the data generated from individual smart home devices in a controlled environment.

1.4. Research Questions

Data generated by the IoT devices can be stored on the IoT device itself and the cloud service. This data is, in turn, synced and viewed by users usually through client applications such as mobile devices. Anyone of these data can be the subject to digital forensics. This research aims to acquire and analyze cloud-native data for Smart Home IoT-related cloud service.

The thesis aims to address the following research questions:

RQ1: Can data be collected in a forensically-sound way from cloud service data from IoT devices related to the smart home environment using APIs?

With this research question, we will discuss methods, techniques, and strategies to get cloud-related data for smart home IoT devices using APIs.

RQ2: **How much data can we obtain in terms of completeness?**

With this research question, we will investigate if the acquired data is full and correct by comparing it with the data generated.

As a case study, we will investigate three IoT-based cloud providers and acquire their cloud-native data. Finally, we will develop a proof-of-concept tool that can download and parse artifacts from these cloud service.

1.5. Thesis Structure (Outline)

Chapter 1. Introduction – provides an overall introduction and background of IoT, cloud service, and its forensic purpose. In addition, it discusses the problem statement (research questions), court cases, and the overall contribution of the thesis.

Chapter 2. Literature Review – presents some of the previous works conducted in the area of cloud forensics and IoT Forensics.

Chapter 3. Methodology – this chapter covers a list of tools and devices used on this thesis, environment setup for the research, data generation, and collection methods, and lastly the process we will follow to conduct the research.

Chapter 4. Technical Aspects of API-Based IoT Cloud Forensics – introduces and discusses the data sources for IoT environment, describe the APIs, its types/design and how to extract it, and acquisition of cloud-native data using the APIs.

Chapter 5. Cloud Data Acquisition Tool Design and Implementation – this chapter describes the overall design and implementation of cloud acquisition tool. The chapter also talks the requirements the tool needs to successfully download data from cloud services, how to configure, run, and set parameters, and the tasks performed by the tool while downloading and after download.

Chapter 6. Case Studies – based on the research and the methods introduced in this thesis, this chapter shows the procedure for the acquisition and analysis of cloud-native data using cloud APIs. Three IoT smart home device cloud services are selected as a case study; Amazon Alexa. Google Assistant and SmartThings. The tool design and implemented in chapter 5 also will be used to acquire cloud service data from these devices.

Chapter 7. Conclusion and Future Work – this chapter presents conclusions of the preceding chapters, discusses the research questions, limitations of the research, and discusses potential future work.

Bibliography. List of all cited research and resources.

Appendices. This section contains a list of any additional resources or extracts of APIs, code snippets, and any other materials that are needed to supplement previous chapters.

CHAPTER 2. LITERATURE REVIEW

In this chapter, we will cover researches that are published related to cloud services and IoT environment forensics. These researches help us in determining what has been done on the field, know the limitations on IoT forensics, and then help us setting research direction. For understanding, we categorized the research into cloud forensics which could be client-side or cloud-side, IoT forensics which could be device level, client-side, or cloud-side, and additional general research such as challenges in IoT forensics.

2.1. Cloud Forensics

In September 2015, Forensic Focus ran a "state of forensics" survey with digital forensic practitioners. The results of the survey indicated that cloud forensics was the biggest concert for investigators [17]. Investigators approach the cloud in two ways to access data stored in the Cloud. They can either attempt to access the data themselves by authenticating as the user or work with the Cloud Service Provider to collect data as written by James and Jang [10].

Further, investigators can acquire cloud data remnants from client applications such as mobile application and browser caches. In this section, we will look past research done on cloud forensics in terms of client-side and API-based acquisition. Besides, we will also outline the limitations of client-side artifacts and the advantage of using API to acquire cloud-centric data.

2.1.1. Client-Side Cloud Forensics

Previous works on cloud forensics focus on its remnants on the client application that is used to connect and control the cloud services. However, these client-side cloud forensics has its limitations – discussed in the next section. Here are some client-side cloud forensics and how our thesis will fill the limitation. We will look into API-based cloud acquisition works and map these works into IoT-based clouds.

Chung et al. analyzed four cloud storage services (Amazon S3, Google Docs, Dropbox, and Evernote) in search of traces left on the client system that can be used in criminal cases [18]. They reported that the analyzed services might create different artifacts depending on specific features of the services, and proposed a process model for the forensic investigation of cloud storage services based on the collection and analysis of artifacts of the target cloud storage services from client systems. The procedure includes gathering volatile data from a Mac/Windows system (if available) and then retrieving data from the Internet history, log files, and directories. On mobile devices, they gathered data from Android and iPhone smartphones then check for traces of a cloud storage service in the collected data.

Hale [19] analyzed the Amazon Cloud Drive and discussed 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 installed on a local system which is provided by Amazon. After analyzing the two methods, Hale found artifacts of the interface in the web browser history, and among cached files. He also found application artifacts in the Windows registry, application installation files, and an SQLite database used to keep track of pending upload/download tasks.

Quick and Choo studied Google Drive, Microsoft SkyDrive, and Dropbox to discover the remnants left on computer and iPhone after a user accessed these services from a computer or phone [20]–[22]. In all of their work, they followed the cloud forensic analysis framework they proposed to approach the analysis of these cloud services. Hash analysis, keyword searches, and examining common file location can be used to determine if the client software provided by these cloud services has been used or not. From the Dropbox analysis, the username can be determined from the browser history (Mozilla Firefox, Google Chrome, and Microsoft Internet Explorer).

From Microsoft SkyDrive, username and password can be determined from forensic images and memory capture respectively. Lastly, from Google Drive, username and password were can be determined from the forensic images. Overall, there is a wide range of investigation points for an examiner to determine the use of these cloud services, such as directory listings, prefetch files, link files, thumbnails, registry, browser history, and memory captures.

2.1.2. Limitation of Client-Side Cloud Forensics

The limitations of client-side artifacts are:

- ➢ The client applications may not contain the latest data or may contain incomplete data.
- ➢ Besides, application artifacts are not stored permanently on the clients, and applications store it on local storage as a cache with no guarantees to its completeness or accuracy. Besides, it could be deleted or overwritten at any time.
- ➢ Lastly, client-side may not contain long time historical data. Getting the last seven days or months of data on client application is almost impossible because of the storage limitation on client devices.

With these limitations in mind, we propose the acquisition of IoT-related cloud-centric data using APIs. Data on the cloud could contain complete and latest data, including long term data [23]. Also, APIs could provide direct access to the cloud data as the providers themselves use it for data transportation between the devices and the client applications.

2.1.3. API-Based Cloud-Side Forensics

Digital forensic research on Amazon Echo (Alexa) by [24] shows that potential digital evidence can be obtained from an IoT device's backend cloud service. Based on their research, Chung et al. revealed unofficial APIs that client applications and the cloud use to communicate. They also designed an IoT forensic tool (cloud-based IoT Forensic Toolkit) to acquire forensically-relevant data from Alexa and its ecosystem. Overall, in their research, they demonstrated that it is possible to acquire cloud data using cloud APIs. Using these APIs, they have got user data such as email,

full name, activities performed by the user and the device, general information such as WIFI SSID and password, device Id, and so on.

Roussev et al. in [25] defined two main problems with client-side artifacts. The first is that the client device may not contain a copy of the cloud data locally. The other problem is that the client-side data could be overwritten when new data is available and may not contain the previous revision of the data. In their research, they argued that the only way to fully address these two problems is to utilize the service provider's official API. Roussev et al. developed a cloud drive acquisition tool called “kumodd” which can perform full API-based acquisition of four major cloud providers: Google Drive, Dropbox, Box, and Microsoft OneDrive. The tool can enumerate and download all files associated with one of the above accounts and all of their revisions. It also acquire snapshots of cloud-native artifacts in standard formats, such as PDF, via the API.

Roussev and McCulley [26] extended the above method, official API-based evidence acquisition, to the private communication APIs (unofficial APIs). Because the above method cannot acquire cloud-native artifacts in their original form since they are not part of the official exposed APIs. As a proof of concept, the authors develop a tool called “kumodocs” which acquire Google Document and Slides artifacts. The tool can extract text content for any revision, the embedded images, and drawings, as well as the history of comments from Google Doc.

As described above, cloud data on client-side could be incomplete, overwritten or could be already outdated. Obtaining the cloud-native artifacts from the cloud services addresses these problems. The above works addressed the problem of client-native artifact acquisitions by using APIs as an acquisition method. However, every company has its cloud API structure and type. Therefore, in this thesis, we will uncover and demonstrate how to acquire cloud data using both official and unofficial APIs with selected smart home devices as a case study.

2.2. IoT Forensics

As stated in the introduction section, our thesis focuses on IoT forensics related to their cloud storage. In this section, we will see what the challenges in IoT forensics are, what has been done until now to fill the gaps, what researches are already done directly related to our thesis, and what are the limitations this thesis trying to fill.

2.2.1. Challenges in IoT Forensics

Yaqoob et al. in [27] outlined various factors related to IoT that affects traditional computer forensics. Big IoT data generated by the IoT-enabled environments are one factor which affects the investigation process. Forensics investigators also face difficulties in collection stage because the data are spread across multiple platforms in the IoT environment; data could reside on the edge devices or clouds. The other challenge is the complexity of the computing architecture – IoT devices come in various hardware architecture and different operating system, and this makes it hard to have a generic investigation framework for investigators. Use of proprietary hardware and software – different vendors and various standards – in IoT devices are the last factor the authors mention in their paper.

In [28], Hegarty et al. pointed out four main phases of digital forensics investigation challenges when investigating the IoT environment. Identifying and detecting the presence of IoT systems and a particular user's data are the main challenge faces for digital forensics in the Identification phase. The challenge in the preservation stage is that the volatility of the evidence is more complex in IoT. The data may be stored in the device locally, in which case, the lifespan of the data could be limited before it is overwritten or compressed. Besides, the data could be transferred and used by another device or transferred to the cloud for aggregation and processing. In the analysis stage, the data from the IoT environment will have to consider the temporal dimension. That is the timelines of the IoT creation, modification, and deletion of data. Presenting the

findings of the IoT analysis is difficult because IoT data will undergo aggregation and processing that can alter the structure and the meaning of the data, or limited memory, battery life or network bandwidth may reduce the quality of the data as the author argues. In the end, Hegerty et al. proposed new techniques to overcome challenges discussed above, which include digital preservation orders or warrants to prevent evidence contamination or overwriting in preservation stage, as well as developing digital forensics tools that can handle and bridge the semantic gap in aggregation challenges. The tool would enable calculation and comparison of the granularity of data from different sources.

Conti et al. in [29] wrote major security and forensics challenges within the IoT domain, and then briefly discussed works published in a special issue journal targeting identified challenges. From the forensics perspective, they divide the challenges in three ways. The first challenge is evidence identification, collection, and preservation. They argue that detecting the presence of IoT systems is a challenge because these devices are designed to work passively and autonomously. Even if we identify related data, collecting could be challenging because there is no documented method or a reliable tool to collect residual evidence from the device in a forensically sound manner. The second challenge is that the analysis and correlation of the evidence collected from the IoT environment. A lack of metadata such as temporal information and a large volume of data collected from heterogeneous IoT environments make it challenging to provide an end-to-end analysis. They presented the last challenge from the attack or deficit attribution perspective. Identifying criminal actors or liabilities of involved parties in the case of an incident is challenging in an IoT environment considering the absence of documented methods and forensically sound tools for collection, preservation, and analysis of cyber-physical systems data.

Considering the above research, the main challenges IoT forensics are the limitations of data in the client side, the data lifespan on the devices including on their client applications such as

mobile apps, and the distributed nature of the environment in which data could reside. This thesis tries to address the identification and acquisition of cloud data from the IoT-related cloud environment. An IoT environment uses the cloud as its central hub to sync data between the devices in addition to storage usage. For this reason, acquiring data from the cloud could solve challenges that the digital forensics field face.

2.2.2. Hardware-Level IoT Device Forensics

Clinton et al. surveyed Amazon Echo smart speakers to identify and analyze the attached surface that can be used to acquire data from it [30]. These researchers identified three attack surfaces to access the Amazon Echo device. These are the SD card pinout, JTAG (Joint Test Action Group) and eMMC (embedded Multi Media Card). These surfaces would allow access into the file system of the Amazon Echo firmware that would allow researchers the ability to reverse engineer binaries to find vulnerabilities, scan the device for hard-coded credentials, and much more. From the digital forensic perspective, Hyde and Moran in [31] presented artifacts extracted from Amazon Echo, Echo Dot and Show using the chip-off method. From the extracted data, it is possible to get WIFI connection information form the device information logs, and registration information, which is the detail of the owner.

In this thesis, hardware/device level data extraction and analysis are not considered. Hardware-level extraction will be added in future work.

2.2.3. API-Based Cloud-Side Forensics from IoT Environment

There is not much research on cloud-side data acquisition for a smart home environment. As described in section 2.1.3, Chung et al. in [24] showed how to extract and use unofficial APIs to obtain Amazon echo data from Amazon Alexa cloud. Besides, Hyde and Moran talked about how to utilize the unofficial APIs in acquiring data from Alexa cloud [31]. In this thesis, we will extend the work on Amazon Alexa cloud and present additional devices to show how to obtain cloud data from smart home devices using official and unofficial APIs.

2.2.4. Others (Scenarios and Theory)

Rahman et al. created scenarios and analyzed the forensic relevance of the data collected by sensors as it is written in [32]. In their scenario, they showed that data collected from different Cookies — Cookies are Mother sen.se sensors — can be applied to different investigation cases. Orr and Sanchez in [33] studied Amazon Alexa to know whether data collected by the device has value, what kind of data the law enforcement could get, and determine if it is helpful for their investigation. Besides, the authors reviewed current legal cases of law enforcement seeking to use Echo generated data in an investigation. Finally, they concluded by suggesting that the Amazon Echo generated data have evidentiary value for law enforcement.

Besides the above work, we will analyze the acquired data for the selected IoT-related providers and devices to describe its evidentiary value and show how it can be used by creating a scenario of each environment.

2.3. Research Direction Affected by Related Works

As it is shown in the above works, APIs are already used by investigators and researchers to acquire cloud-centric data from cloud service providers. However, each cloud service providers are different. In other words, even if APIs are used to acquire cloud data from one provider, it is not possible to use that same APIs in other cloud providers. Hence, each cloud service providers should be studied to understand the APIs, the structure and overall data format it provides. As a result, we decided to perform research focusing on acquisition and analysis of IoT-related cloud service providers. The study includes the technical approach for revealing and using private cloud APIs and tool development for cloud acquisition using the identified APIs. As a case study, we will approach and analyze the three most popular platform for IoT Smart Home environment; Amazon Alexa, SmartThings and Google Home devices and its cloud services.

2.4. Summary

Most of the previous works on IoT environment and cloud storage forensic analysis mainly focus on the traditional approach such as acquiring artifacts from the client devices. However, the approach provides limited evidence as the data in the client side is either incomplete, limited or outdated because it is mostly cache and configuration data. Some researches tried to fill the gap by using cloud APIs to acquire data from the cloud. In addition to these works, we will approach our research in uncovering these APIs and using it to acquire cloud data for IoT environments and devices.

CHAPTER 3. METHODOLOGY

This section documents the procedure, tools and devices used in this thesis. Also, it describes the research environment, which includes setting up the target devices and supporting infrastructure. Lastly, acquisition methods and data generation will be discussed. Figure 2 illustrates the general research methodology and process.

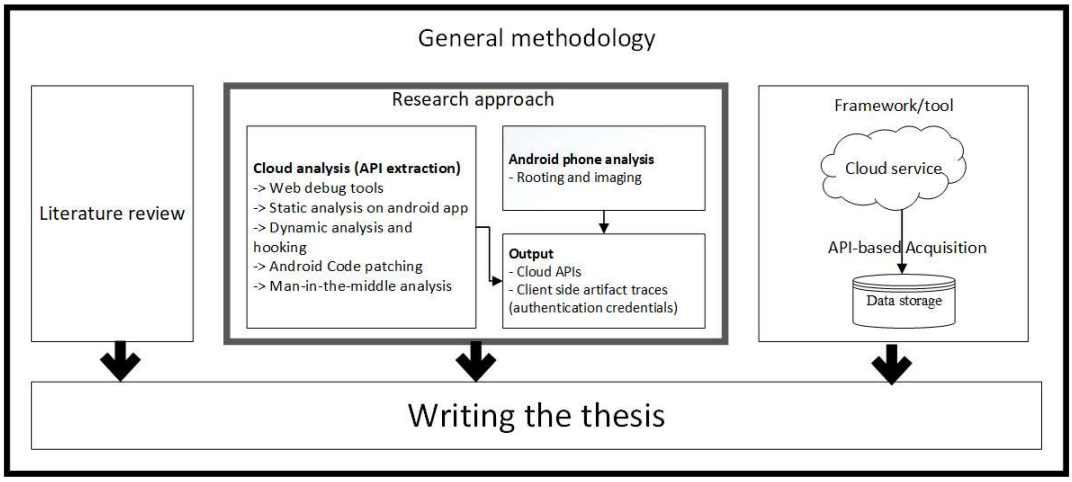

Figure 2: Overall research approach to study the IoT devices and write the thesis

3.1. Tools and Devices Used

For this thesis, the following devices and tools will be used and analyzed (Table 1).

Table 1: Tools and devices used to analyze and conduct the research.

Tools/devices	Description
Devices used	
IoT Devices	1) Echo: S/N: ***0071****30**5 Software version: 599469720 2) SmartThings: (Model) STH-ETH-200 3) Google Home mini: 7C18L5EHSK
Android phones	1) Samsung Note 4 (Android 6). It is a rooted phone to experiment root functionality of the phone such as Imaging and dynamic instrumentation. 2) Samsung S7 (Not rooted) (Android 8) 3) Android Virtual Device (Android 5.1)
Android apps and tools	
Android Application	1) Alexa App (V2.2.2) 2) SmartThings classic (V2.1.7) 3) Google Home app (V2.9.4)
Web tool	Windows 10 + Chrome (Version 63.0.3239)

Rooting and Imaging tools	1) help know and SuperSu – used to root phones and emulators 2) dd – command-line utility used to convert and copy files. 3) Android Debug Bridge (ADB) – is a command-line tool that enables users to communicate with an Android device. 4) Busybox – is a tool installed on a rooted mobile device to execute Linux commands. 5) Netcat – is a networking utility which reads and writes data across network connections.
Analysis Tools	1) The Sleuthkit Autopsy (V 4.4.0) – it is used to analyze the image file. 2) GMD tools (commercial) – MD Next () and MD Red (3.1) 3) Inspackage (V) – Xposed based dynamic analysis tool for Android apps. 4) DroidMon – is the same as the above. 5) SandroProxy – root-based proxy app. 6) Burp Suite Community (V) – Used as a Man-in-the-middle tool to intercept communication between app (Web or mobile) and the cloud. 7) Notepad++ with JSON formatter plugin.
Debugging and Development tools	1) Python (V3.7) for scripting 2) Microsoft Visual Studio Code (V1.30) used as an editor and debugger. 3) Postman for API testing and debugging

3.2. Environment Setup

The research environment setup includes the following stages:

1) Prepare the host machines for the research, acquisition and analysis work. This stage includes installing tools such as rooting, imaging, and analysis tools, freeing storage space and setting up directories on the machine. In this thesis, the host system is Windows 10 x64, and we installed the required tools on it, such as ADB [34], Autopsy [35] and JSON formatting. Table 1 shows the full list of used tools.

2) Install and set up the IoT devices in the room. In our lab, we installed and set up Smart Home IoT devices to generate data for our research. Samsung SmartThings Hub is installed with four of its devices. The door sensors are placed in the front main door and bedroom door, which connects to the middle room. In our scenario, the main room is considered as a dining room and the middle room as a bedroom. The smart outlet from SmartThings is plugged in the bedroom to control appliances such as the smart TV. An Amazon Echo is placed in the bedroom while Google Home is placed in the main room (dining room).

To configure and control these devices, we used Samsung Galaxy Note 4 with Android OS 6.0 which is considered Simon's phone — will be described in the scenario creation; all the applications are registered with Simon name. Table 1 contains the version of each device and its corresponding applications. The following are a general configuration for devices:

- Google Home and Amazon Echo (including phone version) devices are used as voice assistance speaker. On both of the devices, we did not configure voice match on both of the smart speakers.
- The SmartThings motion sensor never functioned.

3) Create accounts for each device, install the device's mobile app on the phones, and setup/manage the installed devices.

4) Execute activities on the IoT devices and their corresponding companion apps. The activities are started as soon as the devices are set up and connected with the internet using the account created in III.

5) Acquisition of phone Images. Acquirer the Android phone to analyze the companion Android application.

3.3. Research Approach

In this thesis, we followed the following approaches to conduct our research.

1) Literature review

Review literature concerning IoT forensics, cloud forensics and related digital forensic if relevant to this thesis.

2) Client-side analysis

After installation and configuration, the devices are up and running. When the apps are configured and utilized, files and caches can be created and stored on the smartphone

application. Authentication information, including cookies or tokens, can be stored on the smartphone and can be acquired for later usage to access the associated cloud service.

Besides, manual reverse engineering, dynamic, and static analysis are some of the methods of analyzing mobile application — described in section 4.3.1. One can decompile and reverse an Android application to study and research what the code or the application is doing. Tools for manual reversing and static analysis can be used to study and research mobile applications [36]. Other than that, the dynamic analysis of a running Android application could be helpful to extract relevant information such as APIs and token [37]. Moreover, we will use manual code injection to the application source code to observe sensitive data (token, device id, secret id) and APIs. Section 4.3 discuss more on the client-side analysis.

3) Cloud (API extraction and analysis)

Because of limited storage on IoT devices, manufacturers tend to use cloud backend services to store device and user data. Client applications communicate with the cloud over the Internet using designated cloud APIs [16], which are used to send and receive data. These communication APIs are one method to get cloud-native data in addition to synced or cached data on client applications. These APIs are either public for users and developers, or unofficial, which is hidden and private. Section 4.3 discuss more on API extraction. Figure 3 shows the process flow and procedure for API extraction.

3.4. Data Generation

For this research, data from the smart home devices will be generated and collected from the installed devices, sensors and client applications. To generate and collect realistic data, we interacted with each deployed device through physical interaction, client applications or automation between the devices and IFTTT [38]. However, data generation and collection will be performed in a controlled manner, and it will be documented including the timestamp and

description of the action performed on the devices. It, then, used to evaluate the acquired data with the generated data later in the evaluation stage.

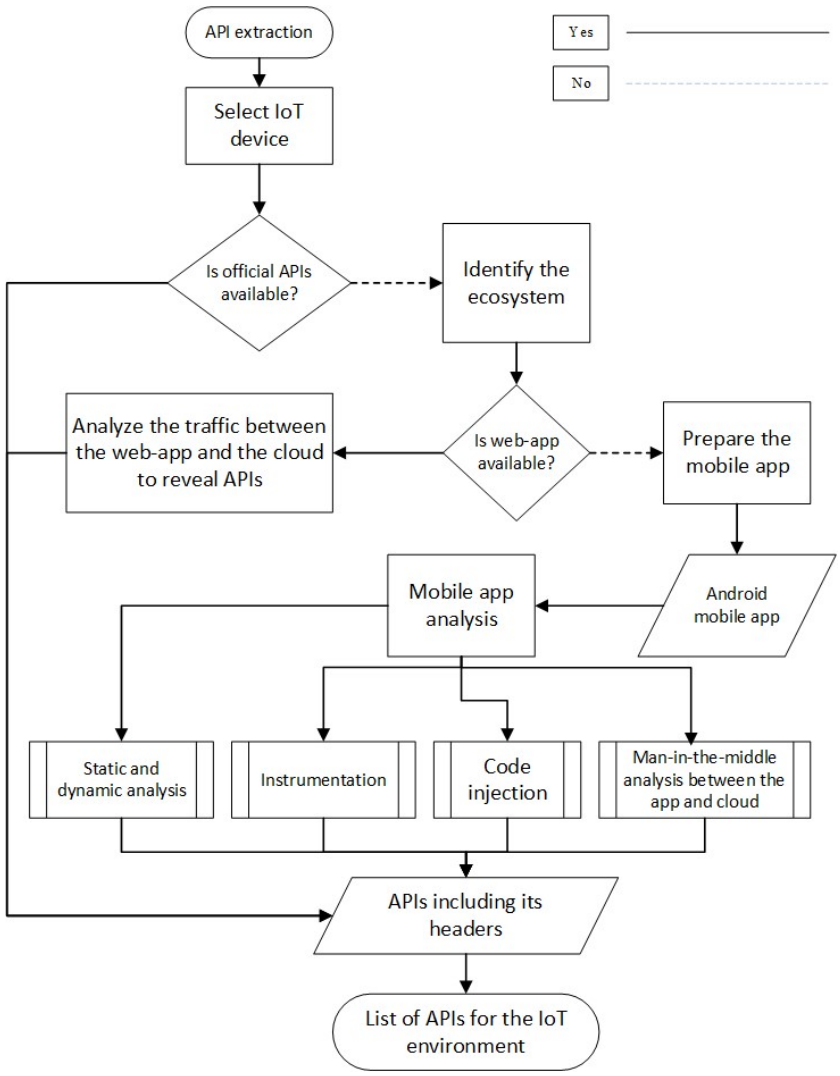

Figure 3: Cloud API extraction process for the target IoT-related cloud services and devices

3.5. Summary

In this chapter, we showed the methodology we used to conduct the research and write this thesis. We listed and documented the tools, procedure, and devices used in the thesis. Also, the chapter described the environment set up, which includes installing and configuring the target environment and its supporting infrastructures.

CHAPTER 4. TECHNICAL ASPECTS OF API-BASED IOT FORENSICS

Cloud forensics will play a significant role in the IoT forensics field, especially since the data generated from IoT environments are already being, or will increasingly be stored at cloud services. One way to obtain cloud data is to use cloud APIs which are provided by various cloud service providers. Besides, when there are no provided APIs, unofficial APIs could be used. In this section, we will demonstrate how to use and acquire cloud service data using APIs. Also, we will describe the overall ecosystem of the IoT environments data sources in terms of Smart homes. Lastly, we will demonstrate how to uncover hidden APIs for a specific ecosystem in addition to official APIs provided by the cloud service providers.

4.1. IoT Ecosystem Data Sources

In IoT ecosystems, the data can reside in the actual devices, such as home appliances, sensors, AI speakers, embedded systems, or cars. Alternatively, due to the limited memory space, valuable information could be sent to the central unit such as cloud servers for processing through the Internet. In addition, most IoT devices use companion applications such as mobile apps and web-apps to control and manage the device itself. Hence, these companion applications can be an additional data source for the ecosystem. In this section, we will explore the potential data sources for Smart Home IoT environment. We divide the data sources into three categories, namely:

- Local data source which contains client apps and IoT device hardware including the hubs and gateways;
- Network-level, and
- External data sources, such as cloud service providers and Internet service providers.

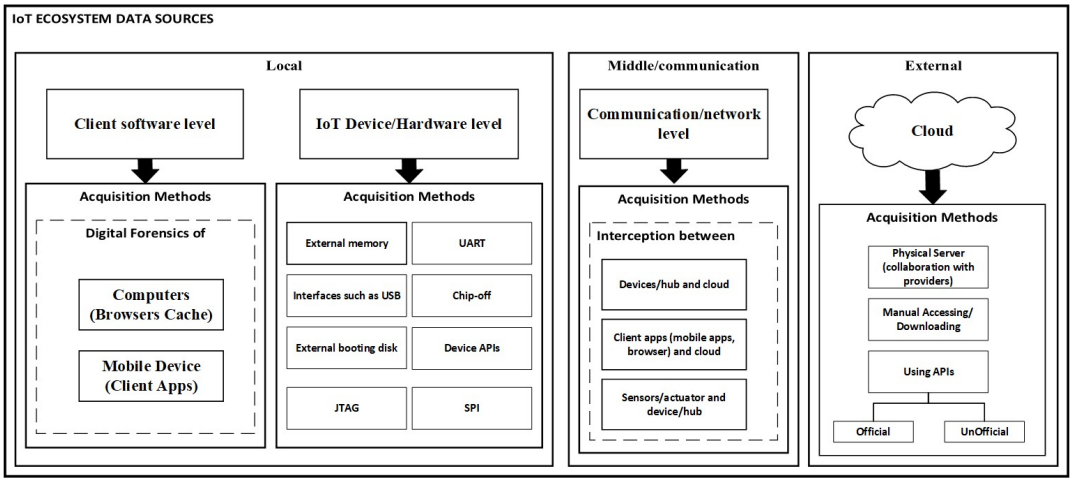

Figure 4: Smart Home IoT devices data source and its acquisition method

4.1.1. Local

As it is seen in Figure 4, this data source category contains two subcategories: the actual hardware (devices) and its companion (client) applications. Smart Home devices and network devices (hubs, routers) are located in IoT devices subcategory, while client category contains software (web-apps) and mobile apps. This category can be close to a crime scene, and investigators need to know what and where to look to find these data sources. Here, we will describe each possible data source in this category.

1) IoT device level

Device level forensics includes the hardware forensics of the IoT/Hub device and its associated sensors. The device level forensics may reveal configuration or cache data saved, or live data in memory, logs, device settings and unsynchronized data with the cloud or other connected devices. Since the device level forensics could destroy the device and its operation, it should be considered as the least option, that means if the other potential sources do not provide the required data.

2) Companion applications (Client)

Companion client applications like mobile devices and computers are essential to use IoT Smart Home devices. Specifically, users can configure settings and manage devices by using dedicated

mobile apps or web-browsers. In the process, a large amount of data can be stored temporarily as local cache data within client applications. Thus, uncovering the client-side artifacts may increase a chance to understand user behaviours through finding artifacts stored before being edited or deleted from the cloud. As discussed in the background section, different researchers did client-side cloud data analysis.

4.1.2. Communication or Network

Devices need a network for transferring and receiving data for different purposes. IoT devices basically use two different options for communication, which are wired network and wireless connections with different kinds of connection methods and protocols. The chosen transfer method always depends on what kind of IoT appliance it is, and what its purposes are. For wireless communications, the IoT environment uses Bluetooth, ZigBee, Z-Wave, COAP, MQTT, and WIFI as a communication protocol. For instance, SmartThings Smart Home ecosystem use ZigBee and Z-Wave protocol for communication between its hub and sensors [4].

Forensics examiner mostly captures network traffic data to collect content data, session data, or statistical data [39]. Network traffic analysis helps to capture raw data before it is structured and modified by the receiving ends to be further processed accordingly. Moreover, it helps investigators to identify the available devices and where they are communicating. Using the network analysis tools such as Wireshark, Burp Suite and other tools, investigators can perform network forensics capture and investigations.

4.1.3. External or Cloud

Data from IoT devices sent and stored in backend cloud servers. Then, it is synched to client applications such as web-apps or mobile Apps to the end users. Data from these client applications can also be sent to the cloud and then to the IoT devices. These data and commands are transported between client applications and the cloud using designated APIs, as shown in Figure 5. In general, aggregated user data, and specific device setup and configuration data could be saved on the

cloud, and if acquired successfully, it could be useful for digital investigators to solve a crime. In the following subsections, we will explore how to acquire this cloud data using these communication APIs. Besides, we will describe how to extract these cloud APIs if not provided by the cloud service providers.

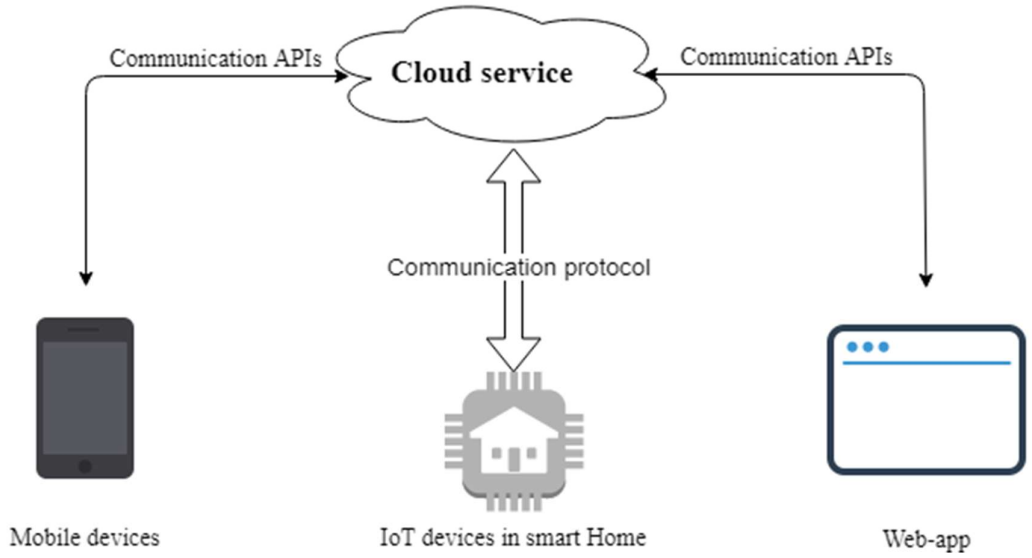

Figure 5: IoT devices ecosystem communications between devices, client applications and the cloud service

4.2. API-Based Cloud Acquisition

As stated above, there are official and unofficial APIs. In this section, we will discuss these APIs, how to use it to acquire cloud data and its limitation in acquiring cloud-native data.

4.2.1. Official APIs

Cloud service providers and IoT device vendors provide well-documented and official interfaces (APIs) through which client applications communicate with their cloud service. Using these APIs, third-party applications and manufacturers integrate their application and product with the cloud provider. For instance, Google Drive [40] and Microsoft OneDrive [41] and others provide APIs. It is good practice to check if the company, IoT device, or cloud service provides any documentation and information to work with Official APIs before proceeding to unofficial APIs.

4.2.2. Unofficial APIs

Most cloud service providers either do not provide official APIs or provide limited APIs that return limited data. However, cloud service providers and IoT devices allow users to set up, manage and view their devices and usage history by using mobile applications or web-apps. That means, there are some communication methods to transfer and receive data between the cloud services and the client's applications. These APIs can be uncovered using various analysis strategies, as shown in section 4.3.

4.2.3. Using APIs for Acquisition

APIs are provided to store and access metadata containing potentially useful information such as how and when data was accessed or modified as well as the data itself. The most direct way to use these APIs is using client applications provided by the cloud providers or third-party tools such as built-in web-browser tools like Chrome's DevTool or Firefox developer tool. However, third-party tools, such as Postman [42] is more convenient. As shown in the literature review section, researchers showed how to use these APIs by developing custom tools to help automate the cloud acquisition process for investigators. Researchers in [24]–[26] demonstrated the feasibility of an API-based approach through the automation of an API-based acquisition and provided a tool for digital forensic investigators. Same as the above researchers, we automated the API acquisition process; chapter 5 discussed detail about the tool.

4.2.4. Limitations of API-Based Acquisition

API-based acquisition from the cloud has three fundamental limitations:

1. It requires a valid authentication method or user credentials — it could be security token a cookie, or an ID/password combination) to access cloud-native data.
2. Based on the providers and user agreement, it may not be possible to recover historical deleted data from the cloud. In addition, some cloud providers do not store user activity data.
3. Since the APIs are private, the providers could change the unofficial APIs without notice.

4. Deleted data may not be recovered from the cloud using APIs

4.3. Uncovering Unofficial APIs

As stated, unofficial APIs are private and hidden. It is used for data sharing between the cloud and the client applications. In the following sections, we will describe how to uncover or extract these hidden APIs in addition to vendor-provided APIs.

4.3.1. Extraction Methods

7.3.1.1. Static Analysis (source code)

In addition to understanding the source code and the overall workflow or architecture of the mobile application, analyzing the source code could be used to look for hardcoded URLs/APIs, usernames and passwords, communication protocols and so on. Some mobile application hardcodes their communication APIs as a text. Looking at these URLs/APIs could help know the hidden communication methods in which the application and the cloud use. In addition to the hardcoded URLs, understanding the communication code or module can help in understanding the security, headers, and the parameters. For example, Amazon Alexa App hardcodes some APIs and its structure in its source code.

7.3.1.2. Website Debugging Tools

There are a lot of web development tools integrated into browsers or stand alone. These tools are not just for web development, but also serve a lot of other functions, of which debugging and network communication interception are two of the most prominent. Using these tools, we can inspect the network communication APIs, its request/response headers, contents, and so on. To uncover hidden communication APIs, we used web debugging tools such as Chrome DevTools and Firefox Developer built-in tools [43], [44]. However, not every IoT device or cloud providers provide browser-based client applications (website).

7.3.1.3. Man-In-the-Middle (MITM)

When the cloud providers and IoT devices ecosystems do not provide browser-based client applications, we have to use other methods such as a Man-In-The-Middle attack [45]. In a Man-in-the-middle attack, we monitor data in the middle between the client applications (mobile app) and the cloud and inspect the communications to analyze and watch the APIs, request/response headers, authentication method, payloads, and the response data. With the help of some tools such as Burp Suite [46] or root based mobile applications, we can monitor and catch network communication between the app and the cloud.

7.3.1.4. Dynamic Analysis (at runtime)

Nowadays, applications implement security method to prevent their application from a Man-In-The-Middle (MITM) attack. Security implementation such as SSL pinning prevents the mobile application not to connect to the cloud or internet when there is a proxy in the middle or when the certificate provided by the device does not match their certificate [47]. In this case, dynamic instrumentation methods are necessary. Dynamic instrumentation is an analysis of the running mobile application without changing the working environment of the application, such as changing the source code or proxying the traffic. Xposed and Frida frameworks are two of the most popular tools that enable dynamic instrumentation on mobile apps [48]. Specifically, we used Inspeckage Xposed module — a module developed to work on Xposed framework — to analyze the target application while running, and intercept and inspect the communication.

7.3.1.5. Manual Code Injection

When all the possible security mechanisms are implemented, such as SSL pinning [47], anti-instrumentation, root detection and so on [49], reversing the mobile app and injecting a code could play a big part. The process is: reverse the mobile application, analyze the source code and select where to inject the code, and put the code into the source code — the source code could be a smali or java source code. In the end, recompile the new source code and install it on the mobile phone.

If successful, the intended added feature will work — for instance, if you put a logging code, the output will be displayed in the logcat window. By doing this, we extracted the APIs that the application uses to fetch/send data from/to the cloud service.

4.3.2. Understanding the API Structure

Every cloud providers and vendors have their own API type, structure and authentication method. However, it could contain common — shown in Figure 6 – URL and its operational method and parameters; request and response headers and body; and its security method [50].

- URL contains operational methods such as POST, GET, PUT or DELETE. The URL may also contain Queries and filter parameters; sorting, limiting data based on date and size, status can be achieved using filtering parameters.
- Headers provide meta-information about a request and a response such as a browser type, content type, and sometimes authentication information.
- Request and response body — contains the data the client wants to send the server, and the data the server sends.
- Authentication — when requesting resources and data from the cloud using APIs, the providers could validate the authenticity of the request and its authorization privilege. The authentication information could be a token, cookie, or a pair of username/password. For investigators, this authentication information sometimes can be recovered in different ways. Tokens and cookies can be obtained using Man-in-the-middle interception, checking in the smartphone mobile app image if the application saves it locally, or extract from browser history if used in the computer browser. However, username and passwords should be recovered from the owner if the above methods are not successful.

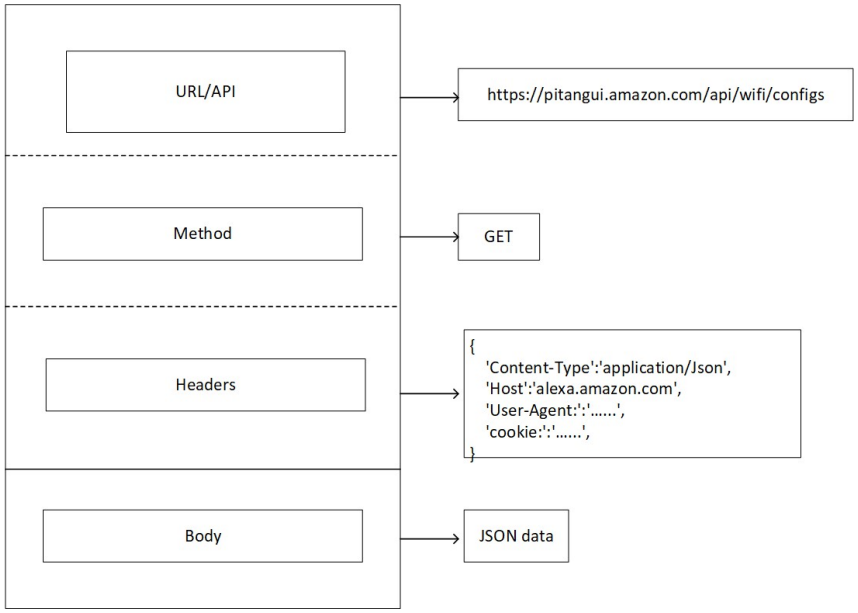

Figure 6: Generic API structure containing URL, method, request/response header which contains authentication cookie, a response data which could be in JSON format

4.4. Summary

In this chapter, we described the overall ecosystem of the IoT in the Smart home environment and its data sources. We categorized the data sources in three: client side, network side and cloud side. In this thesis and chapter, we are interested in the cloud side data source and the method to acquire it. Even though there are different methods to get cloud data, our approach is in using its communication APIs. In this chapter, we demonstrated how to uncover the hidden communication APIs for a specific ecosystem or use public APIs provided by the cloud service providers then use it as a method to acquire cloud-centric data. We used different methodologies to uncovered these communication APIs such as web debugging tools, man-in-the-middle attack, and static analysis, and code injection.

CHAPTER 5. CASE STUDIES

In this chapter, we will study three devices to show how to use APIs in acquiring cloud-native data from their respective cloud services. We have selected Amazon Alexa, Google Home and SmartThings devices and cloud providers based on popularity and availability. All three are the top Smart Home devices used by consumers based on [51].

5.1. Procedure

For each of the case study devices, we approached the overall analysis based on the following procedure or steps — shown in Figure 7 as a process. For each of the steps, a description is provided in the following subsections.

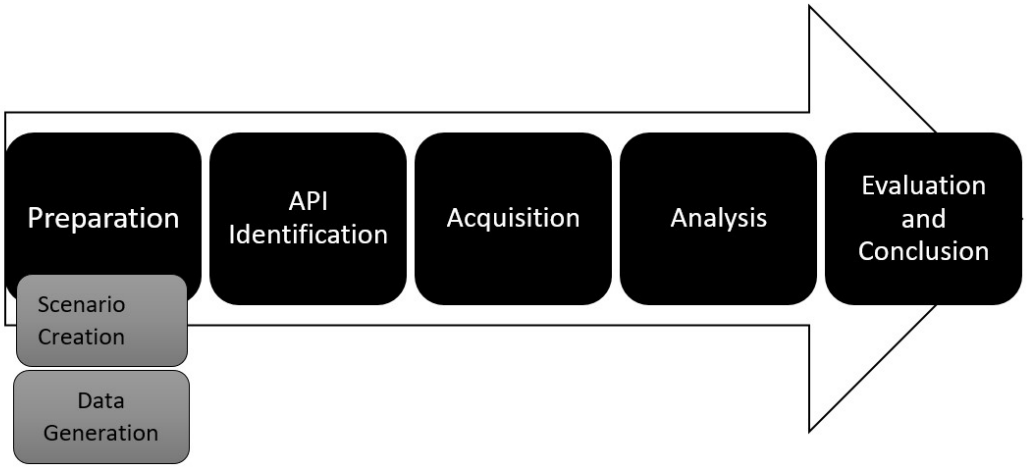

Figure 7: Analysis procedure for the IoT-related case study environment

5.1.1. Preparation

In this stage, we research the overall structure of the device and its cloud and how the data is stored and possible artifacts. Data generation and scenario creation are also defined here.

5.1.1.1. Scenario Creation

In this stage, we will create a scenario to show how the data acquired from each device's cloud can be used in an investigation process — scenario-based analysis.

5.1.1.2. Data Generation

The purpose of the data generation is to show the completeness of the data acquired from the cloud. We will compare the data generated against the data downloaded in the acquisition stage. All the devices are collecting data from our lab since the installation and setup. However, to evaluate the completeness of the acquired data, we have generated data based on the scenario transcript for each device.

5.1.2. API Identification

In this stage, we will be identifying the APIs used in each of the case study devices. Besides, describing the research process, categorizing based on its functionality, and identifying authentication and header information are done at this stage.

5.1.3. Acquisition

As described in the data source section — section 4.1, data can be acquired from different data sources such as client applications, network, and cloud. In this thesis, the cloud-side data will be acquired from cloud services. We used our own designed and implemented tool to acquire cloud-native data from the case study device's cloud service — Amazon Alexa, Google, and SmartThings cloud. The assumption is that account information including username and password, token or a cookie is available. Tokens or cookies could be available in client applications such as mobile phones and browsers.

The tool accepts all the required parameters needed such as evidence directory, the cloud service provider, and authentication methods such as Token, cookie or username and password combination. Then, it filters (if provided), and acquire the cloud-native data and save to the provided directory.

5.1.4. Analysis

In this stage, we will analyze the acquired data and map into the digital forensic perspective to examine what it could mean and if it is valuable to the investigators. In addition, we analyze the acquired data to answer the question asked when creating a scenario. In the end, we discuss the overall result and proof or disproof the hypothesis/questions.

5.1.5. Evaluation and Conclusion

The evaluation stage is conducted to find out if all the generated data is acquired fully. In other words, we measure the completeness of the acquired artifacts based on the data generated from the script. In addition, this stage reviews the overall results and discussions, asked investigator questions, and conclusion, including the scenario-based analysis of the data.

5.2. Case Study 1: Amazon Alexa

Amazon Alexa is cloud-based virtual assistance called Alexa Voice Service (AVS), which is available in Alexa-enabled devices such as Amazon's owned products Amazon Echo [52]. Besides, Amazon now lets users use Alexa skills in a Mobile phone without owning one of its products such as Echo or Dot. Hence, this cloud-based service — in addition to the hardware and phone application — could be a source of potential digital evidence for law enforcement agencies and investigators. For example, as stated in the introduction section, Amazon is getting a request to provide over user recordings and activities from Alexa cloud service. In such circumstances, identifying and obtaining data from its cloud service could be helpful for investigators. In this case study, we will identify its APIs, acquire cloud-based data, and examine the evidence. Besides, we will show the value of the evidence by creating a scenario and analyze the acquired data based on the scenario.

5.2.1. Preparation

The main feature on the ecosystem is that it lets a user perform voice-based commands and actions. Using its library of "skills" - or small custom scripts - users can do online shopping, ask for information like weather, play music, control other smart home devices, and many other features. It also enables a user to call and message other Amazon Echo users with a voice command. The primary purpose of this case study is to obtain these user activities and commands from Amazon Alexa cloud using APIs. Specifically, user command/activity history and calling/messaging data could contain valuable information for digital forensic investigator since it contains rich user data, including audio/voice recordings, as shown in Figure 8.

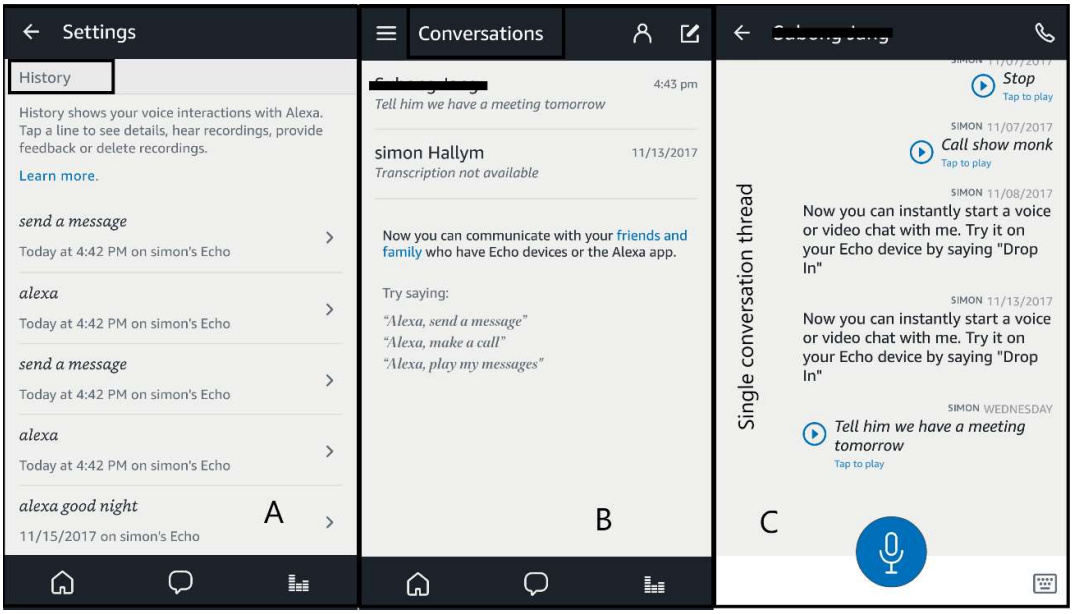

Figure 8: Alexa user voice interaction history and calling/messaging conversations. A) user interaction history. B) list of conversation and call. C) single conversation thread

5.2.1.1. Scenario

For this scenario, we will use the Alexa scenario from 2017 DFRWS challenge created by James [53]. A woman called Betty is killed. There was an Amazon Echo speaker in the house, and police suspect the device recorded something that may contain a clue about the murder case. From the data acquired, police what to answer the following questions:

- Is there anybody else in the house?
- What commands Betty issued to the device at the time of the crime that may contain relevant evidence for the investigation.

5.2.1.2. Data Generation

For the scenario, the data were already generated in 2017. Based on that data, we will analyze and map with the scenario. Specifically, from the Alexa part, the following data is generated for the challenge and scenario.

1. Simon turns on the TV via the Echo with voice
2. Betty turns on music via the Echo with voice
3. Betty turns off music via Echo with voice (record John yelling)
    - Someone keeps yelling "How could you do this to me! You said you would leave him!"
4. Simon uses Echo to call Ambulance (fail, not in JSON file)

5.2.2. API Identification

As it is mentioned in the background section, in [24] Chung et al. studied the overall Amazon Alexa ecosystem and revealed unofficial APIs which Alexa ecosystem uses to communicate with the Amazon cloud. Also, in [31], [54] Hyde and Moran presented newly added APIs for calling and messaging features. Other than the APIs presented in the above works, we identified newly added and updated APIs. To do additional analysis, we used the strategies mentioned in section 4.3 — static analysis on the reversed source code, code injection, man-in-the-middle interception, and dynamic analysis using Xposed/Frida — to uncover its unofficial APIs.

The APIs start with the base endpoint/URL https://pitangui.amazon.com/api, or https://alexa.amazon.com/api. The calling and messaging APIs which start with https://alexa-comms-mobile-service.amazon.com. The remaining path and query string parameters after the base URL describe the main functionality and option of the API to influence

the response. For instance, ***wifi/config*** in the API https://pitangui.amazon.com/api/**wifi/configs** shows that the API returns the Wi-Fi configuration of the device. In Appendix B, we described how to construct the APIs by adding variables and values, what values should be filled in, and how to get these values. Moreover, values and query string parameters placed in {*parentheses*} should be replaced with appropriate values to get the intended result.

For the simplicity of presentation and analysis, we divided the APIs into three categories: General APIs which return general information about the user and device; activities and cards APIs which contain user voice interaction activities; and calling and messaging APIs which return calling and messaging feature data.

5.2.3. Acquisition

A valid authentication — cookie or username/password combination — is needed to obtain data from the cloud using the identified APIs. In our thesis, we used both username/password combination and a cookie recovered from the phone. Using the tool, we managed to download all user data from Amazon Alexa. The downloaded files are saved based on their types such as user activities, cards, calling and messaging artifacts and general information including the user/owner information and devices list, WIFI/Bluetooth information, notifications, and default and user-created lists. However, for this thesis and based on our created scenario, we downloaded general information and user activities such as calling or messaging the owner exchanged, general activities such as commands given to Alexa and so on. Here is the procedure:

- Setting up the parameters such as username/password or a cookie. In our case, a cookie from the mobile phone is extracted and used as authentication to the cloud.
- Provide filter data to download that specific date activity. For the scenario, the date range is provided to download data on the date the crime happened.

- Given time range is from July 16, 2017, to July 18, 2017. The crime happened on July 17, 2017.
  - Start date = July 16, 2017
  - End date = July 18, 2017
- We want the audio file hence we check “include audio” checkbox.

- Download the data, including calling and messaging conversations. The acquired data contains general information such as user account, device preference, lists (to-do, shopping or custom created lists), connected devices, and so on — Figure 9 (A). Besides, the tool downloads and puts the calling/messaging and user activity data, including its voice file in a separate directory. As shown in Figure 9 (B and C), the user activity directory contains all or range of activities based on investigators preference while calling/messaging directory that holds user contacts and conversations(chats).

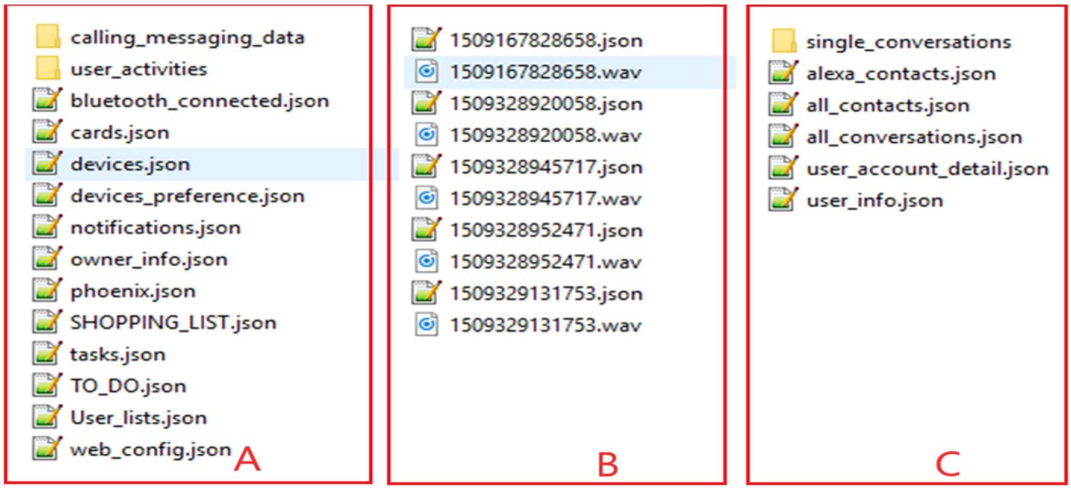

Figure 9: Acquired cloud artifacts categorized based on their function and type. A) all the files. B) activities data, including its voice recording files. C) Alexa Calling and Messaging artifacts

5.2.4. Analysis

As stated by Orr and Sanchez in [33], from timestamps to recording values, the data collected by Alexa has the potential to provide law enforcement with the ability to further their investigation. In this thesis, as shown in the above section, we can acquire cloud-native data from

Alexa cloud service using the unofficial APIs. The data, which is in JSON format, contained UNIX timestamps, text descriptions, lists such as to-do lists, shopping or custom created lists, and transcript of the voice command for user activities, and so on. Furthermore, data such as user conversations, contacts, and voice messages from the calling and messaging are available in the acquired artifacts. These acquired data could help investigators in reconstructing user actions and studying user behaviour in combination with other sources of evidence.

For the scenario, we have analyzed the activity data for any valuable evidence. Between the given range of date, given to the tool, around 38 individual activity items and its audio files are downloaded. From these files, there are recordings and activity entry same as the data generated. Based on the acquired evidence, we can answer the questions asked:

- For the first question, we can say that there was another person in the house at the time. From the recordings, one person was yelling while betty stops the music. Also, probably a different person gave a command to Alexa to call an ambulance. In the JSON file, we can map the audio file and the timestamp of the command.
- The most critical entry was when Betty commands Alexa to stop the music while somebody else was yelling at the background. Other than that, all the commands were basic commands, such as play music, stop the music and so on.
- In the end, her husband heard calling the ambulance. Based on the voice type, it seems the previous person is different from Simon.

5.2.5. Evaluation and Conclusion

All the generated data are acquired using the unofficial APIs presented in this thesis. However, the audio recordings contain more data than the JSON entry, which is a text version of the audio. Overall, the data acquired has the value to answer the question asked in the scenario creation stage. Other than that, we can conclude that essential artifacts can be acquired from Alexa cloud

which could help criminal investigation process and the tool we developed based on our research in this thesis can acquire all the available cloud data.

5.3. Case Study 2: Google Home and Assistant

The Google Home is the physical hardware speaker designed by Google. Same as Amazon Echo, it comes with Google virtual Assistant. Google Virtual Assistant is an artificial intelligence-powered virtual assistant. With Google Home device and its assistant, users can ask for traffic or weather information, a search on the web, call or control other smart Home devices [55]. At Consumer Electronics Show (CES) 2019 [56], Google announced that its Assistant is expected to hit the mark of 1 billion devices. That number includes smart speakers, smart displays, phones, headphones and more.

The Google Home speaker should connect with the internet and its backend cloud services since its head and computation are in the Google cloud — Google Assistant. Whenever a user asks a question or command by voice, Google Home device can conduct web searches and find answers or communicate with other smart devices. The answer then comes back to the user through the device after search/computation in the cloud. From a digital forensics' perspective, this user action and activities can be a great source of potential digital evidence for the criminal investigation process.

5.3.1. Preparation

Any user activities performed will be stored on the Google cloud in user activity page. The service, which is called "My Activity" contains all including voice search or assistant related data. However, any assistant related — it could be from Google Home or mobile phone — data can be filtered using the "assistant" parameter. What the user asked or command, what tasks and to-dos created, the automation performed on/by the device are saved in the Google cloud in the name of

Assistant service. The main goal of this case study is to acquire and analyze cloud-related data from Google cloud related to Google Home device and Assistant.

5.3.1.1. Scenario

The owner of a house reported a robbery to the police on April 27, 2019. He said he was not around for the past two days and when he gets back, his house was robbed. He also reported to his insurance he gets robbed. Police suspect the owner wants to cash in on insurance claims. While searching, police seized a Google home mini from his bedroom. Using his account, they acquired his activities for Google Home service and Assistant. An investigator wants to answer the following questions:

- What happened in those days around the house?
- When was the last time he used the device and the last time he was around his home?

5.3.1.2. Data Generation

Any activity from the house and Google Home mini.

- The Morning Routine "good morning" which triggers other actions.
- Search for traffic information on that day using voice command.
- Reminder to call a friend.
- Search "How to fraud insurance."
- Search "for traffic using mobile phone assistant."

5.3.2. API Identification

Google Home mobile app is used to configure the device and manage preferences and accounts that are used for the device. We used the same methodology described in section 4.3 to analyze and intercept the communication APIs between the app and the cloud.

Based on the analysis, we found some APIs that the app uses. They are used to fetch linked devices and linked the user to the device. In addition, if the user wants to check the activities, the app sends it to another Google service which manages user's activity history known as My

Activity. It is a history of almost everything a user does online, including, sites visited, things and places searched for, as well as activity on each of its products such as Google Home [57]. However, Assistant and Google Home specific activity logs can be acquired using 'restrict' parameter on My Activity API. To clarify the activity data, we have mapped the Table and Figure — Table 2 and Figure 10 — to show the relationship between the activity in the web-based application for Google activity and the API returned data listed in the sample data. All the APIs, including its structure and parameters, are shown in Table 2.

Table 2: Google Home assistant APIs and My Activity history detail returned from API call

Category	***Google Home APIs ('{}' should be replaced with appropriate values)***	***Description***	***Sample data returned by the APIs (protocol buffer type)***	
Home internal APIs	https://clients3.google.com/cast/orchestration/linkeddevices?rt=b **Method:** POST	Return all linked devices with the account holder.	{Id} {Home or group name} {Home type — mini} {D2C*******C936F11757**********7C3F57} Office speaker Google Home Minib	
	https://clients3.google.com/chromecast/emails/app-get-preferences?rt=b **Method:** GET	Provides account holder email, language preferences, and country code	KR en-us* simonhallym@gmail.com	
	https://clients3.google.com/cast/chromecast/home/devices **Method:** GET	Returns setup device IDs with the above account	$64473d1b-9c69-4058-a4dc-ea7aedf431fa B1F4344F0A50568F34AACA62FD4B0A1B D2C293358C936F11757914443A7C3F57	
User's My Activity	https://myactivity.google.com/item?**restrict**=assist&**min**={fromTime-epoch}&**max**={toDate-epoch}&**jspb**=1 **Parameter:** **Min:** minimum time range (from) – it is Unix time format Max: maximum time range (to) – it is Unix time format **Jspb:** returns the activity data in protocol buffer format (close to JSON) **ct:** next batch parameter. it can be retrieved from the values returned in the first request (ct=none)	User data filtered to return assistant related activities	"1552917315632110"	Timestamp (epoch) (Figure 10 (4))
			["hey Google you working", true, "Said", "https://www.google.com/search?q=hey+Google+you+working"]	Transcript of a user's voice command (Figure 10 (1))
			[["I'm always on the clock. The good news is I love my job"]]	Answer from Assistant (Figure 10 (2))
			["Assistant"]	Source of command (Figure 10 (5))
			[["Google Home"]]	Origin of this command (product name) (Figure 10 (7))
			[["From your current location", "https://maps.google.com/?query=<lat>,<lng>"]	Location information (Figure 10 (8))
			["https://myactivity.google.com/history/audio/play/{Id}/{Timestamp}"]	URL of a voice file (Figure 10 (3))
			[["Started by hotword"]]	Information (Figure 10 (6))

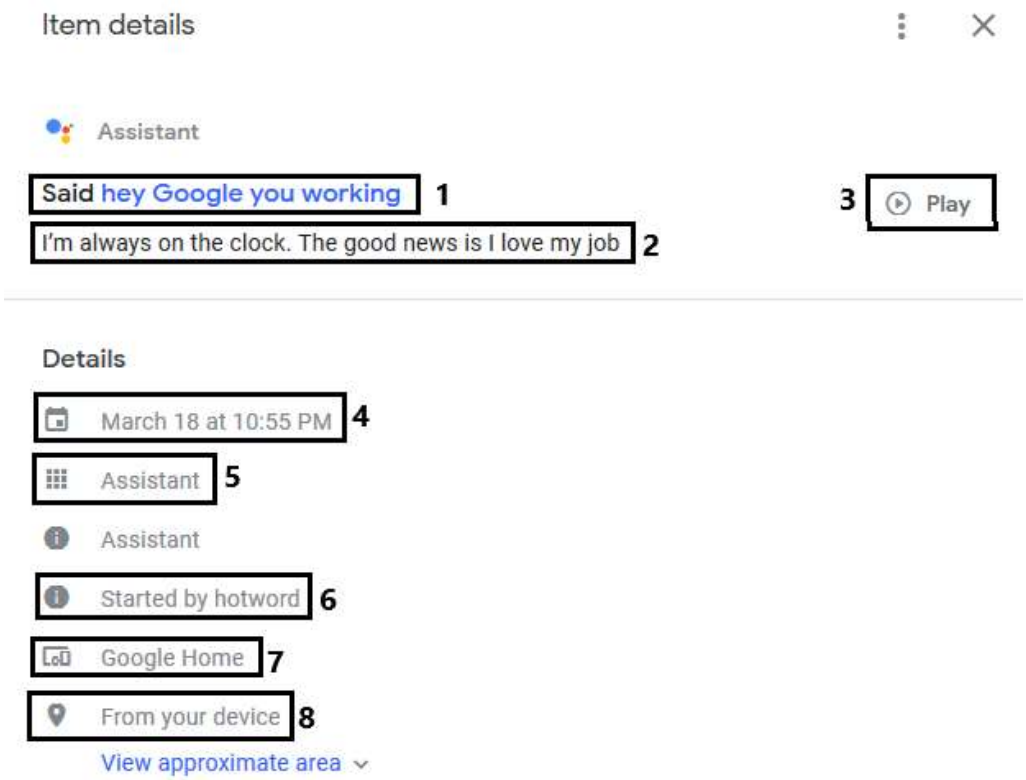

Figure 10: User activity entry in My Activity history API. (1) Transcript of a user's voice command. (2) Answer from Assistant. (3) URL of a voice file. (4) Activity timestamp. (5) Source of command. (6) Trigger Information. (7) Origin of this event (product name). (8) User location information

5.3.3. Acquisition

Google Home app APIs use a bearer token as an authentication method. On the other hand, My Activity APIs needs login information or cookies from the browser. The token can be extracted from the account database mobile phone which contains account information used in that phone. However, the APIs from the speaker does not contain much data except installed Home devices, including its IDs, location and account information. On the other side, the My Activity API returns a lot of information to understand a user's behaviours and activities over time. The response data provide simple and full version log history. The full version response contains user activities including a transcript of user's voice command and answers to the command, link to the voice data, the location where the command issued, and source of the command such as Google Home and Google App; the simple version does not contain this information except the timestamp and

location. This API can be accessed on a browser with a valid credential and returns data in a protocol buffer message format if the *jspb* parameter is set to 1.

Using the tool provided, we managed to download all user activity information from the cloud using the provided API. Here is the procedure:

- Provide the parameters, including token and cookie authentication information. In this case, we used username and password as authentication and downloaded user activity from Simon's Google activity.
- Provide filter parameters to download that specific date activity.
    - The time range is given from April 27, 2019, 18:04 (download started time) to April 27, 2019, 00:00 as the crime has happened on April 27, 2019. Downloading the whole activity takes time and bandwidth as it downloads from the date the user setup Google Home device till to date.
    - We wanted to acquire parsed data, so we checked the parse checkbox for the tool to parse out relevant data.
- Download the data — for the time being; the tool downloads the user activity data using "My Activity" API. If date range parameters are given, it saves in a separate directory; otherwise, it downloads all the available user activity and put it in a single directory. If parsed data is needed, it puts in a separate directory in an excel sheet format. The overall directory — based on the base directory path — looks like Figure 11 (A). The acquired JSON data – originally it is a protocol buffer format – are saved with the file name based on the first and last item's timestamp — Figure 11 (B). The parsed data containing selected values from the JSON data is saved to excel file — Figure 11 (C).

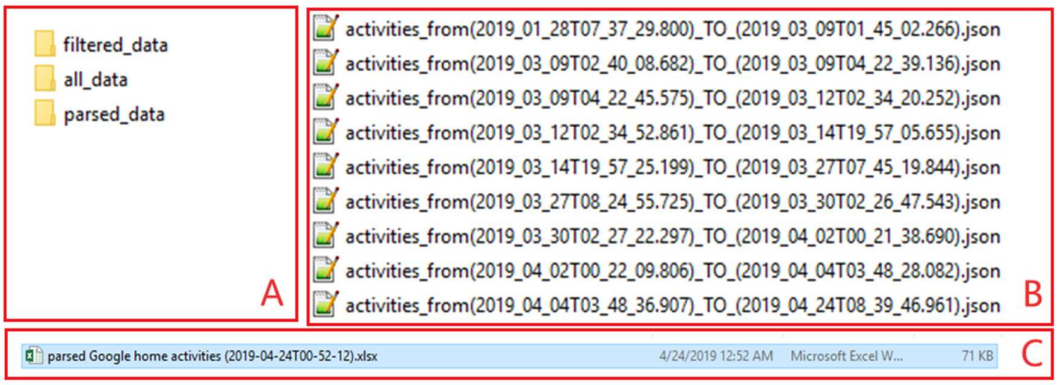

Figure 11: Acquired data from Google Cloud for Assistant and Google Home services. A) the base directory created based on the parameters given. B) downloaded file saved as a JSON file based on a timestamp from items inside. C) parsed data saved to excel sheet

5.3.4. Analysis

As shown in Table 5-1 sample data column, the acquired data contains potential evidence, such as what a user command to the Assistant, at what time (timestamp), what service or device used to command the action such as Google Home or another device, and an actual voice data (MP3 format) if available. Using these values, investigators can reconstruct user activities.

From the acquired data, we have analyzed the user activity data to answer the questions asked by the investigator. The parsed data contains: timestamp [timestamp column] which is a time when the user issue a command to the device; the voice to text transcript of user command [description column]; answer from the device/cloud [answer column]; the source device such as Google Home device [command_source column]; location where the command issued [location column]; and the URL to the voice data if available [voice_url column] — shown in Figure 12. From this data, we can reconstruct and recognize activities around that house that day and if the owner was around. On that day, Simon started his morning routine by saying “Hey Google, good morning”. Then, Google Home performed all his morning activities such as playing news, stating time and reminders. One of his remainders was calling his friend — whom he called can be analyzed using other data sources. Besides, he wants to know how to fraud insurance by asking the device. He also asks about traffic using both his mobile phone (see Google App on command

source column) and Google Home. This digital evidence from the Google Home and Assistant could show that the owner was at his home and could prove, with additional evidence, that he lied about his location and may have done it to get insurance from the insurance company. Specifically, the location data, which points around his house, shows that Simon was around. Also, the command came from Google home, which means Simon asked these questions from inside the house.

eet contains an activity from Google assistant history acquired using activities API. Updated: 2019-04-27T1

	downloaded_date	timeStamp_unix	timeStamp_iso	description	answer	command_source	location
0	2019-04-2718:04:29	1.5563E+12	2019-04-27T01:11:21	what is the traffic looks like today	Here are the top results, starting with one from Google	['Google App']	https://www.google.com/maps/@?api=1&map_action=map&center=37.892887,127.734398&zoom=12
1	2019-04-2718:04:29	1.5563E+12	2019-04-27T01:10:59	Assistant	-		https://www.google.com/maps/@?api=1&map_action=map&center=37.892887,127.734398&zoom=12
2	2019-04-2718:04:29	1.5563E+12	2019-04-27T01:10:56	what is the time today	It's 1:10.	['Google Home']	https://www.google.com/maps/@?api=1&map_action=map&center=37.892887,127.734398&zoom=12
3	2019-04-2718:04:29	1.55629E+12	2019-04-27T01:09:53	how is the traffic to the nearest coffee shop	Sorry, I can't give driving directions for that country yet.	['Google Home']	https://www.google.com/maps/@?api=1&map_action=map&center=37.892887,127.734398&zoom=12
4	2019-04-2718:04:29	1.55629E+12	2019-04-27T01:07:10	how is the traffic today	Sorry, I don't know how to help with that.	['Google Home']	https://www.google.com/maps/@?api=1&map_action=map&center=37.892887,127.734398&zoom=12
5	2019-04-2718:04:29	1.55629E+12	2019-04-27T01:06:02	how can I frowed an insurance	Sorry, I don't know how to help with that. But I'm trying to learn.	['Google Home']	https://www.google.com/maps/@?api=1&map_action=map&center=37.892887,127.734398&zoom=12
6	2019-04-2718:04:29	1.55629E+12	2019-04-27T01:04:40	Reuters TV (U.S.)	-		
7	2019-04-2718:04:29	1.55629E+12	2019-04-27T01:04:36	stop	-	['Google Home']	https://www.google.com/maps/@?api=1&map_action=map&center=37.892887,127.734398&zoom=12
8	2019-04-2718:04:29	1.55629E+12	2019-04-27T01:01:33	play news	Here's the latest news.	['Google Home']	
9	2019-04-2718:04:29	1.55629E+12	2019-04-27T01:01:25	Google Assistant	The time is 1:01 AM. By the way, remember to call friend Have a good one!	['Google Home']	
10	2019-04-2718:04:29	1.55629E+12	2019-04-27T01:01:20	good morning	Hi, Ninaninyono!	['Google Home']	https://www.google.com/maps/@?api=1&map_action=map&center=37.892887,127.734398&zoom=12

Figure 12: Google Assistant activity data extracted from Google APIs then parsed into excel file

5.3.5. Evaluation and Conclusion

All the activity data, including generated data, are acquired using the "*My Activity*" Google API. Hence, investigators can recover user activities, including the command issued Google Assistant using the provided API. However, action and commands sent to third-party devices, and deleted user activities cannot be recovered from the cloud. In these circumstances, client-side data from the mobile app, browser cache, or device data could be useful. Also, actual user voice URL is not provided for some command; hence, it is not possible to acquire it.

Overall, the data acquired, including its values, command descriptions, timestamps, is valuable for investigators to answer the question asked concerning a crime incident. For our scenario, additional data may be needed to show the intent of the owner or a CCTV data to know what

happened to the house. Other than that, we can conclude that important artifacts can be acquired from Google Assistant activity and help criminal investigation as we have shown in our scenario.

5.4. Case Study 3: SmartThings

Samsung SmartThings is one of the Smart Home platforms that turns home into a Smart Home with a SmartThings Hub that wirelessly connects to hundreds of compatible smart devices like lights, speakers, and sensors, and makes them work together [4]. It is a cloud-based service that interfaces with a user's SmartThings hub and client applications. Its design allows for quick and seamless additions of many types of sensors to a home network. Devices are connected to the hub using either the Z-Wave or ZigBee standard [58].

The SmartThings ecosystem consists of three major components: the hub with its connected devices, the SmartThings cloud backend, and the smart-phone companion app – see Figure 13. Using the smartphone companion app, users manage their hubs, associate devices with the hubs, and install SmartApps from an app store.

The SmartThings platform and its ecosystem depend on its backend cloud service. Supported devices, automation, companion applications, and the hub needs to be connected to the SmartThings cloud [59]. In general, user's data is collected and stored on the backend cloud; then, the processed data is available for users through client application — the SmartThings mobile app and web-app. To acquire data from its backend cloud service, we researched the communication between the client apps and the cloud to extract its APIs.

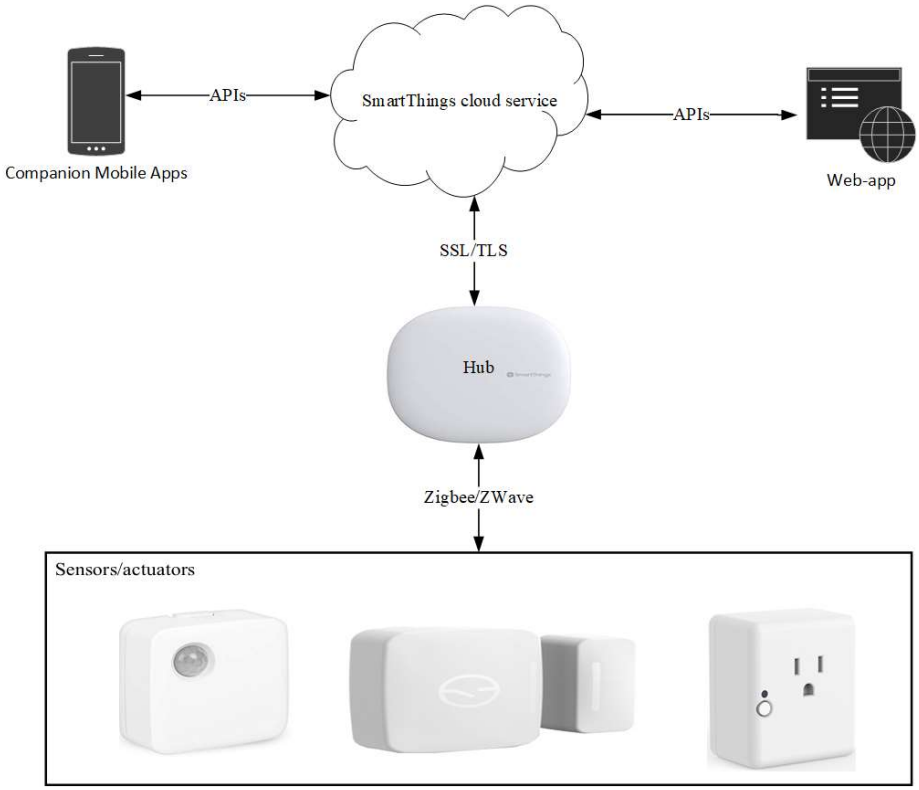

Figure 13: SmartThings setup and its communication between the hub, sensors and the cloud

5.4.1. Preparation

To understand cloud artifacts, we need to understand how the SmartThings structure and compose the environment. It is composed of three different "containers": Accounts (customer), Locations (geolocation), and Groups (rooms/physical space). Below these containers, there are individual devices [59]. Devices are the "things" that SmartApps interact with, while SmartApps are applications that allow users to tap into the capabilities of their devices to automate their lives. Capabilities represent the things a device knows (attributes) and the things they can do (commands). A State of a device is the information about a particular Attribute at a particular moment in time. Table 3 describes what each structure, sensors, or devices data could mean for investigators.

5.4.1.1. Scenario

The fire started at Simon's house. In his testimony, he said he was not around the house on that day. Police think John intentionally set fire and want to answer the following question.

- Was Simon around the house on that day?
- Is there a possibility anyone did it?

Table 3: SmartThings type of possible artifacts and its possible meaning

Type	Possible information	Forensic implication
Accounts (customer)	Name, email, location	Overall information about the user or customer
Locations (geolocation)	Geolocation (latitude, longitude), location name, devices in that location, time zone, aggregated data in that location collected with the devices.	Where the customer is, the devices installed, names and rooms in that location, what happened in that location (state/event information of the devices in that room).
Rooms/groups	List of rooms, list of devices in a specific room, aggregated data in that room collected with the devices.	What rooms the user have/created, devices installed in that room, what happened in that room (state/event information of the devices in that room)
Door/windows sensor	Door/windows activity, temperature, and other attributes (if any) data.	What time a door/window opened/closed, what kind of activities were around the door/windows and at what time.
Motion sensor	Activities/motion in that house	There was or was not a movement in that location at a specific time.
Temperature sensor	Temperature values	Someone affected the temperature of the house, the temperature activities based on the time table.
Power outlet	Power activities/events	Someone power on/off the device, at a specific time, there were activities in that house.
Automation	Routine data such as opening tv or windows in the morning, turn on/off lights at a given time.	The routine was broken at a given day/time, and there was a problem because the automation was not performed.
SmartApps	Intrusion detection, smoke detection, leaks, automatic locks	Intrusion detected at that day/time; someone tried to open the door, the owner was not around and so on.

5.4.1.2. Data generation

Suppose the above-created scenario and the crime happened on Sunday, June 9, 2019, 7: 00 PM Korean time zone (UTC+9); Sunday, June 9, 2019, 10:00 Universal Time Zone (UTC+0). The following actions and events have happened in each of the SmartThings devices.

- At 6:05 PM (9:05 UTC), someone entered the house. The front door sensor generated an open/close event.
- At 6:06 PM (9:06 UTC), the bedroom door sensor registered opening and closing event.
- At 6:06 PM (9:06 UTC), the smart outlet goes from off to on.
- At 6:30 PM (9:30 UTC), the smart outlet switched off.
- At 6:30 PM, (9:30 UTC), the bedroom door sensor registered open/close event.
- At 6:32 PM (9:32 UTC), the front door sensor recorded open/close event.
- At 7:00 PM, June 9 (June 9, 10:00 UTC), a fire started in the house.

5.4.2. API Identification

The client applications and the cloud communicate through cloud APIs which enable applications and users' access to cloud application services and data. These APIs are REST APIs, and the responses are sent as JSON. From SmartThings cloud, we managed to identify three sets of APIs; official APIs which is for developers and companies, unofficial APIs in which the mobile app and the cloud used to exchange data and commands and the APIs from web-app which is used as a managing and development interface for users and developers.

From the official APIs, we used selected APIs to get the list of locations associated with the user, details of each location, list of devices installed in that specific location, specific device details and its current status — for a complete list of official APIs, see the developer portal [60].

The second set of APIs is the unofficial and undocumented APIs. Strategies mentioned in section 4.3 are used to intercept and uncover it. We can extract cloud-native artifacts such as device and user information, history events data (activities on the devices), installed security automation, and other data from SmartThings cloud.

The third type of APIs are from the SmartThings IDE web-app which allows users to view and edit information such as logical Locations, Hubs, Devices, custom SmartApps, and Device

Handlers, as well as view a live log for all installed SmartThings devices and apps [61]. The web-app also used Both the unofficial and this APIs acquire the same set of data, however, the web-app APIs can be used as an alternative and does not return as structured data as the official and unofficial APIs mentioned above.

Appendix A contains All the APIs. For ease of understanding; we categorized these unofficial APIs into the following API categories: account and user information, locations, devices, room/group, and installed SmartApps. From these categories, there are APIs that return only general information, such as account and user information, list of devices and locations, and so on. In another side, some APIs return events/states of each device, aggregated events/states in a user-created location or rooms and installed SmartApps, including its event/states data and notifications. For instance, the API https://api.smartthings.com/elder/{locationId}/api/devices/{deviceId}/**events**?beforeDate={beforeDate}&max=200&all=false returns specific device's (deviceId) event/states data. The events may be what was the temperature in that room/location, if the device detected motion or if the door/windows were opened in a specific time. Appendix A contains the detailed description of each APIs, how to construct the parameters, and how to get the query parameter.

5.4.3. Acquisition

Both the official and unofficial APIs need a Bearer token to access the cloud resources [60]. The mobile phone accounts database contains the token – located at a mobile phone image path */system/users/0/accounts.db*. However, it expires in a day since last generated or refreshed. Additionally, the access token can be generated from the SmartThings web-app if valid username and password provided. To acquire data, we used the tool developed in this thesis.

Same as the other devices, the tool tends to download all the available user data in SmartThings cloud with valid authentication. For this case study, we acquire the devices data based on the date

of the crime — filtered based on the crime data based on the created scenario. Here is the procedure:

- Provide the parameters, including token information. In this case study, we used a Token recovered from the smartphone app.
- Provide filter data to download that specific date for the events
    - The given time range is; from June 9, 2019, 12:00 AM to June 9, 2019. Twenty-four hours of data is requested to know what happened in that house.
        - Start date => June 9, 2019.
- Download the data — In Figure 14, we can see that we can download a lot of different types of data such as location and account information, devices list, available rooms and installed Hub Figure 14 (A). Besides, data for each installed device such as detail, events, is also downloaded – shown in Figure 14 (B and C). Inside of each devices folder, events data for that device is downloaded such as what happened on the device, at what time both in Unix and iso format, and so on.

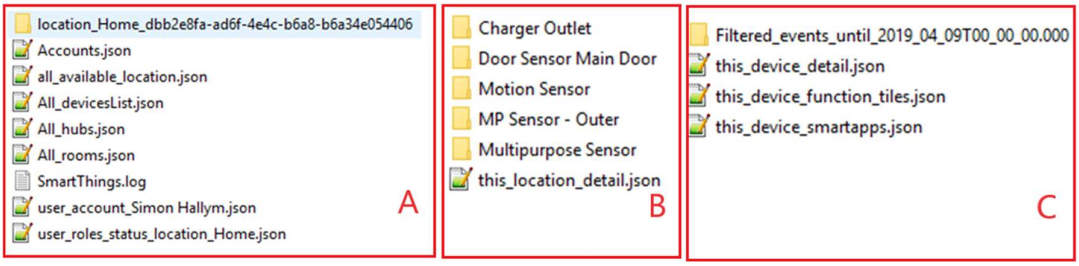

Figure 14: Acquired data from SmartThings cloud using our tool. A) list of data containing general info. B) each device's data in a separate directory. C) events/history data for each device

5.4.4. Analysis

In our acquisition section, we are managed to acquire the artifacts listed in Table 3. More importantly, each device's data, including the events and actions performed on/by the sensors, are obtained.

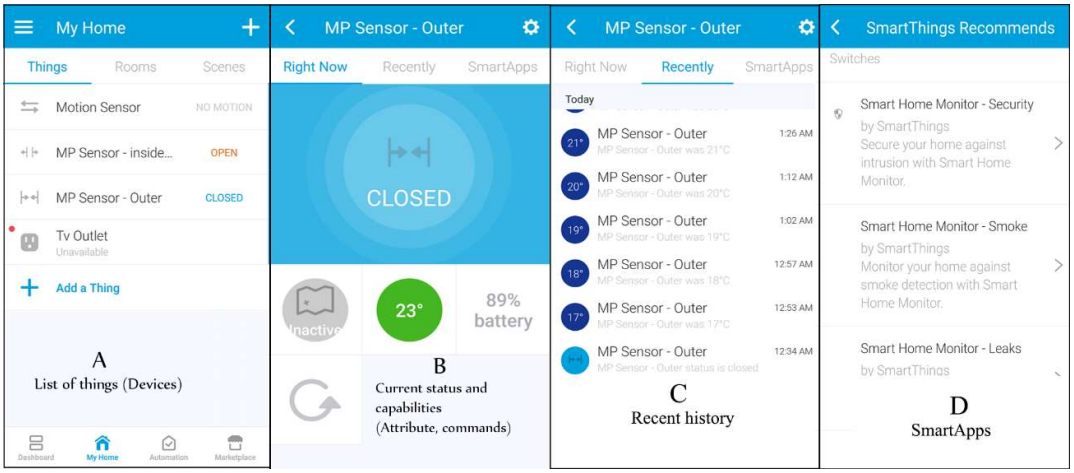

Figure 15: SmartThings classic app interface with devices. A) list of connected devices. B) capabilities of a single device. C) state of the attributes of a device. D) Available SmartApps to automate the devices

Specifically, the data acquired contains, but not limited to, the data shown in Figure 6.9:

- List of things — List of devices are all the devices installed in that user-created location, as shown in 15 (A). It contains values such as the name of the device, its network type (Zigbee/ZWave), its logical locations, and its capabilities — shown in 15 (B). The "*all_devicesList.json*" file contains all this information in a formatted way.
- Events/history data — as shown in Figure 15 (C), events or history data contains recent activity that happened in each of the devices available. The values depend on the capability of the devices such as to measure temperature or sense contacts/motion. In the data acquired, we have got these values in a formatted JSON format; however, it is only the last seven days of data. The JSON file contains interesting values, including the state of that device at a specific time, its description, and so on — shown in Figure 16.

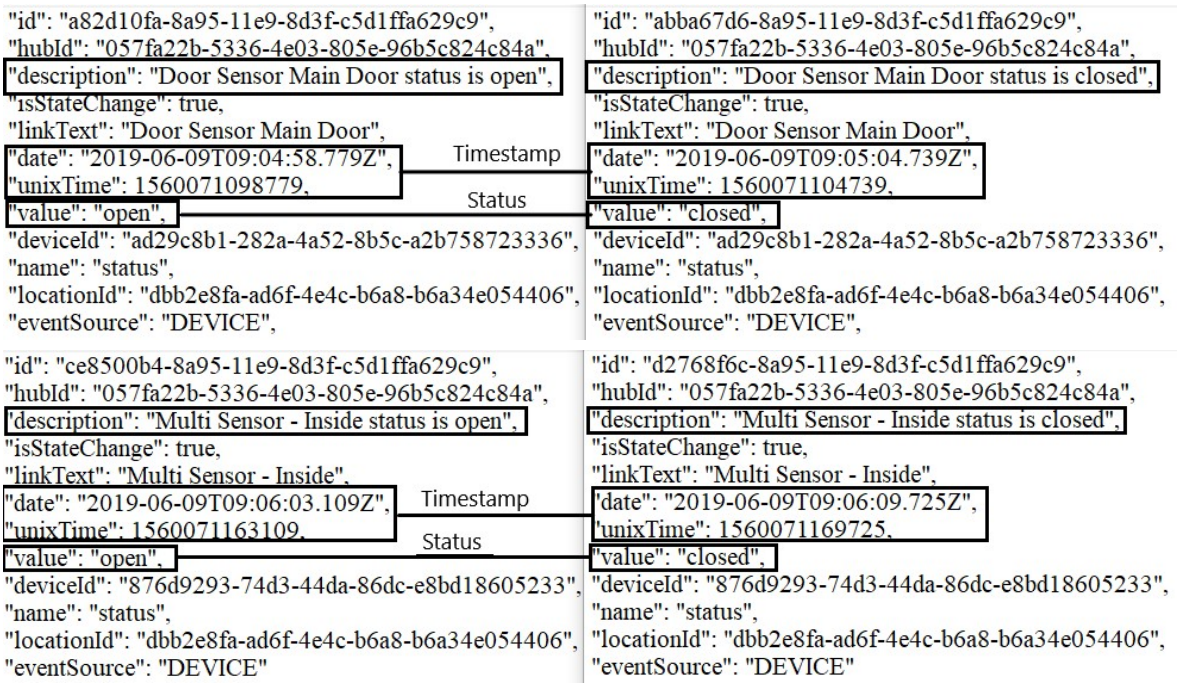

Figure 16: Events or actions data for a door sensor which contains open or closed status and its timestamp

Other than the above data, there are cloud data that contain the rooms the user created, smart app and automation in action, user and account information including the geolocation information, full name and email, and so on. However, for our scenario, we will look into each device event/history information, specifically the door sensors and smart outlet events data. Note that, the timestamp on the application is local time (UTC+9 in our case), while the timestamp on the JSON data is Universal time (UTC + 0).

On the day the event happened (June 9), there was an activity in the house, which seems suspicious because he said he was not around that day. From the sensors data, we can learn that someone was around before the fire broke out. Here is the timeline of that hour before the fire started. At 6:05 PM (9:05 UTC), an action happened on the front door sensor, which is an open/close event. At around 6:06 (9:06 UTC), the bedroom sensor opened/closed while the power outlet (smart outlet) is turned on at the same time. After around 20 min, the bedroom sensor

recorded an open/close event. At the same time, the power outlet turned off. Lastly, at 6:32 (9:32 UTC) PM on the same day, the main/front door sensor registered an open/close event. To answer the investigator question:

- Was Simon around the house on that day? Even if we can not 100% confirm it is Simon, there was someone in the house before the fire happened. As it is shown in Figure 16, the main room and bedroom sensors status (open or close) and its timestamp showed there was an activity before the crime happened. Besides, the power outlet happened to be turned on and off from the SmartThings mobile application.
- Is there a possibility anyone did it? It is possible someone started the fire, or the fire started on accident. However, if there is additional proof that Simon was in the house, it will be more suspicious why he lied to the police and suspected he started the fire on purpose.

In general, investigators could get valuable artifacts from SmartThings cloud data and utilize it to solve crimes as we showed in the above analysis based on a simple scenario.

5.4.5. Evaluation and Conclusion

In terms of the completeness of the data, we managed to download the data generated on section 6.4.3. That means the events from all the devices and sensors can be recovered using the APIs we introduced in this thesis. Moreover, using these APIs, we have managed to acquire all the generated data — and additional — using the tool we designed and developed for this purpose. However, per SmartThings notice, it is good to know that the last seven days of each devices events/history can be acquired [62]. As per the notice, we have tried and failed to acquire data which is older than seven days. Hence, it is better to acquire data as soon as possible if the SmartThings ecosystem seized from the crime scene.

From the above analysis, we showed how the data from the SmartThings cloud could be used to understand the overall activity of the house. Besides, we showed how the data could be used

in the criminal investigation process. Using the data acquired, we have tried to answer the investigator question. However, to prove/disprove Simon was or was not in the house before the crime happened, additional evidence, such as camera feed, location, motion sensor and voice data, may be needed.

5.5. Summary

In this chapter, we acquire cloud-native data using APIs by selecting three popular IoT-related cloud service providers — Amazon Alexa, Google Assistant and SmartThings clouds. In the process, we uncovered their unofficial APIs and used the APIs to download the cloud-centric data. Also, we used a scenario-based analysis and demonstrated how to use the downloaded cloud data in crime investigation. Finally, we evaluated the completeness of the downloaded data based on the action performed and generated in a controlled manner in each of the devices.

CHAPTER 6. DESIGN AND IMPLEMENTATION

Cloud data acquisition needs a tool or automation — manually constructing each API, requesting APIs one by one could be tiresome for investigators — to manage the downloading process from the cloud such as construct each APIs and its corresponding headers, managing the authentication, requesting individual APIs, and parsing out the data. There are web debugging tools to automate this manual work; however, it still takes much time and is not effective for investigators to construct and request each APIs. This section explains the overall design concept and implementation of an integrated tool to acquire cloud-native data from the three IoT Smart Home devices and its cloud service providers.

6.1. Design and Architecture

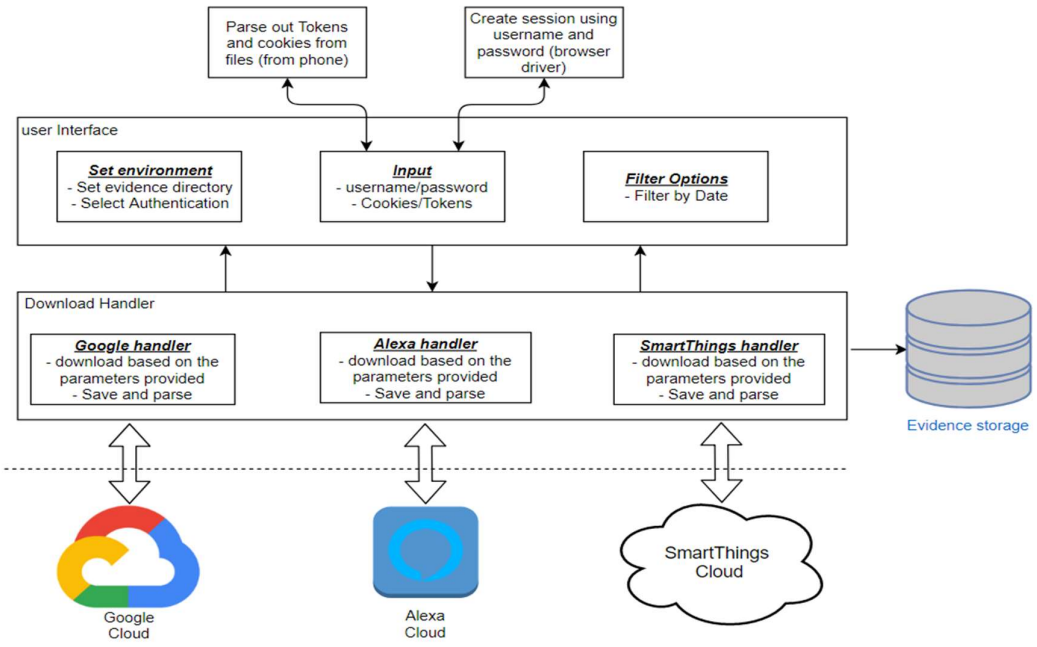

Figure 17: Design and module communication of the cloud downloader tool

The overall architecture of the tool, as shown in Figure 17, consists of two separate modules - user interface and the downloader handler. The user interface is responsible for interaction with the digital forensic analyst. It provides a common interface to execute cloud acquisition

implementation for various IoT Smart Home products. Besides, the user interface module provides basic interface methods for setting up environments, as well as adding and processing user inputs such as setting saving directory for the downloaded data, providing a token or a cookie, setting filtration parameters including date range. Besides, if username and password are provided in place of cookie or token, it extracts cookie by logging in to the intended website using headless Firefox or Chrome browser drivers; headless browser is a web browser without a graphical user interface [63]. In addition to setting user input and parameters, the user interface displays necessary parsed information such as username, email, location, available devices for the selected device environment, and so on.

Based on the input provided on the user interface, the download handler collects acquisition parameters and prepare the acquisition tasks against the cloud target. The module tries to acquire (download) cloud artifacts from the server using the APIs provided for the selected cloud provider – assuming there is a valid authentication information. When the server returns a valid data without error, the module saves them to the provided evidence directory. The handler stores original files to the evidence library; however, it parses out some basic information and parses all the data for Google Assistant data since the data from Google is now in JSON format. Finally, the handler returns basic information to be displayed on the main user interface.

6.2. Implementation and Installation

Based on the proposed design concept, we developed a Python-based program to automate cloud acquisition to help digital forensic investigators and tested on the selected case study IoT-related clouds.

6.2.1. Development Environment

Our tool is developed and tested on Windows 10 (64-bit) with Python 3.6. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. It

supports modules and packages which encourages program modularity and code reuse [64]. Even though the tool is developed on Windows operating system, it should work on any operating system that supports Python 3.4.x and above since Python platform independent language.

6.2.2. Requirements

- Windows machine
- Python > 3.4.x
- Packages:

To install a package in Python, use the PIP installed method — "*pip install package name*". As shown in listing 1, to install selenium, type "pip install selenium" in any command line window.

- Selenium — a web automation framework that can be used to automate website testing and accessing. Here, it is used to login to the cloud provider service and get session/cookie.

```
pip install selenium
```

Listing 1: Installing selenium library for requesting webservers

- requests — HTTP library that enables to send HTTP/1.1 requests using Python. We used it to send API requests to the cloud.
- pickle — a module that implements binary protocols for serializing and de-serializing a Python object structure. Here, it is used to save a web session cookie to a file for future use.
- BeautifulSoup — It is a Python package for scraping and parsing HTML and XML documents. It is used in the tool to parse out HTML values.
- Pandas — Data Analysis Library for python. We used it in the tool to parse data into table format.
- lxml — provides a capability to parse XML and HTML. Here, it is used to access HTML tags and parse out values.

- xlsxwriter — Python module for writing files in the Excel 2007+ XLSX file format. We used it here to write parsed data to excel sheet.
- openpyxl — Python module for manipulating files in the Excel. We used it here to write downloaded data metadata such as time downloaded, path, hash to excel sheet.
- iso8601 — a module to parse ISO 8601 dates. We used it to convert date time value to ISO 8601 format for readability purpose.
- pathlib — it offers classes representing filesystem paths with semantics appropriate for different operating systems. It is used to standardize the directory path for saving and accessing files.

- Chrome and Firefox driver for headless selenium browsing.
- Valid credentials such as token, cookie, or username and password combination are required.

6.2.3. Installation

- Python 3.4.x and above — If Python is not available, download the latest 2.7.x installer for Mac OS X or Windows from https://www.python.org/downloads/. On Linux systems, Python comes pre-installed and ready to use. However, if the python module is less than 3.4, it should be updated to the required version.
- Download and setup browser drivers for selenium such as Chrome or Firefox [65] – Listing 2 contains the drivers download links. After download, put the driver in C drive because the tool looks for the driver in the C drive. One of the Chrome or Firefox browsers should be available on the machine. If not, download and install either the Chrome or Firefox browser.

Firefox: https://github.com/mozilla/geckodriver/releases
Chrome: https://sites.google.com/a/chromium.org/chromedriver/downloads

Listing 2: Firefox and Chrome selenium browser driver download links

- Our tool can be downloaded from its Gitlab repository using the following command:

```
git clone https://github.com/LIFSHallym/Integrated_API_based_cloud_acquisition_IoTDevice
```

Listing 3: Cloning our tool from GitHub repository

- The code can also be downloaded as a zip file if the Git tool is unavailable:

```
https://github.com/LIFSHallym/Integrated_API_based_cloud_acquisition_IoTDevice/archive/master.zip
```

Listing 4: Downloading the zipped source from GitHub

- The text file "requirements.txt" contains all necessary packages. Python package manager pip installs the text file:

```
pip install -r requirements.txt
```

Listing 5: Installing required packages using pip and requirement text file

6.3. Tool Usage

In the following sub-section, we will describe the overall usage of the tool, such as how to use the user interface, how to provide the parameters, and then acquiring the data from the selected cloud service.

6.3.1. Graphical User Interface (GUI)

Once the setup is finished, users can start the main user interface from the command line. The command line should be started from the same path as the source code, or the full source code path should be given, including its name – see Listing 6.

```
- python GUI_IoT_cloudAcquisition.py
- python "path/to/the/source/code/GUI_IoT_cloudAcquisition.py"
```

Listing 6: Running or starting the user interface from the command line

However, if the operating system is Windows, the user interface can be started by double-clicking the "*start_user_interface.bat*" inside the source code. The bat file runs the above command automatically.

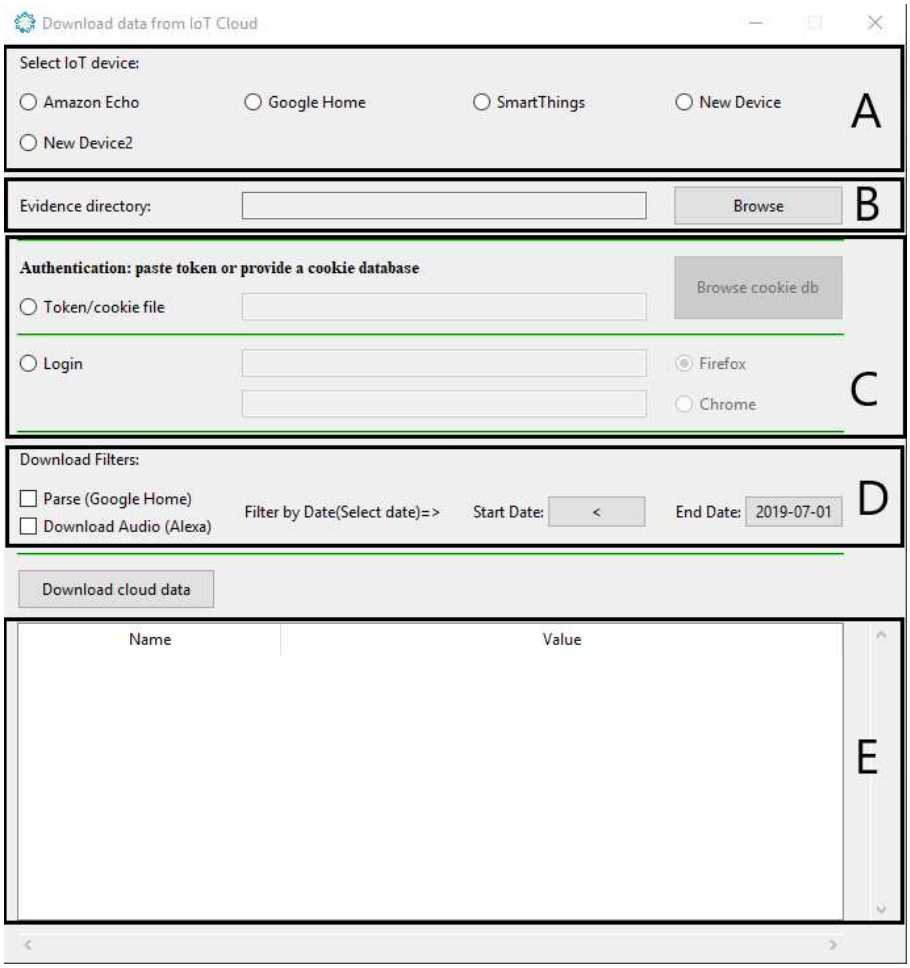

Figure 18: The main user interface of the implemented tool. A) Device/provider selection. B) Directory selection to save the downloaded cloud data. C) Authentication selection such as token, a path to a cookie file, or a username password to extract a session/cook. D) Filtering options. E) Selected information display

The main user interface looks like 18. As it is described in the design section, it has a device selection and user input options. Figure 18 (A) shows where a user selects a device or cloud provider. Figure 18 (B) shows where an investigator chooses a directory to put the downloaded artifacts. A user also should provide an authentication method such as a token, a cookie or a username/password pair — 18 (C). In Figure 18 (D), a user provides a filtration parameter. After setting and selecting all the above parameters and inputs, the user or an investigator should click download cloud data button, as shown in Figure 18 (E). In the end, after finishing the acquisition,

the tool will display selected user information such as owner name, email, location, available connected devices and so on — shown in Figure 18 (F).

6.3.2. Parameter Input

All the parameters should be set to download the cloud data. Otherwise, the program throws an exception or show the error occurred. Here are the parameters detail and how to provide them:

- The user should select a device or a cloud service to download its data. At the moment, the tool supports Amazon Alexa, Google assistant, and SmartThings ecosystem. The tool also can be extended in the future if user or programmers want to add new devices and cloud service providers.
- The user/investigator should provide a directory path to save the downloaded cloud data. If not, it shows an error message.
- Authentication is the most important input because the tool should authenticate to the cloud to access the data. Next section describes the detail.
- The program lets a user provide a parameter to parse, download audio and filter range of data. If the provider is an Amazon Alexa, the tool can download the audio files if the user checks the option to download the audio files. Besides, the data from Google cloud can be parsed to excel sheet if the user select parse checkbox. For all the devices and providers, activity or history data can be filtered based on the given range of date — Figure 18 (D). An investigator may need to download a week data instead of downloading all the available data of all time. To provide date range, click start Date and end date buttons, then, a date selection dialog will appear. Select the date and click the OK button or click Cancel to clear the selection. “Start date” is the date when the data should start while “end date” is the last date of the data. For instance, to return activity/event data from January 1, 2019, to June 30, 2019; select January 1, 2019, for start date, and select June 30, 2019,

as an end date. The parameter “start date” should be before today and less than the “end date” while the “end date” should be today and before.

6.3.3. Authentication

Users should provide authentication information in three ways. If the first authentication options, token/cookie, is selected, a path to a cookie database file or a saved cookie should be provided — Figure 18 (C). Alternatively, a token file should be pasted on the space provided, as shown in Figure 19. If a token or a cookie is not available, a user can provide a valid username and password — Figure 19. With it, the tool can create a session using Chrome or Firefox headless driver and save the session/cookie for later use in the same directory where the tool is located. Keep in mind that not every APIs accept a cookie. If the authentication is invalid or expired, the tool does not download cloud data since the cloud providers need a valid authentication values; at least in our case, all the three need a valid user authentication value.

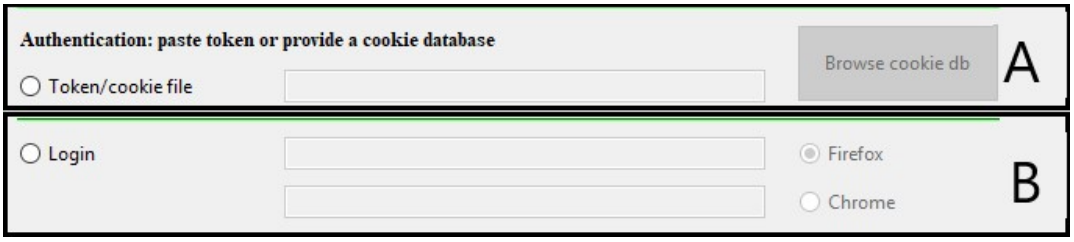

Figure 19: Authentication input. A) Paste token or provide a cookie file. B) Providing username and password

6.3.4. Acquisition and Parsing

If the provided parameters — specifically the authentication — are valid, the tool starts the downloading process. Files downloaded are saved in its original format unless it is Google Home and specified in the filtration parameters. The tool also classifies and put the downloaded file in separate folders based on the type and purpose of the data. For instance, for Amazon Alexa cloud data, general information JSON files are saved in the based directory provided. However, the user activity and calling/messaging data are saved in a separate directory as these files contain rich data — Figure 20 (A). Besides, if the data is downloading for a specific date-range, a new

directory is created for that specific date range data. Besides, as shown in Figure 20 (B), if the information contains an activity/events with a date-range, the file name of the downloaded files includes the start and end date timestamp of the data parsed from inside the file.

For Google Assistant cloud acquisition, the tool can parse out essential information. Since the data from Google Home is unstructured; it is difficult to read and analyze it. For this purpose, the tool can parse main information/values such as the timestamp, the transcript of the user command, the location of the command, and save it in an excel sheet file — shown in Figure 21.

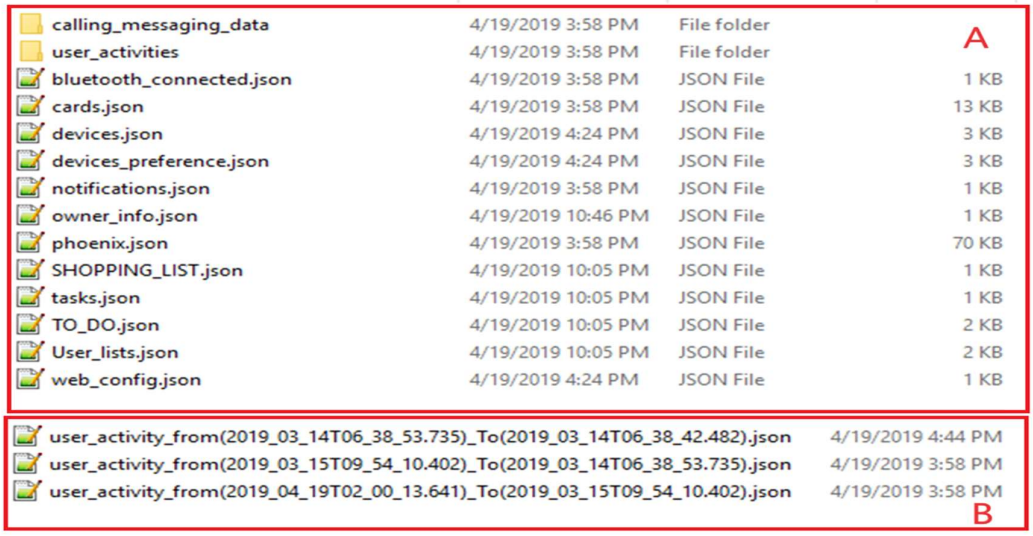

Figure 20: How the tool acquires and store data based on its type and function. A) how the downloaded data is stored. B) file naming based on the date inside the file

	A	B	C	D	E	F	G	H
	eet contains an activity from Google assistant history acquired using activities API. Updated: 2019-04-27T1							
		downloaded_date	timeStamp_unix	timeStamp_Iso	description	answer	command_source	location
0		2019-04-2718:04:29	1.5563E+12	2019-04-27T01:11:21	what is the traffic looks like today	Here are the top results, starting with one from Google	['Google App']	https://www.google.com/maps/@?api=1&map_action=map&center=37.892887,127.734398&zoom=12
1		2019-04-2718:04:29	1.5563E+12	2019-04-27T01:10:59	Assistant	-		https://www.google.com/maps/@?api=1&map_action=map&center=37.892887,127.734398&zoom=12
2		2019-04-2718:04:29	1.5563E+12	2019-04-27T01:10:56	what is the time today	It's 1:10.	['Google Home']	https://www.google.com/maps/@?api=1&map_action=map&center=37.892887,127.734398&zoom=12
3		2019-04-2718:04:29	1.55629E+12	2019-04-27T01:09:53	how is the traffic to the nearest coffee shop	Sorry, I can't give driving directions for that country yet.	['Google Home']	https://www.google.com/maps/@?api=1&map_action=map&center=37.892887,127.734398&zoom=12
4		2019-04-2718:04:29	1.55629E+12	2019-04-27T01:07:10	how is the traffic today	Sorry, I don't know how to help with that.	['Google Home']	https://www.google.com/maps/@?api=1&map_action=map&center=37.892887,127.734398&zoom=12
5		2019-04-2718:04:29	1.55629E+12	2019-04-27T01:06:02	how can I frowed an insurance	Sorry, I don't know how to help with that. But I'm trying to learn.	['Google Home']	https://www.google.com/maps/@?api=1&map_action=map&center=37.892887,127.734398&zoom=12
6		2019-04-2718:04:29	1.55629E+12	2019-04-27T01:04:40	Reuters TV (U.S.)	-		
7		2019-04-2718:04:29	1.55629E+12	2019-04-27T01:04:36	stop	-	['Google Home']	https://www.google.com/maps/@?api=1&map_action=map&center=37.892887,127.734398&zoom=12
8		2019-04-2718:04:29	1.55629E+12	2019-04-27T01:01:33	play news	Here's the latest news.	['Google Home']	

Figure 21: Parsed user activity data from the Google Assistant (Home) data

Finally, the tool contains a log file which tracks and logs every functionality and action of the downloading process — Figure 22. The log file documents what kind of actions performed, at

what time, where the files are saved, including its full path. Other than the log file, the tool also tracks and saves all the downloaded file information such as, the downloaded time, the path of the downloaded data, and the file md5 hash, into an excel sheet file. These files could be useful for debugging as well as reporting purpose.

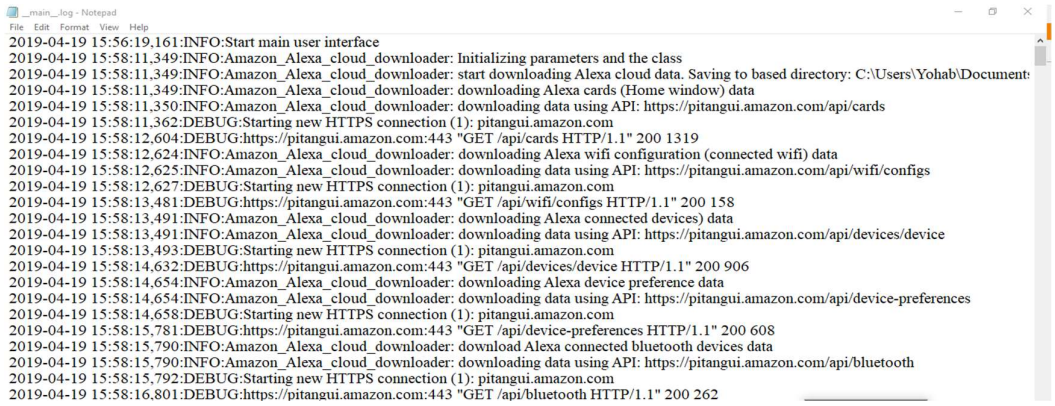

Figure 22: Logging information from the downloading process

6.4. Feature Support

Unofficial APIs are private APIs that can be changed by the providers without notice – discussed in section 4.2. The tool uses these APIs to acquire the cloud-based data for IoT-related cloud environment. If the APIs are updated and changed, the tool generates an error. Hence, we will continue to support the tool and the APIs in case of update from the providers. In other words, we will follow the API updates in the future if any of the case study devices and environments update their APIs, and then update the tool to support the new APIs. Besides, we will release the source code for the tool to the public, and law enforcement agencies and digital forensics communities can update the tool, including researching and updating the APIs.

6.5. Summary

We provide a tool to assist investigators in downloading cloud-centric data from the selected cloud service providers. It is GUI-based easy to configure and to use a standalone tool. Besides, it is designed for future add-ins in mind in order to accommodate any additional devices and cloud

services. User or investigators should provide the necessary parameters including a valid user authentication such as username/password, token or a cookie so that the tool performs as intended. If needed, the tool can parse out basic data to make it easy to understand and construct user activity. Also, it logs every activity of the tool, including its timestamp. Besides, every downloaded file is tracked and logged to excel file, including the downloaded time, the file type, the path to the downloaded file, and its hash value. These files could be useful for reporting purposes.

CHAPTER 7. CONCLUSION AND FUTURE WORK

7.1. Conclusions

Previous works and investigations are done based on traditional approaches such as finding and recovering forensically relevant information from remnants left by client applications or user interaction. However, the client-side application data could be limited, incomplete or out of date as most of the data saved in client applications are cache and configuration files. In this case, cloud service data could be a great source of evidence which could fill the client-side limitation. In this thesis, we showed how to acquire data from IoT-related cloud service using cloud communication APIs; the APIs could be official which is public or unofficial that is hidden and private. The main contribution of the study is acquiring cloud-centric potential evidences from Smart Home environment using these identified APIs and develop a cloud data downloader tool to help investigators.

As a case study, we selected the three most popular smart Home related cloud providers – Amazon Alexa, Google Home Assistant and SmartThings – to extract and utilize the APIs to acquire cloud data. To uncover hidden communication APIs, we used different methods, such as reverse engineering, man-in-the-middle attack, and web debugging tools, code injection, and dynamic analysis. As a result, we are managed to acquire data stored in each of the cloud services.

The data collected by Alexa – from timestamps to user voice data – has the potential to provide law enforcement with the ability to further their investigation. In this thesis, we showed that these data could be acquired from Alexa cloud service using the unofficial APIs. The data, which are in JSON format, contained timestamps, text descriptions, transcript of the voice command for activities, to-dos, and a shopping list of the user. Furthermore, data such as conversations, contacts, and voice messages from the calling and messaging feature. These

acquired data could help investigators in reconstructing events and studying user behavior in combination with other sources of evidence.

From Google cloud, it is possible to download user activity data related to Google Home device and Google Assistance, such as command issued to the device, user information, and audio file (if available) using the APIs presented in this thesis. Same as Alexa, using Google Assistance cloud data, user activity can be constructed including what time the user issued a particular command, what was the intent of the command such as asking traffic/commute information go somewhere, or shopping for a specific item used to commit a crime and so on.

From SmartThings, we showed how to obtain Hub and sensors data, and other metadata from the cloud in JSON format using the official/unofficial APIs. In this thesis, we have managed to obtain potential evidence, such as the activity of the account holder, the events performed on/by the sensors, including the timestamp, and any automation created by the user such as intrusion detection. These potential evidences, in turn, can help to construct the Smart Home activities using the sensors data installed in that particular house. For instance, the motion sensor records the state of a particular space in the house; the record contains values which indicates whether there was a motion or not at a given time – obtaining this data help to answer questions such as whether someone was at home at a given time.

Based on our research, we developed a tool which automates the API-based cloud data acquisition process for the selected IoT-related cloud providers; the tool is also open for the future add-on to add additional devices and environments. It provides a user interface with an option for the investigator to input authentication information, evidence storage directory, and date range to download a range of data based on the time range given. Then, based on these inputs and filter parameters, it downloads and parses data from the target IoT cloud service. One of the challenges in developing a tool for API-based acquisition for cloud-centric data is that each IoT devices and environments use different APIs and structures. Hence, it is needed to research each IoT-related

cloud providers and its APIs, and then develop a tool to acquire data based on the research. All three case study environments in our thesis, for example, use different APIs. The Other challenge is that unofficial APIs could be changed at any time by the providers; hence it needs to continue research and maintenance in case the APIs are changed and updated. As our tool is open-source, it is open for an update from the digital forensic community; we will continue to support and maintain the tool and the APIs if there are any change and update on the selected case study providers.

In this thesis, we have tried to answer the research questions we coined at the start of our research. Here is the comprehensive answer for each of the questions asked in section 1.3:

RQ1: Can data be collected in a forensically-sound way from cloud service data from IoT devices related to the smart home environment using APIs?

As it is stated in this research, the IoT ecosystem works in a connected environment; any information from one device is sent to a cloud and then synced to another companion applications or web-apps. During these communications, data cloud can be saved in each of the involved ecosystems. Hence, data from any of these data source could be a potential data source. However, the most reliable and convenient place to get cloud data is the device's backend cloud storage. In this thesis, we answered the question by showing how to acquire cloud service data using the provided or uncovered communication APIs.

RQ2: **How much data can we obtain in terms of completeness?**

The amount of data provided depends on the cloud service providers police. However, if it is provided to the user using client applications, it is possible that data is saved on the cloud and provided in same way to users. In this case, with the right research on the communication methods and APIs, we can get all the data saved on the cloud except deleted data. In our case study devices, we have managed to get all the data saved in the cloud for each of the devices. However, if the

cloud is not saving data or remove it after some time – SmartThings cloud does not save events data more than seven days of data – it is impossible to get all the data.

In general, based on our research, we can say that Smart Home IoT environments use APIs to transport data between the ecosystems. Hence, these APIs can be used to acquire data from their cloud service that can help digital forensics investigation process. However, because each device and ecosystem contain different APIs, it is necessary to study and research to extract their communication APIs. In our research, all the case study IoT-related cloud providers had APIs, and we have managed to get forensically relevant data from their cloud services using these APIs.

7.2. Future Work

As future work, we plan to implement and generate timeline information from the acquired data. Besides, parsing the acquired data in a way that is easy to analyze, read and report is our future work. Finally, we put a placeholder in our tool for additional IoT devices. We have a plan to look into additional IoT devices and cloud providers and add to the tool. It is also open for anyone interested to add or to improve the tool; we will make the source code open soon when we have finished the final touches.

REFERENCES

[1] K. Lueth, “Why it is called Internet of Things: Definition, history, disambiguation,” 2014. .

[2] “Global Internet of Things market size 2009-2019 | Statistic,” *Statista*, 2019. [Online]. Available: https://www.statista.com/statistics/485136/global-internet-of-things-market-size/. [Accessed: 12-Jan-2019].

[3] E. Oriwoh, “A Smart Home Anomaly Detection Framework,” p. 193, 2015.

[4] “SmartThings.,” *SmartThings.com*, 2019. [Online]. Available: https://www.smartthings.com/. [Accessed: 13-Jan-2019].

[5] G. Clauser, “What Is Alexa? What Is Amazon Echo, and Should You Get One?,” *Wirecutter: Reviews for the Real World*, 2018. .

[6] “Smart Home Lighting,” *Philips Hue*, 2019. [Online]. Available: https://www2.meethue.com/en-us. [Accessed: 13-Jan-2019].

[7] Nest, “Nest Learning Thermostat | Programs Itself Then Pays for Itself,” *Nest*, 2019. [Online]. Available: https://www.nest.com/thermostats/nest-learning-thermostat/overview/. [Accessed: 13-Jan-2019].

[8] “Google Home,” *Google Store*, 2019. [Online]. Available: https://store.google.com/product/google_home. [Accessed: 13-Jan-2019].

[9] V. C. PINACL, “WiFi, Internet of Things, Network Infrastructure,” *Pinacl Solutions*, 12-Feb-2019. [Online]. Available: https://pinaclsolutions.com/blog/2017/cloud-computing-and-iot. [Accessed: 13-Feb-2019].

[10] J. I. James and Y. Jang, “Practical and Legal Challenges of Cloud Investigations,” *The Journal of the Institute of Webcasting, Internet and Telecommunication*, vol. 14, no. 6, pp. 33–39, Dec. 2014.

[11] M. Smith, “Cops to increasingly use digital footprints from IoT devices for investigations,” *CSO Online*, 02-Jan-2017. [Online]. Available: https://www.csoonline.com/article/3154064/cops-to-increasingly-use-digital-footprints-from-iot-devices-for-investigations.html. [Accessed: 28-Feb-2019].

[12] C. Mu-Hyun, “Explosion in digital evidence coming thanks to IoT and 5G: Hancom GMD,” *ZDNet*, 2019. [Online]. Available: https://www.zdnet.com/article/explosion-in-digital-evidence-coming-thanks-to-iot-and-5g-hancom-gmd/. [Accessed: 18-Jan-2019].

[13] S. Kleeman, “Woman Charged After Her Fitbit Contradicted Her Rape Claim,” 2015. [Online]. Available: https://mic.com/articles/121319/fitbit-rape-claim. [Accessed: 12-Jan-2019].

[14] C. Hauser, “Police Use Fitbit Data to Charge 90-Year-Old Man in Stepdaughter’s Killing,” *The New York Times*, 04-Oct-2018.

[15] E. C. McLaughlin, “Suspect OKs Amazon to hand over Echo recordings in murder case,” *CNN*, 2017. [Online]. Available: https://www.cnn.com/2017/03/07/tech/amazon-echo-alexa-bentonville-arkansas-murder-case/index.html. [Accessed: 13-Jan-2019].

[16] “Cloud API.” [Online]. Available: https://www.techopedia.com/definition/26437/cloud-application-programming-interface-cloud-api. [Accessed: 11-Aug-2018].

[17] scar, “Current Challenges In Digital Forensics,” *Forensic Focus - Articles*, 11-May-2016. .

[18] H. Chung, J. Park, S. Lee, and C. Kang, “Digital forensic investigation of cloud storage services,” *Digital Investigation*, vol. 9, no. 2, pp. 81–95, Nov. 2012.

[19] J. S. Hale, “Amazon Cloud Drive forensic analysis,” *Digital Investigation*, vol. 10, no. 3, pp. 259–265, Oct. 2013.

[20] D. Quick and K.-K. R. Choo, “Digital droplets: Microsoft SkyDrive forensic data remnants,” *Future Generation Computer Systems*, vol. 29, no. 6, pp. 1378–1394, Aug. 2013.
[21] D. Quick and K.-K. R. Choo, “Dropbox analysis: Data remnants on user machines,” *Digital Investigation*, vol. 10, no. 1, pp. 3–18, Jun. 2013.
[22] D. Quick and K.-K. R. Choo, “Google Drive: Forensic analysis of data remnants,” *Journal of Network and Computer Applications*, vol. 40, pp. 179–193, Apr. 2014.
[23] P. M. Cassandra Apache, “Internet of Things: Where Does the Data Go?,” *Wired*, 09-Mar-2015.
[24] H. Chung, J. Park, and S. Lee, “Digital forensic approaches for Amazon Alexa ecosystem,” *Digital Investigation*, vol. 22, no. Supplement, pp. S15–S25, Aug. 2017.
[25] V. Roussev, A. Barreto, and I. Ahmed, “Forensic Acquisition of Cloud Drives,” *arXiv:1603.06542 [cs]*, Jan. 2016.
[26] V. Roussev and S. McCulley, “Forensic analysis of cloud-native artifacts,” *Digital Investigation*, vol. 16, pp. S104–S113, Mar. 2016.
[27] I. Yaqoob, I. A. T. Hashem, A. Ahmed, S. M. A. Kazmi, and C. S. Hong, “Internet of things forensics: Recent advances, taxonomy, requirements, and open challenges,” *Future Generation Computer Systems*, vol. 92, pp. 265–275, Mar. 2019.
[28] R. C. Hegarty, D. J. Lamb, and A. Attwood, “Digital Evidence Challenges in the Internet of Things,” p. 10.
[29] M. Conti, A. Dehghantanha, K. Franke, and S. Watson, “Internet of Things security and forensics: Challenges and opportunities,” *Future Generation Computer Systems*, vol. 78, pp. 544–546, Jan. 2018.
[30] I. Clinton, L. Cook, and S. Banik, “A Survey of Various Methods for Analyzing the Amazon Echo,” 2015.
[31] J. Hyde and B. Moran, “Amazon Alexa, Are You Skynet? – Cyber Forensicator,” 2017. .
[32] K. M. S. Rahman, M. Bishop, and A. Holt, “Internet of Things Mobility Forensics,” p. 8, 2016.
[33] D. A. Orr and L. Sanchez, “Alexa , did you get that? Determining the evidentiary value of data stored by the Amazon® Echo,” *Digital Investigation*, vol. 24, pp. 72–78, Mar. 2018.
[34] Android, “Android Debug Bridge (adb) | Android Developers.” [Online]. Available: https://developer.android.com/studio/command-line/adb. [Accessed: 13-Jan-2019].
[35] “Autopsy | Open Source Digital Forensics.” [Online]. Available: https://www.autopsy.com/. [Accessed: 14-Jan-2019].
[36] L. Li *et al.*, “Static analysis of android apps: A systematic literature review,” *Information and Software Technology*, vol. 88, pp. 67–95, Aug. 2017.
[37] K. Yang, “Introduction to Dynamic Analysis of Android Application,” 2012.
[38] “IFTTT.” [Online]. Available: https://ifttt.com/. [Accessed: 09-Aug-2018].
[39] G. C. K. and M. Fasulo, “The Case for Teaching Network Protocols to Computer Forensics Examiners: Part 1,” *Forensic Magazine*, 2013. [Online]. Available: https://www.forensicmag.com/article/2013/05/case-teaching-network-protocols-computer-forensics-examiners-part-1. [Accessed: 02-Mar-2019].
[40] “Google Drive REST API v2,” *Google Developers*. [Online]. Available: https://developers.google.com/drive/api/v2/reference/. [Accessed: 07-Apr-2019].
[41] Gregg, “Microsoft Graph API - OneDrive dev center.” [Online]. Available: https://docs.microsoft.com/en-us/onedrive/developer/rest-api/. [Accessed: 07-Apr-2019].
[42] Postman, “Postman,” *Postman Learning Center*. [Online]. Available: https://learning.getpostman.com/docs/postman/launching_postman/installation_and_updates/. [Accessed: 02-Jun-2019].

[43] “Firefox Network Monitor,” *MDN Web Docs*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Tools/Network_Monitor. [Accessed: 07-Apr-2019].
[44] Google Chrome, “Chrome DevTools Overview - Google Chrome.” [Online]. Available: https://developer.chrome.com/devtools. [Accessed: 09-Aug-2018].
[45] Imperva, “What is MITM (Man in the Middle) Attack,” *Learning Center*. .
[46] “Burp Suite Scanner | PortSwigger.” [Online]. Available: https://portswigger.net/burp. [Accessed: 22-Mar-2019].
[47] “TLS Certificate Pinning 101,” *Nettitude Labs*, 13-Mar-2018. [Online]. Available: https://labs.nettitude.com/tutorials/tls-certificate-pinning-101/. [Accessed: 07-Apr-2019].
[48] “Android Applications Reversing 101,” *evilsocket*. [Online]. Available: https://www.evilsocket.net/2017/04/27/Android-Applications-Reversing-101/index.html. [Accessed: 07-Apr-2019].
[49] “Android Anti-Reversing Defenses.” [Online]. Available: https://mobile-security.gitbook.io/mobile-security-testing-guide/android-testing-guide/0x05j-testing-resiliency-against-reverse-engineering. [Accessed: 03-Jun-2019].
[50] G. Vázquez, “An Introduction to API’s,” *The RESTful Web*, 26-Aug-2015. [Online]. Available: https://restful.io/an-introduction-to-api-s-cee90581ca1b. [Accessed: 05-Jun-2019].
[51] M. Chin and M. Prospero, “Best Smart Home Devices of 2019,” *Tom’s Guide*, 02-Apr-2019. [Online]. Available: https://www.tomsguide.com/us/best-smart-home-devices,review-2008.html. [Accessed: 30-Apr-2019].
[52] K. Wetzel, “What exactly is Alexa? Where does she come from? And how does she work?,” *Digital Trends*, 16-Feb-2019. [Online]. Available: https://www.digitaltrends.com/home/what-is-amazons-alexa-and-what-can-it-do/. [Accessed: 20-Mar-2019].
[53] J. James, “IoT Forensic Challenge, 2017-2018,” *DFRWS 2017-2018 Forensic Challenge*, 2017. [Online]. Available: https://jijames.github.io/DFRWS2018Challenge/. [Accessed: 12-Apr-2019].
[54] B. Moran, “Amazon Alexa Forensic Walkthrough Guide,” 2017. .
[55] A. Grush, “Google Home and Assistant,” 2018. [Online]. Available: https://dgit.com/google-home-assistant-59577/. [Accessed: 24-Mar-2019].
[56] A. Gebhart, “Google Assistant expands to a billion devices and 80 countries,” *CNET*, 2019. [Online]. Available: https://www.cnet.com/news/google-assistant-expands-to-a-billion-devices-and-80-countries/. [Accessed: 14-Apr-2019].
[57] L. Tung, “Google’s new My Activity page now displays your whole online life,” *ZDNet*, 2016. [Online]. Available: https://www.zdnet.com/article/googles-new-my-activity-page-now-displays-your-whole-online-life/. [Accessed: 25-Mar-2019].
[58] “Z-Wave and ZigBee FAQ,” *SmartThings Support*. [Online]. Available: http://support.smartthings.com/hc/en-us/articles/208672926-Z-Wave-and-ZigBee-FAQ. [Accessed: 01-Apr-2018].
[59] “Architecture — SmartThings.” [Online]. Available: https://docs.smartthings.com/en/latest/architecture/#containers. [Accessed: 23-Mar-2019].
[60] “SmartThings API.” [Online]. Available: https://smartthings.developer.samsung.com/docs/api-ref/st-api.html#. [Accessed: 24-Mar-2019].
[61] “SmartThings IDE,” 2018. [Online]. Available: https://docs.smartthings.com/en/latest/tools-and-ide/account-mgmt.html. [Accessed: 24-Mar-2019].

[62] “Device event — SmartThings.” [Online]. Available: https://docs.smartthings.com/en/latest/ref-docs/device-ref.html. [Accessed: 14-Apr-2019].
[63] A. GIRDWOOD, “What is a headless browser?,” 2009. .
[64] “What is Python?,” *Python.org*. [Online]. Available: https://www.python.org/doc/essays/blurb/. [Accessed: 21-Apr-2019].
[65] “Installation — Selenium Python Bindings 2 documentation.” [Online]. Available: https://selenium-python.readthedocs.io/installation.html. [Accessed: 27-Apr-2019].

API-Based Cloud Acquisition and Analysis from Smart Home IoT Environments

2019.
Master's Degree
Brhan, Yohannes Yemane
Department of International Studies
Advisors: prof. Jang, Yunsik, prof. Joshua I. James

The increasing number of IoT devices used in different application domains is changing the digital forensics landscape by providing a variety of potential data and data sources. However, Current forensic tools and techniques have been slow to adapt to new challenges and demands of collecting and analyzing IoT environment artifacts. Like the traditional forensics, data acquisition is possible from the client-side, network or cloud service in the IoT ecosystem. However, unlike computer forensics, IoT forensics does not normally contain much data on the client-side; since the device's storage is usually limited, the client devices may not save much data except some cache and configuration files. In this case, their cloud service could normally be a great source of potential evidence.

Investigators access cloud service data in different ways. They can either attempt to access the cloud service data themselves by authenticating as a user or collaborate with the Cloud Service Provider to collect data. In this thesis, we studied the acquisition and analysis of cloud data from IoT environments to addresses the limitation of client-side acquisition. Specifically, we introduce an acquisition procedure for IoT-related cloud services using Application Programming Interfaces (APIs). Some IoT devices and its cloud services provide APIs for programmers and user while

others not. We use both official APIs given by cloud providers and unofficial APIs in which we used different research methods to uncover it.

There are previous works that showed and used these official and unofficial APIs for data acquisition purposes. However, every IoT devices and cloud providers have their own API type and structure. Hence, we used the three most popular IoT-related cloud providers – Amazon Alexa, Google Home and SmartThings – to show how to use APIs to acquire cloud-native artifacts from their respective backend cloud service. Using the APIs, we can obtain cloud-centric data from each selected case study cloud services. User command history, user activities, audio files and other additional data obtained from Amazon Alexa cloud. It is also possible to download user activity data, including command issued to the device from Google cloud related to Google Home. From SmartThings cloud, we able to download information, including each device events/action performed on the sensors, user-created locations, Hubs, account detail, rooms information, etc.

To help investigators in automating the acquisition process, we designed and developed a python application that connects and retrieve every possible information stored in their respective cloud services. The application is a user interface-based tool with an option to select different filtration parameters, authentication option, data parsing. The tool also provides logging every action and documenting all the downloaded file metadata, including the calculated hash.

Further, to evaluate the completeness of the acquired cloud data, we generated data in a controlled environment with specific set of actions and scripts, then compare the API-based acquired data with the generated data.

Keywords: IoT forensics, Cloud forensics, Amazon Alexa forensics, SmartThings forensics, Google Assistant forensics, API-based cloud data acquisition, cloud acquisition tool.

스마트홈 IoT 환경에서의 API 기반 클라우드 데이터 획득 및 분석

2019.
석사학위논문
요하네스 예멘 브라한
국제학과
지도교수: 장윤식, Joshua I. James

IoT 기기의 적용 범위가 넓어지면서 기기의 수와 함께 제공되는 다양한 데이터와 데이터소스에 따라 디지털 포렌식 분야도 함께 변화해가고 있다. 현존하는 포렌식 도구와 기술들은 IoT 환경에서 생성되는 아티팩트를 획득하고 분석하기 위해 생기는 새로운 요구들과 문제점들에 대해 천천히 적응해가고 있다. 기존의 컴퓨터 포렌식과는 다르게, IoT 포렌식의 경우 클라이언트 측에 많은 데이터를 저장하지 않는다. 보편적으로 기기의 저장용량이 제한되어 있기 때문에, 클라이언트 측 데이터는 캐시나 환경설정 파일들 이외의 데이터는 많이 저장되지 않는 편이다. 이와 같은 상황에서의 IoT 클라우드 서비스에는 잠재적 증거들이 저장되어 있을 가능성이 높다.

본 논문에서는 클라이언트 측에서의 획득 한계점을 다루기 위해 IoT 환경에서의 클라우드 데이터 획득과 분석을 연구한 내용을 다뤘다. 특히 어플리케이션 프로그래밍 인터페이스(APIs)를 사용한 IoT 관련 클라우드 서비스 획득 방법에 대해서 소개할 것이다. 일부 IoT 기기와 클라우드 서비스는 프로그래머와 사용자를 위한 API 를

제공하지만, 그렇지 않은 기기와 서비스도 있다. 그에 따라, 본 논문에서는 위의 내용을 밝혀 내기 위하여 클라우드 서비스 제공자들이 제공한 공식적인 API 와 다른 연구 방법을 통해 비공식적인 API 를 또한 사용하였다.

본 논문에서는 가장 많이 사용되는 IoT 관련 클라우드 서비스 제공자인, 아마존 알렉사, 구글 홈, 삼성 SmartThing 의 API 를 사용하여, 각각의 백엔드 클라우드 서비스를 통하여 생성된 클라우드 고유 아티팩트를 얻는 방법을 보여줄 것이다. 클라우드 포렌식 아티팩트 획득 자동화를 위하여, 본 논문에서 각각의 클라우드 서비스에 연결해 저장되어 있는 모든 정보를 검색할 수 있는 파이썬 응용프로그램을 구상 및 개발 했다. 해당 응용프로그램은 사용자 인터페이스 기반 도구로써 사용자가 다른 필터를 매개변수로 고를 수 있고, 그 이외에 인증 옵션과 데이터 파싱 기능이 있다. 이 도구는 모든 작업을 로그형태로 기록하고 계산된 해시가 모든 다운로드 된 파일의 메타데이터에 기록된다.

또한 획득된 클라우드 데이터의 완성도를 평가하기 위해 특정 스크립트를 사용하여 제어된 환경에서 데이터를 생성한 다음, 연구에서 획득한 데이터와 생성된 데이터를 비교한다.

주제어: IoT 포렌식, 클라우드 포렌식, 아마존 알렉사 포렌식, SmartThing 포렌식, 구글어시스턴트 포렌식

APPENDIXES

Appendix A: SmartThings Cloud APIs (Official and Unofficial)

No	*Name*	*APIs; change the parameter in the {} with the right value*	*Parameters: get the parameters from the one of the APIs stated.*	*Description*
Samsung SmartThings Official APIs				
1	Locations	https://api.smartthings.com/***locations***	--	**Locations**: list of locations where the device is deployed
2	Specific location	https://api.smartthings.com/locations/{***locationId***}	**<locationId>** from API *"no 1"*	Get a specific Location from a user's account.
3	Devices List	https://api.smartthings.com/***devices***	--	Get a list of devices in this location.
4	specific device detail	https://api.smartthings.com/devices/{**deviceId**}	**<deviceId>** from API "*no 3*"	Get specific device's description based on its device ID
5	Devices current status	https://api.smartthings.com/devices/{***deviceId***}/**status**	**<deviceId>** from API "*no 3*"	Get the full status of a device.
6	List installed SmartApps	https://api.smartthings.com/v1/**apps**	**--**	Get a list of Smartapps
7	Single SmartApp	https://api.smartthings.com/v1/**apps**/{***appNameOrId***}	***<appNameOrId>*** API *"no 6"*	Detail of the single Installed SmartApp
8	Token generating URL	https://account.smartthings.com/**tokens**	—	Generate tokens
Samsung SmartThings Unofficial APIs				
9	Accounts	https://api.smartthings.com/elder/{***locationId***}/api/accounts	**<locationId>** from API *"no 1"*	Account information about that specific location, it may be the owner or somebody else that is added to that location
10	Single account detail	https://api.smartthings.com/elder/{***locationId***}/api/**accounts**/{***accountId***}	< **locationId>** from API "no *1*" < **accountId>** from API *"no 9*"	Detail information about specific account holder

11	user roles	https://api.smartthings.com/elder/{***location Id***}/api/**accounts**/{***accountId***}/roles	< **locationId>** from API *"no 1"* < **accountId>** from API *"no 9"*	User roles on this current owner, owner can give permission to one or more uses (employee, relatives).
12	User information	https://auth-global.api.smartthings.com/**users/me**	—	User information such as user name, email, uuid, and full name
13	Location dashboard	https://api.smartthings.com/elder/{***location Id***}/api/locations/{***locationId***}/**dashboard**	< **locationId>** from API *"no 1"*	Dashboard form this specific location. It is "Home" screen on the smartphone app.
14	Location features	https://api.smartthings.com/elder/{***location Id***}/api/locations/{***locationId***}/**features**	< **locationId**> from API "no 1*"*	Enabled features for this location such as CONTACT_BOOK, Device_watch.
15	Notifications	https://api.smartthings.com/elder/{***location Id***}/api/locations/{***locationId***}/**notifications**	< **locationId>** from API *"no 1"*	Notifications for this location
16	More notifications	https://api.smartthings.com/elder/{***location Id***}/api/locations/{***locationId***}/**notifications**?max=20&beforeDate={***date***}	< **locationId>** from API *"no 1"* < **beforeData>** from the last item that is returned from the notification API.	Get more notifications based on the parameters provided. For **example:** to get more than <**max**> notifications (or max size), get "**date**" value from the last notification and put on the ***beforeDate*** parameter.
17	Rooms list/tiles	https://api.smartthings.com/elder/{***location Id***}/api/locations/{locationId}/**rooms/tiles**	<**locationId>** from API *"no 1"*	Get list of rooms created by the users in this location
18	Location Events	https://api.smartthings.com/elder/{***location Id***}/api/locations/{***locationId***}/events?beforeDate={***beforeDate***}&max=200&all=false	< **locationId>** from API *"no 1"* < **beforeData>** from the last item that is returned from the first/latest events. - Leave **beforeDate** empty to return the first max size.	- Events happened in this location such as door closed, motion detected, temperature change, and so on. - Only last 7 days events can be obtained
19	Devices list/tiles	https://api.smartthings.com/elder/{***location Id***}/api/devices/tiles?locationId={***locationId***}	< **locationId>** from API *"no 1"*	Get/show list of devices in this location. Show in tiles format in the application
20	Hubs list	https://api.smartthings.com/elder/{***location Id***}/api/hubs/	< **locationId>** from API *"no* 1"	get list of hubs installed in this location
21	Specific hub	https://api.smartthings.com/elder/{***location Id***}/api/hubs/**{hubId}**	< **locationId>** from API *"no 1* < **hubId>** from API *"no 20"*	detail about the specific hub, which includes the id of zigbee, zwave, id, uptime, and more
22	device detail	https://api.smartthings.com/elder/{***location Id***}/api/devices/{***deviceId***}	< **locationId>** from API *"no 1"*	detail of specific device (sensors, actuator) including its current state

23	device tile	https://api.smartthings.com/elder/{***locationId***}/api/devices/{***deviceId***}/mainTile	<**deviceId>** from API *"no 3" or "no 19"*	Show/get all the actions and attribute this device can perform such as battery level, refresh, restart and more
24	current device SmartApps	https://api.smartthings.com/elder/{***locationId***}/api/devices/{***deviceId***}/pages/smartApps		installed smartApps on this specific device such as detect intrusion, temperature increase, or other
25	device events	https://api.smartthings.com/elder/{***locationId***}/api/devices/{***deviceId***}/events?beforeDate={***beforeDate***}&max=200	< **locationId>** from API *"no 1"* <**deviceId>** from API *"no 3/19"* < **beforeData> (**named **'date')** from the last item that is returned from the first/latest event. - Leave **beforeDate** empty to return the first max size events.	- Events happened in this location using this device such as door closed, motion detected, temperature change, and so on. - Only last 7 days events can be obtained
26	device events (alternative API)	https://api.smartthings.com/elder/{***locationId***}/api/devices/{***deviceId***}/events?max=200&startAfter={***eventId***}	< **locationId>** from API *"no 1"*. <**deviceId>** from API *"no 3/19"* - Get **eventId** from the last item that is returned from the first/latest events which is returned by leaving this parameter empty. - Leave **eventId** empty to return the first max size.	- Events happened in this location using this device such as door closed, motion detected, temperature change, and so on. - Only last 7 days events can be obtained
27	list rooms	https://api.smartthings.com/elder/{***locationId***}/api/rooms	< **locationId>** from API *"no 1"*.	Get a list of rooms created by the users in this location
28	room detail	https://api.smartthings.com/elder/{***locationId***}/api/rooms/{***id***}/roomTile	< **locationId>** from API *"no 1"*. < **id> (room Id)** from API "no 27"	returns room detail such as name, id, location, and more.
29	room tiles	https://api.smartthings.com/elder/{***locationId***}/api/rooms/{***id***}/tiles		returns the list of actions and attributes for the devices in this room
30	devices in specific room	https://api.smartthings.com/elder/{***locationId***}/api/rooms/{***id***}/devices		Get all the devices specific to this room
31	devices without room (ungrouped devices)	https://api.smartthings.com/elder/{***locationId***}/api/devices/tiles?withoutRoom=true&locationId={***locationId***}	< **locationId>** from API *"no 1"*.	Get all the devices not a member of any room (devices without room)

32	room events	https://api.smartthings.com/elder/{***locationId***}/api/rooms/{***id***}/events?beforeDate={***date***}&all=false	< **locationId>** from API *"no 1"*. < **id> (room Id)** from API "no 27" < **beforeData> (**named **'date')** from the last item that is returned from the first/latest events - Leave **beforeDate** empty to return the first max size events.	- Events happened in this rooms such as door closed, motion detected, temperature change, and so on. - Only last 7 days events can be obtained
33	Installed smartApps	https://api.smartthings.com/elder/{***locationId***}/api/smartapps/installations/	< **locationId>** from API *"no 1"*.	returns installed SmartApps in this location
34	Installed smartApps initial data	https://api.smartthings.com/elder/{***locationId***}/api/smartapps/installations/{***installedSmartAppId***}/getInitialData	< **locationId>** from API *"no 1"*. < **installedSmartId>** from API *"no 13*".	returns initial data for the specific installed SmartApps
35	Installed smartApps home	https://api.smartthings.com/elder/{***locationId***}/api/smartapps/installations/{***installedSmartAppId***}/home		returns home (same as the above?) data for the specific installed SmartApps
36	Incidents List	https://api.smartthings.com/elder/{***locationId***}/api/smartapps/installations/{***installedSmartAppId***}/pastIncidentData		returns all the past incidents for the specific installed SmartApps
37	First incident	https://api.smartthings.com/elder/{***locationId***}/api/smartapps/installations/{***installedSmartAppId***}/pastIncidentData?lastIncidentId=0		returns all the first incident for the specific installed SmartApps (current status)
38	past specific incident	https://api.smartthings.com/elder/{***locationId***}/api/smartapps/installations/{***installedSmartAppId***}/pastIncidentDataForId?incidentId={***incidentId***}	< **locationId>** from API *"no 1"*. < **installedSmartId>** from API *"no 13*". < **incidentId>** from API *"no 36*".	returns specific incident
SmartThings web-app APIs - APIs the web-app application use to exchange JSON data				
39	List of devices, hub, and their locations	https://graph-na04-useast2.api.smartthings.com/device/listJson?useTimestamp=1	--	Return devices list, the current state of the devices, the locations, and the hub.
40	Location events data	https://graph-na04-useast2.api.smartthings.com/event/listMoreEvents?all=&source=&max=10&id={***locationId***}&type=location&eventType=&startAfter={***theLastEventId***}	< **locationId>** from API *"no 1/39"* <**startAfter>** is available in the returned JSON data. <**max**> default 10 and 200 is maximum.	Return JSON data which shows what happened in that user location. Maximum< **max>** items will be returned. To get the next < **max>** items, replace **startAfter** parameter with the value

				returned from the last API call.
41	Devices events data	https://graph-na04-useast2.api.smartthings.com/event/listMoreEvents?all=&source=&max=10&id={***deviceId***}&type=device&eventType=&startAfter={***theLastEventId***}	< **locationId>** from API *"no 1/39"* <**startAfter>** is available in the returned JSON data. <**max**> default 10 and 200 is maximum.	Return JSON data what the device reported/sensed. Maximum 200 items will be returned. To get the next 200 items, replace **startAfter** parameter with the value returned from the last API call.
42	Hub events data	https://graph-na04-useast2.api.smartthings.com/event/listMoreEvents?all=&source=&max=10&id={***hubId***}&type=**hub**&eventType=&startAfter={***theLastEventId***}	< **hubId>** from API "no 39" <**startAfter>** is available in the returned JSON data.	Return the status of the hub. Maximum 200 items will be returned. To get the next 200 items, replace **startAfter** parameter with the value returned from the last API call.
Note: 1. LocationId can be found in the smartphone image in the shared preference file if the smartphone is available. 2. The maximum value returned using events API is 200, and the default is 10 (max=200)				

Appendix B: Amazon Alexa Cloud APIs

No	Name	APIs; change the value in the {} with the right value	Note/Description
General APIs			
1	Wi-fi config	https://pitangui.amazon.com/api/**wifi/configs**	Wi-Fi settings
2	wi-fi detail	https://pitangui.amazon.com/api/**device-wifi-details?**deviceSerialNumber=*{deviceSerialNumber}*&deviceType=*{deviceType}*	***DeviceSerialNumber*** and ***deviceType*** value can be acquired using the above API.
3	Devices	https://pitangui.amazon.com/api/devices/***device***	Alexa enabled devices such as Echo, Dot, phone app
4	Device preference	https://pitangui.amazon.com/api/***device-preferences***	list of Alexa enabled devices with their preference and settings.
5	Smart home devices	https://pitangui.amazon.com/api/***phoenix***	Detected compatible smart devices that Alexa can see in the Smart Home
6	Shopping lists	https://pitangui.amazon.com/api/***todos***?type=***SHOPPING_ITEM***&size=100	User created shopping list.
7	Named list	https://alexa.amazon.com/api/***namedLists***	Newly added API that contains user-created lists which combines the shopping, To-Do, and custom lists.
8	Specific named list (Items)	https://alexa.amazon.com/api/***namedLists***/{*itemId*}/***items***?startTime=&endTime=&completed={true/false}&listIds={*listIds*}	***itemId*** and ***listId*** values are available in the above API, replace with appropriate value. Also, make ***completed*** parameter true to return only completed lists. ***Note:*** change equal (=) sign in ***listIds*** with (%3d), its hex value.
9	Bluetooth	https://pitangui.amazon.com/api/***bluetooth***	Bluetooth setting of the device including a list of paired devices
10	To-do lists/tasks	https://pitangui.amazon.com/api/***todos***?type=***TASK***&size=100	User created to-do list.
11	Accounts	https://pitangui.amazon.com/api/***authentication***	Account information of the user, including email, full name, and id.
12	Household information	https://pitangui.amazon.com/api/***household***	User information including full name and email
13	Notifications	https://pitangui.amazon.com/api/***notifications***	User-created timer and alarm.
14	user detail	https://alexa.amazon.com/api/***users/me***	User detail including email, name, country and id.
User conversations, activities and voice commands			
15	All active cards	https://pitangui.amazon.com/api/***cards***?limit=50&beforeCreationTime={ *nextQueryTime/null*}	Changing the parameter ***beforeCreationTime*** with a value ***nextQueryTime*** from the returned JSON data will retrieve the next active card collection. This way, we can get all data only by changing this parameter. Note that leaving

			beforeCreationTime parameter null/empty will return the latest active cards.
16	Individual cards	https://pitangui.amazon.com/api/***cards***/*{cardId}*	***cardId***: the id of a specific card, from the list of cards, returned using the above API.
17	Individual voices	https://pitangui.amazon.com/*{url}*	***URL***: get this value from the above API, there is a ***URL key*** value on the returned JSON.
18	All activities	https://pitangui.amazon.com/api/***activities***?startTime={null/startDate}&endTime={null}&size=50&offset=-1	This API returns the last 50 activities. To get more activities, we should change the value of ***startTime*** on the API with the value of ***startDate*** on the returned JSON data, see next ***cell***. Leave ***startTime*** parameter null to get the latest 50 activities.
19	Individual user activity dialog	https://pitangui.amazon.com/api/***activity-dialog-items***?activityKey=*{id}*	Get the value of ***id*** from each activity from the above API to get the individual dialog. ***Note:*** Change the hash (#) character with %, plus its hex equivalence (%23) from the ***id***. See the sample JSON column of the above API.
20	Single activity	https://pitangui.amazon.com/api/activities/*{id}*	Get the value of ***id*** from each activity from the above API to get the individual dialog. ***Note:*** Change the hash (#) character with %, plus its hex equivalence (%23) from the ***id***. See the sample JSON column of the above API.
21	Activities with Range	https://alexa.amazon.com/api/activities-with-range?startTime={}&endTime={}&size=50	***startTime:*** *this the date until which the data should be returned.* ***endTime:*** *this the date from which the data should be returned.* ***size****: the API returns a max of 50 items Example:* ***[startTime - endTime]*** *(Jan 1, 2019 - June 30, 2019]*
22	Voice	https://pitangui.amazon.com/api/utterance/***audio***/data?id=*{utteranceId}*	***utteranceId*** is a value available in each individual activity and contains voice command of the user. To get the voice data from these activities, append the ***utteranceId*** from these individual activities into this API. See the sample JSON column of the "all activities" API.
Calling and messaging APIs			
23	Account detail	https://alexa-comms-mobile-service-na.amazon.com/***accounts***/	Account info including full name, phone number, and communication id.

24	Contacts	https://alexa-mobile-service-na-preview.amazon.com/users/*{commsId}*/**contacts**	***commsId:*** get *commsId* from "Calling and messaging account detail" JSON.
25	All conversations	https://alexa-mobile-service-na-preview.amazon.com/users/*{commsId}*/**conversations**	***commsId***: get commsId from "Calling and messaging account detail" JSON.
26	Single conversations	https://alexa-comms-mobile-service-na.amazon.com/users/*{commsId}*/**conversations**/*{conversationId}*/messages?startId={}&count={}	***commsId:*** get commsId from "Calling and messaging account detail", or from the "Calling and messaging conversation" returned JSON. ***conversationId:*** get this value from "Calling and messaging conversation", each conversation has its own *conversationId*. ***startId*** is conversation id to start from, then return ***count*** size.
27	User identity	https://alexa-comms-mobile-service.amazon.com/users//*{commsId}*/**identities**	User identity detail including ***commsId*** and ***homeGroupId***
28	Home group contacts	https://alexa-comms-mobile-service.amazon.com/users/{***homeGroupId*** }/**contacts/**	Contacts in the user's home group. Get ***homeGroupId*** from the above API return JSON.