TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technology

Department of Software Science

Marieke Jahn 201684IVCM

FORENSIC DATA ACQUISITION SOFTWARE DEVELOPMENT FRAMEWORK FOR INTEGRATED SMART HOME ECOSYSTEMS

Master Thesis

Supervisor

Pavel Chikul

PhD early stage researcher

Tallinn 2022

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author:	Marieke Jahn	....
		(signature)
Date:	May 16th, 2022	

Abstract

Due to the complex nature of IoT, traditional forensics methods to analyse their data traces are not sufficient. Though, having become a staple in our lives, IoT devices can be a rich source of evidence.
The aim of our study is to explore if it is possible to devise a framework to assist forensic software development for data acquisition in smart home environments.
Within our research, we conduct a case study on an experimental smart home setup, extracting and analysing generated data, deriving a process-oriented and technical framework from the results.
As IoT can serve many different purposes, making one tool to support all of them is not realistic, which was also a main takeaway from our case study, in which we compared different devices and traces left by them. The organisational part framework is based on an agile software development approach, supplemented with additional information such as common file locations and formats, while the practical part consists of blueprint classes written in Python which can be extended for specific data sources.
Validating the framework by developing forensic software along it, indicated that it has a beneficial effect on the development process and can be used to proliferate forensic tools specific to IoT.

List of abbreviations and terms

IoT	Internet of Things
API	Application Programming Interface
UML	Unified Modeling Language
DFRWS	Digital Forensic Research Workshop

Table of Contents

List of Figures vi

List of Tables vii

1 Introduction 1

1.1 Motivation . . . 1

1.2 Research Objective and Contributions . . . 2

1.3 Methods . . . 3

1.4 Scope . . . 3

2 Literature Review / Related Works 4

3 Data Generation 15

3.1 Devices and Setup . . . 15

3.2 Data Generation . . . 17

3.3 Data Acquisition . . . 20

3.3.1 IoT Forensics - Fibaro Home Center Lite and Google Home Mini 21

4 Results 27

4.1 Data Analysis . . . 27

4.1.1 Google (Home Mini / Takeout) . . . 27

4.1.2 Fibaro Home Center . . . 30

4.1.3 Mobile . . . 31

4.2 Common aspects . . . 39

4.2.1 Google Home and Fibaro Smart Home . . . 39

4.2.2 Mobile . . . 40

4.3 Framework Development . . . 42

4.3.1 Framework . . . 49

4.3.2 Code Development . . . 52

4.3.3 Technical Implementation . . . 57

5 Framework Validation 60

6 Discussion 64

7 Conclusion 67

Bibliography 70

List of Figures

1 Schematic of room layout 16

2 Google Takeout export structure 28

3 DFRWS Investigative Process[25] 43

4 Agile vs. Waterfall[28] 47

5 Organisational Framework - Forensic Software Development 49

6 UML diagram hierarchy 59

List of Tables

1 Events that were synthetically produced. 19

2 Fibaro API available functions. 25

3 Google Takeout export contents 29

4 Backup Data Extraction 33

5 Rooted phone logical data acquisition 35

6 Physical acquisition data 37

7 Common data types 40

1. Introduction

1.1 Motivation

Recent years have seen a continuous growth in development and deployment of IoT devices in both the industrial[1] and private[2] sector. Smart homes have become a wide-spread phenomenon, aiding to improve quality of life. Yet, security of IoT devices has fallen short so far also impeding the further adoption in businesses and business processes.[3]
With IoT devices becoming more prevalent in our lives, such as smart watches and smart home appliances, the attack surface grows, as long as security is not a standard part of these applications. With these devices being very interconnected with their users' lives, the stakes are high and the risk becomes very real, but at the same time, this opens up an opportunity for forensic data analysis[4]. While it is crucial to be able to extract and analyze data from attacks targeting IoT devices, crimes have not only been committed through and with IoT. These smart devices can also be a rich source of relevant information to a case when not directly targeted, but being part of the environment in which the crime has been carried out[5]. Smartwatches, smart speakers, smart hubs, smart TVs, and many more smart appliances and sensors collect vast amounts of data about users' behaviour and location, for example. This data can pose as valuable evidence in criminal cases.
Though, when dealing with data from IoT devices, there are a multitude of factors that complicate matters of data extraction and analysis - big volumes of data, heterogeneous architecture, and proprietary soft- and hardware are examples of these challenges. Hence, there is an existing need to be able to overcome these challenges. One solution to face these issues could be to add support for IoT devices and their data traces to forensic tools which automate these processes, yet this is where another challenge is faced: the vast amount of devices and producers of IoT systems makes it sheer impossible for the few commercial forensic tools to keep up with development. Since smart devices are made for very specific purposes and there exist no standards for their development whatsoever, their architecture

varies greatly and parallels across devices are difficult to establish. Additionally, there is a high throughput of IoT devices. Hence, it is up to the forensic community and specialists to develop tools for their specific needs and help contribute to the area of IoT forensic. With combined efforts it is possible to better tackle the issue of IoT forensics lagging behind.

1.2 Research Objective and Contributions

Our research will aim to answer the following questions:

- What is the current state of IoT forensics and related tools?
- What are common aspects and differences across IoT devices and their traces in smart home environments?
- Can these aspects be integrated into a framework to aid forensic tool development for automation of data acquisition and analysis?

By reviewing research on IoT forensics and the analysis of IoT devices and their traces, we will establish an overview of current capabilities and challenges when dealing with IoT environments in forensic investigations, underlining the need for tools supporting automation of extraction and analysis of IoT traces.

The main goal will be to develop such a framework to ease and support the process of tool development for data acquisition and analysis in smart home environments which will aid future developers in their endeavour and encourage the making of such tools. The focus will be on data that can be extracted from mobile applications and through API access.

Within our case study, the data that is generated will be analysed for commonalities across sources such that the framework can be developed with these aspects in mind, to be easily adaptable for other IoT data sources.

To validate the framework and show its applicability and practicality, we will develop a standalone tool that automates part of the data processing with data generated from our case study (Fibaro, Google Home), following our proposed framework.

The data generated during and for this work will be made available for the public for reference and as a possibility for others to conduct their own research on it. The same goes for all tools developed in the scope of our research which will be open-source.

1.3 Methods

To establish the current state-of-the-art and provide background information, we conduct a literature review on IoT forensics and related tools, as well as tool development in this particular field. For the second part of this work which involved data generation, extraction, and analysis we used experimental science by conducting a case study from which the proposed framework is developed. The validation of the framework is done by developing software in Python.

1.4 Scope

Since IoT encompasses a very wide range of environments and devices that vary greatly in architecture and functionality, the focus of this study will remain on integrated smart home ecosystems.

The devices used within our work are Fibaro sensors and home center, a Google Home smart speaker, a d-link IP camera, and an Android smartphone. Data generation and analysis is done on these devices and their respective traces. Due to the complex nature and infrastructure of IoT systems, relevant data can be found across multiple sources, such as the network layer, on the cloud, on mobile phones, and on the IoT devices themselves. For the purpose of our research, we will limit the scope to server-side available data, accessible through an API, as well as artifacts that can be recovered from related mobile phones. Any physically intrusive analysis will not be encompassed in our studies.

2. Literature Review / Related Works

While IoT has become more prevalent in the life of many, ways to analyze data generated and stored by IoT devices and environments have not been developing as fast.
To provide an overview of current trends and possibilities in the domain of IoT forensics, relevant literature has been reviewed and the findings will be presented in the following chapter. This literature will consist of three main topics to be reviewed: firstly, a general look and the state-of-the-art in IoT forensics, including existing solutions and challenges are established. Next, we follow up with existing frameworks in the (IoT) forensics domain that we could adapt to fit our needs, and lastly forensic case studies on IoT devices, which will provide references for our own work.

To create an understanding of the base issue, we first need to present characteristics of IoT devices that create a challenging environment for forensic analysis. Atlam et al. provide a comprehensive overview of IoT from the perspective of security: the authors explain that IoT combines diverse networks that enable heterogeneous devices and platforms to connect and communicate.[6]
These devices can be equipped with all different kinds of sensors, advanced technologies, and software. Smart devices are able to gather, process, and communicate data, to provide different services and applications to improve the quality of life of their users. So, IoT is not a single technology, but merges several, such as Big Data, AI, edge and cloud computing. Only within recent years has IoT spread into multiple domains, ranging from simple household devices to complex and sophisticated industrial equipment and machines. Applications, where IoT can be found, include smart healthcare, supply chains, smart homes, smart grids, smart cars, and smart industries.[7]

The architecture of IoT can be described in four layers: perception layer, network layer, application layer, transport layer. On the perception layer, the data is generated and gathered by various devices, usually sensors, for example, temperature sensors or smoke detectors. The network layer is used by the devices to communicate with each other and the applications. Data is received from the lower layer and mapped to the required format of the upper layer. The application or service layer is where gather data is processed in various ways and presented. Responsible for end-to-end communication over the network is the transport layer, which provides flow control and reliability multiplexing. Various different protocols are used on the different layers.[8]

IoT can be applied in many of our daily activities, in different domains. The personal and social domain allow users to communicate with their environment and other users. This is the domain that our own study will be concerned with, as smart homes are part of the personal domain. The mobility and transportation domain includes anything sensory related to roads and vehicles.[7]

IoT forensics can generally be described as a branch of digital forensics, though due to the aforementioned differences between conventional computers and computer networks and the equivalent counterparts of IoT, there can be several layers defined on which forensics can be applied:[9]

- **Device level**: The IoT devices themselves can store data, which can be extracted in various ways, such as logically or physically. For that, the devices have to be targeted and identified physically. This may include mobile forensics, as mobile phones can be an important part and controller of IoT systems.
- **Network level**: Since IoT is all about communication between the many devices, anything happening on the network layer plays a big role as well. Types, forms, and content of network communication can be analysed, such communication sources, exchanges, communication times, or extraction of valuable information.
- **Cloud level**: One of the most important parts of IoT environments, is the cloud, as the devices themselves do not have a lot of storage or computing power and need to rely on the cloud for that. Hence, a lot of the data that is generated in such environments, ends up in the cloud.

A very recent study[10] delves into the topic of security in IoT. Missing and inadequate security in IoT is one of the main reasons that calls for the adoption of forensics in the field. The authors reviewed about 200 papers, pointing out a multitude of aspects that make IoT environments vulnerable to cyberattacks by adversaries. Omolora et al. explore various levels of security in IoT, analysing existing research findings, presenting challenges, emerging issues, and open issues that are yet to be addressed by research. It has to be considered that while IoT provides many beneficial opportunities for the quality of life, security issues in their application and resulting cyber incidents raise concerns and distrust in both commercial and industrial users and possible users which impedes further adoption of these technologies. The paper first creates a main understanding of the concept of IoT, discusses industrial development trends of IoT, and identifies their security challenges and privacy issues. The authors identify the majorly lacking areas in IoT as security and privacy, due to added complexity in architecture, implementation, maintenance, sustainability, and minimal control, as well as pliability, compared to conventional home computers. They express the opinion it is likely that if nothing is done by stakeholders and the government to intervene with the current trend of IoT development, security being more optional than by design, IoT will remain insecure. The lack of security stems from low-cost and an increasing amount of devices. Main security challenges faced by IoT, as defined by Omolora et al. are:[10]

- insecure web interfaces,
- insufficient authentication and authorization,
- insecure network services,
- lack of transport encryption,
- privacy concerns with stored data,
- insecure cloud interfaces,
- unprotected mobile interfaces and networks,
- software and firmware issues,
- physical security issues,
- environmental constraints e.g., hardware/software, limitations
- cross-device dependencies.

These challenges stem from related vulnerabilities that IoT inhabits, for example weak, hard-coded passwords on client software and firmware, insecure update mechanisms, inadequate privacy protection, lack of device management, and insecure default settings, among other issues.
Another topic that the authors covered in their survey considers forensics in the era of IoT. They explain that data from smart devices can be a rich source of evidence, but the nature of these devices and ecosystems complicates the topic of digital forensics as the usage of IoT devices increases. IoT forensics faces some more challenges:[10]

- growing number of devices,
- relevance of devices,
- non-standard formats,
- multi-tenant cloud infrastructure,
- resulting multi-jurisdictional litigation,
- end-to-end encryption,
- blurry network boundaries and edgeless networks,
- increase in volume of data,
- new technologies,
- physical acquisition of devices,
- need for close coordination among law enforcement,
- need for educated and properly equipped personnel to perform forensic processes in IoT environments,
- distributed digital data processing due to cloud-native and client-side forensics,
- insufficient integrity of data due to being (partially) overwritten or compressed,
- difficulty to provide chain of custody,[10]
- difficulty of admissibility in front of the court,
- lack of forensic tools.[9][11]

New methods and processes to conduct forensic analysis in IoT environments need to be identified through research and developed to accommodate these factors, as the number of IoT-connected devices grows.
The authors define long-term goals for IoT forensics, which should be pursued in the future:

firstly, digital forensic standards should be created and used for IoT and IoT security, which would ease IoT-based investigations. Though, the opinion has been stated that as long as stakeholders and the government do not intervene with the current development of IoT, meaning allowing basically unfinished products on the market, with security not being by design, but rather optional, IoT is likely to remain insecure.
Commonly proposed solutions to these security challenges stated in the paper are: [10]

- securing of IoT network,
- Authenticate IoT devices,
- usage of IoT data encryption,
- usage of IoT security analytics,
- usage of IoT API security methods,
- hardware tests,
- development of secure IoT apps,
- no rushed releases of IoT devices,
- being aware of current IoT security breaches and threats.

While these are often proposed solutions, this does not mean that they are generally adopted. These solutions usually come at a trade-off price with e.g., cost- and time-efficiency.[10]

With IoT forensics being a rather new research topic, a few case studies have been conducted to explore the possibilities and shine some light on the unknown.

In the context of related research, Servida and Casey conducted a similar case study to ours, though with a shifted focus and outcome. The authors conducted case studies on several smart home devices, focusing on information accessible from the network layer and mobile phone (applications).
In the scope of their work, the authors tried to extend existing methods for extraction and analysis of data on mobile phones to IoT devices, developed a scenario for the 2018-2019 DFRWS IoT forensic challenge, and several tools as well as plugins for the forensic tool Autopsy were created. Additionally and incidentally, previously unknown vulnerabilities

on some of the IoT devices were discovered. [12]

In "Challenges and opportunities for wearable IoT forensics: TomTom Spark 3 as a case study" the authors conducted a forensic analysis on data generated by a wearable IoT device (TomTom Spark 3 GPS fitness smartwatch) and its companion app in a set of controlled experiments. The authors pointed out the usefulness of such data, limitations of existing forensic tools, and that in development of future tools the ableness to register all generated data by an IoT device may be useful. For that the memory of the smartwatch itself, the data accessible through the app, and event logs, as well as proprietary activity files were analysed to establish a timeline of events. Forensic tools used in this setup were Cellebrite and FTK Imager, but non-forensic tools such as a database browser and runalyze web application also found application in this case and were used to decode and interpret proprietary files.[13]
The work identifies file paths of evidence sources and reconstructs the user activity by utilization of non-forensic tools. The authors highlight that their methodology was presented on the case study of this particular smartwatch, but that it can be extended to other fitness trackers and IoT devices if tools to support the extraction of relevant files are developed. Their research shows the limit of today's (commercial) forensic tools and helps create references for future investigations.[13]

Becirovic et al. conducted a case study on the manual device level extraction of data on a smartwatch. Prior work is reviewed and the experimental scenario plus its results presented. The aim is to provide a reference to approach the data extraction of the particular IoT device analysed in this work.[14]
For this paper the Samsung Gear S3 Frontier smartwatch was analysed, and a series of events executed to proliferate data for this scenario. These actions were recorded in a table to later compare this with the dataset acquired through analysis. For the analysis directories, where relevant information was saved, were identified and from there copied. In the results, the authors established a timeline that showed all executed events were possible to be reconstructed and even more details could be found than in the initial recording of events. Locations, where important data could be found, was recorded. Some aspects such as GPS ended up not bringing the hoped results of location status and some databases were

encrypted. Through the analysis the authors were able to determine when the user moved their wrist, how they rotated, what applications were used, data about the connected phone and user account, as well as notifications on the watch could be discovered. The work concludes that the forensic investigation of smartwatches is able to uncover important evidence, but definition of privacy levels for IoT data as to not jeopardize the user's privacy completely needs to be defined.[14]

Stated as a work in progress, the paper by Iqbal et al. deals with the feasibility on conducting forensic analysis on Smart Plugs (conducting an experimental case study) and what issues this might entail, also conducting a research review on related work.
In this related work, known vulnerabilities and flaws in some of Smart Plugs and other IoT devices are highlighted, as well as previously conducted forensic analysis studies of IoT devices.
With the experimental setup the goal was to identify what and where those traces can be found after performing certain actions on the Smart Plugs, how such evidence can be captured on the network, and if there are any digital forensic challenges along the way. The setup is described, and activities that were performed for the scenario, recorded.
For the analysis path with relevant data were identified and the files with fitting tools analysed. On the smartphone, relevant information was mostly stored in the databases, but only limited evidence was found as not all performed activities had stored logs. The authors encountered encryption issues on some files and network traffic. Port scans revealed some open ports and possibilities to discover e.g. the model and firmware.
Main challenges pointed out by the research included variety of data formats encountered, calling for many separate tools, encrypted files and network traffic, interpretation of the work was not always possible, and the possibility of missing logs being stored on cloud services.[15]

"Digital Forensic Approaches for Amazon Alexa Ecosystem" aims to contribute by combining cloud-native forensics with device forensics to provide practical reference to future digital investigations. Some related work is reviewed to establish background information, a case study is conducted, and based on this scenario a proof-of-concept tool is proposed. The environment of the Amazon Alexa ecosystem is explained, as well as the proof-of-

concept tool for cloud-based IoT systems which is able to access Alexa's cloud traces and analyse traces of web-based applications on the client-side.
The experiments were performed over a period of two months with two Amazon Echo Dot devices and client- as well as cloud-side analysis conducted. On the client-side paths of relevant artifacts were provided and databases analysed. It was noticed that only very little information was actually stored locally.
The Cloud-based IoT Forensic Toolkit that was proposed provided a user interface and normalises the collected data.[16]

Kim et al. focus on analysing data from companion apps, web interfaces, and APIs of a smart home, an IoT service platform, and proposes scenarios in which digital forensics could make use of this data. For this data (sources) are identified that could be used as evidence, and correlation between the data established.
The study considers previous work analysing such IoT-generated data as reference and considers devices that have not been previously studied. An experimental setup of a smart home was built for this purpose, including several different smart home appliances, such as a Google Nest Hub and a SmartThings Motion Sensor. From this setup an analysis of obtained data from the apps, web interfaces, and API was conducted, singling out the paths and content while also rating the usefulness of the data.
Possible scenarios of smart home data in digital forensics were outlined by the authors: base stations that connect other IoT devices within the home contain information about installed devices, which can be useful in identification. Data collected about movement can assist in establishing a timeline. Call logs (to e.g. Google Nest Hub) can provide additional information.
The study concludes that data acquired from a smart home environment can be useful in forensic investigations and suggests that future research should include a wider range of IoT devices.[17]

We also explored existing frameworks related to IoT forensics and present them for context. Goudbeek et al. present a framework that is specific to the forensic investigation of smart home environments. The main goal of their research is to provide information on potential data generated in smart home environments, and how this data can be acquired.

Background information, as well as challenges faced, when investigating smart home environments are provided to establish a basis for the framework.
The proposed framework consists of seven phases. First comes the preparation off-site. For this step it should be ensured that forensic investigator with appropriate skills and knowledge are available and that relevant technologies (hardware, software) can be provided. The second phase is the search for a home automation system on-site. This entails looking for physical devices, recording device information also in form of pictures or videos. It should also be checked for remote access and access through mobile phone applications. Next, the system should be preserved as close to the way it was found as possible. Phase four sees to creating an understanding of the specific home automation system, in form of e.g., a network topology. A security level check is performed in the next step, meaning establishing if there are any security measures in place on the home automation system, such as access control. A user list with rights should be compiled and status of software patches explored. After compiling all this information, the next two steps comprise the location and acquisition of possible evidence, as well as processing and analysis of said data. The authors then proceed to explain how to apply their framework by applying it to three case studies they conducted. From those case studies Goudbeek et al. concluded that relevant artefacts could be recovered, following the framework. Future work that the authors see is further validation with real-life home systems and the development of tools.[18]

Kebande et al. address how current state of the art IoT can be integrated into the digital investigation process, and the proposition of a generic and holistic framework (DFIF-IoT) for the digital forensic investigation in IoT settings. The paper proposes a framework to help guide the process of data collection and analysis in an IoT scenario. It defines several building blocks – a proactive process, IoT Forensics, and a reactive process. In most scenarios even nowadays, looking at the more commercial IoT environment, a proactive process is not feasible. In industrial settings this might be possible. In the study, IoT forensics is split into three subcategories – cloud forensics, network forensics, and device level forensics. These are proposed aspects in an IoT environment that have the potential to be investigated. The reactive process happens after an incident has occurred and consists of the basic aspects initialisation, acquisitive process, and investigative process. While the proposed framework has not been verified in this paper, it complies with the ISO/IEC 27043: 2015 standard and suggests better admissibility in front of the court. Yet, each step is very generic and is not concerned with the actual difficulties of data acquisition and the investigative process in IoT environments. Since this is only a guideline it may help in structuring and planning an IoT forensic investigation, but references on how to actually extract and analyse IoT traces were not given which is left for future work.[19]

In a later study by Kebande et al., the authors propose an integrated digital forensic investigation framework, focusing on IoT-based ecosystems. For this they first analyse the framework provided in their earlier work[19], forming the basis for their further research which will extend this existing framework. This extended framework is provided in a high-level overview and detailed view. Overall, 9 processes for the Integrated Digital Forensic Investigation Framework for an IoT-based Ecosystem (IDFIF-IoT) have been defined - "Things", Device Connectivity and Communication Network, Readiness Process Groups, IoT Forensics, Digital Investigation Process, Concurrent Processes, IoT Management Platform, IoT Policy, IoT Standards and Protocols. The provided framework complies with the ISO 27043 standard and each component is described in detail to explain what is to be done for each process, as the framework itself is given in flowchart form. The authors evaluate their own work, pointing out the flexibility of it to be easily adopted, but do surrender that at the time of conducting their research, there was not much to go on regarding standards and accepted procedures in IoT. Differences from their previous framework are pointed out and how they have developed it further. In the future, Kebande et al. hope to implement their framework as a prototype, to identify critical forensic aspects and develop their research further.[20]

In a comprehensive study by Hassan et al., the authors provide a high level overview of current IoT forensics and introduces several IoT forensic frameworks. The Digital Forensic Investigation Framework (DFIF) is described as a guide that can potentially provide investigations in IoT environments. The Digital Forensic Framework For Smart Environments (IoTDOTS) is presented as a two stage framework that help scan for relevant information in smart applications during build time and then analysing the data using e.g., machine learning techniques, extracting forensically interesting information. Similarly, the

FSAIoT - Forensic State Acquisition from the Internet of Things - consists of two parts: a central control unit called Forensic State Acquisition Control Unit, and a collection of techniques on how to collect the current state of the IoT device.
Challenges faces in IoT forensics, as well as proposed solutions are briefly described.
The paper concludes that the majority of frameworks focuses on integrating IoT environments into the forensic processes of traditional forensics.[9]

From reviewing relevant literature, we were able to conclude trends in IoT forensics.

- There is a lack of research done on IoT forensics.
- There exist challenges in IoT environments that need further analysis, such as proprietary hard- and software of IoT devices.
- There is a need for standards in IoT systems, e.g. standardized data formats.
- There is a need for development of automation in context of IoT because of big volumes of data.
- Case studies are useful in providing reference for future research and investigations.
- There is a lack of forensic tools supporting traces from IoT environments.

With our research and the framework we propose, we will address some of the current and aforementioned challenges being faced in forensic investigations, as well as lay groundwork for future research and tool development.

3. Data Generation

This part of our research will encompass anything related to data - from the devices used for the experiments and study, over to data generation and acquisition. The data analysis and framework development will be included in the results section.

3.1 Devices and Setup

For generating, as well as analysing and comparing data needed for this study, the following devices were used:

- Google Home Mini
- Fibaro Smart Home Center Lite
- Fibaro Motion Sensor
- Fibaro Window Sensor
- D-Link Securicam
- Xiaomi Redmi Note 7

The devices were setup across two rooms in an ordinary living space and thus over the course of two months natural data was accumulated. A schema of the room layout and setup of the devices is depicted in figure 1:

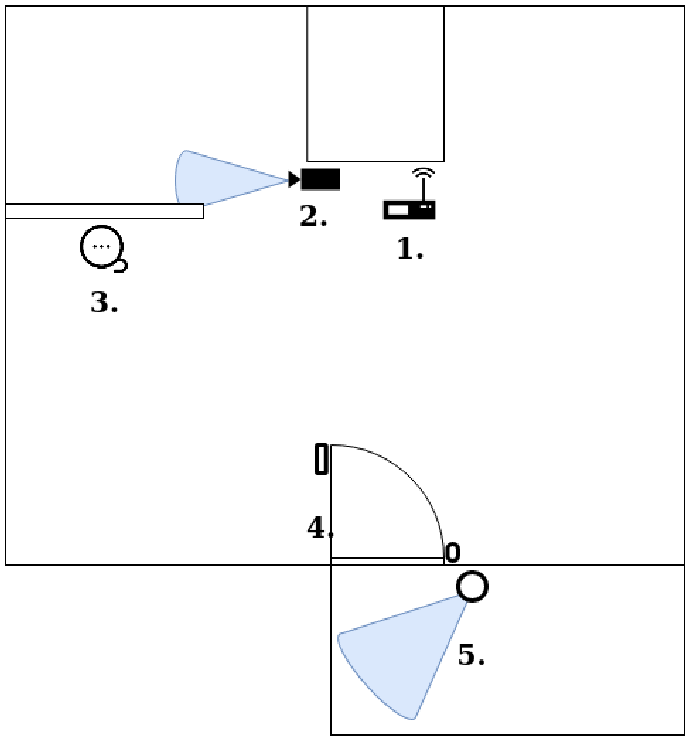


Figure 1. Schematic of room layout

1. Fibaro Home Center Lite
2. dlink camera
3. Google Home Mini
4. Fibaro Door Sensor
5. Fibaro Motion Sensor

The host machines used for the experiments were running Ubuntu 20.10 and Windows 10. Tools used to extract and analyse the data were Andoird Debug Bridge (ADB), SQL Lite Browser, a simple API parser, file converter, and data extractor for Fibaro .

3.2 Data Generation

Naturally, to assess what data will be found where, data has to be generated. This includes triggering different events and behaviours on the sensors, as they would normally occur in real life scenarios. The devices were situationally set up as depicted in figure 1 and each node connected accordingly: the Fibaro Home Center Lite was connected to the local network and acts as a base station for the sensors. The Fibaro sensors (motion sensor, door sensor) were locally connected to the Home Center, and firmware updated as needed. The camera first had to be setup in its own environment to then integrate it into Fibaro. It was connected wirelessly through the mydlink app on the Xiaomi phone. After the setup, the IP assigned to the camera within the network was not available from within the app, but could be discovered from the router's interface. The camera can be accessed over web browser, but works exclusively with Internet Explorer, where additional changes and settings can be made. The video feed can also be watched through the mydlink app on the network, as well as remotely.

After having set up the network camera, it was possible to integrate it into the Fibaro interface via the IP and credentials that were provided within the app. It is possible to watch the video feed from there, but things such as motion detection or alerts are not available from within Fibaro - this has to be configured and received through the mydlink interface.

It was not possible to add the camera to Google Home / Google Assistant, as the camera does not receive security updates anymore and has reached end of life, thus Google will not support it, as it could pose a security risk. Integrating the Fibaro sensors into Google Home was easily achieved by selecting Fibaro Smart Home from a list of devices/services that work with Google Home. Though, the functionality concerning the Fibaro sensors provided within Google Home is much more rudimentary than what is actually possible from Fibaro's native interface - the motion sensor includes a brightness sensor and both sensors are also fitted with temperature sensors. Querying information about those additional measurements was not possible from either directly within the Google Home app nor via voice command to the Google Home Mini. Actually, the Google Home app itself did not offer any functionality regarding the sensors other than providing their info (name, which home and rooms they are assigned to in Google Home, manufacturer, device type,

connected through), so not even the device state (safe / breached) was visible from the app itself. The device state is queryable via voice commands by addressing the devices by their names and devices can also be armed and disarmed (PIN needs to be provided).
Fibaro itself offers a much more broad spectrum of customization and information available.
The Google Home Mini itself was easily and intuitively setup with the Google Home application. To be able to collect all information interesting for this study, certain settings had to be considered though. For example, it was necessary to turn on voice match for voice recordings to be made for the voiced out commands issued to the Google Home Mini. It should also be noted that voices that were not set up for voice match were not recorded in this matter.
All of the devices were connected sharing the same network.

Over the course of a few months, we conducted experiments in the loose sense for this study. The data for these experiments was generated naturally, as the sensors were setup in an environment where they were exposed to our everyday life. Thus this data is quite life-like.
To correctly assess what information is processes and transferred by the respective devices, sensors, and controllers, in addition to the naturally generated data which was not documented, controlled synthetic events were created. These events were created artificially, e.g., by triggering the sensors on purpose, and the time and course of action were documented. The following table depicts these events:

Synthetic Events		
Date	Time	Event
03.04.2022	22:00	Motion sensor breached
	22:11	Door sensor disarmed
		Door sensor armed
	22:16	Door opened (safe -> breached)
		Door sensor alarm disarmed
		Door closed (breached -> safe)
		Door open (safe -> breached)
		Door closed (breached -> safe)
	22:19	Motion sensor breached
	22:24	Motion sensor armed
		Door sensor armed
		Motion sensor breached
	22:25	Motion sensor alarm disarmed
	22:31	Motion sensor armed through Google Home Mini via voice command
	22:33	Motion sensor disarmed through Google Home Mini via voice command
		Issued voice command to Google Home Mini - what's the time
	22:38	Door opened (safe -> breached)
	22:42	Motion sensor breached
	22:43	Motion sensor breached
	22:47	Motion sensor breached
	22:53	Motion sensor breached
	22:54	Motion sensor breached
	22:55	Motion sensor breached
		Motion sensor breached
	23:00	Door alarm disarmed
		Turned on light close to motion sensor / light sensor
	23:04	Motion sensor breached
	23:13	Motion sensor breached
	23:32	Turned off light close to motion sensor

Table 1. Events that were synthetically produced.

3.3 Data Acquisition

Since IoT encompasses many diverse technologies, some limitations need to be applied on what data will be acquisitioned and later analysed, as well as be included in the framework. When it comes to IoT, generally, data can be found on three different levels - the device level, network level, and cloud level. Data from any of these levels can be interesting to investigations, but especially the cloud level will contain big amounts of (relevant) data. IoT devices do not possess a lot of storage space nor computing power, so the cloud is oftentimes used for computing and to store data. Yet, there is little research done on forensic analysis of cloud environments, especially when it comes to IoT. This can have several reasons: The cloud, just like IoT has only become relevant within the recent years and usually forensics need some time to catch up with current trends. But there are also multiple issues that can arise when it comes to data stored on the cloud - multi-legislation scenarios with cloud providers and their storage being located in different countries makes accessing the data in a lawful way, more difficult. It could for example also be the case that more than one person uses the same cloud storage, but there can be no real distinguishment made between what data belongs to whom. This is especially true when it comes to IoT devices. For example, sensors record/sense their environment and do not distinguish (usually) between people. If someone steps into the view of a motion sensor or camera, the sensor will trigger regardless of who it is. It will be difficult attribute some dataset to one person if the sensor is located in a place where several people could have access to it. Though of course this is true not only for IoT data stored on the cloud. Multi-jurisdiction across provided cloud storage can make accessing the data even more complicated (read up on this, I know I had some papers in lit rev that mentioned this stuff). Besides all of these issues, there is also the problem that most of the time the user credentials are needed for accessing the cloud. If devices are acquired that are already logged in, this might not pose as a problem, but if that is not the case, the suspect or involved person does not have to give out the password. Furthermore, data and information found on the cloud might only have limited admissibility in front of the court - oftentimes, especially in the case of smart home environments, the data that gets uploaded to the cloud is either compressed or precomputed/filtered. But to be admissible in front of the court, the data has to be unaltered - as it has been recorded. It might be possible to revert the data back to its original state.

3.3.1 IoT Forensics - Fibaro Home Center Lite and Google Home Mini

The Fibaro Home Center which acts as a hub and base station for the sensors can be accessed from a web interface, as well as the mobile application. From both interfaces it is possible to manually query all information that Fibaro has to offer - about devices, settings, users, etc.

The most interesting information that can be found, exists most likely on the events panel where device state changes are documented; whenever a sensor was breached, disarmed, alarms were triggered, and so on. As there exists an API for Fibaro, including an official documentation, within the extent of this research we developed a simple parser for this API to improve further processability and readability of the data. The following table gives an exhaustive overview of all the available functions that are possible to query from the official Fibaro API. The table was compiled by studying the official API documentation by Fibaro, as well as the list of functions provided for this particular Home Center (Fibaro Home Center Lite) at the URL http://<IP>/api/docs, as not all of the functions provided in the documentation are applicable or exactly the same for each home center. For example, the Home Center Lite contains less functionalities. On that URL it is also possible to try out each function.

The name, URL, and a description for each function will be provided. It should also be considered that most of these queries include subqueries that query for more specific data under its category. This is especially useful and important, for example, for the events panel, as not all events can be queried from just the general /api/events URL. A time frame needs to be defined, translated to Unix Epoch time, and appended to the URL. There exist also other options such as filtering for events of a specific device (by ID).

Fibaro API - available functions		
Name	URL	Description
Settings		

General settings	/api/settings/info	Returns a list of parameters of Home Center controller, such as serial number, soft version or default language, etc.
Backups	/api/settings/backups	Returns a list of saved controller's backups and their parameters like number of devices, rooms or scenes, etc.
Location	/api/settings/location	Returns a list of parameters related to date, time and location configured by user in Home Center interface.
Network settings	/api/settings/network	Returns a list of parameters related to network connection, such as DHCP status, remote access availability or IP number.
General		
Devices	/api/devices	Returns a list of devices, containing the main controller, all added devices, virtual devices and plugins as well, including all their parameters, properties and actions. Number of available data depends on the selected device.
Sections	/api/sections	Returns a list of sections, their names, sort orders, etc.
Rooms	/api/rooms	Returns a list of rooms, their names, icons, sort orders, etc.
Scenes	/api/scenes	Returns a list of all saved scenes and their parameters, such as name, id and sort order.

Users	/api/users	Returns a list of users, their names, types, rights, etc.
Global variables	/api/globalVariables	Returns a list of global variables, their values and parameters.
RGB programs	/api/RGBPrograms	Returns a list of RGB lights programs.
Tracking schedules	/api/trackingSchedules	Returns a list of weekly tracking schedules divided into four parts of the day.
Linked devices	/api/linkedDevices	Returns a list of linked devices and their parameters.
Virtual devices	/api/virtualDevices	Returns a list of virtual devices, their source codes, properties and actions.
iOS devices	/api/iosDevices	Returns a list of added iOS devices and their parameters.
VoIP devices	/api/voip	Returns a list of VoIP clients associated with Home Center end their parameters.
Icons	/api/icons	Returns a list of icons available in the system.
Panels		
SMS notification	/api/panels/sms	Returns number of available sms and list of associated phone numbers.
Location panel	/api/panels/location	Returns a list of predefined locations and their parameters.
History panel	/api/panels/history	Gets an array of objects containing actions stored in the event panel for a specified time period.
Notifications panel	/api/panels/notifications	Returns a list of notifications and their names.

Heating panel	/api/panels/heating	Returns a list of heating zones and their settings, such as temperature sets.
AC panel	/api/panels/cooling	Returns a list of cooling zones and their settings, such as temperature sets.
Humidity panel	/api/panels/humidity	Returns a list of humidity zones and their settings, such as humidity levels.
Alarm panel	/api/panels/alarm	Returns a list of alarms and associated devices.
Drenchers panel	/api/panels/drenchers	Returns a list of added drenchers and their parameters.
Favorite colors	/api/panels/favoriteColors	Returns a list of favorite colors presets, representing their RGBW values.
Fibaro alarm panel	/api/panels/fibaroAlarm	Returns Fibaro Alarm settings list, its properties, conditions, etc.
Energy panel	/api/panels/energy	Returns Energy panel data.
Temperature panel	/api/panels/temperature	Returns Temperature panel data.
Events panel	/api/panels/event	Returns events history from defined time, device states, state changes, their old and new values, etc.
Plugins		
Plugin types	/api/plugins/types	Returns a list of plugins divided into categories and their parameters.
Plugins installed	/api/plugins/installed	Returns a list of installed plugins, their names and predefinition status.
Other		

Login status	/api/loginStatus	Returns a list of parameters related to user's login, such as status, username or type of currently logged in user.
Password reminder	/api/passwordForgotten	Returns a password to your account sending it by e-mail.
Refresh states	/api/refreshStates	Returns refreshment details and performed changes.
Network discovery	/api/networkDiscovery/arp	Find IP and MAC physical addresses for specified network.

Table 2. Fibaro API available functions.

The data/response to requests made on the API are in JSON format. We developed a parser that converts the responses from JSON to csv format, as this is a standard format oftentimes used throughout investigations and it is also possible to be (more easily) read by human beings and has good properties to be processed further. The GitHub repository containing this parser can be found here.

The activity recorded on the Google Home Mini, or rather all activity related to the particular Google account, can be viewed under the Google account's My Activity tab. The data itself can be viewed there, but not downloaded for offline viewing or processing. This can be done under Google's Takeout. The data that can be downloaded from there comes in different formats, for example, HTML, CSV, or JSON. Many different kinds of data can be included in this export, for example the locations history from Google Maps, Mail, Google Pay, though the most interesting when it comes to smart home integration is data associated with the Home App and My Activity. The Home App data includes information about devices, rooms, and homes, as well as a history of events that happened in these homes. Both in JSON format. Included are also audio recordings of the issued commands.

After the data is selected, an export will be created that can then be downloaded. Though again, all of this is only possible with access to the user credentials or with a device that has already been authenticated and even then, the password is still required to download the export. Most of the data comes in JSON format and for this could again be a parser developed that converts JSON to csv.

4. Results

4.1 Data Analysis

We analysed the data that we generated and extracted in the previous steps, to find out relevancy, as well as commonalities and differences across devices and manufacturers.

4.1.1 Google (Home Mini / Takeout)

When exporting the data from Google Takeout, several data fields can be selected. For the purpose of our research not all have been selected, but only those that could be of interest in regards to data generated by our IoT devices, namely My Activity and Home App. For actual forensic investigations, any of the data found in My Takeout could be of interest and should be exported accordingly, though the exporting process could take a lot more time in that case. As stated by Google this could take several hours or even days, so prioritizing data can prove helpful for a timely progression of the investigation, seeing as all the exported data then would also need to be analysed and useful information separated from not so useful.

After a successful export the data can be downloaded and is arranged in folders for each data type included. For our particular export, the structure was as follows:

```
Takeout
│
├─ Home App
│   │
│   ├─ GoogleNestPartnerConnections.json
│   ├─ HomeApp.json
│   ├─ HomeHistory.json
│   └─ SoundSensing.json
├─ My Activity
│   │
│   ├─ Android
│   │   └─ My Activity.json
│   ├─ Assistant
│   │   ├─ [several files of voice commands].mp3
│   │   └─ My Activity.json
│   ├─ Google Play Store
│   │   └─ My Activity.json
│   ├─ Maps
│   │   └─ My Activity.json
│   └─ Search
│       └─ My Activity.json
└─ archive_browser.html
```

Figure 2. Google Takeout export structure

A few of these files are actually empty for us, namely ***Takeout/Home App/GoogleNest-PartnerConnections.json***, ***Takeout/Home App/HomeHistory.json***, and ***Takeout/Home App/SoundSensing.json***.

The other files are included in the following table with descriptions of their contents:

	Directory	Content
1	Takeout/Home App/Home-App.json	Information about Home owners (emails), names and devices of rooms. Further info about devices, such as model, type, software version.
2	Takeout/My Activity/Android/My Activity.json	Information about what services/apps and URLs Android devices accessed, including access time.
3	Takeout/My Activity/Assistant/<various voice command files>.mp3	Files are named by date and time of voiced out commands. Only recorded if voice match is turned on.
4	Takeout/My Activity/Assistant/My Activity.json	Includes transcribed voice commands, their initiation times, how the vc was started, the service that was queried, as well as the answers and notifications sent by the assistant.
5	Takeout/My Activity/-Google Play Store/My Activity.json	Information about all activity within the Google Play Store with timestamps, including searches, installations, and places visited.
6	Takeout/My Activity/Maps/My Activity.json	Timestamped information about when the user interacted with Google Maps.
7	Takeout/My Activity/Search/My Activity.json	Timestamped information about Google searches and webpages visited from Google, including searches made from the Google Assistant/Google Home Mini.

Table 3. Google Takeout export contents

All files will also be included in GitHub for a more detailed look.

As is evident, Google records and stores a lot of information about their users and the users' behaviour. Assessing the data that we were able to export and taking into account that Google has a lot more data stored that we chose not to export to keep the focus on IoT and the smart home environment, there is still a mentionable amount of related data. Especially the voice commands, the recordings, transcriptions, and answers by the Assistant can prove very valuable in investigations. From voice recordings it can become apparent if a person was alone, who voiced out the command, and other cues can be delivered through voice. It is especially interesting that voice commands are only recorded in mp3 format if voice match is turned on, and the person speaking the command is the one whose voice has been matched. This has likely been implemented due to privacy concerns. What is even more interesting is that the PIN that is used to disarm the Fibaro sensors is included in the transcript, as well as voice recordings. So, technically, if someone were to gain access to a Google account that is being used with integrated Fibaro Home Center, and some alarms were turned off / sensors disarmed via voice command, then that person would now easily be able to disarm any sensors/alarms of this smart home. All names of smart devices connected to the Google Home app can be discovered in artifact 1 (from table above), thus each one of them could be addressed separately. One thing that falls into the eye though is, that while there is a lot of information about the Google user's internal activity, but what could already be seen when integrating Fibaro into Google Home, there are not many actions that can be performed on the sensors. The possible actions are defined in artifact one, under "supported_traits". For the Google Home Mini, there are several traits defined: the Assistant, Cast, Call, VideoCall, RemoteDucking. For the Fibaro door/window sensor, there are two traits defined - OpenClose and ArmDisarm. Since the sensor cannot be exactly opened or closed, that functionality is most likely to query the state, if opened or closed, the other function to arm and disarm the sensor. The motion sensor only includes the ArmDisarm trait.

4.1.2 Fibaro Home Center

By parsing the RESTful API of the Fibaro Home Center Lite, a lot of data could be uncovered. Not all of this data may be useful for any investigations, but all of it should definitely be considered, as even RGB lights programs could be of interest.
As can be seen from Table 2 there are many functions that can be queried for. Each function was parsed, and each response from the server was written to a JSON file with our parser. Each file/server response includes very detailed information on the given section/topic. Usually, timestamps are given in Unix Epoch time format.
As the sensors and devices are integrated directly into the Fibaro interface, all of the data that the sensors generate and receive is being gathered and stored here.

Information that is most likely interesting to any investigation in a smart home environment was found by querying the following paths:

- **/api/settins/info**: These general settings include standard information that can be relevant e.g., for identifying devices. For example, the serial number, name, and MAC address of the home center can be found here.
- **/api/settings/network**: The network setting provides the IP address of the home

center.

- **/api/devices**: Querying this provides a list of all related devices, as well as additional information about those, such as serial numbers, names, in which room the device is installed, and the device types.
- **/api/users**: Returns a list of users, containing names, email addresses, and user rights.
- **/api/panels/event**: A list of events will be returned, including timestamps, the involved device, and what happened/changed.

4.1.3 Mobile

The acquisition of respective data on the mobile phone is not "IoT forensics" per se, but is a substantial part of investigations in smart home environments, and nowadays in general as well and should not be excluded. Thus, we decided to cover at least the aspect concerning IoT devices and integration, though of course a user's smartphone can have a lot more to offer and data to get from.

To access the data on the phone, it is more often than not required to know its PIN (to unlock the screen). If no PIN or biometrics are applied by the user, this of course makes things easier for the investigation. Usually, interacting with the phone is not encouraged for investigators , but especially in smart home environments, it may be necessary to assess the situation correctly in a timely manner. Identifying IoT devices can be a real challenge, and though some solutions for this problem have been proposed, those solutions may not always be accessible, and it may become necessary for investigators to interact with the phone directly at the crime scene or location of interest to establish devices in the network and possibly locate and secure them. It can be crucial to act fast in IoT environments. The special architecture of IoT devices implies a bigger possibility of volatile data - because of low computing power and memory, data is overwritten more often than on conventional home computers or phones for example. Interacting with the location of interest could thus eliminate relevant data - consulting a phone that may be connected to IoT devices of interest can be a safer first choice to assess the situation and identify possible devices. Though, of course interacting with the phone may also trigger data changes, first of all, on the phone, and secondly across IoT devices.

This is generally less likely though. To minimize state and data changes across the network and phone, it is recommended to perform a logical or physical acquisition of the phone instead of interacting with it directly or to interact with it directly as little as possible.

Logical acquisition

For the purpose of our experiments, to access as much data from the phone as possible, we rooted the phone, however it is still possible to access data that is available on user level permissions/privilege without rooting. Hence, logical data acquisition of the phone is possible on the fly to some degree. For a first look on what apps are installed to get an idea about possible devices that may be present, interacting with the phone itself is also an option.
To root the phone, the bootloader has to be unlocked (if it is not already) which is not possible on every phone, as vendors can decide if they want to enable users to install custom ROMs on their phones. For example, Huawei does not give out codes to unlock the bootloader anymore.
From the backup that we created with ADB it is actually possible to access
Rooting of the phone was done following a guide[21] on the xda-developer forum. When unlocking the bootloader, it is usually the case that all existing data on the phone is deleted. With ADB (Android Debug Bridge) it is possible to create a backup beforehand, (with e.g., ***$adb backup -shared -apk -all -f backup.ab***) that can be pushed back onto the phone after unlocking the bootloader (with ***$adb restore <path-to-backup-file>***). To connect the phone to ADB it is necessary to unlock the developer options, turn on USB debugging, and unlock OEM. This way we are able to unlock the bootloader and root the phone without losing all the data that this whole process was even started for. We used TWRP (Team Win Recovery Project) to root the phone. TWRP is a custom recovery image for Android devices that lets users flash their phones with third-party firmware. We used Magisk Manager to grant root permission to apps and services. SuperSU is another popular option for such kind of management, but imposes changes on the file system that could impact forensic soundness and the integrity of data.
While the backup we created with abd can on one side be used to restore data on the phone after unlocking and rooting it, but it can also be analyzed for data. By adding a valid

tar header to the file, it becomes extractable. For this we used the following command:

```
( printf "\x1f\x8b\x08\x00\x00\x00\x00\x00" ;
tail -c +25 backup.ab )|tar xfvz -
```

The following table gives an overview of artefacts that we were able to extract this way - with a focus on data generated by or related to the IoT devices and sensors, but also concerning the phone in general and data that can be interesting for forensic cases (that could relate or be related to IoT).

Index	Path	Description
1	shared/0/DCIM/Camera	Pictures (JPG) and videos (MP4) that were taken with the phone in question will appear here.
2	shared/0/DCIM/Screenshots	Screenshots that were taken with the phone in question will be saved in this location.
3	shared/0/Pictures/mydlink	Pictures that were taken within the mydlink app will appear here (in JPG format).
4	shared/apps/com.google.android.-apps.chromecast.app/sp/com.-google.android.apps.chromecast.-app_preferences.xml	Some values concerning Google services will be saved here. For example, the related email address could be read from here.

Table 4. Backup Data Extraction

As the phone had not been rooted beforehand, only a limited amount of data was available, contributing to the fact that the file we extracted these artifacts from was only a backup and no legitimate/valid data extraction method.

While not all these artifacts are recorded by a sensor node of our experiment, we felt it still worth mentioning some of these more generally mobile forensics related aspects. As, for example, pictures and videos that the user took and recorded (Index 1) could be essential to solving a case/incident, but also reveal more information about IoT devices - e.g. their location and setup. The same goes for screenshots (Index 2); they can give insights to various aspects of the user's behaviour, interests, plans.

Images specific to the mydlink app will be saved in a specific folder, as seen in Index 3. These pictures have to be taken manually from within the app. They could potentially be of importance, as different people could have been captured, certain changes, or other developments. While there is no guarantee that there will be any images of significance, it

is worth checking these files out.
Index 4 is related to the Google services and in more life-like scenarios where the phone is actually in use, this file may be populated more densely. Its content concerns mostly the phone's environment variables, with the most notable data being the email address associated with the Google account, hence also the Google Home service.

After we rooted and flashed the phone, it was possible to get a shell with root permissions through adb on the phone ($adb shell and then #su). This way we could explore the file structure and look for artifacts of interest. Since we are considering mostly companion apps and data generated by them, the /data/ directory is one the most relevant for us. Particularly, the directories /data/data, /data/system, and /data/misc have been investigated. We copied these directories onto a local machine by using the command $adb pull <directory-path>. The following table gives an overview of data we deemed the most interesting for our research and investigations.

Index	Path	Description
1	/data/data/com.android.chrome/app_-chrome/Default/Cookies; History; Login Data; Top Sites; Web Data	Database files (.db) that contain the respective information of user's behaviour for the Google Chrome browser
2	/data/data/com.android.chrome/app_-Chrome/Default/Preferences	Text file (data in JSON format) that contains information about preferences set in the Google Chrome browser by the user. If the user is logged into Chrome, the email address will be available here.
3	/data/data/com.android.providers.-contacts/databases/contacts2.db	Database file that contains information about calls and contacts.
4	/data/data/com.android.providers.-media/databases/external.db	Database file that contains information about locations of all media files on the phone.
5	/data/data/com.google.android.-gm/databases/downloader.db	Database that contains files that were downloaded via Gmail.
6	/data/data/com.android.vending/-databases/localappstate.db	Database file that contains installation dates of apps that have have been associated with the connected account.
6	/data/com.android.vending/-databases/suggestions.db	Database file that contains keywords that were typed in the search bar of the Playstore.
7	/data/system/packages.list	List of installed applications.
8	/data/system/packages.xml	Information about installed applications and their permissions.
9	/data/misc/wifi/WifiConfig-Store.xml	Information about Wifis/SSIDs that the phone has been connected to.
10.	/data/system/sync/accounts.xml	Information about accounts used for apps and services and their sync ability.
11.	/data/system/usagestats/0/-daily/1649608573940	Information about usage statistics, such as application name and timestamp of last usage. Also available for monthly and yearly statistics.

Table 5. Rooted phone logical data acquisition

Most of the data addressed here is again generally more of the mobile forensic kind rather than very much related to IoT. Nevertheless, these are all things that could be relevant for investigations and may also provide further information about IoT devices used in the smart home ecosystem. For instance, usage statistics (Index 11) and installed applications (Index 7 and 8) can give insights to the users' behavioural patterns, since when and what (at least the brand or associated app) smart home devices have been installed, and how much they are used/configured. Much of these different kinds of data can later also be used for cross-validation - to check for possible alterations or inconsistencies. For example, if a call was made or initiated from a smart device, such as the Google Home, that event will be recorded in Google's My Activity. From there it can be deleted, but the same information about the call should still be available here which the user might not be aware of, or of course vice versa.

Physical acquisition

Other than what might be expected when hearing "physical" acquisition, it is not necessarily a physical procedure of dismantling the phone, for example. A physical acquisition in terms of mobile forensics is simply a bit-for-bit copy of the device's storage media. This is something that can be achieved through invasive and non-invasive methods. Non-invasive are usually software-based methods, e.g., through the Android Debug Bridge that we already utilized for the logical acquisition of the phone. A complete copy of the physical storage can be done by using the dd utility (stands for copy and convert and dd because cc was already taken for c compiler). Usually the idea would be to copy the phone's internal storage onto an SD card, but instead we directly established a connection from the computer to the phone using a TCP connection. Since for a connection to be established, a service, in this case netcat needs to be listening on a port, this utility needs to be installed on the phone first, as it is not natively included. Through Magisk Manager third party modules can be installed. In this case we used BusyBox. Thus for this method we are using, it is necessary to root the phone, as this third party service needs to be granted root access rights to function properly. To find out which partition(s) includes the whole internal storage, we estimated by size. The command $ cat /proc/partitions will show contents of a file that contains partition block allocation information that will help identifying which partition we will need to copy onto our local machine for analysis. There were two partitions close to the size of the internal storage of the phone (32 GB) - mmcblk0 and dm-1. Both were copied over. To establish the TCP connection, the following steps were followed:

- computer: *$ adb forward tcp:9999 tcp:9999*
- phone: *$ dd if=/dev/block/ | busybox nc -l -p 9999*
- computer: *$ nc 127.0.0.1 9999 > img.dd*

To analyze these images, we used the forensic software Autopsy. In this particular case, the machine running Autopsy had to be Windows, as the needed functionality was not given in the version running on Unix. After feeding both sources to Autopsy, one of the images - mmcblk0 - delivered no results/information whatsoever. From this result we determined this partition to be encrypted. Android implements encryption by default since Android version 6. Our phone has Android version 9 PKQ1.180904.001 running, hence this feature is in effect. We were not able to retrieve any useful information from this partition. The other partition delivered better results, as it was not encrypted:

Index	Path	Description
1	Carved Files tab (Autopsy)	Any files that were deleted from the phone could potentially be recovered here.
2	/data/system_ce/recent_images	Includes screenshots (.png) of last used applications.
3	/data/system_ce/recent_tasks	Includes XML files of last used applications.
4	/data/system_ce/snapshots	Includes screenshots (.png) of last used applications.
5	/data/com.google.android.apps.chromecast.app/files/home_graph_dG90by5oYXJyaXNzb25AZ21haWwuY29t.proto	Includes information about the Google Home environment.

Table 6. Physical acquisition data

A benefit that physical extraction brings with itself is the possibility to carve for files. This way, deleted files may be recovered, as well as files present only in unassigned file spaces discovered. The only files we will not be able to discover this way are damaged or already (partly) overwritten files, as we simply do not have the data to repair them

(unless duplicates exist, but that's not the point of this study). The carving of files can be very interesting in the context of IoT because of their nature and special architecture - the devices do not have a lot of computing power nor big amounts of memory, thus anything stored on them (or the cloud - to save space) usually gets overwritten within a much shorter period of time than on other, more conventional devices such as personal computers. This also means, that if for example, hypothetically, a picture or screenshot of the videofeed of a network camera was automatically taken in response to a triggering event, then saved by the app on the phone, but eventually deleted after a certain timespan or manually deleted by an individual, it might still be recoverable through physical data acquisition and data carving. These files will not include any metadata usually, but can still be of high forensic value. Data in index 1 symbolize these carved files, which of course can also be files of any kind. Sleuthkit Autopsy natively includes the functionality to carve for files on forensic images.
Files we ordered into index 2 and 4 both contain screenshots from when the last applications were used on the phone. Those last applications can be cross-validated with the files from index 3 (in .xml format) that contain the name of the last used applications. The screenshots can give information about the interface and obviously apps that were last used.
Index 5 gives more information about the Google Home environment, for example the name of the home, name of the rooms, associated email, type of connected devices and functions of that device.

As can be seen, physical acquisition brings some novel data to the discussion which is not available in logical acquisition. We only listed data here that we had not been able to extract in the previous logical acquisition. Since the physical acquisition is a full bit-for-bit copy of the complete internal memory, everything that we extracted logically before this point, was also available in the memory dump of this step.

4.2 Common aspects

4.2.1 Google Home and Fibaro Smart Home

The first thing that needs to be mentioned is that while we only used the Fibaro API to parse data from there, Google Home does actually have an API as well. But, for the data stored by Google and its services, it was not necessary to aggregate it, as the possibility to export it in a reasonable file format already exists. For the sake of data acquisition and our research, it was not necessary to further complicate this process, though it would have been possible to access the data from the API as well. So, this difference in accessing the data should not be taken as a difference that might influence tool development - it was mostly a decision made for brevity and simplicity. Nevertheless, we want to point out that APIs are actually a very common approach to request and interact with data in IoT environments and most smart home service providers include an API for developers to implement their home automations[22].

The data we were able to access differed somewhat in structure - Google provides a much more broad spectrum of services and this might be the reason why data on and from smart home integration is less detailed with Google. Since Fibaro aims to provide smart home automation, the data aggregated from Fibaro directly is more in-depth and sensor-centric which is needed for this kind of automation. Google Home does not provide this kind of functionality, at least with the Fibaro devices and integration, and also acts like a middle-man - it does not receive the data from the sensors themselves, but from the Fibaro API. Google has its own products (like the Google Home among others) that provide more functionality and only somewhat supports other brands, mostly because it is so prominent and widely used, that a lot of services depend on integration (and most likely pay a price) so that people will even consider using their service. Available functions implemented by Fibaro for the Google Assistant can be found online to assess the level of integration[23]. Because of that difference in what each service/provider actually is and does, there is a noticeable difference in the provided data. The data formats from the Google Takeout exports were rather generic and general - the directories were arranged for the different services/categories Google provides, but each mostly only had a file called "My Activity.json" that includes different kinds of information. For example, the HomeApp.json contains all

data that could loosely be associated with the smart home itself - devices, names, owners; while Fibaro has smaller subsections, as devices, rooms, and users are each separately queryable. While there is a difference, that also means that the data is available on both interfaces - Fibaro and Google Home; the difference in structure just needs to be taken into account.
That said, both data sources contain data of comparable information, namely:

Type of data	Location Google	Location Fibaro
Device data	/Takeout/Home App/HomeApp.json	http://<IP>/api/devices
User data	/Takeout/Home App/HomeApp.json	http://<IP>/api/users
Home data	/Takeout/Home App/HomeApp.json	http://<IP>/api/settings/info
Room data	/Takeout/Home App/HomeApp.json	http://<IP>/api/rooms
Mobile device data	/Takeout/My Activity/Android/My Activity.json	http://<IP>/api/linkedDevices; http://<IP>/api/iosDevices; http://<IP>/api/voip; http://<IP>/api/devices
Events data	/Takeout/My Activity/Assistant/My Activity.json	http://<IP>/api/panels/event

Table 7. Common data types

As can be seen from the commonalities across the Google Home/Assistant services and the data that is provided by the Fibaro Home Center Lite, there is data describing the same or similar information. This data is not always 100 percent the same, which can be explained by the fact that the devices used, specifically the Google Home Mini and the Fibaro Home Center Lite have big differences in their functionalities. As mentioned - the Google Home Mini/Assistant acts more like a middle-man or bridge between the user and the application that employs and integrates the actual sensors for easier access and control for the end-user, that being the service provided by Fibaro. The Home Center acts as a hub for the sensors, receiving and processing all data that is generated by them. Some information that could be very crucial for investigations is not stored or processed by Google.

4.2.2 Mobile

From mobile acquisitions it became apparent that on Android there are a lot of similarities. Even between the different services of Fibaro and Google Home/Assistant, as data is managed in apps which are somewhat universal and this could be considered the same kind of source of data.

From the data acquisition on mobile it became apparent that on Android there are more similarities than across APIs and whatnot, as Android handles data more uniformly in applications and such. This way, even if data is accumulated from different apps, it is most likely going to end up in similar places. Though, this does not necessarily mean that same types of data is stored or that different apps generate/store the same information.

The data possible to extract from mobile was less specific to the different services (Fibaro, Google Home), but instead gave insights about general things such as (last) app usage, calls and contacts, installed applications, and WiFis the phone has been connected to, as can be seen in tables 4-6. Hence, data possible to analyse from this source is more user-behaviour-centric. Nevertheless, most information found here does not give a whole lot of details about the smart home environment, especially concerning Fibaro. Actions made on the sensors and events triggered are also not accessible/visible from the mobile phone. As storage is limited, only the essential (and manually saved) data is stored.

4.3 Framework Development

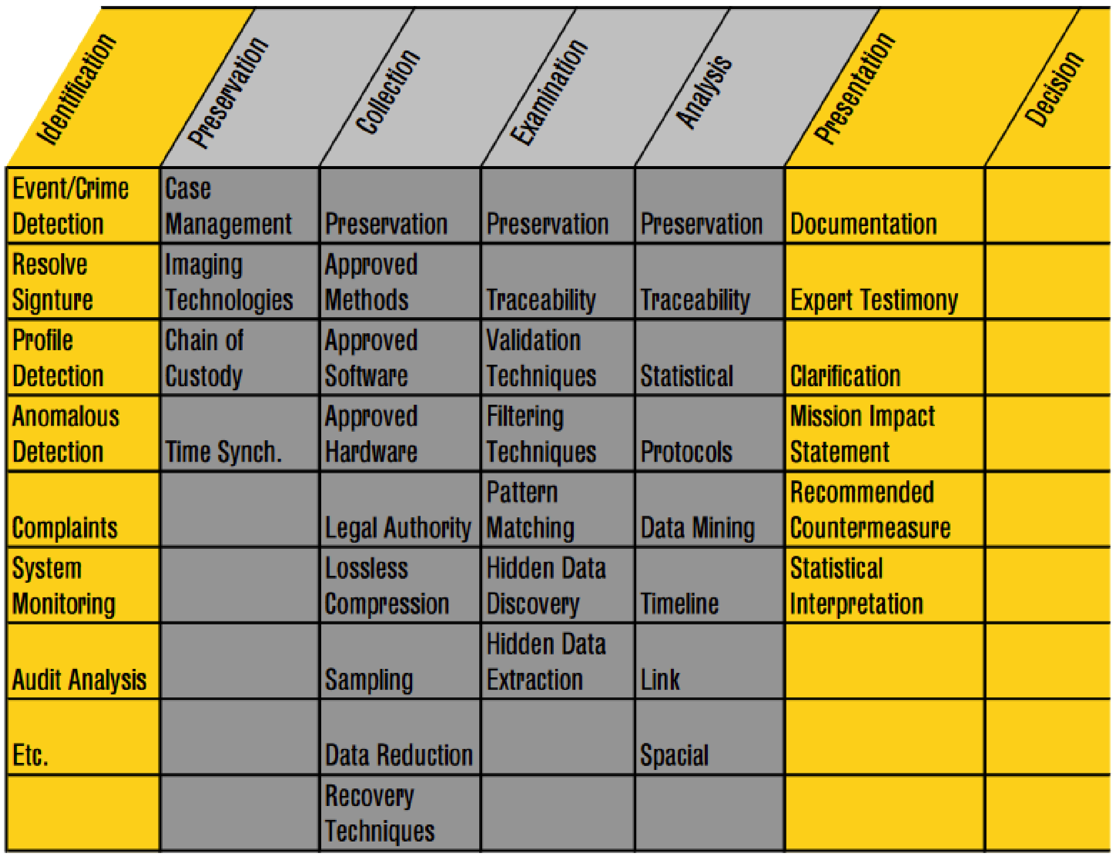
In this section, we are going to present our proposed framework and explain the process of developing it, as well as providing explanations for decisions taken.
This framework aims to provide future researchers and investigators with a guide to kickstart automation in forensic investigations in smart home environments, specifically to further forensic software development for data acquisition when dealing with smart homes. The framework we propose consists of two parts - one being an organisational guide that helps to define and structure processes needed for the software development project, the other being a more applied and technical part of the framework in form of blueprint or base classes containing predefined functions that can be applied to various (data) sources, needing only small adjustments to fit the source-specific case. First, we present the organisational framework, then moving on to introduce the concept of the applied framework.

To identify which phases of an investigation our framework can be applied to, we will first establish these phases and how they are typically defined. The two most notable models that define the general investigative processes in digital forensics are published by NIST (National Institute of Standards and Technology) and the DFRWS (Digital Forensic Research Workshop). The following is the investigative process as presented by NIST[24]:

- **Collection**: In this step, it is first of all essential to identify and record all potential evidence sources that could be relevant to the incident. Next, the data from those evidence sources needs to be collected in a preserving and forensically sound manner to ensure integrity.
- **Examination**: Once the evidence has been collected, the data needs to be examined and assessed. Evidence should be separated into relevant and non-relevant to the case while still ensuring its integrity.
- **Analysis**: In the next step, the information given by the evidence separated in the examination step is analysed to try and find answers to questions such as "what happened?", "where?", "who was involved?", "when?", and so on (5WH questions).
- **Reporting**: Finally, the evidence needs to be prepared and results presented, includ-

ing methods and tools that were used during the investigation.

In comparison to that, we take into consideration the investigative process by DFRWS which offers a more detailed visual:



Identification	Preservation	Collection	Examination	Analysis	Presentation	Decision
Event/Crime Detection	Case Management	Preservation	Preservation	Preservation	Documentation	
Resolve Signture	Imaging Technologies	Approved Methods	Traceability	Traceability	Expert Testimony	
Profile Detection	Chain of Custody	Approved Software	Validation Techniques	Statistical	Clarification	
Anomalous Detection	Time Synch.	Approved Hardware	Filtering Techniques	Protocols	Mission Impact Statement	
Complaints		Legal Authority	Pattern Matching	Data Mining	Recommended Countermeasure	
System Monitoring		Lossless Compression	Hidden Data Discovery	Timeline	Statistical Interpretation	
Audit Analysis		Sampling	Hidden Data Extraction	Link		
Etc.		Data Reduction		Spacial		
		Recovery Techniques				

Figure 3. DFRWS Investigative Process[25]

As can be seen, an investigation can consist of many phases, which are also not always completely separable. For example, in the model by DFRWS, preservation is a step, which is also included in basically each other phase. It is necessary to preserve the evidence and ensure its integrity throughout the whole investigation process to in the end be able to actually use it as evidence in front of the court. For the scope of our research, we are specifically taking into account and developing for the phase of evidence collection/acquisition, extending towards examination of the data, specifically filtering and data extraction. Nevertheless, the framework may still be applicable to other steps of the investigative process, though the validation of this assumption and further development and additions to the framework are left for future research. The decision to focus on this particular part of the process stems from the necessity for automation in big data processing, whether it be acquisition or analysis. With our research we are aiming to propose and lay a base framework for future studies to build upon and extend as needed. Hence, we are starting with the initial step that involves data processing. Additionally, some solutions have been proposed for the discovery and identification of devices and possible evidence in a smart home environment.

As there exist quite a few framework in the forensics domain already, taking a look at existing work, we can take inspiration and derive a model that fits our own premise. While a framework like the one we propose does not exist, there are manuals describing the general forensic actions to be taken during an investigation, from start to finish. Some of these are also specific to Internet of Things, for example Islam et al. describe a comprehensive approach to to a digital forensic investigation framework for Internet of Things to give guidance for forensic investigations in such environments and to reduce dependencies on cloud service providers and network logs in an ongoing investigation. Their framework is provided in form of a flowchart with some additional elements[26].

Thus, understandably so, while there exists a multitude of different kinds of frameworks, IoT environments have only seen an increase in research in the last few years and the focus of those newly developed frameworks concerning IoT has been on how to adapt the existing investigative processes to include and better prepare for IoT as part of a crime scene and evidence source, or have rather specific use cases and research interests, and consider forensic readiness, while our study focuses on practicality and actuality. Tool development as a framework is also not something that has seen much traction, as there a few bigger commercial tools and companies that focus on tool development. While there is not one forensic tool that satisfies all needs, it is usually possible to supplement different options to one's use case to cover as many aspects as possible. But when it come to IoT there has not been made much progression in tools to include such devices and environments, as it is not an easy feat, taking into account the nature of IoT devices, the many different kinds and manufacturers there exits, the big throughput of devices, as well as a lack of standards, and further complications introduced through proprietary software, hardware, and communication.

Because of these challenges faced in IoT forensics, tools and their manufacturers have not been able to keep up with integrating support for smart devices and the changes that come with a smart home environment.
Additionally, when considering IoT and IoT forensics, it should be taken into account that the mobile component, and thus also mobile forensics plays a big role. Data is spread across different locations, including the phone(s) that is usually being used to control the smart home variables. Though from the data we were able to recover from the phone during our research, the most interesting parts (as we think) of the data such as triggers and breaches of the actual sensors, were not recorded on the phone itself and only accessible on cloud-/server-side through the API, or in the special case of Google, from the Google account itself (Google Takeout).

Mobile forensics is a category and topic that has been around a while longer than IoT forensics and there exist tools that support the analysis of data and images from these devices, such as Sleuthkit Autopsy, FTK Imager, Encase Forensics, and some tools even specialize on extraction from mobile devices, such as Cellebrite Touch or Oxygen Forensic Suite[27]. While mobile forensics face some of the same challenges as in IoT, e.g., a magnitude of different devices, in mobile the variability between devices is not as extreme as it is the case across IoT devices since phones are still going to be phones and nowadays there are just a few prevalent operating systems. But IoT devices can have very different, very specific purposes. The differences are less severe, which makes data acquisition a bit more predictable.
Despite all this, information on and from the phone can be and is still an important source of evidence for the investigation and should definitely be considered when collecting data. Most smartphones nowadays give great insights into their owner's life and more. Thus, while not our research will also take into account the mobile component of data acquisition.

We want to mention here that it is also very much possible to extend our framework further to include more data sources, such as the network component. For example, the local network could be analysed for additional information. It is even possible that otherwise unavailable data is transmitted in plain text, such as login information. The network aspect was outside of the scope of our research due to time limitations, but it could be a valuable addition to the framework.

As the framework we are proposing is basically software development, it is possible to use an existing model or methodology for software development as a basis which the framework should follow. There were a few things to consider which change the usual software development premise: While there do exist some major producers of IoT devices or smart home appliances, there is a wide variety not only of devices, but also producing companies. As already established, there will be minor and major differences across products and producers, also when it comes to data generation, storage, and APIs for example. Some devices and producers may be more common than others, but in many cases tools will be developed for and due to current needs, not considering what may be best for the future. This may happen under time pressure of investigations or research projects, and the tools/automation may fit very specific needs and is most likely not going to be developed for a very general or broad purpose. Thus, reusability may be limited and existing tools along this framework may first need significant adjustments to be useful for other cases. Nevertheless, existing tools will create a fantastic basis and code base to build upon, even if tools are not actively maintained.
Realistically speaking, it is also unlikely that developers of such forensic tools, especially when the tool is being developed for an ongoing investigation, will have much time or thought to spare on code quality and code reviews. The language being developed in, will most likely be decided by the skill level of the developer and the use case. Wanting our framework to be life-like, realistic, and practical, we are taking these factors into account. Considering existing methodologies that we could apply, we conducted a short comparison of the waterfall and agile methodologies to determine which could better fit our needs.

While one may think at first look that having several iterations of development within our premise - time pressure and the need to get results as quickly as possible - may not be the best idea. But looking into it more closely, a more agile approach to project

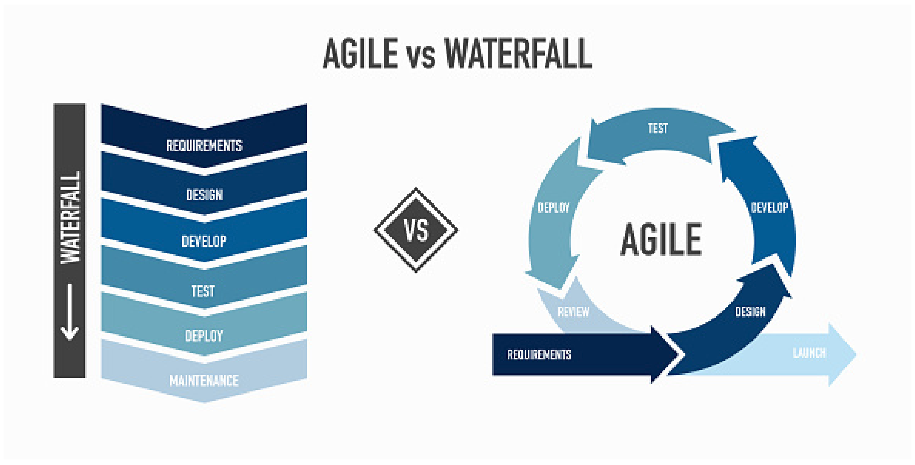


Figure 4. Agile vs. Waterfall[28]

management can benefit the development in several ways. The very rigid and linear waterfall development is less than ideal for the general use case for our framework, as we expect it: each step has to be 100 percent completed before moving on to the next stage - it is not intended behaviour to go back to some previous step to adjust or modify. Having a linear development structure can be favourable in projects where clear requirements and objectives can be defined. Such projects may adapt the waterfall method very well, but for our purposes such a set-in-stone structure can mean delays and slowness in the development. As already expressed, time can be critical when it comes to forensic investigations, especially with IoT involved. Additionally, requirements may not be completely clear or definable when first starting the development process and certain aspects may only arise during a later stage. It may also happen that during implementation it is found out that certain requirements are not possible to be met. While the agile methodology generally focuses on an approach to extensive testing, feedback, and attaining quality code, it can be applied in a way to benefit our framework - smaller subsets of requirements can be taken and implemented to realize if the requirements are attainable or need reworking. In turn, several iterations of "feature" releases can be completed, so developers may focus on most (time) critical requirements, with a MVP (minimum viable product) release, proving first functionalities earlier than for example with the waterfall methodology. Hence, we concluded that approaching our framework with the intention of using agile methods as a basis, made the most sense.

Though, for our purposes and in general the purpose of forensic tool development, these models can be somewhat reduced, as the software development is not so customer-centric and certain steps of the process may require less efforts than in usual software development. In the following, we will present the first part of our framework based on the agile methodology and explain each step.

4.3.1 Framework

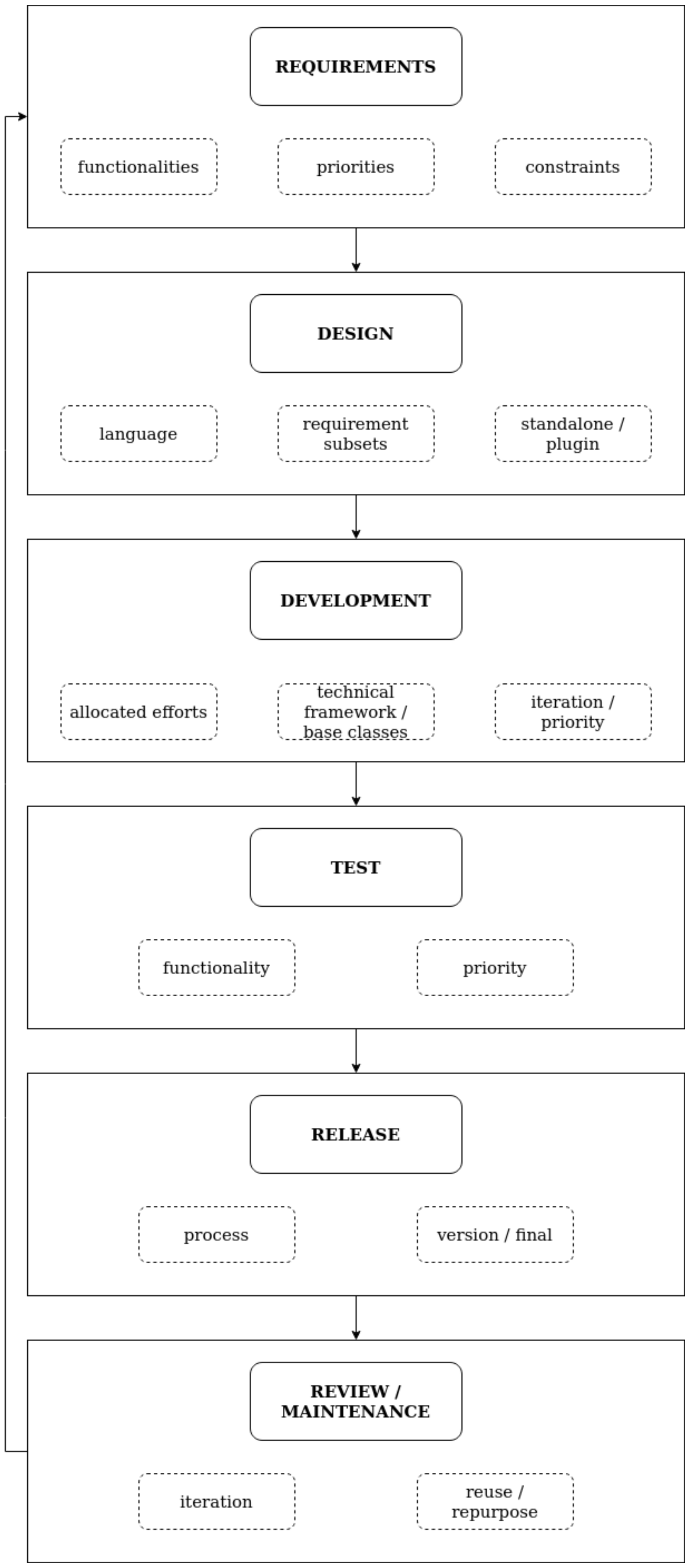


Figure 5. Organisational Framework - Forensic Software Development

We kept all steps that are defined in the usual agile model, but want to emphasise that these phases may hold different weight, depending on the project. This is something that should be decided individually and it needs to be considered. We will go more into detail about this in the following:

- **Requirements**: Requirements are easily the most important part of the development cycle. Setting clear and precise expectations for the project and precisely the tool to be developed can help with expectations and faster progression in actual development. Requirements will mostly concern the tool (development) itself, nevertheless the project as a whole should be considered. Most important aspects to consider when defining requirements are time constraints, necessary functionalities, and priorities. Time constraints can come from ongoing investigations that need to progress in a timely manner, or the possibility of losing valuable evidence if relevant data is not extracted soon enough. Quite obviously, requirements should focus on functionalities the finished product aka tool should offer. When defining requirements, it can be beneficial to assign priorities. Since there will most likely be several iterations and some functionalities may need to be prioritized, as mentioned possibly due to time limitations or different levels of relevance of the data. Generally, requirements could be looking at the scope of the project, supported devices or producers, which data sources will be considered, where the data will be taken from (e.g., API, mobile acquisition), what kind of data processing will be done, what the output should look like. Requirements should cover everything considered to be relevant to the project, and can also be adjusted after each iteration.
- **Design**: It is possible that in general for forensic software development, the design phase does not require a lot of effort. This does not mean design is not important, as it should still be thought through. Things such as the programming language may be chosen by the developer(s), considering their skill level or preferences. Other factors may include the use case, as a research project can look very different from a project within the scope of an investigation - time constraints may be different, thus also the priorities. Priorities should be defined within the requirements and applied in the development step. The decision (made in requirements preferably), if the tool should be developed as a plugin for an already existing (forensic) software,

such as Autopsy, or if a standalone tool will be the result of the project should be made. It is of course possible to do both within the same project, but time and efforts should be considered when making this decision, as developing plugins brings more constraints with itself.

- **Development**: The development phase will be supplemented and supported by the technical part of the framework which will be introduced shortly. The actual development will depend a lot on defined requirements and the project itself. For each project, it should be decided how much time and effort should be allocated to the design, and in which iteration it should be considered. The priority level of the design can vary greatly from project to project, depending on e.g., the use case the tool is being developed for. In case of several runs or iterations of the development cycle, those requirements that have been prioritized should be taken into account. Subsets of requirements should be pushed for development first, which can also help assess if requirements are well-defined and realistic, or if changes need to be made. It may also happen that one run of the cycle may fail completely, if development of set requirements proves not possible within the scope of the project.
- **Test**: In most circumstances, after development, at least functionality testing should be done, to ensure correct behaviour. Testing should be subject to prioritization as well - in some cases, for some functionalities, more extensive testing may be necessary, for example, if a UI is being developed. It is important to decide this for each use case, run, and even functionality. Some functionalities with high priority within the project may (need) go through several testing phases. Prioritizing testing for some functionalities may be necessary, while leaving less pressing testing for later iterations.
- **Release**: In the environments in which these tools will be used and released, will usually not need deployment or launch as such. First, it should be decided if there will be a release in every iteration, like a version release cycle, or if only the final product will be made available if a release is planned at all. It is of course possible to not release the tool and only use it for whatever purpose it was intended to be developed for. Though, we really encourage "releasing" the developed software in any way. Version release can be useful if e.g., different iterations actually implement different functionalities. The process of the release should be defined, for example,

where (e.g. GitHub, a research conference), at what milestones, progress, or time the tool will be released.

- **Review / Maintenance**: After each iteration, there should be a review of if the subset of requirements has been fulfilled accordingly, any requirements should be changed, or if there were any occurrences throughout the run that could be improved in the next iterations or need adjustments in general. Additional review and maintenance can happen both internally or externally, as after the release of the developed software, other developers, or same developers for that matter, may decide to reuse an existing code base that fulfills a similar purpose. Then, the cycle will start anew - requirements will be (re-)defined, and functionalities adjusted and repurposed accordingly to the needs.

Having defined and explained the necessary steps included in our framework, we are going to have a detailed look into the development part, as we have gained some insights from the manual analysis of the data generated by our experimental smart home setup. We hope to provide guidance on possible locations where relevant data could reside, giving example file paths from our case for others to apply them to their own. These file paths will also be compiled in a list in the appendix for a better overview and for others to add to it more easily.

4.3.2 Code Development

In real life settings, it is only realistic that the author of the tool to be developed will choose a programming language that they prefer or if developing a plugin, the language best supported (API access) by the already existing software. While it would be reasonable to ask authors to follow good coding practices and use languages that are common and fit the task, but while short on time and with investigations needing to progress, we understand that what works and is known to the developer/investigator is best and will be used. In any case, it would be good to at least keep this in mind, even if it is not something that can be asked or set as a hard requirement.

Mobile

If a mobile phone has been recovered that relates to the smart home or person of interest. Basically, the analysis can be done with most forensic software of choice (e.g., Autopsy), as physical dumps of most Android devices will be recognized and parsed there, since they are in a way just small Unix systems. But, the part of data acquisition is something that could be considered for development if wanted or needed, for example for reuse purposes. In this case, since Android Debug Bridge will most likely need to be utilized, an automated script for the data extraction as it was manually demonstrated by us, could be written. It should be kept in mind that for physical data acquisition the phone needs to be rooted and this is a process that is usually specific to phone model, and difficult to automate, though this can be attempted if for example, the same kind of model is expected to be needed to be rooted repeatedly.
It should also be noted that the automation of for example, server-side requests made through an API is usually a more fruitful work, as reusability is more realistic. APIs are a common thing, especially with smart home environments and even different models of hubs or base stations that gather the data from the sensors, will usually have very similar API access within one manufacturer/vendor. Referring to the data and tables from the data acquisition section, we found locations on the phone, where relevant data was stored. We will give these locations as recommendations to be considered when developing tools.
The camera roll or rather any folders where pictures or downloads are stored, can be a rich source of information. Be it in general of things that were taken pictures of with the phone and pictures that were downloaded to the phone, or what we could observe - pictures that were taken within apps, like those of smart cameras (in our case mydlink app). These pictures may be in the common downloads directory, or a folder might have been created by the app itself. Interesting directories of this kind in our case were found under the following paths. It should be taken into account that these paths are only representation of a case study and should not be taken as absolutes. Please refer to our phone model and compare.

- **shared/0/DCIM/Camera** - This directory included pictures and videos taken with the phone.
- **shared/0/DCIM/Screenshots** - Screenshots that were taken with the phone will be

stored here.

- **shared/0/Pictures/mydlink** - Pictures that were taken within the app, in this case mydlink were stored here.

As stated, this framework concerns itself with data that is possible to be extracted from various sources which is related to IoT and smart home environments. Hence, the phone will only be considered from this aspect, though a complete data acquisition and analysis of the phone may bring about other information or data relevant to the investigation or research.

Generally, data regarding apps will be stored in the data partition of the phone of which an extraction will require physical acquisition (a full bit-by-bit copy of the internal file system) methods. The most interesting artefacts are usually found in the directories **/data/data**, **/data/system**, and **/data/misc**.
Common locations to look for information concerning applications are:

- **/data/data/com.android.<app name>**
- **/data/system/packages.list** - This is a list of all installed applications.
- **data/system/packages.xml** - This file will also contain information about installed applications and their permissions.
- **/data/system/usagestats/0/** - Subdirectories contain data about usage statistics of applications.
- Additionally, **/data/system/sync/accounts.xml** contains information about accounts that are used for apps.
- **/data/system_ce/recent_images & snapshots & recent_tasks** - Information about last used applications.

Locations to consider for Google services can for example be:

- **/data/data/com.android.chrome/app_chrome/Default/** - Subdirectories contain database files about user's behaviour for the Google Chrome browser.

- **/data/data/com.google.android.gm/databases/downloader.db** - Database with files downloaded via Gmail.

In case of IoT devices produced by Google and Google Home integration of other smart home accounts/manufacturers’ products, there are likely additional file paths, for example:

- **/data/com.google.android.apps.chromecast.app/files/** - Information about the Google Home environment.

The **/data/misc** directory can contain information about general settings, e.g.:

- **/data/misc/wifi/WifiConfig-Store.xml** - Data stored about wireless networks that the phone has been connected to.

This may provide additional insights.

API

The data sharing and processing across smart home environments is usually handled by using APIs, though there may not always be support for a private person to use that API and no (official) documentation available.
For this section we would like to refer to Table 2 which describes the functions that were available for the Fibaro API through which we extracted the information as needed.
For starters, when first identifying the products present in the smart home environment, it should be researched if the specific manufacturer or producer provides API access and documentation. Should that be the case, which is very likely, the documentation should be explored to figure out available functions and information that can be queried, as well as considered which of that information or data should be prioritized.
Another factor that should be explored is the availability of the API - if it is something only available on the local network or if it can also be accessed remotely. It is likely to be the former, which should be kept in mind for development and usage.
Most of the time, request are easily written and executed in python. A script that was quickly developed within the scope of our research can be found at the linked GitHub and serve as an example. In our case, requests were made over HTTP and the responses were in JSON format.

In case an API is not available, it should be explored which is the easiest and quickest way to extract the data without too much interaction with the smart home environment itself. For example, Google Home does actually have an API from which data could be requested, but since the Google Takeout exists it was less time consuming to simply access the data needed from there, as the file formats were common and could be easily fed to forensic software for analysis. It is usually not as easy and straightforward to develop data acquisition tools and automation when there is not a standardized way to access the data, though it definitely is not impossible. If it is not possible to automate the process of evidence collection with our framework, it may be possible to develop a tool for the next step - data examination/analysis. For this step, the developer could for example try to automate the sorting of data into relevant (to the case or in relation to IoT) and irrelevant along our framework. It is no set in stone which step of the process can automation be developed for with it, even if it was initially intended for data acquisition.

4.3.3 Technical Implementation

In this section, we will provide insights on implementation and development of a practical part and adoption of the framework. This will further aid the development stage of the earlier presented organisational-focused software development framework (Figure 5).

A substantial part of the framework we are proposing is to include a technical implementation of it. Meaning, a blueprint in form of base classes is provided that implements generally useful functions. These classes and their functions are constructed from references given by our case studies, taking into account what information can usually be found, where, and in what form. With these base classes we are laying the basis for easier adoption of forensic tools and inclusion of data sources not covered in our research. We conducted our case study with the goal in mind to answer our research questions and provide support for investigations and the research community. The reason to look for commonalities in data builds on the construction of this technical part of the framework. Aspects that are common can be summed up and integrated in this proposed technical blueprint in form of classes. In our scope, we will construct a prototype implementation, writing Python classes for certain tasks that could be useful in investigations and research. These tasks could consist of automated API requests, saving the servers responses in the desired file format, or a quick filtering of the necessary and interesting information from the extracted source files.

The idea is to have one Python file where the functions are defined. The functions will be more generic than an implementation for specific tools and require arguments that will provide the level of customization needed for the smaller differences in the common aspects we established across IoT sources. These classes can then be used to implement the needed functionalities, only needing small adjustments or passing arguments. This will speed up necessary investigative processes by a great margin and provide future tool development with a solid base to be built upon. Basics will be given by the framework and extended as needed. Functionalities will aim to unify file formats and aggregate information in one place for easier accessibility and analysis.
The base classes will be inherited by extending classes - the ones for specific use cases,

e.g., Fibaro or Google Home/Assistant. Certain variables are required to be defined by subclasses of this base class, so that necessary and source- or product-specific information will be applied, to make use-specific applications of these blueprint classes.

Basic functions can be implemented in the base class which we were able to identify as commonly applicable across the data sources and products used in our case study. We derived these functions from analysis of the data we generated, looking at aspects that are common or similar from different sources (Table 7). We identified following functions as useful:

- **API requests**: As APIs are generally a common occurrence with smart homes, having a baseline for such requests can accelerate data acquisition.
- **File format conversion**: For further processing it is essential to have files stored in a uniform format and some file formats may be preferred in investigations or research.
- **Timestamp conversion**: Converting timestamps into human readable formats is essential for investigations and presentation in front of the court.
- **Data extraction**: Big volumes of data which needs to be analysed for relevant information waste a lot time and efforts if this has to be done manually. Extraction of a few values that are interesting for the case or research can save this wasted potential.

The classes can be constructed as visualized by the following UML diagram:

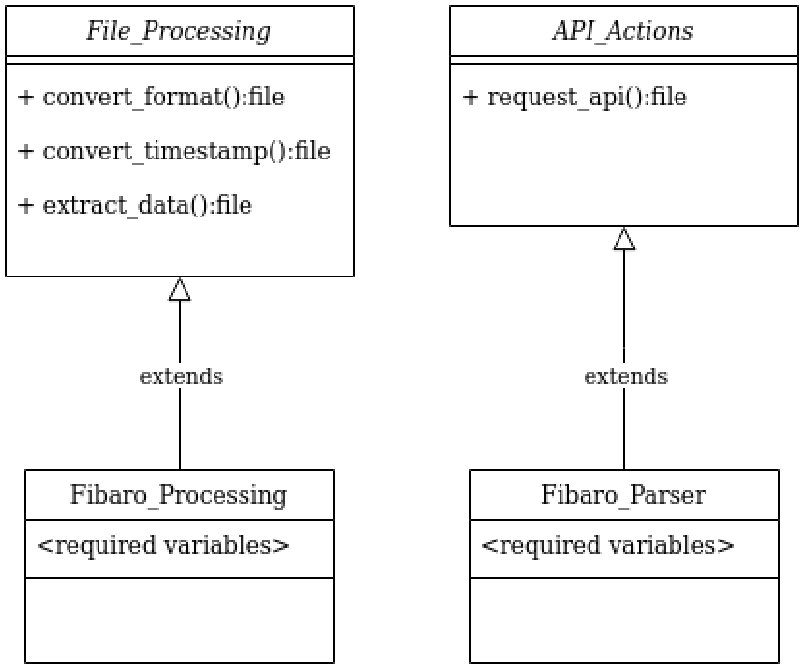


Figure 6. UML diagram hierarchy

Two base classes are constructed - one holding functions that perform actions on files, and another where actions/requests for the API are defined. These classes are separated for a better overview, each of them having the potential for extension.
In this example, subclasses are given which are extending the base classes. They inherit the functions of the base classes and contain the specific required variables that are responsible for the customisation to fit the use case, here, Fibaro.

The developed base classes that implement these functionalities can be found on my GitHub.

5. Framework Validation

To validate our proposed framework, we developed a standalone tool, following the organisational part of the framework and applying the technical part of it. In this section we are going to describe the process of software development along the framework and using the base classes to inherit given functions which should require only minimal effort to adjust to the specifics of different data sources.

If time allows, we actually recommend developing modules for Autopsy, because it is a widely used forensic suite. Sleuth Kit being a collection of utilities for data extraction and analysis, and Autopsy providing a graphical interface to Sleuth Kit to analyse extracted data. Individual developers are able to add functionalities to Autopsy in form of plugins that can be written in either Java or Python. There are different kinds of modules which can be developed:

- **Ingest Modules**: Analyse data and content from different data sources and relay results to the backend, for the user to be seen. This is the most common extension point for Autopsy.
- **Report Modules**: Report modules are usually run at the end of investigations, after the data analysis, to produce a report.
- **Content Viewers**: These are graphical modules and allow the user to view and analyse files in different ways. The content viewer is located in the lower right part of Autopsy.
- **Result Viewers**: In contrast to content viewers, result viewers show information about a set of files and are located in the upper right part of Autopsy. This is the least common type of module to be developed for or extended.

The developed modules can be added to the Sleuth Kit's GitHub repository for individual add-on modules where they will be openly available in a centralised manner for the community to download and install.

In the following, the validation has been documented and resulting code has been published on GitHub. In the following, we are going to describe how the project and code was developed while adhering to the framework, documenting the processes and decision.

Requirements

Following the framework, the first step towards the goal is to define clear requirements that the finished product should meet. As for our scope - mainly the validation of the framework - the tool does not need to be extremely sophisticated, but should serve a purpose. We focused priority on functionality, considering less the environment in and for which the software is being developed.

We decided to develop the tool for Fibaro. Since we already wrote a program to extract the data via API, wanting to add something novel, for this tool we apply the framework to the step of data analysis and (pre-)process the data that we extracted. In particular, we will be dealing with JSON files, from which we want to extract some specific information. Following the framework, the first step is to define requirements that we want the tool to fulfill.
We are going to be working with the JSON files that we extracted by making requests to the Fibaro API. For the scope of our work, we are going to define the following requirements. The tool should extract the following information from the files:

- Serial number of the home center (top priority)
- Name of the home center and rooms (top)
- IP and MAC address of the home center (top)
- Software version of the home center (top)
- Home center users and devices/sensors (top)
- Single out file that is storing events (top) and convert timestamps to humanly readable (mid priority), convert to csv (mid)

Assigning some priorities to the functionalities - all data we want extracted, including identifying the events file, should take top priority and be included in the first run if possible. It matters less in what format this data is provided in the first version, but a later run should include a kind of report or csv table with this information. The file and time conversion can be added after the first priorities have been fulfilled. Generally, if this was a bigger project, including more functionalities, they would be grouped into several subsets, based on priority. In our case, since we have only a few functions to implement, it will not take a lot of time nor a lot of iterations.

Furthermore, for our purposes a UI is not necessary, as the software we are developing can be counted as a pre-processing step or further data extraction, and will possibly still go through some more analysis.

Design

Within the design phase, basically everything needed to set the environment for development is defined and software architecture built. The software is defined by the requirements we set, and because it will be just a cli tool, the graphical aspect was not considered extensively. The programming language to be developed in is Python, which we chose due to personal preferences, skill level, and due to the fact that the provided base classes are written in Python which we are trying to validate. Requirement subsets were defined, in our case simply by priority, though estimation of effort and other factors can also be taken into account when defining these subsets that are to be implemented in different iterations.

Development

Generally, the efforts within a team will be allocated and the software in accordance to the requirements and priorities, and defined software architecture developed. Since this was a one-person-job, this step simply included the implementation of design and requirements. Since we were already working on data that we analysed, there was not much need to go into detail in the analysis again. But important findings that are discovered during the process may be documented here or in the process of defining requirements, for which it

might be necessary to already do preliminary research on the data sources.
For the actual development we used the File_Processing base class as shown in Figure 6 to inherit the extract_data() function. Only few adjustments were needed to make the blueprint fit for the case of JSON files generated by Fibaro.
We ended up doing two iterations - the first implemented all top priority functionalities and the second one added the second subset of requirements - the conversion of time and file, as well as output being in csv instead of just on the command line. The iterations went through all processes, though in the first two steps of the second run, nothing was changed.

Test

Testing was done only to check for all implemented functionalities and adjustments made in the code, in case of errors or unintended behaviour.

Release

For backup purposes, accessibility, and synchronisation, we decided GitHub to be the place for the release. It is a central platform where code can be deposited and accessed. We uploaded the code for every run, so technically there exist two versions.

Review

After each run, we reviewed the code and checked if all requirements and priorities had been met in the developed code, which was the case.

6. Discussion

In this part, we are going to summarize our results.
Firstly, the literature review we conducted helped us establish a good basis for our own research. We presented how IoT systems work differently than conventional computers, introduced general security flaws and vulnerabilities in IoT, and identified current challenges and possibilities in IoT forensics. The most severe issues which complicate forensic investigations in IoT environments and need to be overcome are big volumes of data, heterogeneity, and volatility of data. Furthermore, existing frameworks in the forensic domain were explored to assess if it would be possible to apply our own use case to an existing work and adapt it accordingly to our needs. In the end, this was not possible, as only frameworks for the general forensic investigative process could be found, some integrating the IoT component, but no software development frameworks that could apply in our case. And finally, we reviewed research that presented case studies on IoT devices, which provided us with a great deal of references. From this information we were able to establish why our own study holds value and contributes in terms of research, as well as in practical scenarios such as investigations.

From our research experiments with a life-like smart home environment we were able to generate data for a case study. This study included the mostly manual extraction and analysis of IoT-related data from the mobile phone, and on server-side from Fibaro through API requests, as well as Google Takeout for relevant data for the Google Home. For the API requests and responses a simple parser, as well as file converter was developed to ease access and interactions. From this case study, we gained insights about locations of data generated by IoT devices which could be interesting for investigations, and possibilities to access and extract this data.
From analysing the data, we establish that common aspects could be found across the data and data sources. When considering the comparison between data gathered from Google

Takeout and by making request to the Fibaro API, some of the same information could be found, which can be relevant in investigations. Table 7. provides an overview of this data - there were certain files that provided the same information, even if they were from the different sources. Other commonalities were data locations on the phone, as the data was generated by applications which have a standard way of storing information. Differences, less in data sources but extraction methods of the data from the Google Home and Fibaro Home Center - data from Fibaro could be easily accessed through a RESTful API, resulting in JSON responses, while the information from Google could be downloaded through the Google account which included various formats.
Additionally, even though there exists integration for Fibaro into Google Home, the data generated by Fibaro that was accessible through Google Home was much sparser than the amount that was actually created. This is likely due to the fact that Google only needs access to a few functions to control the sensors, and does not aim to read and process all information which is not relevant for integration of the sensors.
Google being a somewhat monopolistic provider of certain services, is a special case, resulting in these differences. Google mostly does not produce smart devices itself, but offers other producers to provide integration to Google Home - a convenient central unit from which the smart home can be controlled. Nevertheless, APIs are a common thing provided by the producers, as automation plays a big role in the smart home setting and IoT in general.

Taking into account the findings of our case study and data analysis, establishing common factors across data and data sources, we developed a two-part framework for software development based on the agile methodology, supplemented with a technical implementation of base classes with predefined and easily adjustable functions fitting the commonalities we were able to identify. To better fit the use case of the framework, the implementation of agile methods was adjusted accordingly, putting more focus on some steps and less on others.
Results of our studies showed that there indeed are common aspects across different smart devices and producers, as well as that providing general guidance on possible file paths, formats, or API requests can be useful in providing initial pointers to guide development in the right direction, and help with accelerating the development process, as well as getting

started faster.

We validated our proposed framework and blueprint classes written in Python by developing a tool that provides automation in data acquisition and pre-processing on top of files extracted from the Fibaro API, following the processes of the framework, extending the base class, and using the relevant functions provided by this class.

As pointed out by relevant literature, the need to be able to analyse data from IoT settings grows, along with the the need for automation in this area. With our framework we provide a basis to address major challenges in IoT forensics by guiding and encouraging tool development to automate data processing step in forensic investigation.

7. Conclusion

With our research, we were aiming to establish the current state of IoT forensics and related tool development, and find common aspects of data traces left by different IoT devices in smart home environments. The main goal was to then combine these aspects into a framework to aid forensic tool development in automation of data acquisition, with a possibility to extend this to further steps of forensic investigations.

By conducting a literature review, we were able to identify the current state of IoT forensics, where research is lacking, and how our work can contribute. The general consensus is that IoT forensics is a very little researched topic that needs to gain traction, as the necessity to analyse data from such environments grows and challenges faced need to be overcome.

The experimental setup of a smart home ecosystem we constructed, provided us with the possibility to conduct a case study. Data generated within that case study provided us with insights about common aspects and differences across IoT devices and their traces. Commonalities were prominent especially on mobile, though different data sources (Fibaro API and Google Takeout) did include some of the same information, even if formats differed. We incorporated these results into our framework prototype, also taking into account findings from the literature review. Commonalities are considered in the organisational part, as well as the technical implementation included in the framework.

The organisation-centric part is an adjusted version of the agile methodology better fitting the use case of forensic software development, while the technical part is a practical implementation of Python base classes, consisting of functions that can be used and easily adjusted for several data sources.

In the end, we were able to provide a first version of a framework prototype that enables to accelerate and ease forensic data acquisition software development in smart home environments. With this framework, we hope that some of the challenges usually faced in IoT can be overcome to some extent, and further development of the framework can be conducted. Validation of the proposed framework happened by developing a tool from

the data we generated, following the framework, even applying it more to the step of data analysis rather than extraction. Thus, along with the framework, we contribute by making all of the data generated in our research available, providing the community with as many references as possible - this includes data we extracted during our case study, as well as any developed tools, which will be provided open-source.
Hence, we conclude our research questions to be successfully answered, even if the framework is at this time just a prototype.

The biggest limitation we experienced during our research turned out being bound to certain hardware that we had available. This should be considered when consuming our results. As already mentioned, the developed framework is simply a prototype in its beginning and by far no exhaustive study has been conducted. Adding more case studies to this work, in turn developing the framework further to be able to accommodate other data sources is something we hope will be achieved in future research. Be it file paths, example API requests, file formats, or additional functions in the base classes. Providing this information in a centralized document or guide can save a lot of time in the development. Additionally, we are hoping to see an extension of the framework idea to the network layer of IoT, as there is still a lot of data to be gather from traffic which could be very interesting to investigations[12]. As this was not within the scope of our research it was not covered within our work, but we want to mention it is definitely just as important, as for example, due to missing security measures, login data could be submitted in plaintext. With the same notion, we believe and think it to be possible to apply our framework to other steps of the forensic investigative process. As was already shown, we were able to use the framework on the examination step, even though it was developed with data acquisition in mind. If the framework is extended accordingly, this could be a real possibility to be considered and validated in future research.
Concerning the technical part of the framework, we see a lot of potential for adjustments, and extension. As further case studies are conducted and possibly other common aspects uncovered, additional classes and functions that serve as blueprints can be added. As our current framework is only a prototype, we encourage and hope for further development. Something we think that would be especially interesting and useful for future endeavours is the adoption of base classes for the development of Autopsy modules.

We hope that with our framework and incentive, open-source development will be encouraged, to bring the community as a whole forward and focus efforts on where they are needed, not wasting resources on repetitive tasks. If one-time use projects, specific to investigations can be avoided or at least reduced by publishing any developed software, which in turn can then be used by others and repurposed to fit other needs without having to start from zero, then there has already significant progress been made. It is likely that many investigators or researchers of this topic face the same problems. Working as a community can help a great deal in furthering research in this particular, but really any area. A team effort will bring research and investigations further. At this point, dealing with IoT forensically is simply not feasible. Challenges need to be overcome and the few commercial tools will not be able to do so anytime soon. We hope that in the future IoT will become more regulated, also in terms of security, but staying within a realistic perspective, this will not happen anytime soon. Hence, we have to focus on what is possible right now. We prepared a base to support researchers and developers in their endeavours, and hope it will stimulate growth and development among the community.

Bibliography

[1] Statista and Martin Placek. *Industrial Internet of Things (IIoT) market size worldwide from 2020 to 2028*. [Accessed: 12-05-2022]. URL: https://www.statista.com/statistics/611004/global-industrial-internet-of-things-market-size/.

[2] Statista and Lionel Sujay Vailshery. *Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2030*. [Accessed: 12-05-2022]. URL: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/.

[3] Jayashree Mohanty et al. "IoT Security, Challenges, and Solutions: A Review". In: *Progress in Advanced Computing and Intelligent Engineering*. Ed. by Chhabi Rani Panigrahi et al. Singapore: Springer Singapore, 2021, pp. 493–504. ISBN: 978-981-15-6353-9. DOI: 10.1007/978-981-15-6353-9_46.

[4] João Valle et al. "Using Traces from IoT Devices to Solve Criminal Cases". In: June 2020. DOI: 10.1109/WF-IoT48130.2020.9221265.

[5] Casey C. Sullivan. *How the IoT Is Solving Murders and Reshaping Discovery*. [Accessed: 12-05-2022]. URL: https://www.logikcull.com/blog/how-the-iot-is-solving-murders-and-reshaping-discovery.

[6] Hany F. Atlam et al. "Internet of Things Forensics: A Review". In: *Internet of Things* 11 (2020), p. 100220. ISSN: 2542-6605. DOI: https://doi.org/10.1016/j.iot.2020.100220. URL: https://www.sciencedirect.com/science/article/pii/S2542660520300536.

[7] Sita Rani et al. *Threats and Corrective Measures for IoT Security with Observance of Cybercrime: A Survey*. 2021. arXiv: 2010.08793 [cs.NI].

[8] Fadele Ayotunde Alaba et al. “Internet of Things security: A survey”. In: *Journal of Network and Computer Applications* 88 (2017), pp. 10–28. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2017.04.002. URL: https://www.sciencedirect.com/science/article/pii/S1084804517301455.

[9] Mohammad Hassan, Ghassan Samara, and Mohammad Fadda. “IoT Forensic Frameworks (DFIF, IoTDOTS, FSAIoT): A Comprehensive Study”. In: *International Journal of Advances in Soft Computing and its Applications* 14.1 (Mar. 2022), pp. 73–83. DOI: 10.15849/ijasca.220328.06. URL: https://doi.org/10.15849%5C%2Fijasca.220328.06.

[10] Abiodun Esther Omolara et al. “The internet of things security: A survey encompassing unexplored areas and new insights”. In: *Computers Security* 112 (2022), p. 102494. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2021.102494. URL: https://www.sciencedirect.com/science/article/pii/S0167404821003187.

[11] Ibrar Yaqoob et al. “Internet of things forensics: Recent advances, taxonomy, requirements, and open challenges”. In: *Future Generation Computer Systems* 92 (2019), pp. 265–275. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2018.09.058. URL: https://www.sciencedirect.com/science/article/pii/S0167739X18315644.

[12] Francesco Servida and Eoghan Casey. “IoT forensic challenges and opportunities for digital traces”. In: *Digital Investigation* 28 (2019), S22–S29. ISSN: 1742-2876. DOI: https://doi.org/10.1016/j.diin.2019.01.012. URL: https://www.sciencedirect.com/science/article/pii/S1742287619300222.

[13] Liam Dawson and Alex Akinbi. “Challenges and opportunities for wearable IoT forensics: TomTom Spark 3 as a case study”. In: *Forensic Science International: Reports* 3 (2021), p. 100198. ISSN: 2665-9107. DOI: https://doi.org/10.1016/j.fsir.2021.100198. URL: https://www.sciencedirect.com/science/article/pii/S2665910721000293.

[14] Seila Becirovic and Sasa Mrdovic. “Manual IoT Forensics of a Samsung Gear S3 Frontier Smartwatch”. In: *2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. 2019, pp. 1–5. DOI: 10.23919/SOFTCOM.2019.8903845.

[15] Asif Iqbal et al. “Smart Home Forensics: An Exploratory Study on Smart Plug Forensic Analysis”. In: *2020 IEEE International Conference on Big Data (Big Data)*. 2020, pp. 2283–2290. DOI: 10.1109/BigData50022.2020.9378183.

[16] Hyunji Chung, Jungheum Park, and Sangjin Lee. “Digital forensic approaches for Amazon Alexa ecosystem”. In: *Digital Investigation* 22 (2017), S15–S25. ISSN: 1742-2876. DOI: https://doi.org/10.1016/j.diin.2017.06.010. URL: https://www.sciencedirect.com/science/article/pii/S1742287617301974.

[17] Soram Kim et al. “Smart Home Forensics—Data Analysis of IoT Devices”. In: *Electronics* 9 (July 2020), p. 1215. DOI: 10.3390/electronics9081215.

[18] Arnoud Goudbeek, Kim-Kwang Raymond Choo, and Nhien-An Le-Khac. “A Forensic Investigation Framework for Smart Home Environment”. In: *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. 2018, pp. 1446–1451. DOI: 10.1109/TrustCom/BigDataSE.2018.00201.

[19] Victor R. Kebande and Indrakshi Ray. “A Generic Digital Forensic Investigation Framework for Internet of Things (IoT)”. In: *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*. 2016, pp. 356–362. DOI: 10.1109/FiCloud.2016.57.

[20] Victor R. Kebande et al. “Towards an Integrated Digital Forensic Investigation Framework for an IoT-Based Ecosystem”. In: *2018 IEEE International Conference on Smart Internet of Things (SmartIoT)*. 2018, pp. 93–98. DOI: 10.1109/SmartIoT.2018.00-19.

[21] senerex. *[All-in-One] Redmi Note 7 (lavender): Unlock Bootloader, Flash TWRP, Root, Flash ROM*. [Accessed: 12-05-2022]. URL: https://forum.xda-

developers.com/redmi-note-7/how-to/one-redmi-note-7-unlock-bootloader-t3890751.

[22] Joy Culbertson. *12 Popular Home Automation APIs*. [Accessed: 12-05-2022]. URL: https://www.programmableweb.com/news/12-popular-home-automation-apis/brief/2020/09/06.

[23] Fibar Group. *Google Assistant – FIBARO Integration*. [Accessed: 12-05-2022]. URL: https://manuals.fibaro.com/knowledge-base-browse/google-assistant-fibaro-integrations/.

[24] Karen Kent et al. "Guide to Integrating Forensic Techniques into Incident Response". In: *NIST Special Publication* (Jan. 2006).

[25] Collective Work. "The Digital Forensic Research Conference DFRWS USA". In: 2001.

[26] Md. Jahidul Islam et al. "Digital Forensic Investigation Framework for Internet of Things (IoT): A Comprehensive Approach". In: *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*. 2019, pp. 1–6. DOI: 10.1109/ICASERT.2019.8934707.

[27] Asier Martínez. *Tools for carrying out forensic analyses on mobile devices*. [Accessed: 12-05-2022]. URL: https://www.incibe-cert.es/en/blog/mobile-forensic-analyses-tools.

[28] iam2mai. *Agile vs Waterfall methodology for software development life cycle diagram*. [Accessed: 12-05-2022]. URL: https://www.shutterstock.com/image-vector/agile-vs-waterfall-methodology-software-development-1861899391.