TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Simon Victor Jean Laurent Brun 184683IVCM

COMPREHENSIVE DIGITAL FORENSICS ANALYSIS OF SMART HOME ENVIRONMENT.

Master's thesis

Supervisor: Hayretdin Bahsi
PhD
Pavel Tšikul
PhD student

Tallinn 2020

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: Simon Victor Jean Laurent Brun

[22.04.2020]

Tallinn 2020

Abstract

This thesis is written in English and is 79 pages long, including 7 chapters, 5 figures and 14 tables.

IoT environment is an ecosystem that becomes more and more common as time passes. Forensic investigations are more likely to happen in this type of ecosystem than ever. Yet it is still a rather unknown discipline, because of the versatility and heterogeneous nature of IoT. They are, however, a great source of forensic valuable data for digital forensic analysis if they are studied properly. We are aiming to provide a better understanding of such environment, by putting in place an experimental setup and scenario, and by analyzing each component of our environment. The system consists of a smartphone, two IP cameras, a Google Home, and several kinds of sensors, such as motion sensors, flood sensors, and door sensors. We explain in our results what artifacts it is possible to retrieve and how. We also give metrics of importance for valuable data regarding a possible criminal case, as well as providing prioritization guidelines for the different investigation steps.

Acknowledgements

I would like to thank my supervisor and co-supervisor, respectively, Hayretdin Bahsi and Pavel Tšikul, for helping me during my research, more specifically with the acquisition of the devices, the theoretical knowledge as well as the practical know-how needed to write this thesis. I would also like to thank Alejandro Guerra Manzanares for his technical experience and insights he could exchange with me. I would also like to thank all Taltech staff who helped me and allowed me to use a room in Taltech in order to successfully conduct my experiments.

List of abbreviations and terms

TUT	Tallinn University of Technology
IoT	Internet of Things
OS	Operating System
SSID	Service Set IDentifier
API	Application Programming Interface
ADB	Android Debug Bridge
CSRF	Cross Site Request Forgery
TLS	Transport Layer Security
SSH	Secure SHell
APK	Android application PacKage
NBT	Next Best Thing
OOFI	Object Of Forensic Interest

Table of Contents

List of Figures

List of Tables

1 Introduction

Nowadays, more and more devices are connected to the Internet, especially with the growing number of IoT devices. According to *IoT-analytics*[1], 7 billions of IoT devices were connected to the Internet in 2018. An unavoidable consequence is that more IoT devices are going to be present in everyone's home, which means more smart hubs, smart sensors, and smart devices in general. These devices collect, process, and send out a lot of user data such as Internet searches, usernames, the temperature of one's home/room, the presence of people, etc. The fact that these devices become more common in home environments means that they become more present in crime scenes as well. This implies that the forensic investigation of these devices would possibly reveal capital information for forensics experts and law enforcement agencies regarding criminal cases. We can point out two criminal cases where the Amazon Alexa smart speaker was present on the crime scene and was helpful in solving the case. For instance, in 2015, when "James Bates was charged with first-degree murder in the […] death of Victor Collins" [1], found unanimated in Bates' bathtub. After Collins' death, the prosecution requested Amazon to provide the data that could have possibly been recorded by the Amazon Echo present on the crime scene. Another example of a criminal case involving a smart speaker, is, on the night of the 12th of July 2019, in Florida, when Adam Reechard Crespo and Silvia Galva, had an argument in their bedroom, ending up with a spear's blade through Galva's chest [2]. Charges were pressed against Crespo for murder. An Amazon Alexa was present in the bedroom at the time of the death, and the police have obtained a warrant in order for Amazon to hand over the possible recordings; which they did, giving the police more evidence about what happened this night.

Research objective:

The goal of our work, is to show, that it is possible to recover forensically valuable data from a smart home environment, at every stage of an investigation. We are aiming to

[1] https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/

provide retrieved artifacts and their specificities, stating at which point of the investigation they were retrieved, from where, and what is their forensic value regarding a possible murder case. We also aim at proposing digital forensics procedures that could be followed when such environments are being encountered and requested to be investigated. Our work is based around an experimental setup, emulating a smart home environment. The main components of this setup are a Google Home smart speaker, a smartphone running on Android, and smart sensors, including motion detectors (among others) and security cameras for instance. We mainly focused on the mobile forensics point of view, as well as the network and IoT point of view, for our experimental investigation. First, we present the architecture and the specificities of the experimental setup that was put in place in order to do this investigation. This covers the devices that were used and their details, as well as the tools (and version) that were used. We describe also how the environment is connected together and which protocols are used. The room layout where the experiment was taking place is described. In order to make our data collection and analysis relevant, we imagined and incorporated a murder case/scenario. The events that happened are also described in the first part of this paper.

The second part focuses on the actual procedure followed in order to retrieve as much information as possible from this environment, using as much forensically sound techniques as possible. We describe each step that has to be taken for each forensic technique used, which forensic artifacts/data can be recovered and how; organizing them from the least intrusive method to the most intrusive. The digital forensics disciplines present in this paper are: Mobile forensics, by using Logical acquisition, Logical acquisition with root access, and Physical acquisition; Network forensics, and IoT forensics, by using APIs and the device's application.

The last part of this thesis consists of a presentation of our findings, their forensic value, how and at which stage of the investigation they were recovered; as well as a discussion part including advice about the prioritization of the investigation steps, digital forensics procedures, and possible improvements.

Scope:

In the scope of our experiment, we include, as explained previously, different forensic efforts. We focus on the Mobile forensics, the Network forensics, and the IoT forensics.

We are exploring these disciplines of forensics collection and analysis, but, as they can be broad, we limit our study to software-based, non-intrusive collections. Any collection or analysis that might require to physically disassemble any device in order to access the chip falls out of our scope. The Cloud forensic discipline is also out of our scope.

Contribution:

In our work, we gave a more realistic take on a forensic analysis done in a lab. We imagined and implemented a fake murder scenario in our environment. It allowed us to sort the retrieved artifacts by metrics of importance regarding our case, which, in the end, impacted the digital forensics processes; such as the order in which the data collection and analysis should be made, as well as the actions and decisions that should be taken on the crime scene. Our work also contributes to the study of the Google Home smart speaker and environment, by providing a more systematic study. We give details of the steps to follow and the tools to use, in order to retrieve as much information as possible from this type of device. We also provide an analysis of the Fibaro environment, by studying a Fibaro controller and several Fibaro sensors, this type of environment has not been analyzed yet at the time of our writing.

2 Related works

As stated by S. Sathwara, N. Dutta, and E. Pricop [3], one of the biggest challenges in IoT forensics is to preserve data in a heterogeneous environment, regarding software and hardware. That is why in our research we had to reproduce a heterogeneous environment in order to be as close as possible from the reality of the IoT world. The forensic challenges in the IoT environment also come from the fact that, as pointed by A. Goudbeek, K.-K. R. Choo, and N.-A. Le-Khac [4], the lack of standardization is a problem for forensics experts. Indeed the lack of standardization and methodologies [5] forces the experts and the researchers to provide tools, procedures, and more use cases in order to have a better understanding of IoT environments. While conducting our research, we discovered many proposed frameworks[1] made by different authors. Most of these researches were trying to propose a general framework that would apply to all of the IoT environments. For instance the FSAIoT framework presented in [6] is used in order to collect logs from various IoT devices and thus present the changes in the state of the devices. It would be then used to retrace what happened from the sensors' point of view during a crime for instance. However, the authors indicate that it is not robust enough and generalizable enough (at the time of their writing), and that they didn’t implement any forensic soundness checker. The framework described in [3], presents the major steps that should be achieved in a digital forensics investigation and develop in which way the IoT environment impacts these steps. Identification becomes more difficult due to the “lack of detailed log and fingerprints” [3]. Preservation is more challenging in a IoT environment because of the “Lack of tools to preserve information from sensing devices” [3] and the Analysis suffers from the “limited information stored in logs and caches identification” [3]. It describes also which type of data has to be expected depending on where it has been retrieved in the IoT ecosystem. It is however more presented as a theoretical framework and does not include any real-life or experimental

[1] A framework can be seen as a set of tools and procedures that are applied to a forensic investigation in order to maximize efficiency.

scenario. Another framework was proposed in [4] which is well detailed, and in which the authors develop a seven phase framework for analyzing Home Automation Systems. It includes preparation off-site and on-site, preservation of the environment, understanding of the system, security level check, location and extraction of evidence and analyze of data. However, only case studies were dealt with but no real-life example, which is a research gap yet to fill. The IoTDots framework [7] includes the ability to extract data from a wide range of hardware and software and from a heterogeneous environment, such as found in IoT ecosystems; however, they do not include every programming language and APIs for example. The authors have developed a framework as well as a tool capable of "automatically analyzing and modifying smart apps to detect and store forensically-relevant data from smart devices, apps, and users." [7] thanks to machine learning techniques.

Some other frameworks revealed themselves interesting regarding our work, and more specifically the theoretical ones. We can for instance note the "Digital Forensic Procedure" [33] described by M. Harbawi and A. Varol, which is based on efficiency. The authors describe a procedure that allows the investigators to gain time by using the "Last-on-Scene (LoS) algorithm" [33], which "assumes that the last thing to be found in communication should be investigated first" [33]. They also propose a time-based forensic analysis, where data should be extracted only around the time when the crime started; and a zone-based investigation workflow, where the investigation of the IoT devices should be performed first within the "Personal Area Network (PAN)" or "Zone 0", and then should be conducted, only if necessary in the Zone 1 and 2, referring respectively to the "Intermediate Area Network (IAN)" and the "External Area Network (EAN)" [33]. Another interesting framework is proposed by the authors of [34], where they discuss the relevance of the evidence in regards to a scenario. They explain that it would be important for IoT related crime cases to "rank evidence by degree of relevance or importance" [34], which we think is really true considering the amount of data IoT devices can generate. The authors also identify different zones, which are similar to the zones denoted in the paper discussed previously [33]. A very important aspect of IoT forensics is "the question of knowing *where to look.*" [34]. To answer this question, the authors propose the "Next Best Thing (NBT) Triage Model" [34], which is useful for forensic investigation when the IoT device that could contain the evidence is not available for the investigators. In that case, they would use the “NBT Model” where devices that are “either directly connected or somehow related”[34] to the device of forensic interest could potentially contain evidence.

As mentioned earlier, our investigation is focused on a Google Home environment. Google Home forensics has been the subject of several studies already, however, the procedure and the findings were not conducted in a very systematic way. S. Tristan, S. Sharma, and R. Gonzalez studied the Google Home and the Alexa speaker [8] but as said before, the study is not precise as they did not include a detailed procedure to follow. They instead give some general statements and little information about tools and recovered artifacts, which does not really help investigators to find interesting data. More practical work has been achieved by S. Engelhardt [9] as the author included detailed steps that he followed for analysis and practical findings, but we also think that it does not provide a thorough enough analysis of such environment/device as the author just gives results of the analysis without giving details on the methods. Forensics studies have also been performed on other brands of smart speakers. We can note for instance the Almond+ environment, which was analyzed by the authors of [22]. The analysis they present there is quite thorough as it includes different investigation steps for each way of interacting with the Smart home device. They gathered evidence by using the Almond + “via the touchscreen”[22], “via the companion app”[22], or thanks to “Local network and Cloud assessment”[22] for instance. Although Google Home seems to be less verbose than the Almond+ environment, we think it is important to analyze it as it a major actor in the smart home and smart speaker market, as opposed to the Almond ecosystem. Many studies have also been done on the Amazon Echo, embedding the smart assistant going by the name of Alexa. The authors of [23] propose a framework for IoT environments that tries to be as close as possible to a traditional approach of digital forensics. Including the steps needed for “identification, acquisition, analysis, and presentation of potential artifacts of forensic interest”[23]. They are then confirming their framework thanks to a use case on the Amazon Echo. The work presented earlier in [8] and [9] also both include a part dedicated to the analysis of one Amazon Echo device.

Other IoT devices have been forensically analyzed by several authors, such as wearable devices. Smartwatches have been forensically analyzed. J. Gregorio, B. Alarcos, and A.

Gardel [24] have analyzed the CAWONO DZ09, and other low-cost smartwatches, showing that it is possible to recover forensically valuable artifacts directly from these watches, such as “contact data, call records, instant messages, multimedia files[…] without requiring access to the connected smartphone”[24]. It has also been shown by J. Rongen and Z. Geradts that it is possible to recover data from the Google Glass such as “Pictures, Videos, Contacts, […] Locations and destinations [...] Connected devices”[25] and so on thanks to different forensic techniques. We can note the use of software-based methods like Logical and Physical acquisition thanks to ADB, as well as hardware-based methods like JTAG or Chip-Off. In 2020, authors of [27] showed that in the case of an iPhone and an Apple Watch (which are widely adopted by people globally), information is synced on both paired devices. They showed that the “Apple Watch content is constantly being backed up on the paired iPhone”[27]. Which can be a source of valuable data for investigators. They also point out that on the Apple Watch, some apps are “stand-alone apps”[27], while some others are “iPhone-based apps [that] require the companion device to be nearby since the app on the Watch only contains the user interface”[27]. It is thus important in this type of environment to take into account every device that is synced or linked together and to analyze them separately.

Our work is focused on the Mobile forensics point of view, indeed the mobile phone is the device that holds the most user data that could be used as evidence in a court. We have to keep in mind, as stated by Lee Reiber [10], that Mobile forensics is a really specific exercise. Indeed mobile phones are most of the time turned on and thus communicates with a lot of different media; which greatly differentiates them from computers and thus Computer forensics, where write blockers can be used. Communication occurs when the phone is connected to the Wi-Fi or when it is connected to a computer using a USB cable. This forces us to be as careful as possible not to alter the data collected, because the mathematical fingerprint (or hash) of the phone will never match the one expected, as the system is constantly changing [10, pp. 25-43]. We also have to take into account, as explained by the author, the fact that mobile devices evolve quickly, as well as the different version of the Android Operating System (which we study here); which means that artifacts that could have been found on Android phone in the past, may not be present on Android phones nowadays. It is however important to perform other types of forensic methods and not just Mobile

Forensics. Thanks to the work of R. Rizal, I. Riadi, and Y. Prayudi [21], we could identify more forensic fields that are relevant to explore for our case. The authors pointed out three forensic levels of forensics inherent to the IoT forensics: the “Device level forensics”[21], where the “forensic investigator needs to collect data first from the local memory contained in the IoT device” [21]. The “Network level forensics”[21] which is done through the analysis of the “network traffic logs” [21]. And lastly the “Cloud level forensics”[21] which they consider as being “one of the most important part in the IoT forensic domain […] Due to the fact that […] data generated from IoT devices and networks are stored and processed in the cloud”[21]. We thus focus on the Network forensics and the IoT forensics in this paper as well. The *Device level forensics* is not part of our scope as it needs a physical access to the device chip.

3 Methods, materials and experimental setup

In this part of the thesis, we describe the experimental environment that is used during our forensic experiment. We first describe the devices and the tools used. We then focus on the actual physical layout of the devices in the room and on the murder scenario that we imagined in order to create events. Lastly, we cover what would the event look like from the sensors/devices' point of view.

3.1 Devices and Tools

In this subchapter, we provide a description of the Operating systems, of the devices that we used during this experiment as well as the tools used in order to extract and analyze the data. It includes the brand of the devices, their model number, and some of their specifications as well as their versions. The purpose of the tools is briefly described along with their version.

3.1.1 List of Operating Systems used:

Operating System	**Version**	**Hosted on**
Windows 10 Education	1903 – 18362.657 (x64)	Virtualbox 5.2.34_Ubuntu
Linux Kali	5.4.0-kali3-amd64	Virtualbox 5.2.34_Ubuntu

Table 1: List of Operating Systems

For our experiment, our host machine was running on Linux Mint 19.1. On this machine, we hosted the two main operating systems that we were using during the experiment: Windows 10 and Linux Kali. We chose to use these OS because all of the digital forensics software are compatible with at least one of them. Also Linux Kali has a variety of forensic/security tools already installed in the basic package, which makes easier the acquirement of the software needed.

3.1.2 List of devices and tools:

Editor	**Tool name**	**Version**	**OS used**	**Usage**
The Sleuth Kit (4.8.0)	Autopsy	4.14.0	Windows	Used for analysis of physical extraction
Pawel Psztyc	Advanced REST Client	15.0.0	Windows	Used to communicate with the Google Home API
Erik Hjelmvik	NetworkMiner	2.5.0.0	Windows	Used to analyse network packets
Google/Android	Android Debug Bridge (ADB)	1.0.39	Linux	Used to interact with the Android phone
Nikolay Elenkov	Android Backup Extractor	V20180521	Linux	Used to convert Android backup to a tar archive
Gerald Combs	Wireshark	3.2.1	Linux	Used to analyse network packets
Mauricio Piacentini/ Pete Morgan	SQL Lite Browser	3.11.2	Linux	Used to analyse databases retrieved from the Android phone

Table 2: List of tools

3.1.3 List of the devices:

Brand	**Model**	**Version**	**Specifications**	**IP address**
Google	Google Home Mini	System Firmware: 191169 Cast Firmware: 1.44.191160	X	192.168.8.103
Fibaro	Home Center Lite	4.570	Z-Wave controller version 4.33	192.168.8.104
Huawei	B535-232 4G Router	10.0.1.1	2.4 GHz and 5 GHz	192.168.8.1
Fibaro	Flood Sensor	3.3	X	X
D-Link	932LB	2.18	IP Camera	192.168.8.111
Fibaro	Door Sensor	3.2	X	X
Fibaro	Motion Sensor	3.3	X	X
Xiaomi	Mi Home Security Camera 360° 1080P	3.4.6_0213	IP Camera	192.168.8.117
Xiaomi	Redmi Note 7	MIUI version: Global 11.0.4 Android version: 9 PKQ1.180904.001	Internal Storage: 32 GB RAM: 3 GB CPU: Octa-core Max 2.20 GHz	192.168.8.101

Table 3: List of devices

The detailed configuration of all the devices is not necessarily relevant, however, we believe that it is important to explain briefly the configuration of the two IP cameras.

First, the Xiaomi Mi Home Security, as stated above, is an IP camera, it has to be connected to the power supply constantly and communicates using Wi-Fi with the Smartphone of the user through an application: *Xiaomi Home*.[1] We configured the camera to monitor the scene 24/7, with high sensitivity, meaning that the camera detects every movement, even the smaller ones. The alarm interval, meaning the minimum time between two movement detection triggers is set to 3 min (the lowest possible for the camera). The Mi camera allows us to take snapshots and videos manually through the app, which are saved on the internal memory of the phone directly. There is a possibility to install an SD card on the camera in order to record and save the video stream continuously (it is for instance possible to save around 10 days of video stream in 720p and 6 days in 1080p for a 64GB SD card). We did not have any SD card installed during our experiment, however, whenever a movement is detected, the MI camera saves 8 seconds of video in the buffer memory of the device[2], that can later be saved manually by the user into the internal storage of the phone. Otherwise, it will disappear after 8 days. It is possible to control the camera through the app, from the same network, or from a remote connection.

The D-Link Camera is also an IP camera, it also has to be connected permanently to the power supply and uses Wi-Fi to communicates with the application on the mobile phone (*mydlink Lite*[3]) or with the web interface. Unlike the Xiaomi camera that can only be monitored and controlled via the Android app, the D-Link camera can also be configured through a web browser (in a much more extensive manner): it is compatible with *Internet Explorer* exclusively. The D-Link camera is configured to detect motion. The motion detection is set to detect movement 24/7, with a sensitivity of 50%. Whenever a motion is detected by the camera, it sends 6 frames of image to the email address of the user (3 frames before detection and 3 frames after). It is possible to configure and view the camera stream from the same network or from a remote connection.

See Appendix 1 for images of the devices used in our experimental setup.

[1] https://play.google.com/store/apps/details?id=com.xiaomi.smarthome&hl=fr
[2] This information has been found thanks to experiments
[3] https://play.google.com/store/apps/details?id=com.dlink.mydlink&hl=fr

3.2 Room experimental setup

In this section, we explain the experimental setup that was put in place. We had at our disposal the devices that were presented in the previous section and we wanted to create a setup that could be credible with all these devices present in a room. We present the layout of the room, and the position of the different devices. We also describe the scenario that we came up with, in order to generate the data and to make the situation more credible.

3.2.1 Room layout

The layout of the room used for our experiment, is displayed as a figure below.

The room we had at our disposal did not actually contain a bed. A desk was normally present at this location, we just imagined the presence of a bed for the sake of the scenario.

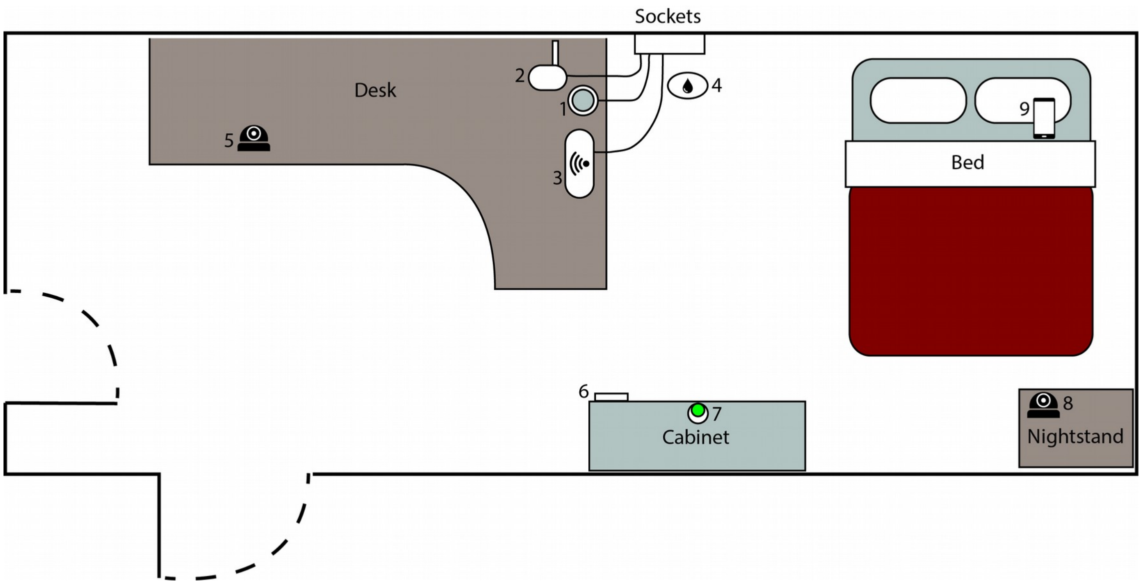


Figure 1: Room layout

Here is the list of the devices and their related number on the figure above:

Number associated from Figure 1	**Device name**
1	Google Home Mini
2	Fibaro Home Center Lite
3	Huawei B535 4G router
4	Fibaro Flood sensor
5	D-Link 932L Camera
6	Fibaro Door sensor
7	Fibaro Motion Sensor
8	Xiaomi Mi Security Camera 360°
9	Xiaomi Redmi Note 7 Android Phone

Table 4: Devices associated to their number in Figure 1

The devices were placed in order to fit some possible needs that the user of the room could have, here is an explanation about this environment:

The user has a Google Home mini, such as many people (according to Statista[1], 24.9% of the global smart speaker share was owned by Google, in the last quarter of 2019) that is used to bring more comfort into his daily life, such as playing music, creating events in his calendar, doing Google searches and creating a grocery list for example; using a simple voice command, activated by the keywords “Hey Google” or “Ok Google”. It is placed on his desk so the voice activation can work from anywhere in the room.

The Fibaro Home Center Lite is a Z-wave controller, that allows the user to manage his Z-wave devices (in our case, all of the Fibaro sensors) and to track their activity. As mentioned, the controller communicates with the sensors with the Z-wave protocol and with the user’s phone or computer using Wi-Fi.

The Fibaro Flood sensor is placed on the floor, close to the main plug socket, in order that if a flood occurs, the alarm would go off and thus alert him of a flood happening close to the electricity supply.

The D-Link 982L Camera is placed on the desk, facing one door, in order to know who is coming through the front door, or from the door on the left hand side of the room.

The Fibaro Door sensor is placed on the cabinet door in order to know when the door was opened of closed. The user is keeping important documents in this cabinet, which is locked by a key.

The Fibaro Motion sensor is placed on top of the cabinet and detects every movement in his range, including movement of people opening the cabinet.

The Xiaomi MI Security Camera 360° is placed on the user’s night stand, it is used to monitor everybody who approaches his bed, it can be also used to monitor his sleep.

[1] https://www.statista.com/statistics/792604/worldwide-smart-speaker-market-share/

3.2.2 Communications in the experimental environment:

It is also very important for our work to understand how our environment and all the different devices and sensors communicate with each other, and which technology and protocols they use. We present it thanks to the following figure:

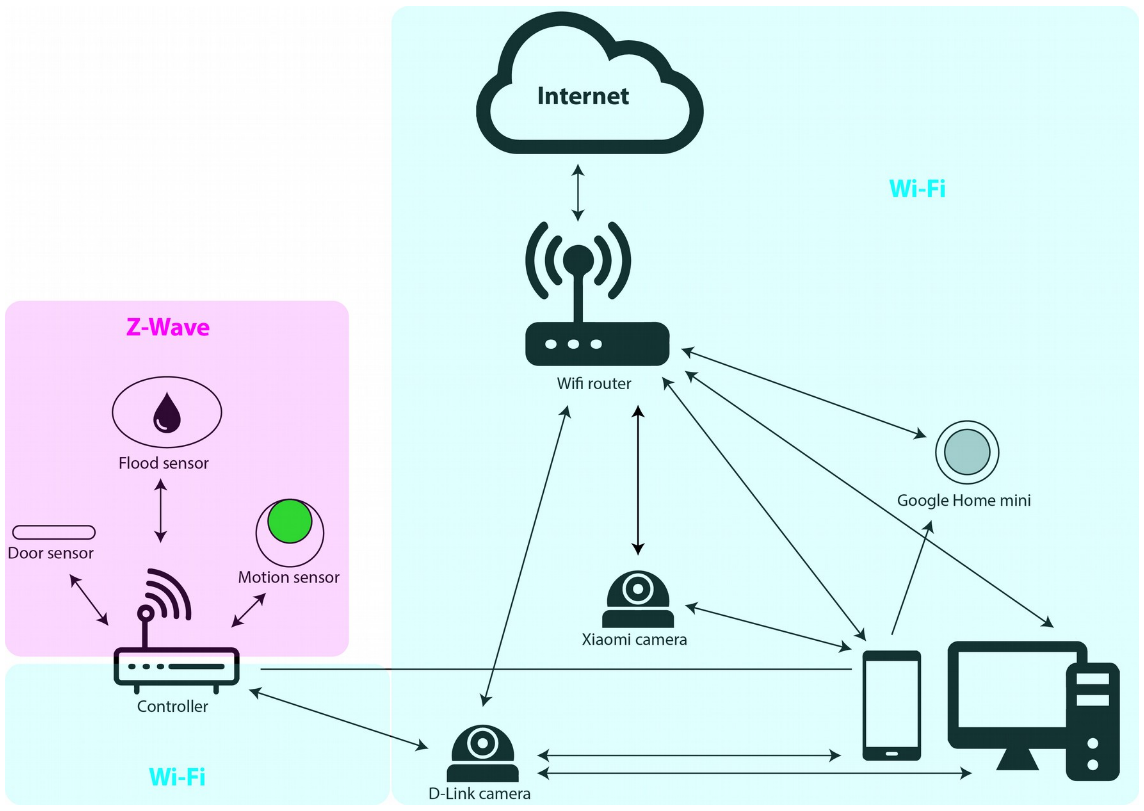


Figure 2: Communication within the environment

As we can see from the figure above, two main protocols were used in this environment, Wi-Fi and Z-Wave, the Z-Wave protocol was only used between the Fibaro controller and the Fibaro sensors. We can notice that nearly all the devices are communicating with each other.

3.2.3 Experimental scenario

Here we explain the experimental scenario that we used in order to produce the data related to our fictitious case. This scenario is the base for our data extraction and analysis. It is used to produce events and data that could give better credibility to our case. This scenario could happen in real life, even though it was made and tailored for our needs and our time period, thus rending it somehow maybe too utopian; as it would appear to be a perfect murder scenario from a forensic point of view (understand: easy to retrace what happened).

The base scenario is as follows:

The victim is taking a nap in his bed, with all of his devices turned on, the room is completely monitored. The attacker enters the room with a knife in his hand from the left door and goes directly in front of the bed where our victim is sleeping. The victim wakes up and stands up, trying to fight the attacker, who stabs him, the victim falls on the floor in a blood puddle. The murderer then takes the key from the man and goes in front of the cabinet to open it. He opens it, takes some important files (in our case, a book), and close the cabinet. He leaves the room from the same door he came from.

As we can see by reading this scenario and by keeping in mind that all the devices were turned on and monitoring the scene, we can assume that the sensors recorded some events:

- When the attacker goes from the door to the bed, he is detected once by the D-Link camera and by the Motion sensor.
- When he murders the victim, he is detected by the Xiaomi Camera.
- When he opens the cabinet, he is detected by the Motion sensor, the opening of the door is also being detected by the Door sensor.
- The Door sensor detects when the door is closed again.
- The attacker is detected while leaving the room, by the D-Link camera again, this time with the book in his hand.
- After the attacker left, the blood puddle spread on the floor and is detected by the Flood sensor.

3.2.4 Events from the sensors point of view:

All the events from the scenario described above, can be also be seen from the sensors' point of view, such as presented in Figure 3, with the corresponding timestamps. The events have been created by the author himself, by simulating the murder scenario. The timestamps are thus correct, they have been recovered by the author. Besides the events and data generated by the author for this specific case, the environment was tested and data have also been generated unaccommodating this scenario (as a normal user using his phone). The environment has been used for a period of approximately two weeks, where the author was doing Google searches, downloading files, using the Google Home, and the sensors. Some of the data that have been generated during this time period are also presented as they could also be used as evidence in different scenarios.

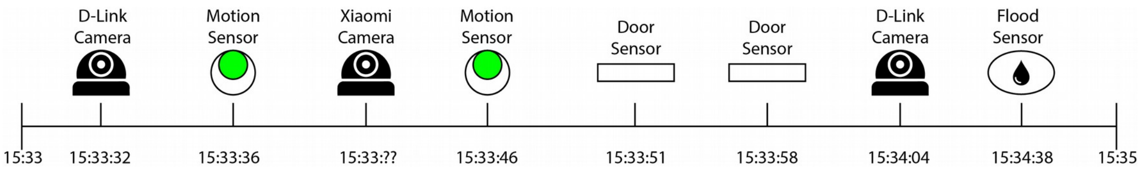


Figure 3: Events seen from the sensors point of view

In the table below, we summarize the events from the sensor point of view with the timestamp and the state that the concerned device is in:

Device	**Time**	**State of sensor**
D-Link Camera	15:33:32	Motion detected, mail sent
Motion Sensor	03:33:36 PM	Motion detected
Motion Sensor	15:33:44	State back to safe[1]
Xiaomi Mi Camera	15:33:??[2]	Motion detected, video buffered
Motion Sensor	15:33:46	Motion detected
Door Sensor	15:33:51	Door open
Door Sensor	15:33:58	Door closed
D-Link Camera	15:34:04	Motion detected, mail sent
Motion Sensor	15:34:09	State back to safe
Flood Sensor	15:34:38	Flood detected

Table 5: Events from sensors point of view

[1] State back to safe: means that the motion detection does not detect any movement at that time.

[2] The two interrogation marks are here because the Xiaomi Camera does not record the precise seconds when the detection happened.

4 Results

In this part, we present the procedure and the steps that were taken in order to retrieve information about the murder scenario that was described in the previous part. The data collection is not limited to this murder scenario, as more data can be retrieved, they are also presented in this part and put in perspective whether they are relevant for this particular case or not. The investigation work is divided in different parts concerning different forensic techniques and disciplines. First the Mobile forensics investigation is tackled, then the Network forensics and finally the IoT forensics. With the purpose of ranking the data in order of relevance regarding our case, we came up with some metrics:

- High: Data that are directly concerning our case and that can incriminate or could incriminate the suspect, or data that are not directly concerning the case but which are forensically important (for checking the integrity of files for example).
- Medium: Data or configuration data that could help to identify the user's behavior, habits, location, or environment.
- Low: Configuration data that are not linked to the user and that only gives information about the environment.

We are referring to these metrics throughout the whole document while presenting the results.

4.1 Mobile Forensics

In this section, we present which evidence was recovered from the mobile phone, where these artifacts were found, and how they were found. Mobile phones are extremely important in crimes. Smartphones nowadays carry a lot of user data and information and are capable of processing them, such as computers. A smartphone investigation becomes important when the device found on the scene is the phone of the criminal, or the victim’s one, or even, in a cyberattack, as a vector to perform another attack at a larger scale. Extracting evidence from a mobile phone can highly help professionals to conduct a suspect or to retrace the events that could have happened. Valuable artifacts found on mobile devices include “software applications, data, and information such as documents, e-mail messages, Internet browsing history, Internet chat logs[…] photographs, images files, databases” [29]. As explained by L. Reiber [10, pp. 157-168], there are several ways of collecting phone information. The important data acquisition phases include Logical acquisition and Physical acquisition. Considering our case, we should proceed first with the Logical acquisition and then with the Physical acquisition, as described in Reiber's book as well [10, p. 237]. During our experiment, the phone has not been locked by any passcode or password. We assume that the phone has been found turned on on the crime scene. Thanks to the *Mobile device evidence processing workflow* found in [10, p.237], that can be found in Appendix 2, we see that we have to collect the device first logically and then physically. It is also important to note that the phone we used during our experiment was not rooted and had his bootloader locked. As explained by Android developers [11], a “bootloader is a vendor-proprietary image responsible for bringing up the kernel on a device. It guards the device state and is responsible for initializing the Trusted Execution Environment (TEE) and binding its root of trust”. The bootloader is essential for the smartphone to turn on and boot on the operating system without any problem. However, depending on the manufacturer and the year of the release of the phone, the bootloader can either come locked or unlocked. Manufacturers intentionally lock the bootloader of their products in order to prevent users to install custom ROMs on it, as explained by C.Hoffman [12]. They also enhance the security of the device. That being said, the locked bootloader is a problem for forensics investigators, because, in order to root the phone, a procedure necessary to extract all the information from a suspect’s phone, the bootloader should be unlocked. We discuss these issues in the following part as part of the rooting process of the phone.

4.1.1 Logical acquisition

In this part, we explain which artifacts were recovered thanks to a logical acquisition of the phone, first while the phone was unrooted (and the bootloader locked) and then when the phone has been rooted and the bootloader unlocked. The term *Logical acquisition* has been defined already several times by different authors. For instance, as pointed by Lee Reiber [10] in his book, different definitions have been made in the past by different organizations. We can note for example, that in, 2007, according to NIST, “a logical acquisition implies a bit-by-bit copy of logical storage objects that reside on a logical store.”[26]. The author also gave his definition of a logical acquisition in his book: “A logical collection on its face should be interpreted as the extraction of user data from the mobile device without the collection of a device’s file system” [10, p.158]. As seen in [13, p. 196], “A logical technique extracts allocated data and is typically achieved by accessing the file system”[13]. H. Srivastava and S. Tapaswi [14] argue that a “Logical acquisition, on the other hand, acquires data by accessing the filesystem”.

Logical acquisition on a non-rooted phone

As explained previously, the smartphone that we had at our disposal during our experiment was not rooted[1]; which means that we could only recover data that were accessible at a user’s level privilege, and thus, not all of the user’s data. However, we could still recover a decent amount of data using an Android Debug Bridge[2](ADB) logical acquisition. In order to have a functioning connection with ADB, we first needed to enable the *USB Debugging*, this option is available in the Developer options of the phone.

We could then connect the phone to our computer with ADB installed. The phone should be recognized by the ADB tool. We could then do an ADB backup extraction, by typing in the terminal emulator *#adb backup -shared –apk -all -system -f backup.ab.*

It was necessary to unlock the phone after typing the command, in order to confirm the backup of the data.

1 “Rooting enables users to perform higher privileged functions on a device than are ordinarily possible under regular user mode.[…] Root access can also be used legitimately for forensic investigators to extract data from a device” [15].

2 https://developer.android.com/studio/command-line/adb

As a result of this ADB backup, a file called backup.ab was created on our computer, this file has to be converted into an extractable format, such as tar. To accomplish such a task, we used Android Backup Extractor[1]. With the following command: *#java jar abe.jar unpack /<path>/backup.ab <path>backup.tar*

This command enabled us to convert the file to an extractable tarball, which when extracted, enables us to access the following data:

[1] https://github.com/nelenkov/android-backup-extractor

Index	Path of file / folder	Type of file	Data retrieved	Relevant regarding case	Which sensor is concerned?
1	shared/0/DCIM/Xiaomi/local/284034774/	Pictures and videos (.jpg and .mp4)	Any pictures or videos that could have been saved from the Mi Camera App	High (if the videos are saved)	Xiaomi Mi Camera
2	shared/0/DCIM/Camera/	Pictures and videos (.jpg and .mp4)	Any pictures or videos that could have been taken by the user with his phone	Medium	Phone camera
3	shared/0/DCIM/Screenshots/	Pictures (.jpg)	Any screenshot that the user could have made	Medium	Phone screenshots
4	shared/0/Download/	Anything	Any file that the user downloaded	Medium	Phone downloads
5	shared/0/MIUI/Video/thumbs-mall/	Pictures (.jpg)	Thumbnails of videos that the user saved manually through the Mi Home App. Even if the videos were later deleted, the thumbnails of the videos were still present in this direc-tory	High (if the videos from the Mi App were saved at one point)	Xiaomi Mi Camera
6	shared/0/MIUI/debug_log/pow-erinfo/result_reason	Log file (.txt)	Logs about the kernel abnormal reboots	Low	Phone
7	shared/0/Pictures/mydlink/	Pictures (.jpg)	Snapshot taken by the user through the My D-Link App	Medium	D-Link camera
8	apps/com.google.android.app-s.chromecast.app/sp/com.-google.android.apps.chrome-cast.app_preferences.xml	XML file (.xml)	Boolean variables and their values regarding different pa-rameters from Google Home	Low	Google Home
9	apps/com.android.providers.cal-endar/db/calendar.db	Database (.db)	Informations about the Google calendar, such as entries, timestamps, metadata, reminders, etc.	Medium	Google calendar (Phone)

Table 6: List of artifacts retrieved during the logical acquisition

As we can see from Table 6, the recovered artifacts from the logical acquisition can be of capital importance from a forensic point of view. We elaborate now about the different artifacts recovered from the previous table.

- The pictures and videos we are referring to in Index 1 of Table 6 are those that could have been saved from the Xiaomi Home App into the user's phone. This means that the videos and images had to be saved manually into the phone, needing a human interaction for this. In the scenario that served us as a base, described in 3.2.3, the murderer would have been recorded by the Xiaomi Camera, but as the phone's user would be dead, he couldn't have saved this video on his phone. However, if the police and forensic investigators are present on the crime scene in the 8 days following the murder (because the buffer memory of the Xiaomi Camera is of 8 days, then the videos are deleted unless manually saved), they could immediately seize the phone, and, by manually interacting with it, they could save the video of the murder for a later investigation. In this case the artifact retrieved is of capital importance for solving the case. These videos and images can however be important as well in a different scenario in order to study the habits of the user, and to know who is potentially detected on a regular basis in his/her home environment.
- The files referred in Table 6 with Index 2, 3 and 4, being the pictures taken by the user from the phone, the screenshots made or the downloaded files, are not relevant in our hypothetical case but could be quite important for any other case, in order to know more about the habits of the user, which pictures he took with his phone, which files he downloaded or which screenshots he took.
- In the same fashion as for the files concerned by the first bullet point of this part, the pictures that can be found at the path in Index 5, could be of capital importance. Once again, if the videos from the Xiaomi Home App have been saved manually (by the user himself or by the forensic expert). The advantage of looking into this folder is in the case where the user saved a video on the internal storage of his phone and then deleted the video. In that case, some residual pic-

tures can be found on the phone, where thumbnails of the videos (a few frames usually) are still present. This is really valuable in the case where the video has been saved in the past, and then somebody erases the video in order to cover some tracks.

- The *powerinfo* folder presented in Index 6 of Table 6 contains timestamps of the phone's reboot, we also see the way the phone was rebooted, by pressing long on the power button, or via a reboot requested by the phone. A sample of the recovered file can be found in Appendix 3. For example we can see on the *kernel reboot* line the values *keypad, long_power_key, usb_chg* or *reboot*.

- The comments that we would like to make on the files from Index 7 are about the same that we did for the first one. We can see in this folder the pictures that the user took manually with the *mydlink lite* app. These pictures could be very valuable in some cases, where a suspect could take pictures of people without their consent, but in our case, as the phone does not belong to the suspect and the phone's user is dead, not a lot of valuable pictures could be found there.

- Some information can be retrieved from the file in Index 8 of Table 6. This information concerns more the environment that is linked to the phone and more specifically to Google Home. This XML file gives us the email address associated with the Google Home account, as well as the presence of an audio device. In our case the variable *audioDeviceDiscovered* is set to *true* indicating the presence of an audio device, in our case, the Google Home. The variable *assistantDeviceDiscovered* is also set to true, indicating once again the presence of the Google Home. The content of this XML file can be found in Appendix 4.

- The file presented in Index 9 is a database file, by opening it with a database browser, such as *DB Browser for SQLite*, we can recover some valuable information about one person's life. This database contains information about the calendar of the user, we can thus recover the name of the event present in the Google calendar, the starting date, the ending date, the time zone, the instances of the events, the reminders that have been set. If we take the example of an event, here is what we can see:

Title	**_id**	**sync_id**	**Datestart**	**Dateend**	**Time zone**	**Has Alarm**
Patrick’s birthday	2	Pq9vg5q098cd-p3j882n-mpgutn0	1581165200000 **8th of February 2020 at 2:00:00 PM GMT +02:00**	1581166800000 **8th of February 2020 at 3:00:00 PM GMT+02:00**	Europe/ Tallinn	1

Table 7: Recovered data from the calendar database

A sample of this database opened in *DB Browser for SQLite* can be found in Appendix 5.

All these recovered artifacts could be important for any forensics investigation, regarding our hypothetical case or other, as explained above.

Logical acquisition on a rooted phone

As stated in part 4.1.1, the phone at our disposal for the experiment was not rooted, meaning that we do not have the full control of the phone (the root privileges) and thus we cannot access every data on the phone. The fact that the bootloader was blocked as well was preventing us to root the phone. In this part, we first describe briefly the unlocking procedure of the bootloader, the rooting of the phone, and then the logical acquisition with the root permissions.

- The bootloader unlocking:

The unlocking of the bootloader is an essential step in order to root the phone and thus retrieve more data. However, the action of unlocking the bootloader is not insignificant from the forensics point of view. Indeed the unlocking of the bootloader will erase all the data from the phone, and perform a factory reset. That is why it is important to have the approval from the person in charge of the investigation before doing such type of actions, as some data might be lost during this step.

However, it is still possible to preserve data, which is what we did during our experiment. First of all, we did an ADB backup, with the same command described previously in section 4.1.1. We put it again here: *#adb backup -shared -apk -all -f backup.ab.* This backup command allows us to preserve the applications and their configuration, the user data present on the internal storage, and the system applications.

It is also possible if a Google account is linked to the phone to do a Google Drive backup. This is useful to have as many data backuped as possible. The Google Drive backup might not be possible in every situation because the credentials of the Google account have to be known by the investigator. In that case, only the ADB backup is possible. That is why it is important to perform the logical acquisition of the smartphone without root access before going on with this step. Every smartphone manufacturer has a specific procedure to follow in order to unlock the bootloader of their product. The procedure can usually be found on the Internet. We found the procedure to follow on the xda-developers forum [16]. It is important to note that this procedure is only possible from a Windows computer, as the unlocking tool is only available on Windows at the time of our writing. It was necessary to request an unlocking procedure from the Xiaomi website, download the tool they provide, and launch it from our computer, with the phone connected to it. It was then a matter of following the steps in order to unlock the bootloader.

After the unlocking of the phone's bootloader, we could now push the data back into the phone by using the following command with ADB: *#adb restore <path-to-the-backup-file>*. The action has to be confirmed on the phone.

- The phone rooting:

The rooting steps for the Xiaomi Redmi Note 7 were the ones followed in [16] as well. Which were consisting of downloading the TWRP image for Xiaomi Redmi Note 7, booting the phone into recovery mode, and pressing Install to flash the phone. Once rebooted, we could flash it with Magisk, it is important for us to enable the superuser access on the phone, in the Magisk application created (Menu → Superuser → enable shell). Magisk has been chosen over SuperSu because as explained by the xda-developers [17], Magisk does not change the filesystem of the phone, rending it more forensically sound.

- The logical acquisition of a rooted phone

  Having root privileges allowed us to have a shell on the phone, in order to interact directly with it and navigate the filesystem at we please. We could copy and extract data thanks to the *#adb pull* command. By using the previous command on the following directories: */data/data/*, */data/system/*, */data/misc/*; we could retrieve the files and the artifacts presented in Table 8 below.

Index	**Path of file / folder**	**Type of file**	**Data retrieved**	**Relevant regarding the case**	**Which sensor is concerned**
1	/data/data/com.android.chrome/app_Chrome/Default/Preferences	Text file (.txt)	Preferences of the chrome web browser, possible to retrieve the email address	Medium	Chrome web browser for Android
2	/data/data/com.android.providers.contacts/databases/contacts2.db	Database (.db)	Calls, conference calls, contacts	Medium, in order to know more about the background of a phone's user	Phone
3	/data/data/com.android.providers.media/databases/external.db	Database (.db)	Path of all the media files present on the phone	High, used to correlate the path with the files found	Phone
4	/data/system/packages.list	List (.list)	List of application installed	Low	Phone
5	/data/system/packages.xml	XML file (.xml)	List of applications installed and their permissions	Low	Phone
6	/data/misc/wifi/WifiConfigStore.xml	XML file (.xml)	List of SSID where the phone was once connected	Medium	Phone
7	/data/system/sync/accounts.xml	XML file (.xml)	List of account usernames for apps	Low	Phone
8	/data/system/usagestats/0/daily/1582036066197	XMLfile (.xml)	Application name and last time active timestamp, sorted daily, weekly, monthly and yearly	Medium	Phone
9	/data/data/com.google.android.gm/databases/downloader.db	Database file (.db)	Database containing files that were downloaded from Gmail.	Medium	Gmail on Android
10	/data/data/com.gooogle.android.gm/files/downloads/	Pictures (.jpg)	Thumbnails of the pictures	High	D-Link camera

	7f20bb9789b68266545eff-fefe74feaf/attachments/		taken by the D-Link Camera that were sent to the email address.		
11	/data/data/com.xiaomi.smarthome/cache/	Videos (.mp4)	Cached videos from the Xiaomi home app	High	Xiaomi Mi Camera
12	/data/com.android.chrome/app_chrome/Default/	Database file (.db)	Databases for the Google Chrome web browser: Cookies, History, Login Data, Top Sites and Web data	Medium	Google Chrome web browser for Android
13	/data/com.android.vending/databases/localappstate.db	Database file (.db)	Installation date of the applications of the account associated	Low	Phone
14	/data/com.android.vending/databases/suggestions.db	Database file (.db)	Keywords that the user typed in the search bar of the Android/Google app store	Medium	Google Play Store on Android

Table 8: List of recovered artifacts from a logical acquisition of a rooted phone

As we did for the previous table of artifacts, we elaborate on what is present in table 8.

- Index 1: In this file, presenting the preferences of the Google Chrome web browser for Android, not a lot of information are present or useful, we can however retrieve the email address which is associated with the Google account.
- Index 2: This file could become pretty useful in a forensic investigation, it would provide us the date and time of the calls and the number that was called. The same information would be provided for the conference calls. The contacts saved, if any, are also present here.
- Index 3: This file has really high importance from a forensic point of view. It contains the path of all the media files from the user. Having all these paths allows us to correlate them with the actual path of the files, in order to check

their integrity. This file gives us the exact path of all the pictures that were taken with the phone for instance. So we know that by looking at this location, we should find this particular file. If the file is indeed present at this location, we can assume that the integrity of this particular file has not been violated.

- Index 5: As explained in Table 8, this file gather the applications installed on the phone and their permissions. We can for example see that for the application *com.google.android.gms*, which correspond to Gmail; we have the following permissions allowed: *android.permission.REAL_GET_TASKS, android.permission.ACCESS_CACHE_FILESYSTEM, android.permisssion.REMOTE_AUDIO_PLAYBACK, android.permission.DOWNLOAD_WITHOUT_NOTIFICATION.* The name of these permissions are self explanatory and allows us to understand which applications are allowed to communicate with which component or other application of the phone.

- Index 6: In this file, we have access to all the SSID information the phone was once connected to, such as the name of the SSID, in our example *HUAWEI-B535-9C7C* the value of the Pre-Shared Key: 5L0LRJ2TQF4, in our case, this Pre-Shared Key is the actual Wi-Fi password of the Huawei router. If the password is modified, it is still possible to find it in this file, in plain text. Some configuration variables are also accessible, such as the “IpAssignment” set as *DHCP*, or the “ProxySetings” set to *NONE.*

- Index 8: This folder contains several files, sorted in the following fashion: daily, weekly, monthly, and yearly. In these folders, XML files are present, their names are in the form of a timestamp, and thus give us information about what is the content of the file. For instance, by opening the folder *daily* and converting the timestamp of the second file (1582036066197), we find that it is the following date: Tuesday, February 18, 2020, the date of the murder in our scenario. An investigator could then investigate this XML file in order to find which application was used for the last time on this particular date.

- Index 10: As stated in the 3.1.3 section, the D-Link camera sends 6 frames (3 frames before and 3 frames after detection) of video stream to the email address of our user every time a motion is detected in its field of view. The folder presented here, contains thumbnails of the pictures that were sent to the email address of the user, if and only if the user opened the concerned email at least once. The opening of the pictures themselves in the email is not necessary in order to populate this folder. It could be then possible for an investigator to save or (at least) click on the email with the incriminating frames in order to save them as thumbnails on the phone. The pictures would stay on the phone, even if the mail they were attached to were deleted afterwards.

As we could see from the experiments and the results above, we could see that valuable data can be recovered from both the Logical extraction when the phone is unrooted as well as when it is rooted. For example, in a non-rooted logical acquisition, some valuable data include pictures, videos, and thumbnails of videos saved in the phone from the Xiaomi Camera, as well as pictures saved thanks to the D-Link Camera. Some other files can give out information about the habits of the user, such as the calendar and the reboots information. On the other hand, having the phone rooted and performing a Logical acquisition then allows an examiner to access more files, including more thumbnails, pictures, and videos from various sources like Google Gmail, the Xiaomi Camera, and the D-Link Camera as well as phone calls history. Other information such as Wi-Fi information, Google Chrome preferences, applications installed, and statistics about application usage gives a better understanding of the environment and the habits of a user. Databases of keywords searches, chrome history, and overall file location are also useful in various cases.

It is also worth noting that, in our opinion, it is really important to take a look physically at the device itself because some data, such as the videos from the Xiaomi Camera are persisting on the application for only 8 days, but by saving them manually on the phone, we can access them without any limitations. Forensic investigators have to take into account this information and thus prioritize the data collection. We thus suggest to take a look at the phone manually upon arrival at the crime scene, looking for some specific applications in order to have an idea about the persistence of the data; for example, in case we find a Xiaomi Home App, it is good to open it, and save all the monitored videos on the internal memory, in order to preserve this potentially important data. Some mails can also be opened in case of a mydlink Lite application installed, for the same reason as stated previously, in order to make sure that the data will stay on the phone and can't be altered by a third party. After all these steps have been taken into account, we can proceed to the Logical acquisition, and then the Physical acquisition.

4.1.2 Physical acquisition

A Physical acquisition can, in the same way as a Logical acquisition have multiple definitions depending on investigators or organizations. As stated by the NIST in 2014, in their SP 800-101R publication, a physical extraction is defined as "extracting and recording a copy or image of a physical store"[28]. A physical acquisition is a bit-by-bit copy of a physical store. It is also important to note, that, opposed to a Logical acquisition, a physical acquisition "accesses the lower areas below filesystem and is thus capable of retrieving the data that has been marked as deleted"[14]. There are several ways to do a physical collection of a phone, such as JTAG, Chip-off, for a hardware-based collection, or software-based [18]. Methods requiring a "disassembly of the phone for access to the circuit board" [10, p.165] is considered to be *Invasive* (like JTAG or Chip-off), as opposed to the Software based which are considered *Non-Invasive.* As in our case, the phone was still in a working state, we could use the *Non-Invasive* methods which require usually less equipment and are preferred in our case. The Physical collection of a phone requires root privileges. It should thus be done after the Logical acquisition if this last one is possible. We decided to do a bit-by-bit copy of the internal storage of the phone thanks to the *dd* utility. It is normally possible to use *dd* in order to make the image of the filesystem of the phone and to direct the output to an empty SD card on the phone, that the examiner would have put in beforehand. We did not have an SD card at our disposal during our experiment, so we decided to transfer the file thanks to a TCP connection between the phone and the computer. This process is briefly described below.

The problem we were facing with the TCP connection copy was the fact that the *netcat* utility is not included by default on Android, we thus had to use a third-party app to accomplish this task. For this purpose, we used BusyBox[1], fortunately, it was possible to install it thanks to the Magisk Manager app of the phone after the rooting,: Magisk Manager → Menu → Modules → Busybox for Android NDK. It was then necessary to have a shell access to the phone, we used *# adb shell.* In order to know which partitions we want to image, a useful command is the following: *# cat /proc/partitions*. In order to copy the partitions we are interested in into our computer through a TCP connection we used the following commands:

- On the computer: *# adb forward tcp:9999 tcp:9999*
- On the phone: *# dd if=/dev/block/<the block we're interested into> | busybox nc -l -p 9999*
- On the computer: *# nc 127.0.0.1 9999 > img.dd*

By seeing the partitions, we can figure out which one we need with the size usually, which has to be close to 32 Go, as it is the size of the internal memory of the phone we had at our disposal. Two partitions stood out, first the *mmcblk0* and the *dm-1*.

As we could see from the TWRP (one of the utilities that allowed us to root the phone) logs (see in Appendix 6), a new block was created after decryption: *dm-0*. In reality, another block was created after decryption, which has a size close to the size of the internal storage of the phone: *dm-1*. We imaged both *mmcblk0* and *dm-1* for the sake of completeness. The resulting image of *mmcblk0* was 29.1 Go large, and the one from the *dm-1* was 21.7 Go.

We then ran *Autopsy (*on Windows) on our images, with the following ingest modules: *File Type Identification, Embedded File Extractor, Exif Parser, Interesting Files Identifier, PhotoRec Carver* and *Android Analyser;* in order to carve data from unallocated space (where the deleted data usually lies).

The image of the *mmcblk0* block contains, as expected, no information to retrieve, because, from "Android 6.x to Android 9.x, encryption is the default" [10, p.431]. Our device is running on Android 9 PKQ1.180904.001, and as it can be seen from Appendix

[1] https://play.google.com/store/apps/details?id=stericson.busybox&hl=fr

7, our device is encrypted (even though no passwords were set on the phone). With a physical acquisition of the encrypted partition, we could recover no interesting artifacts because the /userdata partition was encrypted and thus, impossible to investigate. However the unencrypted image was much more verbose. Here are the data that we could retrieve:

Index	**Path of file /folder**	**Type of file**	**Data retrieved**	**Relevant regarding the case**	**Which sensor is concerned**
1	/data/system_ce/ recent_tasks	XML files (.xml)	Last applications used	Medium	Phone and Potentially Xiaomi Camera, D-Link Camera and Fibaro controller
2	/data/system_ce/recent_images	Images (.png)	Screenshots of the last applications used	Medium	Phone and Potentially Xiaomi Camera, D-Link Camera and Fibaro controller
3	/data/system_ce/snapshots	Images (.png)	Screenshots of the last applications used	Medium	Phone and Potentially Xiaomi Camera, D-Link Camera and Fibaro controller
4	/data/com.xiaomi.smarthome/files/device/ cache/ device_list_ba51de85a374 b126ada4fc63fa63a265	Plain text file	Information about the Xiaomi Camera	Medium	Xiaomi Camera and Huawei Router
5	/data/com.google.android.apps.chromecast.app/ files/home_graph_dG90by5oYXJyaXNzb25AZ21h aWwuY29t.proto	PROTO file (.proto)	Information about Google Home environment	Medium	Google Home
6	Carved files	Pictures (.jpg)	Pictures that were sent to the email address from the D-Link or that were saved on the phone with the Xiaomi Camera App	High	D-Link Camera and Xiaomi Camera

Table 9: List of recovered artifacts from a physical acquisition of a rooted phone

- Index 1: This folder includes several XML files, named such as *43777-164.task* that contain the name of the last applications used, like *com.xiaomi.smarthome, com.android.settings,* or *com.miui.gallery.*

- Index 2 and 3: These two folders contain screenshots automatically taken while the user was using the last used applications that we can see in the *recent_tasks* folder. By correlating the screenshots present in these two folders with the XML files present in the *recent_tasks* folder, it is possible for an examiner to know which app was last used, and how its interface looks like.

- Index 4: This file is a text file, where we can retrieve information concerning the Xiaomi Camera, such as its MAC address and IP address, as well as the MAC address of the router it is connected to. This file can be interesting to get more information about the environment in which the phone and the Xiaomi Camera are.

- Index 5: This file, similarly to the previous one, is giving us information about the Google Home device and its environment, such as the name of the Home set in the Google Home app, the Address of the Home, the Location, the email address, the name of the room, the devices associated and the email address associated, the type of speaker (in our case the Google Home Mini) and its capabilities (the commands that it is possible to use with it, which are not relevant for a forensic investigation).

- Index 6: Because we used data carving tools included in Autopsy on an image from a physical extraction of our phone, we could recover a lot of deleted pictures. The pictures that we could recover with Autopsy were the ones that we already recovered in the Logical acquisition with and without rooting as well as the deleted ones. We could thus retrieve nearly all the pictures that were once present on the phone. The ones not possible to recover would be the damaged pictures. The specifically interesting ones are the ones that were taken thanks to the D-Link Camera, stored and then deleted automatically as well as the ones that were saved manually and deleted manually thanks to the Xiaomi Camera app. It is really important to consider these data. Because, as we can imagine, in our scenario, after being recorded by the Cameras, the murderer could delete all the incriminating pictures of him on the phone or on the Gmail account, and while doing the digital forensic analysis, the experts could still recover numerous pictures and thus incriminates the suspect.

As we could see thanks to the physical acquisition, the majority of the files that could be recovered were already possibly recovered during the previous phases (Logical acquisition with and without root access), however, the physical acquisition allows us to do several major things. First, it allows us to recover the pictures still present on the phone, as well as the ones that have been deleted, which was impossible to do with only the Logical acquisition. Second of all, tools such as Autopsy are much easier to use and allow us to have an automated analysis and thus to skip the manual analysis of the files on the phone, which saves us time. Third and lastly, it is still possible to find all the files that we found in the previous analysis in the program. This gives us the opportunity to double-check and correlate the path and the folder where the files were found and thus to ensure the integrity of the files, giving the examiner's proof better admissibility in court.

To summarize this part, during the phone analysis, several important things have to be noted. First, in the environment described, it is not possible to retrieve all the relevant data with only one type of extraction, we thus advise to use them all. Each analysis relies on different techniques, levels of privileges, and intrusiveness in order to be successful. Once the phone has been seized, it is important for the investigator to manually interact with it in order to save important artifacts that could be deleted automatically, we are here referring to the videos buffered in the internal memory of the Xiaomi Camera, that have to be saved on the internal memory of the phone not to be deleted automatically. The pictures sent by the D-Link camera to the Gmail address of the user can also be viewed through the Gmail app; some valuable pictures can be seen there. If the murderer already accessed these emails in order to cover his tracks, it will be possible to recover the pictures saved. The next step would be to do a Logical acquisition of the unrooted phone, for this, a manual analysis of a full ADB backup should be performed. Several pictures and configuration files can already be recovered at this stage. In the case of the Xiaomi Redmi Note 7, as the phone is unrooted and the bootloader locked, the unlocking of the bootloader would cause a complete loss of data. It is up to the person in charge of the investigation to decide whether this action should be done or not. If the investigation should be pursued, backups should be performed beforehand, such as a full ADB backup and a Google backup if possible. After the phone has been factory reset, the backups should be pushed back onto the phone. The rooting procedure doesn't reset the data if done properly. The *Magisk* rooting tool has to be preferred over *SuperSu* as it does not impact the filesystem. After the rooting, the Logical acquisition can be performed manually, by the means of getting an ADB shell on the Android phone and copying manually the interesting files on the investigator computers for further investigation. It includes databases, additional pictures, and configuration files. Lastly, a physical acquisition should be performed, thanks to *dd* and *Autopsy*, in order to retrieve deleted files and pictures as well as to correlate the data with what was found during the Logical acquisition. As we could see, a lot of valuable information can be retrieved about our environment and case, only thanks to the analysis of the phone.

4.2 Network Forensics

Some artifacts recovered during the previous steps of our investigation already allowed us to have information about the network structure of our environment. Configuration files, such as WifiConfigStore.xml were giving out network information (name of SSID to which the phone was once connected to). However, our previous analysis did not include any packet analysis. We present this step in this part of the thesis. The problem with network analysis arises in a murder case such as ours because it is not possible to analyze the network communications occurring at the exact time of the murder. It is however possible and recommended if possible to take a dump of network traffic or dynamically analyze the traffic after the murder occurred, in order to have a better understanding of how the different devices communicate between each other, and to retrieve possible interesting data. We used several tools for this, such as *Wireshark*, *Network Miner*, and *tcpdump*.

In order to perform this analysis, the investigator should be on the crime scene to analyze the environment. Our steps involved the use of *tcpdump*, a Linux utility that can be run on the phone itself. The phone has to be rooted in order to use *tcpdump*. The *tcpdump* binary[1] has to be downloaded and transferred into the phone, by using an *adb* push command, such as the following: *#adb push tcmdump /data/local/tmp/tcpdump.* It is good to use the tmp folder of the phone because every user has the right to write in it.

[1] https://www.androidtcpdump.com/

It is then necessary to get a shell on the phone (with *#adb shell*). This allows us then to capture the packets with the following command: *#tcpdump -i any -s 0 -w dump.pcap*

The investigator should then interact manually with the phone in order to produce data and traffic. Because we are dealing with a smart home environment, and we are aware that the phone is communicating with the Google Home, the Fibaro controller, and both cameras, we can open each of these apps to generate data, and connect to the associated account if possible. This is usually not a problem as the credentials were most likely saved by the user, as a result, the applications do not require a new credentials input from the user side. In a more general way, an investigator should open all the apps on the phone that he suspects to be connected to the smart home environment, such as camera application, home security application, home monitoring, etc. Once all these steps have been done, we can stop the *tcpdump* utility. With the .pcap file at our disposal, we can now analyze it in *Network Miner* and/or *Wireshark*.

The amount of relevant data is much lower than what we could retrieve with the Logical and Physical acquisition of the phone, but we could nevertheless recover some relevant information.

- The D-Link camera communicates through HTTP (TCP) port 80, allowing us, if knowing the credentials, to use the graphical interface on a computer, which gives out much more relevant data, such as the configuration of the motion detection, the email address where the pictures are sent to and the port used to communicate with Gmail servers. The network settings such as the MAC address, the IP address, the DNS server set for the D-Link camera can also be retrieved, as well as the firmware and hardware version.

- The Fibaro controller communicates through HTTP (TCP) port 80, allowing us like previously, to use the graphical interface on a computer and thus accessing more detailed data about the state of the sensors throughout the time (when was which sensor triggered, and for how long for instance), as well as the users using the controller and their respective rights.

- The phone and the Fibaro controller passes the credentials thanks to basic authentication, we can get the credentials in plain text when passing our .pcap file in *Network Miner* or *Wireshark*. A screenshot is presented in Appendix 8. The request where the Fibaro credentials can be retrieved is in an HTTP packet sent from the phone to the Fibaro controller at the following address: *<IP address of the controller>/api/loginStatus?action=login* as seen in Appendix [9].

- The user name of the D-Link camera account was also retrieved, but not the password.

By retrieving this information, an examiner can have and provide a better understanding of the smart home environment and the interactions between the devices. In our case, by retrieving plain text credentials, it would even allow him/her to gain access to devices present on the scene (such as the Fibaro controller) in order to have access to more data, as presented in the next part about IoT forensics.

As we stated in this part, in order to perform a tcpdump for network analysis, a phone has to be rooted. Which involves doing some work on the phone prior to doing the network analysis. In the case where the investigator sniffs the network with *Wireshark* on his computer (which does not require the phone rooting), and interact with the phone and the Fibaro app to initiate the authentication process (with the credentials saved), the investigator will not be able to retrieve the credentials for the Fibaro controller. The only case where this would be possible is if the authentication process would take place on the web interface of the controller. The problem there, is that the credentials are unknown to the investigator when prompted for them on the web interface of the controller. They will thus not be transmitted through the network and thus will not be seen on the packet analysis. We would then encourage to perform the network analysis thanks to *tcpdump* on the rooted phone because only sniffing the network will not provide credentials. This consideration imply that the investigator would need to perform the network analysis after having the phone rooted, forcing the investigator to come back to the crime scene after the initial seizure of the devices. The analysis of the IoT devices should be done immediately after (as explained in the next section).

It is important to note, that for this analysis to be successful, the investigator must use the rooted user’s phone, launch the *tcpdump* utility on it, and then interact with it by opening every app that could connect to a sensor and authenticate to it if it is possible. By analyzing the dump file created, an investigator could then retrieve precious information, especially credentials. During our experiment, we also tried to sniff the network with a computer and with *Wireshark*, but this method did not allow us to recover the credentials. Several configuration and hardware were tested but no results were convincing. This step may require more testing than we could perform.

In the case the credentials from the Fibaro controller are already known by the investigator, the network analysis is not mandatory, as the credentials are the most valuable artifact retrieved during this analysis. It is indeed possible that the credentials could be guessed, such as the default credentials *admin:admin* that might have not been changed; or because the Fibaro controller is opened in a web browser, it is possible that the user saved his Fibaro credentials directly in his web browser (the investigator would need in this case to use the user’s computer to connect to the Fibaro controller web interface), allowing the investigator to access directly the Fibaro controller web interface without needing to know the credentials.

4.3 IoT forensics

In our lab environment, the other main IoT devices are the Google Home Mini, the Fibaro controller, and the Fibaro sensors: Flood sensor, Door sensor, Motion detection sensor. The Google Home Mini is a smart speaker that is used in order to make the user’s life better. The user can for instance manage simple tasks through voice commands by talking to the Google Home, such as “Ok Google, add an event in my calendar”, or “Hey Google, What time is it?” and so forth. Having information about this vocal commands as well as other specifications of the Google Home can potentially help an examiner to understand more the habits of a user and the configuration of his/her home environment. The Fibaro sensors have basically the functioning of a switch. They all have two states: 1 and 0. These Fibaro sensors are really simple in their shape, functioning, they are battery powered and use the Z-Wave protocol to communicate to the Fibaro controller. The Fibaro controller is used to manage all the sensors, and to monitor their status. The Fibaro sensors could be investigated directly, with hardware forensics methods, which were not applied by us as it does not fall under our scope of study. The other way to investigate the Fibaro sensors as directly as possible would be to analyze the Z-Wave communications occurring, which we didn't do either because we did not have any device capable of sniffing Z-Wave communications. The only device directly related to the Fibaro sensors in our environment is the Fibaro controller, which we analyzed. In this part, we present the analysis of the Google Home mini and the Fibaro controller, including as a result the investigation of the Fibaro sensors.

4.3.1 Google Home

There are several ways to extract data from the Google Home with a software-based technique (because any hardware-based technique where the devices are opened do no fall under our scope of study). The first way to recover information is to interact directly with the phone connected to the Google Home, thanks to the Google Home app[1]. It is possible, as with the PROTO file found during the physical acquisition phase (*/data/com.android.google.apps.chromecast.app/files/home_graph_dG90by5oYXJyaXNzb25AZ21haWwuY29t.proto*), to see the settings of the Google Home such as the name of the Home, its address, the email address of the Google account associated to the Google Home, the name of the Room, the other devices associated. It is also possible to recover the Google shopping list of the user, by simply browsing the Google Home application manually. By doing this, it is also possible to discover some useful information, such as the one stated above, as well as the IP address of the Google Home Mini, or the languages that have been set for it (two languages maximum can be set). The rest of the information we can get highly depends on which services the user uses from Google and if he/she set them in the app or not. For example, if the user set a nickname for him, or his birthday, phone numberm or home address, it would be possible to recover this data from the Google Home app. Reservations for flights, events ,or hotels could also be seen, if the user reserved them with the same email address. All this information allows us to have more information in the user's background, his habits, and the potentials events and trips he was planning.

1 https://play.google.com/store/apps/details?id=com.google.android.apps.chromecast.app&hl=fr

What can also be quite interesting from the investigator's point of view, is the vocal commands that the user used with the Google Home. All the commands that the user used, are stored in Google My Activity, accessible through the phone with the Google Home App, or with a computer, at the following address: *myactivity.google.com*. When used on the phone, My Activity allows us to have access to every voice command that the user pronounced. For each command, we know what the user said, at what time and date, where, and we also have access to the Google Home answer. If the recording setting is activated, it is also possible to hear the voice of the user saying the command. This could help an investigator to know whether the user was alone, if he was the one speaking, etc. It is also important to note that it is possible for the user to delete all of this information from My Activity, afterward Google gives a message saying that this event is not associated in any way to the user's account. If Google My Activity is looked upon a web browser, it is also possible to retrieve which app he used and at what time as well as the vocal commands described previously.

By knowing the IP address of the Google Home, which is an information easily retrievable by going in the settings of the Google Home app, or by doing a nmap scan on the network where the Google Home is, we can use the Advanced REST Client[1], in order to make GET requests to the Google Home. By making GET requests on the Google Home, thanks to the HTTP protocol, on port 8008, we can have access to several files from the Google Home, and thus retrieve more information. By using this technique with the previously described parameters, we can retrieve useful information:

Index	**Path of file**	**Data retrieved**	**Relevant regarding the case**	**Which sensor is concerned**
1	http://*GoogleHome-IpAddress*:8008/setup/eureka_info	Configuration about the Google Home	Medium	Google Home
2	http://*GoogleHome-IpAddress*:8008/setup/supported_timezones	Time zones supported by the Google Home	Low	Google Home
3	http://*GoogleHome-IpAddress*:8008/setup/supported_locales	Locales supported by the Google Home	Low	Google Home

Table 10: Data retrieved from the Google Home

[1] https://github.com/advanced-rest-client/arc-electron/releases

- Index 1: This file is the most verbose file that it is possible to retrieve from the Google Home. We can extract information such as the BSSID, the build version, IP address of the Google Home, Locale, Country code, Latitude, Longitude, MAC address of the Google Home, Name, Noise level, Public Key, SSID, and the Time Zone. More variables can be seen in that file, but we limit our findings to the ones we judge as valuables.

- Index 2 and 3: These two files are interesting in terms of capabilities of the Google Home, even though it is not useful for our case, these files give respectively the supported time zones and the supported locales of the Google Home.

The three previous files described above are the only ones possible to retrieve with this configuration (8008 port), because in early 2020, Google pushed a new update on every Google Home devices, thus rendering a lot of files inaccessible through port 8008, resulting in 403 errors (forbidden). If we want to explore more files, we can do several things, as described by rithvikvihu on Github [19]. As explained, Google changed the port and implemented security features. In order to access them, it is necessary to "change the port from 8008 to 8443 (HTTPS), to change the protocol from HTTP to HTTPS, to add a new header cast-local-authorization-token"[19]. A token is also needed. This is a simple, task, we need to pull a file thanks to *adb* (a file that we used earlier) which is:

/data/com.google.android.apps.chromecast.app/files/home_graph_dG90by5oYXJyaXN zb25AZ21haWwuY29t.proto

Then, the token from this file has to be retrieved thanks to a website the author of the GitHub source made[1]. The curl request needed in order to view the new information would look like this: *#curl –include –insecure H "cast-local-authorization.token: <value of the extracted token>" https://GoogleHomeIpAddress:8443/setup/assistant/ alarms*

By making these changes, it should be possible to retrieve these additional files and in-formation:

[1] https://rithvikvibhu.github.io/gh-web-proto-decode/

Index	Path of file	Data retrieved	Relevant regarding the case	Which sensor is concerned
1	https://*GoogleHome-IpAddress*:8443/setup/assistant/alarms	Date and time of the alarms setup by the user	Medium	Google Home
2	https://*GoogleHome-IpAddress*:8443/setup/bluetooth/status	Information and configuration about the bluetooth devices	Low	Google Home
3	https://*GoogleHome-IpAddress*:8443/setup/bluetooth/get_bonded	Information about bonded devices trough bluetooth	Low	Google Home
4	https://*GoogleHome-IpAddress*:8443/setup/bluetooth/scan_results	Scanning results for the bluetooth discovery of the Google Home, inlcuding TVs, mobiles	Medium	Google Home
5	https://*GoogleHome-IpAddress*:8443/setup/configured_networks	Networks to which the Google Home is connected to	Low	Google Home
6	https://*GoogleHome-IpAddress*:8443/setup/scan_results	List of all the access points available nearby for the Google Home	Medium	Google Home

Table 11: Data retrieved from the Google Home with the updated configuration

In our case, we could retrieve data from the file presented in Index 2, 5, and 6, as the others appear to be empty. In order to know which files were possible to read from the Google Home, we used the information available on the Github page of rithvikvihu [20], where the author gives all the files possibly readable from the Google Home.

The information presented in the table above allows us to gain information about the user habits, especially the one concerning the alarms set, in our scenario, we could maybe see the alarm the victim set previously for the end of his nap, and correlate this with the time of the murder; as well as some information related to the configuration of the Google Home, such as access points connected and Bluetooth devices connected, and also, the devices in range, waiting for a connection.

4.3.2 Fibaro controller and sensors

In the same fashion as we retrieved data from the Google Home, two ways of retrieving data are used for the Fibaro controller. As explained previously, the Fibaro controller is the “hub” of all Fibaro Z-Wave sensors. We can thus recover their state and data related to them thanks to the web interface of the Fibaro controller. The web interface of the Fibaro controller can be accessed by browsing to the IP address of the Fibaro controller, which is once again easy to retrieve using a nmap scan on the network. In order for somebody to access the web interface of the Fibaro controller, the credentials of a user account are needed. We could also retrieve this information from the Network analysis described in section 4.2. Once logged in the Fibaro controller web interface, it is possible to gather data about all of the sensors present on the scene and their configuration, as well as the configuration of the users and their rights. We can retrieve information on which room the sensors are located, their current state as well as their past state. Z-Wave can have two states: 1 or 0, or Up and Down. By going in the Event Panel in the web interface of the Fibaro Controller, it is possible to have a precise timeline of when the sensors were in the state Up (event detected) and when the sensors went back to the state Down. This information is extremely important for a forensic examiner because it will allow him to retrace a possible scenario according to the sensors triggers and the time of their triggering. For instance, in our base scenario, by examining the events of the sensors, we can understand that when the Motion Sensor was triggered at 15:33:36 correspond to the time when the murderer passed in front of the cabinet to kill our victim. The same would apply to the door sensor, opened at 15:33:51, and closed at 15:33:58, depicting the fact that the murderer opened the cabinet to steal something and closed it again afterward. These past events are saved for a long time on the Fibaro controller (we do not have any precise information about how long the logs are saved or how much logs are saved). We can also see the current state of the sensors on the front page of the web application as well as the video stream of the D-Link Camera because this last one was linked to the Fibaro controller.

It is also possible to retrieve data from the Fibaro controller, by making HTTP requests on port 80 in a web browser:

Index	Path of file	Data retrieved	Relevant regarding the case	Which sensor is concerned
1	http://FibaroControllerIpAddress/api/settings/info	MAC address of the controller, Z-Wave region, language, sunset hour, sunrise hour, software version, version stable or not	Medium	Fibaro controller
2	http://FibaroControllerIpAddress/docs/#!/settings/getBackup	Type of backup, timestamp when the backup has been performed, number of devices backuped, number of rooms	Low	Fibaro controller
3	http://FibaroControllerIpAddress/api/settings/location	Time Zone, NTP server, date, units	Low	Fibaro controller
4	http://FibaroControllerIpAddress/api/settings/network	IP address of the controller, network mask, gateway, DNS server, remote access allowed or not, DHCP or static IP	Low	Fibaro controller
5	http://FibaroControllerIpAddress/api/devices	For each device we have: Type of device, sunrise hour, sunset hour, Z-Wave version, address mail associated to the Fibaro device, Last password change, Location and previous location, type of user (superuser or not), Device token and UID, Push notifications enabled or not, Battery level, Armed or not, Creation timestamp, Modified timestamp.	Medium	Fibaro controller
6	http://FibaroControllerIpAddress/api/sections	Sections created, in our case "4th floor"	Low	Fibaro controller
7	http://FibaroControllerIpAddress/api/rooms	Name of the room(s), in our case "Office"	Low	Fibaro controller
8	http://FibaroControllerIpAddress api/users	ID of the user, mail, privileges, rights.	Medium	Fibaro controller
9	http://FibaroControllerIpAddress/api/iosDevices	Phones linked, not just IOS	Medium	Fibaro controller
10	http://FibaroControllerIpAddress/api/panels/location	The location of the Fibaro controller	Low	Fibaro controller
11	http://FibaroControllerIpAddress/api/panels/event?last=30&type=id	Last events that the sensors recorded, here, the last 30 events.	High	Fibaro sensors
12	http://FibaroControllerIpAddress/api/panels/event?from=1582032780&to=1582032900&type=time	Here it is the same as the previous entry in the Table, the difference is that we are analysing all the events that happened in a time period	High	Fibaro sensors

Table 12: Data retrieved on the Fibaro controller and sensors

The information that it is possible to retrieve from the Fibaro controller includes a lot of configuration variables, like the number of rooms set up in the smart home environment or the privileges of the user. However, the most important data to retrieve, from the web interface (in the Event Panel menu) or thanks to HTTP requests, are the state of the sensors and their timestamps. As we can see, in the Index 11 and 12 of Table 12, we can analyze the state of the sensor (Up or Down), at a given time. The investigator could then, by knowing approximately when the murder occurred, frame this time in the events present on the Fibaro controller, as it can be seen in Appendix 10 and 11. For example, in our case, we know that the murder happened around 15:33. We then made the following HTTP request, as presented in the Index 12 of Table 12:

http://FibaroControllerIpAddress/api/panels/event?from=1582032780&to=1582032900&type=time

And we ended up with the result showed in Appendix 11. The two timestamps that we put in our request correspond respectively to the 18th of February 2020 at 15:33 GMT+02:00 (1582032780*)* and to the 18th of February 2020 at 15:35 GMT+02:00 (1582032900*).* From this web page, we can see the state of the sensors and the time they were triggered. For instance, we see that the door sensor changed its state from *oldValue:0* to *newValue: 1* at 1582032831, corresponding to the 18th of February 2020 at 15:33:51 GMT+02:00, which is the exact time the door was opened by the murderer.

From our IoT forensics analysis, we could see that the data that it is possible to retrieve have a rather low impact on our case. This is true for the configuration files that can be recovered through the APIs or the HTTP requests. This information is giving out details about the environment and the configuration of the IoT devices, but not much on the user-generated data on these devices. However, we can see that one of the most important sets of data is also acquired in this part. We are referring here to the events that happened from the sensor point of view, gathered on the Fibaro controller. These events give out highly important data, as they would help an investigator to retrace the events and thus what happened, or at least, proposing interpretations of what could have happened.

5 Discussion

In this part, we summarize the results in two recapitulation tables. The first table presents which are the most important artifacts that can be retrieved and from where, from which device, at which state of the investigation, and with which level of priority. The second table puts in perspective at which state of the analysis, it is possible to recover which type of data. We then present some considerations about the results of our experiment, by identifying possible problems or bottlenecks that could arise during a similar analysis. Improvements and future works are tackled as well in the last part of this thesis.

Prioritization	**Forensic technique**	**Most important artifacts**	**Information retrieved**
1	Manual	In the Xiaomi app or the Gmail app	Videos or images saved by the cameras
2	Logical acquisition (un-rooted phone)	shared/0/DCIM/Xiaomi/local/284034774/	Videos from the Xiaomi Camera if they were saved manually
		shared/0/MIUI/Video/thumbsmall/	Thumbnails of the videos that the user saved manually and/or deleted
3	Logical acquisition (rooted phone)	/data/data/com.gooogle.android.gm/files/downloads/7f20bb9789b68266545efffefe74feaf/attachments/	Thumbnails of the pictures taken by the D-Link that were sent to the Gmail account
		/data/data/com.xiaomi.smarthome/cache/	Cached videos of the Xiaomi Camera
4	Physical acquisition	Carved files	Pictures that were sent to the email address from the D-Link or pictures and videos that were saved on the phone with the Xiaomi Camera App
5	Network interaction and analysis	Network Miner → credentials	Credentials for the Fibaro controller and username of D-Link camera
6	IoT	http://**FibaroControllerIpAddress**/api/panels/event?from=**Timestamp1**&to=**Timestamp2**&type=time **OR** Event Panel in the web interface	Events and state of the Fibaro sensors from one time to the other
		https://**GoogleHomeIpAddress**:8443/setup/assistant/alarms	Date and time of the alarms set up by the user

Table 13: Recap chart of the important artifacts retrieved

	Manual	**Network with root**	**IoT**	**Logical without root**	**Logical with root**	**Physical**	**IoT with root**
Xiaomi Camera pictures				x	x	x	
Xiaomi Camera videos				x	x	x	
Xiaomi Camera thumbnails				x	x	x	
D-Link snapshots				x	x	x	
D-Link pictures	x			x	x	x	
D-Link thumbnails					x	x	
Phone pictures				x	x	x	
Calendar				x	x	x	
Chrome preferences					x	x	
Contacts and calls					x	x	
Wi-Fi Configuration					x	x	
Application installed and usage stats					x	x	
Last applications used						x	
Credentials of the Fibaro controller		x					
Fibaro sensors states			x				
Google Home configuration			x				x
Google assistant alarms							x
Google Home bluetooth and Wi-Fi status							x

Table 14: Recap chart of the retrieved types of artifacts

We think it is important for forensic investigations of this type, to keep in mind that data collection has to be prioritized. As shown in Tables 13 and 14, in our case, a forensic investigator should first seize the phone, examine it manually at the scene, to save if needed potentially important data thanks to the applications already installed. Secondly, the examiner should perform in the lab, a Logical analysis of the phone without the root access, and with the root access afterward, followed by the physical acquisition. In the case where the investigator can go back on the crime scene, he could gather more information by performing a Network and an IoT devices analysis, using the abilities of a rooted phone to extract more data. In the case the investigator cannot go back to the scene a second time, it could be possible to seize the Fibaro controller to investigate it in the forensic lab, as even if the controller is turned off, the sensor events are still accessible. The Network and IoT analysis would give out really important information, such as credentials for the Fibaro controller, which will be useful to recover the state of the sensors' events afterward, which are one of the most important artifacts recovered. It would also allow the investigator to recover alarms set on Google Home for instance. Our work, by exploring manually, with the help of free tools exclusively, our use case environment, aiming at proving that significant information can be gathered from all of the devices gives out satisfying results regarding the scenario tailored for this experiment.

Important considerations arose during our experiment and from the result of our work. Especially regarding the collection of the data that could be done at the scene or in the forensic lab. The amount of data collected might differ depending on the possible warrants that the police would be able to get to go on the scene. Because during our investigation, and especially for the Network analysis and the IoT analysis, the investigator would need to go back to the scene in order to do these collections (network and IoT analysis), after the phone has been rooted in the lab. In the case where it is not possible for the investigator to go back to the scene after the first seizure phase, a lot of really important data might not be retrieved at all. An investigator could try to do a network analysis directly at the scene (during the seizure phase) by sniffing the network with a tool such as *Wireshark*, trying to retrieve credentials. We could not prove that this method works for retrieving the Fibaro controller credentials, as we did not succeed when implementing it, but we might have had the wrong hardware or configuration, even though we tried several ones. We, however know that it is possible to retrieve the credentials thanks to the method presented in the Network forensics in part 4.2. The other solution would be to seize the controller as well to do the Network and IoT forensic analysis in the lab, allowing the investigator to retrieve the data even if no second warrant cannot be obtained. The latter method should be preferred when trying to recover as much user data as possible, without having to be physically at the scene. The change of location of the devices might however disrupt some variables in the configuration files of the controller for example (the Wi-Fi SSID might change). Nevertheless, we argue that in our case, the user data should be the priority, and thus seizing the Fibaro controller in order to perform an analysis in the forensic lab should be accepted.

The Network forensic method we used also requires us to use the user’s phone, which is an evidence in itself, to recover data. This does not sound forensically foolproof, as using an evidence to collect more data implies more interaction with the device that it could be judged acceptable. One could also argue that even though the method could pose some forensic problems, it allows the investigation to take a step forward by revealing capital information, rendering it more acceptable. Also the fact of using an evidence as a tool to recover data could be considered acceptable because the data collected is also user related.

6 Additional discussion

Another discussion point we would like to confront is the importance of the scenario, or the context in which the crime happens. Indeed, during our work, we ranked the steps and the forensic valuable artifacts that can be retrieved regarding our scenario (the murder case). The ranking was performed regarding the order of the steps and the importance of the data retrieved. But it is important to note that if the scenario is different, the order of steps is going to differ as well. In this part, we come up with a hypothetical approach concerning a new scenario, and the differences that it would lead to.

First, let us consider another scenario, this time not concerning a murder, but a cybercrime. In this new scenario, we imagine an attacker, targeting an individual, using the same environment as described throughout this thesis, with the goals of:

- Stealing the pictures of the user’s phone.
- Access the live stream of the D-Link camera.
- Access the Fibaro controller in order to study when the sensors are triggered, the attacker could thus understand the habits of the users to perform a physical attack in the future. Depending on the devices present at the scene, the attacker could manage some of them.

The scenario would go as follow:

The attacker compromises the router of the house, allowing him to have access to the internal network. The attacker would then gain access to the D-Link camera, thanks to a CSRF vulnerability possibly exploitable in the firmware version 2.13.15. (CVE-2017-7852) [30], allowing him to access the live stream of the camera. The attacker would then compromise the Fibaro controller, using a flaw in the certificate validation during TLS connections over SSH. The attacker could perform a Man in the Middle attack by having his own server, impersonating the original Fibaro server, and forging his own certificates, which are going to be accepted by the Fibaro Home Center. The TLS intercepted requests are also vulnerable to command injection, allowing an attacker to “open an SSH backdoor to Fibaro Home Center Lite” [31]. The attacker can then retrieve the root password by intercepting it and cracking it, thus having root access on the Fibaro controller. These types of attacks would work on the firmware version 4.170 [31]. The Kaspersky CERT also showed in their article [32] that, by using different methods and vectors of attack, it is possible to retrieve useful information such as:

- One’s “password in cached form with an added salt”. [32]
- “The precise coordinates of the home where the device was located”. [32]
- “The geolocation of [one’s] smartphone”. [32]
- One’s “email addresses used for registration in the Fibaro system”. [32]
- “All data about IoT devices (including ones not belonging to Fibaro) that were installed by our colleague at home, with device model, username/password in text form, IP addresses of devices in the internal network, etc.”. [32]

While gaining the superuser rights, which are normally not accessible for a Fibaro system user, they showed that an attacker could possibly manage not only the Fibaro appliances but also any piece of IoT equipment that would have been linked to the Fibaro controller. This could include, as pointed by Kaspersky team “alarms, window/door/gate opening and closing mechanisms, surveillance cameras, heating/air conditioning systems, etc.” [32]; which could indeed lead to serious consequences for people’s security and health.

After compromising the Fibaro controller, the attacker would forge a malicious Android application that looks similar to another app the user has (like the Xiaomi app for instance). He would then perform a phishing campaign, luring the user into thinking he is “Downloading the new version of the app”. The user installs the malware on the phone via the app without knowing. The attacker extracts the pictures from the phone thanks to the created backdoor on the phone.

As we can see from the scenario described above, the storyline is completely different from the one we described along with this thesis, and this implies thinking differently, from a forensic investigator's point of view. The order of the investigation steps would differ to offer the best understanding of the attack and the best probability to recover relevant artifacts. Depending on the hacker modus operandi, his skill set, and whether his attack is still going on at the time of the forensic analysis or not, we could identify different steps to take.

First, the investigator could perform a live Network analysis. In the case the attacker is still conducting his attack during the network acquisition, one could see the network packet traffic going from the D-Link camera to the attacker’s IP, from the Fibaro controller to the attacker's IP and from the Phone to the attacker’s IP. In the case the attacker already ex-filtrated the environment, nothing would be retrieved at this step. Secondly, the investigator could perform a mobile analysis. By recovering files, such as the */data/com.android.vending/databases/localappstate.db* (already present in the thesis, part 4.1.1), the investigator could know when all the applications have been installed, including the malicious app installed on the phone. By extracting this application, and running it in a sandbox, he could understand the behavior of the malware (which files were created, which ones were targetted for extraction, etc.), as well as which connections to the internet were made. In our case, most probably the attacker’s IP. This step would be the malware analysis step, which we did not include in our first case, as the scenario did not feature any malware use. The last step would be to request connection logs from the ISP that happened on the router. It is also important to note that in this scenario, it would probably not be useful to do any IoT forensics, as the attacker did not do any effective change on the Fibaro controller, other than accessing it. The Fibaro controller only displays the last login from the users. So if in order to check the access logs, an investigator connects with the same user account than the attacker, only the investigator connection would be seen in the logs. The unauthorized accesses would more likely be revealed in the Network analysis. Also, depending on where the APK of the malware is stored, the investigator might need to root the phone in order to get access to it, and then analyze it. Indeed, the access to the pictures on the Android device is not subject to root privileges. However, the backdoor installation would most probably need root privileges (to open ports, or to allow the malware to run even after reboot).

The metrics we proposed during our forensic analysis in the first case would have to be modified too. We could, for example, say that, in case of a cybercrime, an example of metrics would be as follow:

- High: Artifacts identifying the third party (with IP address, name, or country for example) that connected without authorization to the target's machines.
- Medium: Artifacts showing that a connection has been made by a third party on a device. Artifacts showing that files have been transferred to another host. Proofs that a piece of software is executing malicious activities.
- Low: Unusual access from the internal network or anything that cannot be linked to any third party.

With these new metrics, and considering the possibly recovered data from the explanation paragraph above, we can rank them in the same manner as we did for the whole thesis.

- The connections from the different devices (Phone, D-Link camera, Fibaro controller) to an external IP address, presumably the one from the attacker, could be ranked as High or Medium, depending on how much it identifies the criminal.
- Network logs, proving that files have been transmitted to another host could be ranked as High or Medium, once again depending on how accurately the investigator can identify the attacker.
- The malware in itself, by showing that it is malicious and by explaining what it is used for and how it works, could be considered as Medium.
- The file */data/com.android.vending/databases/localappstate.db* can be considered as low, as it only shows which applications are installed on the phone and when. As in our example, the user purposely installed the application by being tricked by the attacker.

- The email used to mislead the user into downloading a malware can be considered High or Medium, depending on how the email was crafted, and how it helps the investigator to identify the attacker.

As we could see through this alternative scenario, the order of the steps for the analysis of the devices from the same environment can highly differ. Moreover, the artifacts recovered are different and the metrics must be redefined depending on the case.

By means of our work, including the experimental scenario and investigation, we put in place, as well as the hypothetical scenario described above; we could elaborate on important variables that have to be taken into consideration when performing digital forensics investigation in an IoT environment. Given all this information, we present different flowcharts that summarize our thoughts about the procedure to follow. The following flowchart is a general procedure that an investigator can follow when investigating an IoT based environment.

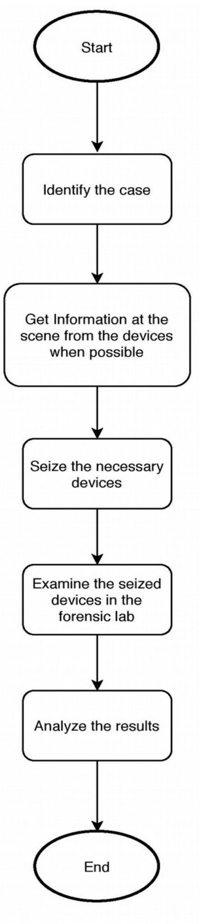

Start
Identify the case
Get Information at the scene from the devices when possible
Seize the necessary devices
Examine the seized devices in the forensic lab
Analyze the results
End

Figure 4: General procedure flowchart

The next flowchart is tailored to be more specific and corresponds thus more to the case we were dealing with throughout this thesis. It includes both types of scenarios covered here and assumes that the IoT environment investigated includes at least a phone and a sensor (camera, motion sensor, smoke detector, etc.).

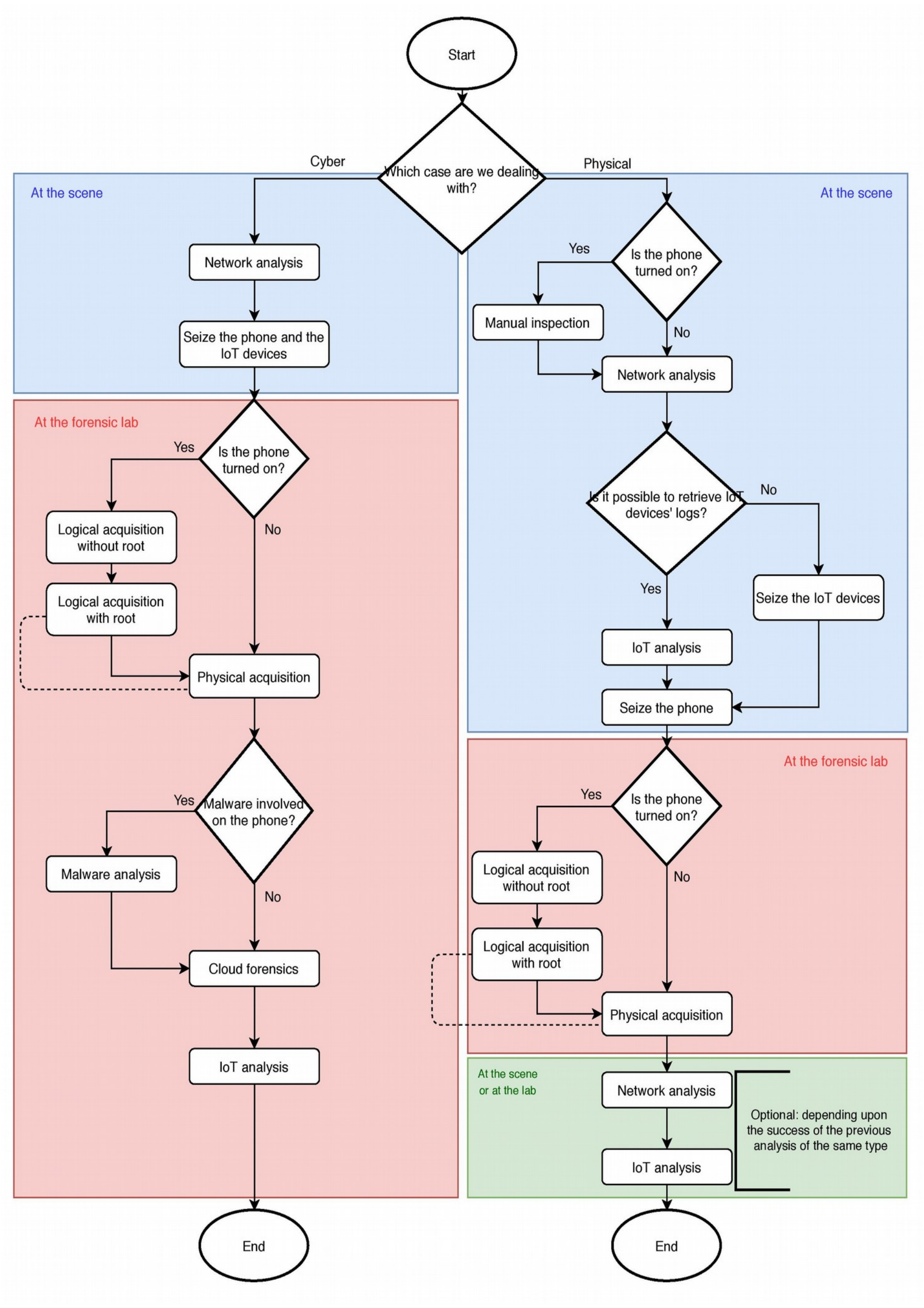


Figure 5: Procedure flowchart for physical or cyber crime

Here we elaborate on the figure presented above.

First, we distinguish two possible types of criminal case: a physical crime, such as a murder or a burglary, and a cyber crime. We assume that most of the criminal activities that could be performed in the context of a smart home environment fall under one of these two categories.

- Physical: As explained during this paper, when a turned on phone is found upon arrival at the scene, an investigator should have a look at the phone and interact manually with it (what we call “Manual inspection” in Figure 5). This would allow him/her to know which applications are installed on the phone, and thus understanding how is it connected with the smart home environment. The investigator could also start gathering evidence by checking potentially interesting locations on the phone, such as emails, pictures or phone calls. Moreover, as explained during our experiment, the manual inspection would allow the investigator to open the application linked to the smart devices and save potential artifacts (pictures or videos from camera for example) in order to keep them for later investigation. By reading the flowchart of Figure 5, we can also see that the seizure of IoT devices would be useful only if it is not possible to retrieve logs from the IoT devices. We are referring here to any type of logs that could give information about the status of the sensors at a specific time, or any other relevant information about IoT devices’ status. If this type of information can be accessed at the scene, it is not necessary to seize the devices. If, however, this information is not accessible, it is advised to seize the devices and to analyze them at the lab later (IoT analysis depicted in the green area of the flowchart). It is also possible, if the scene is easily accessible by the investigators, to go back to the scene and to do the IoT analysis there, after having gathered additional information on how to access the valuable information. The same is applied for the Network analysis. An investigator can perform a Network analysis at the scene upon arrival, and try to gather evidence by sniffing the network with his/her computer for example. If the information gathered are considered satisfactory, then it is not useful to perform a second Network analysis. However, if the information gathered is not satisfactory, one can come back at the scene at a later stage, or perform a network analysis in the lab if devices have been seized, to gather more information thanks to another technique (for example the tcpdump thanks to a rooted phone, as described in section 4.2). The part depicting the different types of acquisition of the phone present in Figure 5 (Logical and Physical) is a summarized version of what has been showed in the *Mobile device evidence processing workflow* by Lee Reiber [10, p.237] found in Appendix 2.

- Cyber: In a cyber criminal case, the priority is not the pictures or the videos that could portrait and thus incriminate a suspect. That is why the manual inspection is not mentioned in this branch of the tree. Because in this type of attack, everything is happening online, and the user is gaining access from the outside network to the internal network, the network analysis is the priority as it could show an investigator connections coming from different networks. The phone acquisitions are performed later, in order to look for evidence on the phone file system for corrupted files or access logs. The malware analysis is an extra step that could be useful in case a malware is suspected to be used. Identifying it and running the malware in a sandbox could be useful for the investigator to trace back the attacker thanks to the possible outbound connection the program is trying to make. It would also allow him/her to have a better understanding on the malware's behavior. The Cloud forensics step could reveal important information, thanks to the cloud and/or internet service providers about possible connections to the home environment. Finally the IoT analysis could potentially contain evidence, user connection logs, or proofs that the devices have been exploited by a third party.

The framework we introduced earlier, written by E. Oriwoh, D. Jazani, G. Epiphaniou, and P. Sant [34], by presenting different aspects that are worth exploring when doing a digital forensics investigation, gave us the opportunity to confront our results to their theories. They, for example, suggest the use of a hypothetical scenario in order to have a more tangible situation and to be able to differentiate which recovered artifact is relevant or not regarding the case. As we implemented this solution for the same reasons, we could indeed show that it is an important issue, because even though more data were retrieved, the ones that would help us solve the case had to be identified and labeled as important. It is also crucial to explain why the important artifacts are relevant in a particular case. Indeed as we could see from the comparison of the two scenarios we imagined, one important artifact within a particular scenario, could be of poor interest in a different one.Also, the framework distinguishes three types of zones where forensic investigation can be done. They more specifically distinguish the Zone 1 as being the Internal network, the Zone 2 as the “Middle” network, including gateways, and the Zone 3, which is the outside or external network. The authors propose two different approaches regarding the investigation of these zones: either in “parallel (all Zones investigated at the same time) or a Zone of greatest priority can be identified [34]”. In our work, we focused exclusively on the Zone 1, as we judged it the most important one. Nevertheless, the investigation of all three zones would provide a better understanding of the environment we emulated.

7 Conclusion and Future work

As our goal was to provide a comprehensive digital forensics analysis of a smart home environment, by retrieving artifacts throughout the whole process of investigation; and by giving importance to the data retrieved. We can conclude that despite the highly heterogeneous nature of the IoT environment that we put in place for our experiment, we could recover numerous forensic artifacts, thanks to several methods of forensic analysis. We presented three types of analysis, Mobile forensics, Network forensics, and IoT forensics. Each of these methods included several sub-steps, that are important to follow in order to retrieve the maximum amount of data. We could also see that not all the devices give out the same amount of data. Google Home for instance, while being a device that is intrinsically linked to a lot of a user's life, did not give as much information as expected. On the other side, the Mobile phone, which is interfacing with all of the devices, holds a lot of user's data and configuration that allows the investigators to have a deep understanding of one's life. The Fibaro controller and sensors, while not necessarily containing a huge amount of data, keep logs of all events happening to the sensors, and thus to the home environment, giving out a lot of details on possible scenarios occurring in a criminal case involving physical interactions.

Future work

As stated throughout this paper, the conditions in which the experiment has taken place are ideal for a forensic investigator, as the events retrace exactly what happened and as the phone has no password set for instance. An improvement of this work would be to test the same environment, with a more complicated scenario or events, and with harder conditions for an investigator. We could then display the differences between what it is possible to recover in the case presented here, and with the more "difficult" case. The user could for example have a passcode on his phone (for his lock-screen), thus rendering some steps more difficult (or even impossible in some cases), as data would be encrypted by using his passcode. The user could also use difficult passwords, thus impossible to crack, that would always be different in all of his devices. He could also never save passwords on any web browser or on his phone, forcing the investigator to type the password every time he wants to access an application, that would render passwords recovery steps difficult for the investigator. The user could also set every application permissions to the minimum, and thus giving the environment a less connected topology, where every device would be (nearly) totally independent from one another and thus reducing the flow of information on the devices and on the network, rending them more difficult to retrieve. Another improvement that should be tackled, in order to have a deeper understanding, and providing more precise guidelines, would be to study the life cycle of the logs. This would be particularly useful for the Fibaro controller. Trying to understand how and where the Fibaro logs are stored, if they are replaced or not after a certain amount of time or after a certain log size have been reached, would be of great importance in the prioritization of the forensic acquisitions, as well as in the understanding of the behavior of Fibaro products. Lastly, an improvement of our work, could be to implement some more aspects of theoretical frameworks. For instance, we could use the “Next Best Thing (NBT) Triage Model” [34] described by the authors, and see its effects on the data collection and analysis. We could for example, after the data have been generated on the sensors, perform an investigation while removing some devices from the scene, and that would be left uninvestigated. This would guide us towards using the NBT Triage Model in order to find potential evidence of the missing device on another “Object of Forensic Interest (OOFI)” [34]. The investigation of all the zones that could potentially contain evidence (internal network, gateways, and cloud) would also be an important improvement within the proposed framework.

References

[1] N. Chavez, “Murder charge dropped in Amazon Echo case,” *CNN*, 02-Dec-2017. [Online]. Available: https://edition.cnn.com/2017/11/30/us/amazon-echo-arkansas-murder-case-dismissed/index.html. [Accessed: 08-Mar-2020].

[2] K. Epstein, “Police think Amazon's Alexa may have information on a fatal stabbing case,” *The Washington Post*, 03-Nov-2019. [Online]. Available: https://www.washingtonpost.com/technology/2019/11/02/police-think-amazons-alexa-may-have-information-fatal-stabbing-case/. [Accessed: 08-Mar-2020].

[3] S. Sathwara, N. Dutta and E. Pricop, "IoT Forensic A digital investigation framework for IoT systems," 2018 10th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Iasi, Romania, 2018, pp. 1-4.

[4] A. Goudbeek, K.-K. R. Choo, and N.-A. Le-Khac, “A Forensic Investigation Framework for Smart Home Environment,” *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.

[5] Q. Do, B. Martini, and K.-K. R. Choo, “Cyber-physical systems information gathering: A smart home case study,” *Computer Networks*, vol. 138, pp. 1–12, 2018.

[6] C. Meffert, D. Clark, I. Baggili, and F. Breitinger, “Forensic State Acquisition from Internet of Things (FSAIoT),” *Proceedings of the 12th International Conference on Availability, Reliability and Security - ARES 17*, 2017.

[7] L. Babun, A.K. Sikder, A. Acar and A.S. Uluagac, “IoTDots: A Digital Forensics Framework for Smart Environments”, 09.2018

[8] S. Tristan, S. Sharma, and R. Gonzalez, “Alexa/Google Home Forensics,” *Studies in Big Data Digital Forensic Education*, pp. 101–121, 2019.

[9] S. Engelhardt, “Smart Speaker Forensics”, *Business/Business Administration*, 2019

[10] L. Reiber, *Mobile forensic investigations: a guide to evidence collection, analysis, and presentation*. New York: McGraw-Hill Education, 2019.

[11] "Bootloader : Android Open Source Project," *Android Open Source Project*. [Online]. Available: https://source.android.com/devices/bootloader. [Accessed: 08-Mar-2020].

[12] C. Hoffman, "The Security Risks of Unlocking Your Android Phone's Bootloader," *How*, 20-Jun-2017. [Online]. Available: https://www.howtogeek.com/142502/htg-explains-the-security-risks-of-unlocking-your-android-phones-bootloader/. [Accessed: 08-Mar-2020]

[13] A. Hoog, *Android forensics: investigation, analysis and mobile security for Google Android*. Amsterdam: Elsevier/Syngress, 2011.

[14] H. Srivastava and S. Tapaswi, "Logical acquisition and analysis of data from android mobile devices," *Information and Computer Security*, vol. 23, no. 5, pp. 450–475, Sep. 2015.

[15] J. Grover, "Android forensics: Automated data collection and reporting from a mobile device," *Digital Investigation*, vol. 10, 2013.

[16] "[All-in-One] Redmi Note 7 (lavender): Unlock Bootloader, Flash TWRP, Root, Flash ROM," *XDA Developers*. [Online]. Available: https://forum.xda-developers.com/redmi-note-7/how-to/one-redmi-note-7-unlock-bootloader-t3890751. [Accessed: 20-Mar-2020].

[17] "Magisk vs SuperSU," *xda*, 20-Mar-2020. [Online]. Available: https://www.xda-developers.com/magisk-vs-supersu/. [Accessed: 20-Mar-2020].

[18] Z. Jovanovic, "Android Forensics Techniques," *International Academy of Design and Technology*, Jan. 2012.

[19] rithvikvibhu, "GHLocalApi Update," *Gist*. [Online]. Available: https://gist.github.com/rithvikvibhu/1a0f4937af957ef6a78453e3be482c1f. [Accessed: 05-Apr-2020].

[20] *Google Home*. [Online]. Available: https://rithvikvibhu.github.io/GHLocalApi/. [Accessed: 06-Apr-2020].

[21] R. Rizal, I. Riadi, and Y. Prayudi, "Network Forensics for Detecting Flooding Attack on Internet of Things (IoT) Device," *International Journal of Cyber-Security and Digital Forensics (IJCSDF) 7(4)*, 2018.

[22] A. Awasthi, H. O. Read, K. Xynos, and I. Sutherland, "Welcome pwn: Almond smart home hub forensics," *Digital Investigation*, vol. 26, 2018.

[23] S. Li, K.-K. R. Choo, Q. Sun, W. J. Buchanan, and J. Cao, "IoT Forensics: Amazon Echo as a Use Case," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6487–6497, 2019.

[24] J. Gregorio, B. Alarcos, and A. Gardel, "Forensic analysis of Nucleus RTOS on MTK smartwatches," *Digital Investigation*, vol. 29, pp. 55–66, 2019.

[25] R. J and G. Z, “Extraction and Forensic Analysis of Artifacts on Wearables,” *International Journal of Forensic Science & Pathology*, pp. 312–318, 2017.

[26] W. Jansen and R. P. Ayers, “Guidelines on cell phone forensics,” 2007.

[27] G. Dorai, S. Houshmand, and S. Aggarwal, “Data Extraction and Forensic Analysis for Smartphone Paired Wearables and IoT Devices,” *Proceedings of the 53rd Hawaii International Conference on System Sciences*, 2020.

[28] R. Ayers, S. Brothers, and W. Jansen, “Guidelines on mobile device forensics,” 2014.

[29] M. B. Mukasey, J. L. Sedgwick, and D. W. Hagy, “Electronic Crime Scene Investigation: A Guide for First Responders, Second Edition,” *NIJ special reports*, 2008.

[30] “Vulnerability Details : CVE-2017-7852,” *CVE*. [Online]. Available: https://www.cvedetails.com/cve/CVE-2017-7852/. [Accessed: 28-Apr-2020].

[31] “Serious flaws found in multiple smart home hubs: Is your device among them?,” *WeLiveSecurity*, 22-Apr-2020. [Online]. Available: https://www.welivesecurity.com/2020/04/22/serious-flaws-smart-home-hubs-is-your-device-among-them/. [Accessed: 28-Apr-2020].

[32] P. Cheremushkin, P. Cheremushkin, Mark, and Kaspersky Lab, “How we hacked our colleague's smart home,” *Securelist English*. [Online]. Available: https://securelist.com/fibaro-smart-home/91416/. [Accessed: 05-May-2020].

[33] M. Harbawi and A. Varol, “An improved digital evidence acquisition model for the Internet of Things forensic I: A theoretical framework,” *2017 5th International Symposium on Digital Forensic and Security (ISDFS)*, 2017.

[34] E. Oriwoh, D. Jazani, G. Epiphaniou, and P. Sant, “Internet of Things Forensics: Challenges and Approaches,” *Proceedings of the 9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2013.

Appendixes

Appendix 1 – Images of the devices used

Google Home mini

Fibaro Home Center Lite

Huawei B535

Fibaro Flood Sensor

D-Link 932LB

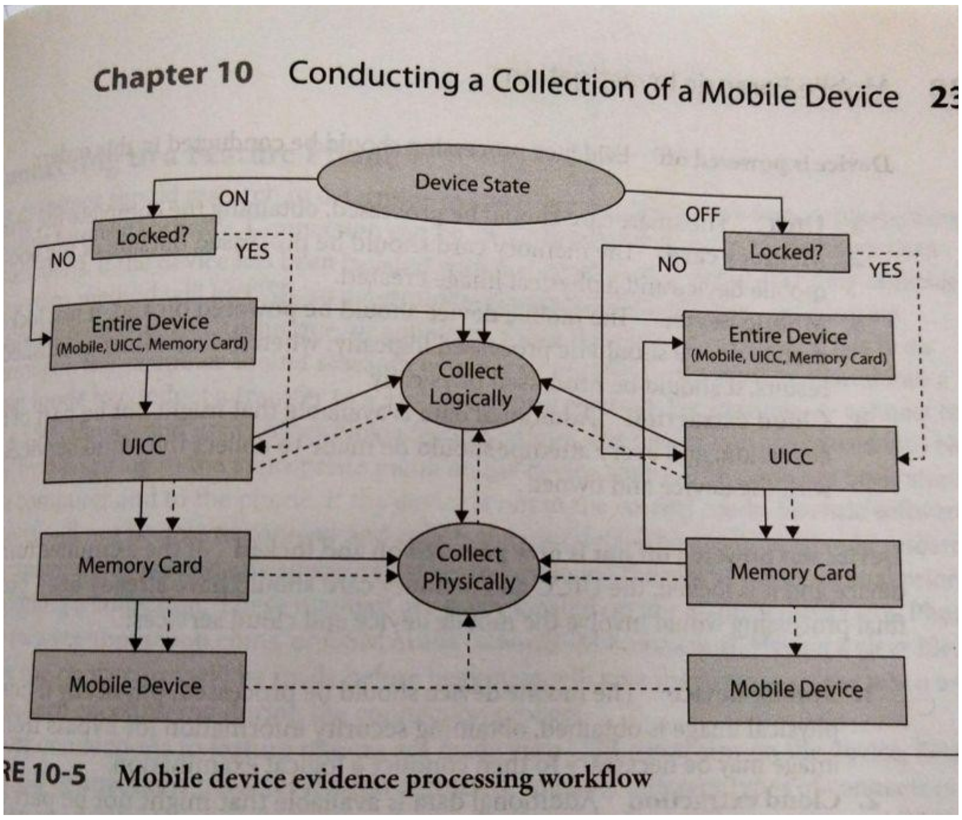
Fibaro Door Sensor

Fibaro Motion Sensor

Mi Home Security Camera 360°

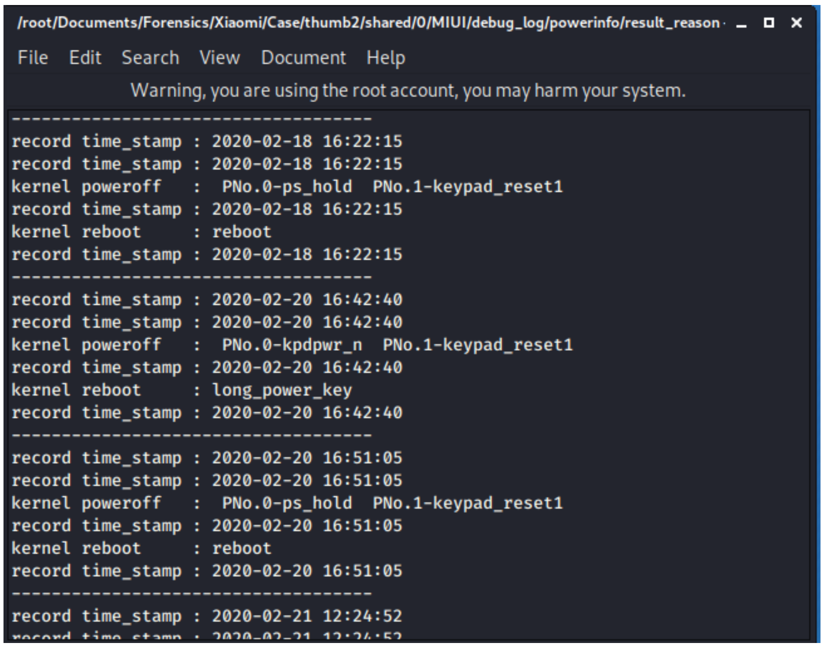
Xiaomi Redmi Note 7

Appendix 2 – Mobile device evidence processing workflow


Chapter 10 Conducting a Collection of a Mobile Device 23

RE 10-5 Mobile device evidence processing workflow

Appendix 3 – Sample of the file *shared/0/MIUI/debug_log/powerinfo/result_reason*


/root/Documents/Forensics/Xiaomi/Case/thumb2/shared/0/MIUI/debug_log/powerinfo/result_reason
File Edit Search View Document Help
Warning, you are using the root account, you may harm your system.

```
------------------------------------
record time_stamp : 2020-02-18 16:22:15
record time_stamp : 2020-02-18 16:22:15
kernel poweroff    :  PNo.0-ps_hold  PNo.1-keypad_reset1
record time_stamp : 2020-02-18 16:22:15
kernel reboot      : reboot
record time_stamp : 2020-02-18 16:22:15
------------------------------------
record time_stamp : 2020-02-20 16:42:40
record time_stamp : 2020-02-20 16:42:40
kernel poweroff    :  PNo.0-kpdpwr_n  PNo.1-keypad_reset1
record time_stamp : 2020-02-20 16:42:40
kernel reboot      : long_power_key
record time_stamp : 2020-02-20 16:42:40
------------------------------------
record time_stamp : 2020-02-20 16:51:05
record time_stamp : 2020-02-20 16:51:05
kernel poweroff    :  PNo.0-ps_hold  PNo.1-keypad_reset1
record time_stamp : 2020-02-20 16:51:05
kernel reboot      : reboot
record time_stamp : 2020-02-20 16:51:05
------------------------------------
record time_stamp : 2020-02-21 12:24:52
```

Appendix 4 – XML file

com.google.android.apps.chromecast.app_preferences.xml

```xml
– <map>
    <int name="appVersion" value="21802090"/>
    <boolean name="history_refresh_needed" value="true"/>
    <boolean name="feed_immediate_refresh_ready" value="false"/>
    <long name="hatsResponseTimeMs" value="1582026455024"/>
    <boolean name="is_child_account" value="false"/>
    <int name="prefs_version" value="1"/>
    <string name="THIRD_PARTY_AGENT_INFO_FILE_CREATION_TIMESTAMP_toto.harrisson@gmail.com">1583755811474</string>
    <boolean name="showHatsResponse" value="false"/>
    <boolean name="hendrixDiscovered" value="true"/>
    <boolean name="assistantDeviceDiscovered" value="true"/>
    <string name="structuresWhereInviteManagerChipWasDismissed">2d9e99b6-f087-45c7-b59d-43a4d4e14f1c</string>
    <string name="setup-salt">7f8880dc-87c5-418d-a689-dcf57c56661b</string>
    <long name="live_card_received_time" value="1581424290322"/>
    <boolean name="content_default_getapps" value="false"/>
    <boolean name="content_whatson_enabled" value="true"/>
    <boolean name="live_card_refresh_needed" value="true"/>
    <boolean name="content_getapps_enabled" value="true"/>
    <string name="live_card_consistency_token"/>
    <boolean name="feed_refresh_needed" value="false"/>
    <boolean name="audioDeviceDiscovered" value="true"/>
  </map>
```

Appendix 5 – Sample of the calendar database

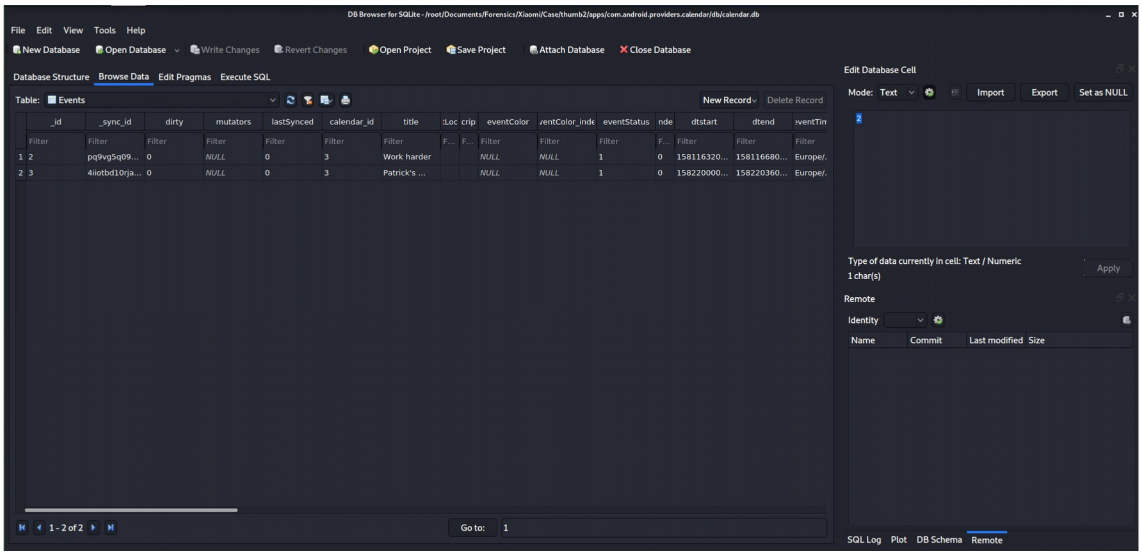

DB Browser for SQLite - /root/Documents/Forensics/Xiaomi/Case/thumb2/apps/com.android.providers.calendar/db/calendar.db

Table: Events

	_id	_sync_id	dirty	mutators	lastSynced	calendar_id	title	:Loc	crip	eventColor	ventColor_inde	eventStatus	nde	dtstart	dtend	eventTir
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	F...	F...	Filter	Filter	Filter	F...	Filter	Filter	Filter
1	2	pq9vg5q09...	0	NULL	0	3	Work harder			NULL	NULL	1	0	158116320...	158116680...	Europe/.
2	3	4iiotbd10rja...	0	NULL	0	3	Patrick's ...			NULL	NULL	1	0	158220000...	158220360...	Europe/.

1 - 2 of 2   Go to: 1

Edit Database Cell

Mode: Text

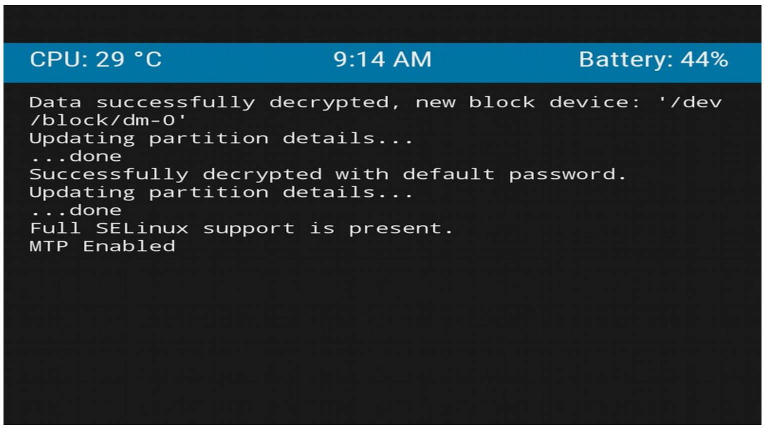
2

Type of data currently in cell: Text / Numeric
1 char(s)

Remote

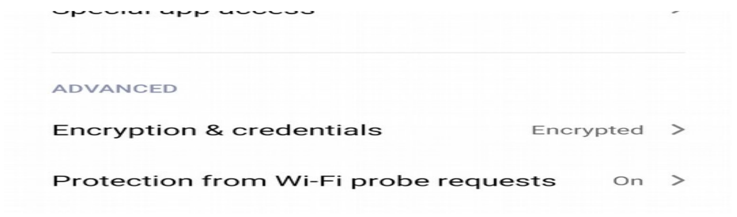
Name	Commit	Last modified	Size

Appendix 6 – Logs from TWRP



```
Data successfully decrypted, new block device: '/dev
/block/dm-0'
Updating partition details...
...done
Successfully decrypted with default password.
Updating partition details...
...done
Full SELinux support is present.
MTP Enabled
```

Appendix 7 – Encryption on Xiaomi Redmi Note 7 on Android 9.x

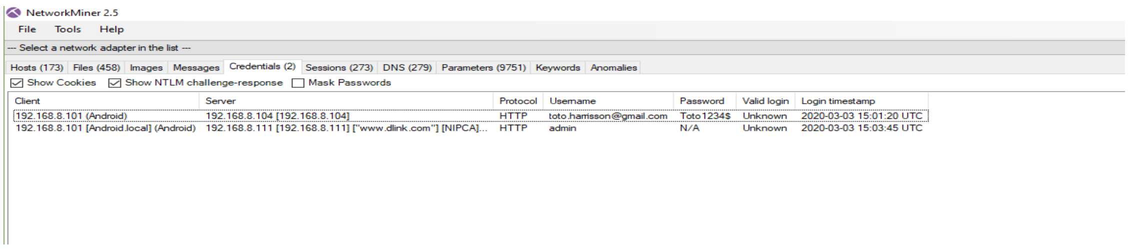


ADVANCED

Encryption & credentials   Encrypted >

Protection from Wi-Fi probe requests   On >

Appendix 8 – Credentials recovered on the network for Fibaro controller thanks to Network Miner

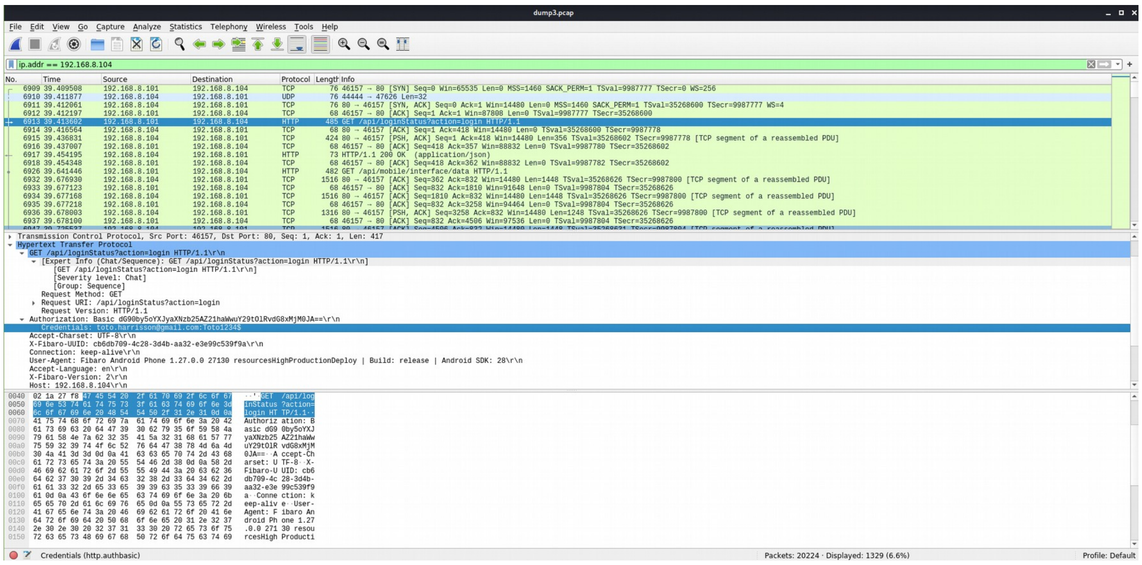


NetworkMiner 2.5
File Tools Help
--- Select a network adapter in the list ---
Hosts (173) Files (458) Images Messages Credentials (2) Sessions (273) DNS (279) Parameters (9751) Keywords Anomalies

[✓] Show Cookies [✓] Show NTLM challenge-response [ ] Mask Passwords

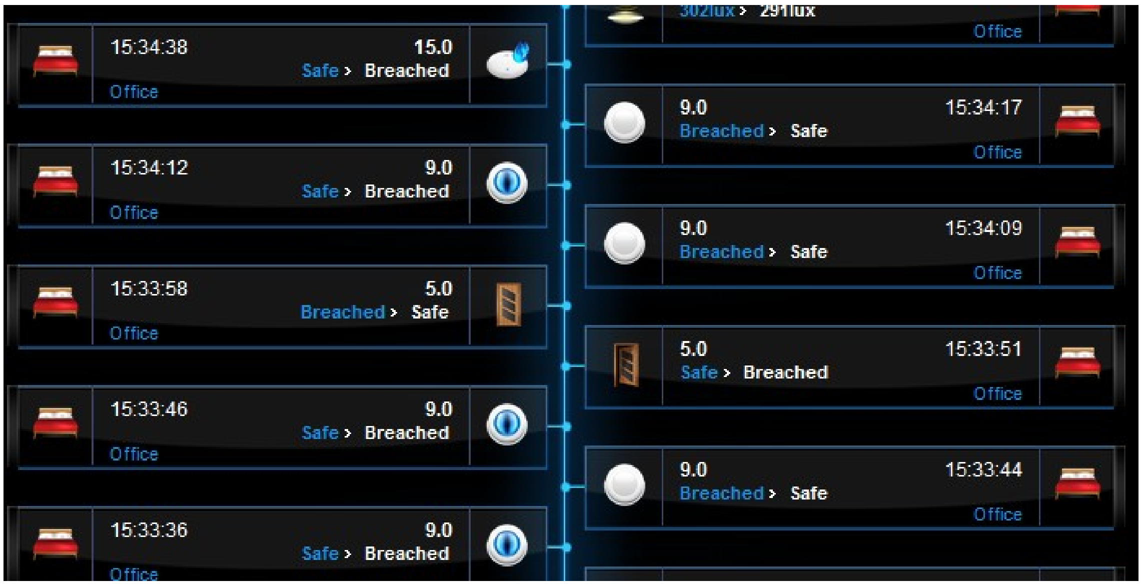
Client	Server	Protocol	Username	Password	Valid login	Login timestamp
192.168.8.101 (Android)	192.168.8.104 [192.168.8.104]	HTTP	toto.harrisson@gmail.com	Toto1234$	Unknown	2020-03-03 15:01:20 UTC
192.168.8.101 [Android.local] (Android)	192.168.8.111 [192.168.8.111] ["www.dlink.com"] [NIPCA]...	HTTP	admin	N/A	Unknown	2020-03-03 15:03:45 UTC

Appendix 9 – Credentials recovered on the network for Fibaro controller thanks to Wireshark

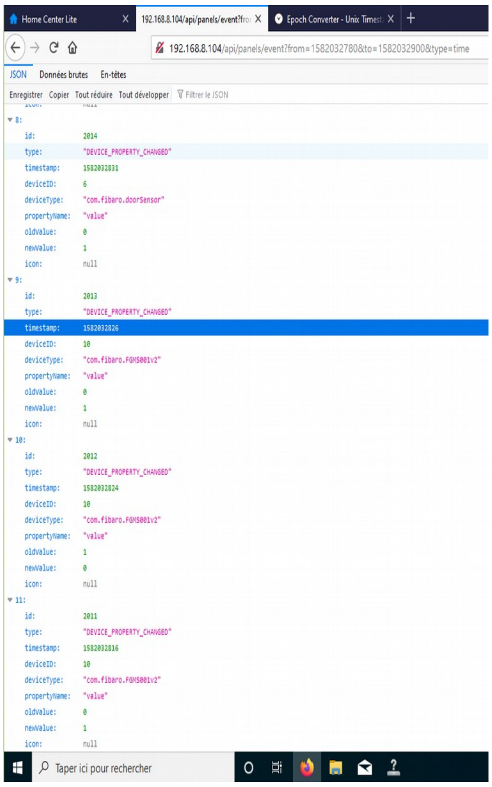

dump3.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
ip.addr == 192.168.8.104

No.  Time       Source         Destination    Protocol Length Info
6909 39.409508  192.168.8.101  192.168.8.104  TCP        76 46157 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=9987777 TSecr=0 WS=256
6910 39.411877  192.168.8.104  192.168.8.101  UDP        76 44444 → 47626 Len=32
6911 39.412061  192.168.8.104  192.168.8.101  TCP        76 80 → 46157 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=35268600 TSecr=9987777 WS=4
6912 39.412197  192.168.8.101  192.168.8.104  TCP        68 46157 → 80 [ACK] Seq=1 Ack=1 Win=87808 Len=0 TSval=9987777 TSecr=35268600
6913 39.413602  192.168.8.101  192.168.8.104  HTTP      485 GET /api/loginStatus?action=login HTTP/1.1
6914 39.416564  192.168.8.104  192.168.8.101  TCP        68 80 → 46157 [ACK] Seq=1 Ack=418 Win=14480 Len=0 TSval=35268600 TSecr=9987778
6915 39.436831  192.168.8.104  192.168.8.101  TCP       424 80 → 46157 [PSH, ACK] Seq=1 Ack=418 Win=14480 Len=356 TSval=35268602 TSecr=9987778 [TCP segment of a reassembled PDU]
6916 39.437007  192.168.8.101  192.168.8.104  TCP        68 46157 → 80 [ACK] Seq=418 Ack=357 Win=88832 Len=0 TSval=9987780 TSecr=35268602
6917 39.454195  192.168.8.104  192.168.8.101  HTTP       73 HTTP/1.1 200 OK  (application/json)
6918 39.454348  192.168.8.101  192.168.8.104  TCP        68 46157 → 80 [ACK] Seq=418 Ack=362 Win=88832 Len=0 TSval=9987782 TSecr=35268602
6926 39.641446  192.168.8.101  192.168.8.104  HTTP      482 GET /api/mobile/interface/data HTTP/1.1
6932 39.676930  192.168.8.104  192.168.8.101  TCP      1516 80 → 46157 [ACK] Seq=362 Ack=832 Win=14480 Len=1448 TSval=35268626 TSecr=9987800 [TCP segment of a reassembled PDU]
6933 39.677123  192.168.8.101  192.168.8.104  TCP        68 46157 → 80 [ACK] Seq=832 Ack=1810 Win=91648 Len=0 TSval=9987804 TSecr=35268626
6934 39.677168  192.168.8.104  192.168.8.101  TCP      1516 80 → 46157 [ACK] Seq=1810 Ack=832 Win=14480 Len=1448 TSval=35268626 TSecr=9987800 [TCP segment of a reassembled PDU]
6935 39.677218  192.168.8.101  192.168.8.104  TCP        68 46157 → 80 [ACK] Seq=832 Ack=3258 Win=94464 Len=0 TSval=9987804 TSecr=35268626
6936 39.678003  192.168.8.104  192.168.8.101  TCP      1316 80 → 46157 [PSH, ACK] Seq=3258 Ack=832 Win=14480 Len=1248 TSval=35268626 TSecr=9987800 [TCP segment of a reassembled PDU]
6937 39.678100  192.168.8.101  192.168.8.104  TCP        68 46157 → 80 [ACK] Seq=832 Ack=4506 Win=97536 Len=0 TSval=9987804 TSecr=35268626

▸ Transmission Control Protocol, Src Port: 46157, Dst Port: 80, Seq: 1, Ack: 1, Len: 417
▾ Hypertext Transfer Protocol
  ▾ GET /api/loginStatus?action=login HTTP/1.1\r\n
    ▾ [Expert Info (Chat/Sequence): GET /api/loginStatus?action=login HTTP/1.1\r\n]
        [GET /api/loginStatus?action=login HTTP/1.1\r\n]
        [Severity level: Chat]
        [Group: Sequence]
      Request Method: GET
    ▸ Request URI: /api/loginStatus?action=login
      Request Version: HTTP/1.1
  ▾ Authorization: Basic dG90by5oYXJyaXNzb25AZ21haWwuY29tOlRvdG8xMjM0JA==\r\n
      Credentials: toto.harrisson@gmail.com:Toto1234$
    Accept-Charset: UTF-8\r\n
    X-Fibaro-UUID: cb6db709-4c28-3d4b-aa32-e3e99c539f9a\r\n
    Connection: keep-alive\r\n
    User-Agent: Fibaro Android Phone 1.27.0.0 27130 resourcesHighProductionDeploy | Build: release | Android SDK: 28\r\n
    Accept-Language: en\r\n
    X-Fibaro-Version: 2\r\n
    Host: 192.168.8.104\r\n

0040  02 1a 27 f8 47 45 54 20  2f 61 70 69 2f 6c 6f 67   ··'·GET  /api/log
0050  69 6e 53 74 61 74 75 73  3f 61 63 74 69 6f 6e 3d   inStatus ?action=
0060  6c 6f 67 69 6e 20 48 54  54 50 2f 31 2e 31 0d 0a   login HT TP/1.1··
0070  41 75 74 68 6f 72 69 7a  61 74 69 6f 6e 3a 20 42   Authoriz ation: B
0080  61 73 69 63 20 64 47 39  30 62 79 35 6f 59 58 4a   asic dG9 0by5oYXJ
0090  79 61 58 4e 7a 62 32 35  41 5a 32 31 68 61 57 77   yaXNzb25 AZ21haWw
00a0  75 59 32 39 74 4f 6c 52  76 64 47 38 78 4d 6a 4d   uY29tOlR vdG8xMjM
00b0  30 4a 41 3d 3d 0d 0a 41  63 63 65 70 74 2d 43 68   0JA==··A ccept-Ch
00c0  61 72 73 65 74 3a 20 55  54 46 2d 38 0d 0a 58 2d   arset: U TF-8··X-
00d0  46 69 62 61 72 6f 2d 55  55 49 44 3a 20 63 62 36   Fibaro-U UID: cb6
00e0  64 62 37 30 39 2d 34 63  32 38 2d 33 64 34 62 2d   db709-4c 28-3d4b-
00f0  61 61 33 32 2d 65 33 65  39 39 63 35 33 39 66 39   aa32-e3e 99c539f9
0100  61 0d 0a 43 6f 6e 6e 65  63 74 69 6f 6e 3a 20 6b   a··Conne ction: k
0110  65 65 70 2d 61 6c 69 76  65 0d 0a 55 73 65 72 2d   eep-aliv e··User-
0120  41 67 65 6e 74 3a 20 46  69 62 61 72 6f 20 41 6e   Agent: F ibaro An
0130  64 72 6f 69 64 20 50 68  6f 6e 65 20 31 2e 32 37   droid Ph one 1.27
0140  2e 30 2e 30 20 32 37 31  33 30 20 72 65 73 6f 75   .0.0 271 30 resou
0150  72 63 65 73 48 69 67 68  50 72 6f 64 75 63 74 69   rcesHigh Producti

Credentials (http.authbasic)        Packets: 20224 · Displayed: 1329 (6.6%)        Profile: Default
```

Appendix 10 – Panel Events from the web interface of the Fibaro controller


302lux > 291lux
Office
15:34:38 15.0 Safe > Breached Office
9.0 15:34:17 Breached > Safe Office
15:34:12 9.0 Safe > Breached Office
9.0 15:34:09 Breached > Safe Office
15:33:58 5.0 Breached > Safe Office
5.0 15:33:51 Safe > Breached Office
15:33:46 9.0 Safe > Breached Office
9.0 15:33:44 Breached > Safe Office
15:33:36 9.0 Safe > Breached Office

Appendix 11 – Events recovered thanks to an HTTP request



Home Center Lite | 192.168.8.104/api/panels/event?fro | Epoch Converter - Unix Timest

192.168.8.104/api/panels/event?from=1582032780&to=1582032900&type=time

JSON Données brutes En-têtes

Enregistrer Copier Tout réduire Tout développer Filtrer le JSON

```
▼ 8:
    id:            2014
    type:          "DEVICE_PROPERTY_CHANGED"
    timestamp:     1582032831
    deviceID:      6
    deviceType:    "com.fibaro.doorSensor"
    propertyName:  "value"
    oldValue:      0
    newValue:      1
    icon:          null
▼ 9:
    id:            2013
    type:          "DEVICE_PROPERTY_CHANGED"
    timestamp:     1582032826
    deviceID:      10
    deviceType:    "com.fibaro.FGMS001v2"
    propertyName:  "value"
    oldValue:      0
    newValue:      1
    icon:          null
▼ 10:
    id:            2012
    type:          "DEVICE_PROPERTY_CHANGED"
    timestamp:     1582032824
    deviceID:      10
    deviceType:    "com.fibaro.FGMS001v2"
    propertyName:  "value"
    oldValue:      1
    newValue:      0
    icon:          null
▼ 11:
    id:            2011
    type:          "DEVICE_PROPERTY_CHANGED"
    timestamp:     1582032816
    deviceID:      10
    deviceType:    "com.fibaro.FGMS001v2"
    propertyName:  "value"
    oldValue:      0
    newValue:      1
    icon:          null
```