TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technologies

Faisal Sumaila 194223IVCM

EXTRACTION AND ANALYSIS OF FORENSIC ARTIFACTS FROM AUTOMOTIVE MAINTENANCE APPLICATIONS

Master's thesis

Supervisor**:** Dr. Hayretdin Bahsi

Tallinn 2021

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: Faisal Sumaila

14.05.2021

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia Teaduskond

Faisal Sumaila 194223IVCM

AUTOTÖÖSTUSE HOOLDUSRAKENDUSTEST KOHTUEKSPERTIISIKS VAJALIKE JÄLGEDE EKSTRAHEERIMINE JA ANALÜÜS

Magistritöö

Juhendaja: Dr. Hayretdin Bahsi

Tallinn 2021

Abstract

Automotive applications are part of the growing innovations in IoT devices gaining popularity. They collect data from the ECU via the CAN when connected to the OBD-II port of the vehicle and transmit it to their corresponding application on smartphones via Bluetooth or Wi-Fi connection, thereby making them a potential source of forensic artifacts. Consequently, we are looking to explore which kind of data can be extracted from automotive applications and whether this data is of any evidentiary value. We experimented using three automotive applications; Zus Smart Vehicle Health Monitor Mini, GoFar, and Veepeak OBDCheck Bluetooth OBD2 Scanner. These were tested on three cars; Volkswagen Golf (2012), Mini Cooper (2018), and Toyota Corolla (2020). We show the possible artifacts obtained and how the data extraction is done. We also offer which extraction method is a priority for an investigation to get more artifacts, and present a process diagram to follow when collecting data from automotive applications.

This thesis is written in English and is 87 pages long, including 6 chapters, 17 figures, and 11 tables.

Annotatsioon

Sõidukite diagnostika rakendused on osa IoT-seadmete kasvavast populaarsusest. Need seadmed on ühendatud sõiduki OBD-II liidesega ja koguvad andmeid sõiduki ECU-st CAN-i kaudu ning edastavad need vastavasse nutitelefoni diagnostika rakendusse Bluetoothi või Wi-Fi kaudu, muutes need potentsiaalseteks kohtuekspertiisi esemeteks. Antud töös uurime, milliseid andmeid saab sõiduki diagnostika rakendustest eraldada ja kas neil on ka tõendusjõud. Töös katsetasime kolme diagnostika rakendust, Zus Smart Vehicle Health Monitor Mini, GoFar ja Veepeak OBDCheck Bluetooth OBD2 Scanner. Valitud rakendusi katsetati kolmes sõidukis, Volkswagen Golf (2012), Mini Cooper (2018) ja Toyota Corolla (2020). Töös näitame võimalikke leitud kohtuekspertiisi esemeid ja nende kogumise protsessi. Samuti soovitame nende prioriteetset kogumise meetodit, et leida võimalikult palju kohtuekspertiisi esemeid ning esitame soovitusliku protsessi diagrammi, mille järgi diagnostika rakendustest andmeid koguda.

Antud magistritöö on kirjutatud inglise keeles ja koosneb 87 leheküljest, 6 peatükist, 17 joonisest ja 11 tabelist.

Acknowledgments

I would like to sincerely thank my supervisor Dr. Hayretdin Bahsi, for his academic guidance, constructive feedback and for allowing me to use his car for this thesis. I would also like to thank my colleague Oluwatosin Samuel Soremekun who was always with me during the test drives for the data collection of this study. Lastly, I dedicate this thesis to my wife, Roselyn Mullet-Sumaila, and my children Fareeda and Farhaan for being my inspiration and motivation to complete this thesis work.

List of abbreviations and terms

GPS	Global Position System
V2I	Vehicle to Infrastructure
V2V	Vehicle to Vehicle
V2C	Vehicle to Cloud
V2X	Vehicle to Everything
IoT	Internet of Things
VIN	Vehicle Identification Number
OBD	On-Board Diagnostics
EOBD	European On-Board Diagnostics
iOS	iPhone Operating System
ECU	Engine Control Unit
CAN	Controller Area Network
ISO	International Organization for Standardization
Wi-Fi	Wireless Fidelity
RAM	Random Access Memory
UFED	Universal Forensic Extraction Device
NIST	National Institute of Standards and Technology
JTAG	Joint Test Action Group
OS	Operating System
ADB	Android Debug Bridge
USB	Universal Serial Bus
RPM	Revolutions per Minute
XML	Extensible Markup Language
TWRP	TeamWin Recovery Project
TCP	Transmission Control Protocol

Table of Contents

List of Figures

List of Tables

1. Introduction

The usage of mobile devices while driving cars has become prevalent, with a study in 2017 showing that 88 out of 100 drivers make use of their phones for different reasons such as sending or reading messages, browsing social media applications, looking at GPS maps, and playing music, just to name a few [1]. Automotive applications are software that serve as a bridge between vehicles, the internet, and smart devices such as smartphones. These applications provide drivers with a notification or information about the integral parts and health conditions of their vehicles and other features so that measures are taken on time to prevent malfunctions. The automotive application developing industry has been projected to reach a value of $166 billion in the next four years as of this time [2].

Automotive applications work in diverse categories: "V2I (Vehicle to Infrastructure), V2V (Vehicle to Vehicle), V2C (Vehicle to Cloud), and V2X (Vehicle to Everything)" [2]. The applications to be analyzed in this thesis work are in the group of V2C (Vehicle to Cloud); however, emphasis will be placed on exploring the smartphones on which these applications have been installed to ascertain what valuable data could be stored in the smartphones that use these automotive applications. Some information such as the GPS coordinates, vehicle information, vehicle engine, and internal information could be extracted from these applications on the smart devices. This thesis work will explore the possibility of the kind of information that can be obtained from 3 automotive applications that have been tested on three different brands of cars.

Automotive applications are part of relatively new software in terms of IoT device interactions with vehicles. The kind of data stored on the smartphones by these applications could prove vital for traffic investigations, vehicle insurance investigations, or criminal investigations. Most mobile forensic studies have focused on messaging and social media applications. Since these automotive maintenance applications provide the driver with a warning in the event of an impending breakdown of engine parts, track the trip locations undertaken, and maintain fuel logs, they could be a rich source of forensic data. The need for a study on what kind of data of evidentiary value is extractable from these applications is a valuable endeavor of which this thesis work will delve into.

Additionally, there is a possibility that these applications communicate differently with the car model. So, it is worth exploring the difference in the data obtained in various car brands to ascertain this information. Automotive applications are relatively new, and as mentioned earlier in this paper, the industry is growing exponentially. Investigators may not currently know some essential details about these applications. They, therefore, may not know which kinds of data can be collected from these applications and how this data can be collected. Thus, our work provides investigators with a guideline to follow to obtain possible artifacts and how to analyze them. Furthermore, in a world with many mobile apps, and IoT devices, triage analysis could be used to help reduce the burden building up on forensic investigators as sending each device to the lab is not feasible in terms of additional workload that may not end up with necessary results. As mentioned in [41], fewer forensic analysts in law enforcements worldwide, has caused a delay in the retrieval of evidence from digital devices which is leading to an accumulation of devices that need to be analyzed. In order to make this process more efficient, there needs to be a way to prioritize the process of evidence collection and analyses. This is where the importance of triage comes into play as mentioned in [42] [43]. Generally, studies dealing with artifact analysis of IoT devices or mobile applications do not discuss much about the triage step. This is one of the gaps our work aims to fill.

Our study will be adhering to the principles and core processes of digital forensics during the work time frame since we aim to retrieve forensic data. Digital forensics as a science that encompasses the phases of identifying, preserving, recovering, analyzing, and presenting facts about evidence found on computers or digital storage media [3, p.25].

The first stage of the digital forensics process, identifying, is a crucial stage that entails identifying artifacts from various devices present to forensic investigators at the scene of a crime [3, p.25]. The next stage after that is the preservation of the identified artifacts to safeguard the integrity of the artifacts. This phase involves the extraction of the data from one device to another, which is a crucial stage in forensic investigation. An alteration in the extracted artifacts will render it inadmissible for a case [3, p.25]. Next, an analysis of the retrieved artifacts is undertaken to identify evidence that points to the perpetrator of the crime and is reported and presented as evidence to a court in the adjudication of a case [3, p.25].

1.1 Research Questions

Previous works [4] [5] [6] by various researchers focused on the extraction of forensic data from messaging and social media applications on smartphones. Other research, such as [7], focused on retrieving data from infotainment systems that interconnect with smartphones. Our work will focus on automotive maintenance applications and will aim to answer the following research questions:

- What kind of vehicle and user data can be extracted from the automotive applications installed on the smartphone, and whether these extracted data are of any evidentiary value to forensic investigations?
- Are there any differences in the data extracted per car model or brand?

1.2 Goal and outcome

The outcome of our research will be to determine the information extracted from these applications are of evidentiary value and whether there are differences in the data extracted in terms of the car's model or brand. Another outcome of our work will be the creation of a guideline that can be followed by digital investigators or forensic analysts to know what kinds of valuable data can be obtained from automotive applications and how/where to extract this information. The range of data our research is interested in finding include:

- GPS coordinates
- Vehicle information details such as VIN, speed, etc.
- Fuel consumption logs
- Mileage information
- Vehicle internal and engine health information

GPS coordinates pinpoint the location that the vehicle either started from or last stopped. This information is important for investigators during a case which involves the need to track the car’s movement to obtain detailed information about both the car and the person in relation to the case. The automotive application in essence provides information that connects an “external object (car)” to a person. This information can be used to help defend or incriminate a suspect in a criminal case. Also, vehicle information details such as the VIN of the car is important to

a forensic investigation. The VIN is the unique vehicle number that identifies the car's make and model. This information in our optimism, can provide a level of attribution that the investigator can pin the mobile phone user to the vehicle. The vehicle's speed on a trip can be useful in a traffic incident where cameras are not available for an investigator to analyze. We are aware that other means such as using a data retrieval tool[1] can be used to access the Event Data Recorder (EDR) by which investigators can obtain information such as the speed history that was saved by the car before the incident. Using such a tool may require taking the car to the lab for the analysis, and in cases where time is of critical importance, our study may provide another possible way to look for this information directly from the mobile phone prior to the detailed analysis of the car. Other data such as fuel consumption and mileage logs when obtained by an investigator can help back up the previously mentioned artifacts. The fuel and mileage information per trip can also help clear some doubts when a forensic analyst presents the distance traveled by the car which matches with the amount of fuel that was consumed on that same trip. Lastly, the vehicle's engine and other internal health information provide details on the state of the car during the trip. This information could be used for instance by vehicle insurance investigators when trying to find data on the state of the car at a particular point in time. Based on the aforementioned, we wish to identify the artifacts stored on the smartphone that has the automotive applications installed which will be valuable for forensic investigations.

1.3 Scope

Our work will focus on the GoFar Automotive Application, Veepeak OBDCheck Bluetooth OBD2 Scanner and Zus Smart Vehicle Health Monitor Mini (Gen 4). These applications were chosen because of their availability to us since some of the automotive applications can only be obtained based on location. Furthermore, the number of users of the applications was considered so that our research findings will make an impact. Zus application has over 1 million units installed worldwide [8] while GoFar has been installed by drivers in over 50 countries and logging in over 95 million kilometers so far [9]. Additionally, we selected applications that use the On-Board Diagnostic II port of the car to communicate with the application, while the connection between the smartphone and the application will be via Bluetooth connection. Furthermore, we will not use volatile memory forensic methods in our work because we are

[1] https://boschcdrtool.com/

not expecting to find the potential artifacts, we have enumerated in section 1.2 on RAM. As it is widely known, passwords or credential values would be valuable to get from live memory but in our case, this is not needed. Lastly, our work targets the smartphone that has these applications installed on it and therefore extraction of the data will be performed using mobile forensic methods. We do not cover cloud forensics in our study as the automotive applications do not provide us with an interface to directly access the data on the cloud.

1.3.1 Contribution and Novelty

In the course of searching for the literature that is discussed below, we could not find research works done on extracting data from automotive applications. This provides our work with a chance to contribute to the field of forensics with the disclosure of our results and findings. Our findings can serve as another option for forensic examiners to acquire evidence about a vehicle by examining the smartphone and more specifically, the automotive application installed. As other researches on mobile forensics and applications based on our literature review do not analyze triage activities in detail, our study looks into the triage issues and provides guidelines.

1.3.2 Limitations

As a limitation, the research does not analyze security vulnerabilities of automotive applications. We also limited our work to 3 specific automotive applications and 3 different vehicle brands. We used 3 cars in our analysis to ascertain if there are differences, which could lead to further investigations in the future and also due to the timeframe of this work we are not looking at more vehicles and applications.

1.4 Ethical Concerns

Although the aim of our study does not cover vulnerability research on these automotive applications, any vulnerabilities that we may come across in these applications during this thesis work will be appropriately reported to stakeholders and manufacturers. Additionally, user information of which exposure will be a privacy concern, will be censored by blurring or covering the sensitive information with a black colored textbox accordingly in this paper.

1.5 Thesis Outline

Our study will be presented in the layout structure below. Each chapter summarizes the contents to be found:

- Chapter 1 – Introduction of our topic, the relevant questions it aims to answer, rationale behind why this topic was chosen, scope and the limitation of the thesis are stated.
- Chapter 2 – A brief explanation of the theoretical background, definition of forensic terminologies, and related literature or works is given here.
- Chapter 3 – The methodology used for our work, our experimental setup, and the process flow to be used for the data collection.
- Chapter 4 – Presentation of the results of the data extraction methods we used.
- Chapter 5 – Discussion and analysis of the extracted data from our work.
- Chapter 6 – The conclusion of the thesis work and the proposed future work that can be done are indicated here.

2. Background

This section of the thesis discusses the theoretical and technical definitions that will be used throughout this work to give a clear comprehension of our work. Additionally, the chapter in its subsections highlights previous works done in the field of mobile and android forensics. Lastly, the extraction of forensic artifacts in other areas is outlined to draw parallels between these works and how some of them relate to our study.

2.1 Automotive applications

Maintenance or automotive applications as they are also called are software that comes with an OBD-II dongle to connect to the car's OBD-II female port. Their primary goal is to gather the vehicle's internal diagnostics and present this information to the driver via the corresponding mobile application that is installed on a smartphone. These applications are becoming popular as drivers are using them to track and economize fuel consumption, driving patterns or behaviors to tell of their speeding or sharp acceleration. This information helps in proactively fixing an issue in the car before it becomes a major problem that breaks the car down. There are several automotive applications on the market, some of which are GoFar, Zus Smart Vehicle Health Monitor, Veepeak OBDCheck, Scan Master to name a few. These applications are compatible with both Android and iOS systems. The development of the OBD standards started in the USA as a means to monitor the vehicle engine functions, gas emissions and to be used to report issues or failures in the electrical parts of the car [10]. This was the first standard which is now obsolete and giving rise to the current version which is the OBD-II standard. The OBD-II standard came into use in 1996 and so vehicles manufactured from this date come equipped with this port [11]. In the EU, the EOBD standard which also uses the same interface and connectors as the OBD-II is required in diesel manufactured cars from 2003 and likewise in gasoline cars manufactured from 2001 [10].

Today's vehicles come with an ECU that is responsible for the efficiency in the way the engine of the car functions based on the data it collects from the sensors in the car [12]. In essence, the ECU can for example tweak the fuel consumption based on the data it has collected from the vehicle sensor. OBD-II applications use protocol standards to communicate with the ECU. There are about 5 protocols for the OBD-II standard; SAE J1850 PWM (Pulse-Width

Modulation), SAE J1850 VPW (Variable Pulse Width), ISO 9141–2, ISO 14230 KWP2000 (Keyword Protocol 2000), and ISO 15765 CAN with the differentiating factor among these being their electrical pin positions [10]. The communication protocol standard widely used is the ISO 15765-4 - CAN which has been mandated to be used in cars manufactured in the USA since 2008 [13]. The ISO 15765-4 - CAN protocol is seen as the standard that will be used by all vehicles for OBD-II communications with the ECU as it can reach a speed of 1 Mbps [11]. We do not explain in detail how each of these protocol standards functions or communicates with ECU, however, we will explain how the OBD-II applications work in general with their corresponding mobile applications when connected with the car.

As mentioned in the opening paragraph of this section, the OBD-II standard communicates with the ECU through the OBD-II port of the vehicle. The automotive applications come with a dongle that is connected to the vehicle's OBD-II port. A mobile application either from the automotive application manufacturer or a compatible application is downloaded on the smartphone. The smartphone and the OBD-II dongle are then connected via a Bluetooth or wireless connection. If the Bluetooth or wireless connection is not turned on, an error is reported to the user on the phone and a prompt to turn either service on. The automotive applications we used in our thesis all use Bluetooth connections. Once the connection is established successfully between the dongle and the phone, an existing user must log in to the application otherwise, a new account needs to be created for new users. The user then configures the application by providing basic information about the car such as the make, model, year, current odometer reading and engine capacity. The application will now begin to monitor and transmit vehicle information such as acceleration, speed, GPS as well as any trouble codes that point to a faulty condition in the car. This information is presented to the driver via the application's dashboard on the phone. We present a flow diagram as shown in Figure 1 to outline how the application connects to the smartphone via Bluetooth or Wi-Fi.

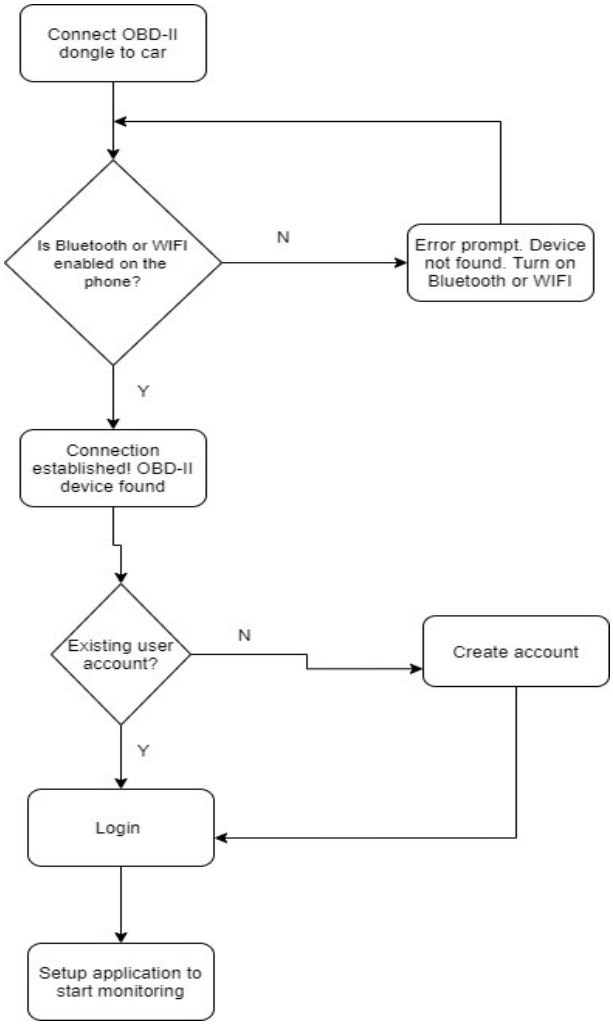


Figure 1. OBD-II basic connection overview

2.2 Theoretical and Technical Background

This area of our study briefly enumerates some key terms used in cybersecurity and digital forensics, and it aims to provide information that guides non-technical or non-cyber security minds to have a better grasp of our paper. These terms will be used predominantly in the subsequent chapters of our work, and it is worthwhile to explain them to clear any ambiguity of the said.

2.2.1 Volatile Memory

Volatile memory of which the most common type known, RAM, is the memory used by computers and mobile devices to keep applications and the operating system running while

these devices are turned on and functioning, and in essence, information in volatile memory is lost when the device is turned off.

2.2.2 Non-Volatile Memory

Non-Volatile memory denotes permanent memory storage. The information stored in this area of memory is neither lost nor changed when the device is turned off.

2.2.3 Forensic Artifacts

In this paper, we used the term forensic artifacts to denote data that can be extracted from the applications tested which have value as evidence to be used in an investigative case or court of law.

2.2.4 Triage

This is a process in Digital Forensics investigations where the activities done to obtain data of evidentiary value are classified on which is to be done first based on their priority [14]. This in turn helps investigators know which devices or where on the device they will need to look upon arriving at a crime scene, so that important information can still be extracted at the scene before detailed analysis is done in the laboratory.

2.3 Related Works and Literature

We present in this section of our work, some past works in the domain of mobile and android forensics. Furthermore, our focus on these works was on the acquisition methods they used and how we can utilize these methods in our study. Additionally, we provide an overview of infotainment and cloud forensics to which are systems that work with mobile devices, to gain some insights from these areas and how applicable they can be to our work. Lastly, the literature review that we present shows that although various mobile and android applications have been forensically examined, the automotive applications have not yet been researched and is an area that needs to be investigated as we are doing.

2.3.1 Mobile Forensics related works

Domingues et.al in their work [15], examined the evidentiary data that is made in Windows 10 individual machines when an application called “Your Phone” system is installed on the computer. They were able to find valuable information such as messages that are 30 days old as well as the database that stores the phone contacts of connected smartphones, which is obtainable from the SQLite 3 database in the Windows 10 machine, and also indicated that data obtained from the Your Phone system can become valuable forensic artifacts in the event that the associated smartphones are not available to be probed into for evidence [15]. This work used the Sleuth Kit Autopsy tool to analyze the extracted data which is a tool that is to be used in our work although they customized it with some python scripts. Their research also analyzed an android version of the application just like the automotive applications to be studied for our work.

Barmpatsalou et.al in their research looked into the trends of mobile forensics, both existing and impending, to find some areas of improvement in the mobile forensic domain from the past 7 years as at when the article was written in 2018 [16]. Their study also makes reference to the investigation process of mobile forensics derived from this work [17]. Their study also enumerates that while every stage during a forensic study for mobile gadgets is important, there is some sort of imbalance in devotion to respective parts in terms of research, and this they said is because each phase’s importance varies; “This is due to the fact that not every stage is equally important for all fields” [16]. The methods of acquisition stated in their work is important to our research as the extraction of the artifacts from the smartphone will include both physical and logical data acquisition methods. The acquisition methods used in their research is shown in Figure 2 below:

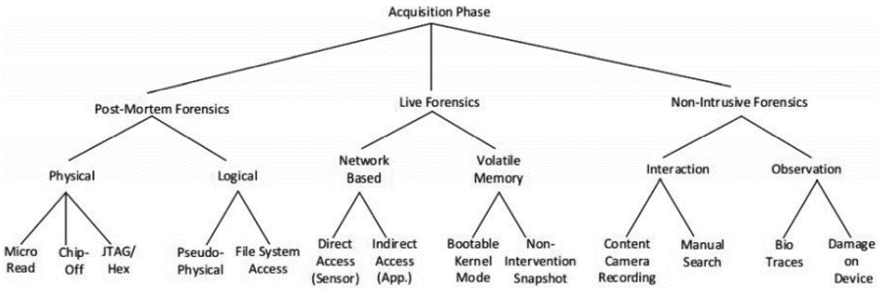


Figure 2. Detailed Acquisition Phase [16]

In another related work [18], the researchers provided insights into how to spot software that hides information or vault applications from the iOS App store. The study further discusses an application that was developed by the researchers that is able to identify the vault software and also perform an extraction of the data concealed in the application from an iOS device. Additionally, they also discussed how forensic tools like Cellebrite UFED can perform logical and physical extraction from a device. The researchers mentioned that both the logical and physical extraction methods are widely chosen, and logical acquisition allows the retrieval of user information from the system files with the exception of deleted data [18]. This study was able to recover artifacts extensively, however, the researchers used Cellebrite UFED which is an expensive commercial tool that is not available for our work. We will use open-source tools in our data extraction to obtain as much information as we can from the mobile application.

Zhang et al in the work [19] presented studies and findings on eighteen Android vault applications which they reverse engineered while analyzing the artifacts formed in them Their findings showed that over 60 percent of the software garbled their code; 27 percent prevented the process of reversing due to their inherent libraries on the concealed data; about 55 percent were obtained without having privileged root access or control to the device; just over 30 percent stored images unencrypted; 44 percent did not also encrypt videos; almost 39 percent saved passwords in plaintext and not encrypted and 27 percent had password file exchanged with one that was made by the researchers [19]. Their work focused on the security aspect of Android applications and revealed that compromise can occur within the application thereby providing a way for forensic experts to obtain valuable information in the course of an investigation. Our work will be looking to extract data from automotive applications on the smartphone in both rooted and unrooted modes, and unlike their study, our analysis will not be made from a security perspective of the applications.

2.3.2 Triage related works

In their work [43], they proposed a triage model for carrying out a thorough forensic examination onsite to get results in a short period. Their model offers an alternative approach to the traditional method, where examination and analysis phases were required to be taken to a lab by introducing the need for an investigator to decide on the scene as to what information is of more importance and where to find them quickly. According to the researchers, the triage model has been largely successful, and the results have not been challenged in court. Although

their study introduced the triage process, which is of interest to our research, it focused on general crime-related investigations. On the other hand, our work proposes a more specific guideline to the extraction and analysis of automotive or maintenance applications.

Similarly, Kao et al. in [44] discuss two types of digital triage; the live and dead. According to them, live triage is conducted on devices that are still powered on by obtaining relevant information as quickly as possible and finding data that might be of evidentiary value. In such cases, the investigator may look at resources such as memory, connections, and other live evidence. Concerning dead triage, they stated that this method is generally carried out in the lab by acquiring the evidence, performing forensic analysis, and generating a case report. The methods discussed in their study can be helpful to our work in terms of our data acquisition phase. However, our work differs slightly from theirs since we will extract and analyze the automotive application folders after we acquire the phone's image.

As discussed in [45], the increase of digital evidence in criminal cases have led to an accumulation of incomplete forensic investigations. To deal with this situation, it has become common for first-responders at an investigation scene to employ triage in an attempt to ensure only relevant devices are obtained, which in turn reduces the number of items to be examined. According to their study, the quality of an on-site triage is based on the perception and depth of knowledge an investigator possesses about the device in question. With the shortage of forensically sound professionals, the majority of the personnel do not have much experience and might negatively impact the triage process. Their work then proposes a system with which investigators arriving at the scene may base the device ranking upon, and score them appropriately. Although their work does not focus on extraction of artifacts from automotive applications, the guidelines provided can be helpful to our thesis during the data acquisition phase as the steps we will take to collect the data and analyze them will be documented into a guideline that can also serve as a way for investigators to introduce a form of triage process to be adhered upon arriving at the scene of crime or incident.

In contrast to the work of [45], [46] in their work introduced a blueprint for applying triage based on machine learning instead of human judgement. This requires that carefully curated and classified inputs from various sources be supplied to an inference engine. Like the work of [45], their main objective was the reduction of human input, hence limiting human error when conducting on-site triage. This is because the intelligence is now provided by a machine. Similar to authors previously cited in this section, their work does not focus on extraction of

forensic artifacts from automotive applications like our work does. Instead, it focused on creating a framework for machine-based intelligence which might be helpful when triaging devices at a live scene. Their work however describes a case study of the classification of child pornography communications on a mobile phone, which can guide us when extracting artifacts in our work.

2.3.3 Android application related works

In the research [20] by Mahajan et al, a forensic study was conducted on the internal memory of 5 mobile devices running 3 different Android operating systems to know the kind of data that can be extracted. The targeted data were the information relating to "WhatsApp" and "Viber" messages as well as media files, and they found Android devices to store a large amount of evidentiary data that is obtainable by forensic experts. They also used the UFED (Cellebrite Universal Forensic Extraction Device) as a tool to conduct a File System Acquisition where folders, as well as files, were extracted. This for the researchers was a way to fully grasp the directory and file structure of the aforementioned Android devices [20]. The File system acquisition method when used is capable of retrieving information such as media and system files, databases, logs, passwords, contacts, calls, and web history from the file system of a device [21]. The methodology used in this reviewed research [20] can be useful to our own research as the goal in our study is to extract data from the phone in regard to installed automotive applications just like the way WhatsApp and Viber were installed.

In another research, Gomez et al emphasized pertinent information that is attainable by forensically examining the Windows 10 IoT vital operating system's nonvolatile memory storage [22]. The researchers used an experimental method to establish a test environment to collect data and then subsequently extract the data by undertaking three diverse examinations and extractions. A diagram of their methodology is depicted below in Figure 3.

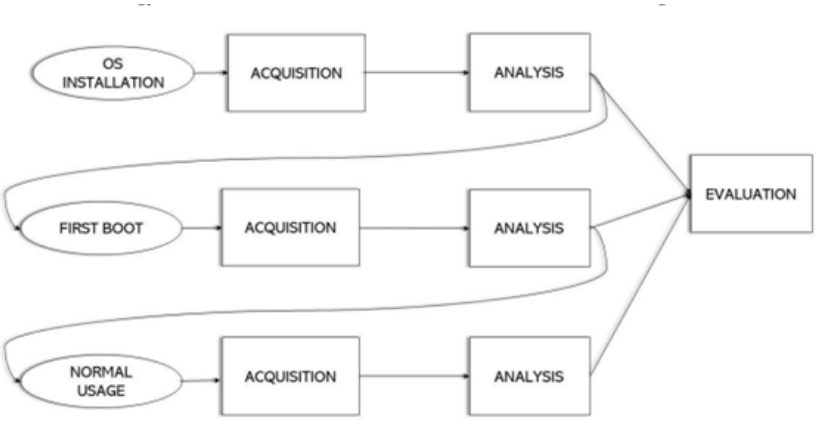


Figure 3. Process flow during forensic analysis [22]

The research also lists some forensic tools including FTK Imager, Autopsy, AnalyzeMFT, all of which help with analyzing the extracted data. Their research thereby provides useful information for our thesis work by giving us a followable guideline in the event when user activity in the vehicle is being simulated to generate information to be stored and later extracted for analysis.

2.3.4 Infotainment Forensics related works

A research by A.K. Mandal et.al presented in their work [23] a static analysis of the Android Auto and On-Board Diagnostic (OBD-II) applications with respect to their vulnerabilities and how these applications can be an entry point of compromise. Their research paper focused on the areas of risks in the Android Auto application and shows how the Controller Area Network (CAN) can leak sensitive information, and also mentioned that attaching OBD devices directly to the car's system poses some form of danger, in the sense that the CAN, based on how it was made provides no protection against any form of alteration or manipulation [23]. This study, although focused on the extraction of data from an infotainment system, points to the vulnerabilities of the OBD devices. Our work will not be analyzing the extracted data from a vulnerability standpoint, but to see how much information these OBD devices leave on the phone which can provide alternatives for forensic investigations in the event the vehicle is not readily available for analysis.

In another research [24], two diverse kinds of infotainment systems were analyzed to get invaluable forensic artifacts. The research aimed to address which kind of forensic artifacts could be mined from the in-vehicle infotainment systems, and the results showed that one

infotainment system stored more data in comparison to the other [24]. The researchers used Berla's iVe tool which is a commercial tool available to the military, law enforcement, and a few governmental institutions. Their work, although focusing on infotainment systems of vehicles, provides us with acquisition methods such as the logical and physical methods to extract data. Our work however will be extracting the data in reverse form, where we aim to acquire data about the vehicle from the smartphone that is using the automotive application.

J. Lacroix in his work [25], also presented methods used in the collection of data from vehicle infotainment systems. The research focused on infotainment systems of 4 different car brands, namely Volkswagen (Passat and Tuareg), Audi (Q5 and Q7) Ford (Fiesta and Focus), and Dodge Durango. Furthermore, the study outlined the use of logical as well as physical data acquisition methods to extract the forensic artifacts. This study also made use of the iVe tool to analyze the forensic artifacts obtained. The kind of data which is of forensic value is also mentioned and extracted in this research such as GPS information, Contacts, Call logs, SMS data, User Account Information, Media Playlist, Emails, and multimedia contents [25]. Our study will not focus much on some user data such as calls, contacts, and messages. We aim to find information such as GPS information that the vehicle, through the application may transmit to the smartphone.

2.3.5 Cloud Forensics related works

The area of cloud forensics is quite new compared to mobile and also android forensics. Using the cloud to expand the storage capabilities of mobile devices and applications has become ubiquitous thereby making the finding of valuable data on the cloud a vital endeavor. Choo et al in their work [26] highlighted the three (3) main areas to concentrate on when conducting cloud forensics. These stages include data residing on the devices, data in the process of being transported, and data that is stored on the servers, which means that it is needed to examine all three areas for artifacts when partaking in cloud forensics [26]. Our study aims to focus on the first part of this area which is to analyze the automotive applications that have been installed on an end device (smartphone) to extract information about the vehicle which has been outlined in the first chapter of this paper that these applications store on the phone.

Cloud computing when accessed from the area of improving storage capacity of mobile devices and applications has been immense. It, however, poses major hurdles for forensic investigators when artifacts are being extracted to be used for legal purposes. As Zowoad et al in their work

[27] mention three (3) drawbacks of the cloud in terms of digital forensics. The use of a single resource by a multitude of users or devices presents a challenge for investigators during data extraction to ensure that the data from different devices have not been joined together [27]. This will violate data integrity which then renders a forensic artifact useless as a piece of evidence in a law court. Their work also mentions that due to the absence of a physical device where investigators can extract data as it is done in traditional computer devices, the amount of valuable data that is available to be extracted is minimized [27]. Lastly, they highlighted the issue of how much the artifact collected from the cloud can be trusted since there exist a possibility of the data being altered by the cloud service provider or perhaps collusion between the service provider and a criminal to obstruct some vital information that forensic examiners may need to successfully prosecute a case [27]. Our work will focus on the data that is directly transmitted and stored on the phone, and therefore our findings will provide investigators with good evidence about the vehicle which has not been put on the cloud yet. However, our study does not cover cloud forensic because the automotive applications do not provide an interface to access the data that is stored on the cloud from the application.

2.3.6 Summary

In the course of our search for related work that would help us successfully conduct our research, we were able to attain numerous information from previous works undertaken in the area of digital forensics for mobile devices, android applications, infotainment systems, and the cloud. We are aspiring to apply the various data acquisition methods and analyses used in these past works.

3. Methodology, experimental setup, and data collection flow process

In this chapter, the methodology that was used for our study will be briefly explained and the rationale behind our choice. We will also outline the materials and tools that were used for our experimental setup and for the extraction and analysis of the data. Lastly, a process flow diagram that captures the data acquisition process that we followed for our thesis is also mentioned in this chapter.

3.1 Methodology

Our work is going to be done using the extraction methods recommended in the work [28, p.17] by Ayers et al which is a layered level of proposed extraction or acquisition methods for mobile forensics. We are choosing their methodology because our study is focusing on extracting information from a mobile device and their approach gives us a guide with which we can approach the data acquisition. As enumerated in [28, p.16], it is advisable to ensure that the data extraction is conducted from the lower level moving up. This as explained by the researchers, extracting data using for instance the chip-off method, which is level 4 in the extraction hierarchy, and then returning to do a Hex Dump or level 3 acquisition method may not be feasible as the lower-level tools may not work as intended. There is a possibility of losing data on the phone or damaging the phone entirely if the investigator does not have the required expertise or the right process is not followed when performing the extraction at this level. Their work, therefore, highlights the importance of ensuring the proper procedures and tools are utilized to avoid the loss or changes in the data. We are starting our data acquisition with the manual extraction method as it is the first level that requires fewer tools to acquire data from the automotive applications, and then progress to the logical method with the physical method being the final level in our extraction methodology. We will not be using the chip-off method in our study as we have already mentioned, this poses a possibility for data loss or damage to the smartphone we used for our study. Additionally, the smartphone we will be using is still in a working condition and since chip-off is a method used as a last step in case the mobile device is not functional when the acquisition process has to be done. This is not the case in our experiment.

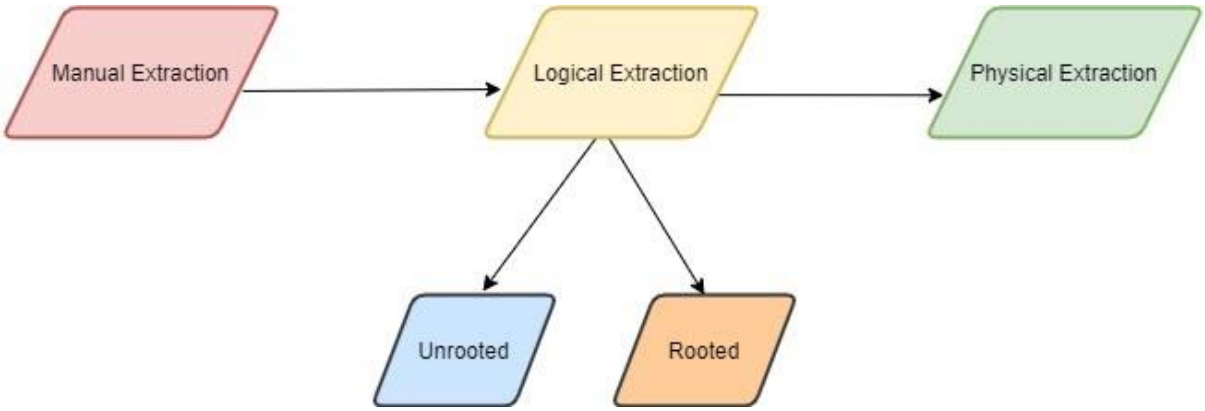


Figure 4. Order of extraction methods used in our data acquisition

3.1.1 Manual Extraction

This extraction method will be used as the first way or first level to obtain information regarding the vehicle and any user-related data such as GPS location on the automotive applications. This will involve interacting with the smartphone and browsing through the user interfaces of the GoFar, Zus Smart Vehicle Health Monitor, and Veepeak OBDCheck Bluetooth OBD2 Scanner applications. As postulated by Ayers et al, some hindrances of this method are the fact that a lot of time is needed to extract the data in the event of large amounts of data, and there is a risk of accidental manipulation of the data during this extraction method [28, p.17]. The former drawback is not applicable to our study; however, we must be wary of any inadvertent modification of the data while we browse the interfaces of these applications for artifacts. This method will also provide us with the chance to get a high-level view of the information that can be gained before any specialized extraction tool or software is applied. This extraction method can provide investigators with vital information for an optimized forensic triage, as browsing through the automotive applications require no special tools to do. This can help with possibly obtaining essential data prior to taking the smartphone to the laboratory for deeper forensic analysis.

3.1.2 Logical Extraction

The second level of extraction to be used in our study is to connect the smartphone to our forensic workstation by either using a wired or wireless connection to execute the logical extraction as indicated again by Ayers et al [28, p.17]. To go by the definition of NIST in 2007, "a logical acquisition implies a bit-by-bit copy of logical storage." [29]. Additionally, Srivastava et al defined logical acquisition as the method that provides the forensic investigator access to data by getting into the file system of the device [30]. We are going to conduct the logical extraction of the phone in two (2) ways; the smartphone will be unrooted and then rooted. The process of rooting is a way to unlock the bootloader of the smartphone and gives a much higher privilege to the user. We aim to identify the artifacts that can be collected from the automotive applications with the phone being either rooted or not.

3.1.3 Physical Extraction

The final method we will use in our work to obtain artifacts from the smartphone regarding the automotive applications is the physical acquisition or extraction. According to the NIST Special Publication 800-101 of 2014, physical extraction can be termed as "extracting and recording a copy or image of a physical store" [31]. This will therefore be synonymous with a logical extraction apart from the bit-by-bit copy being made of the physical store instead of a logical one. As stated in their work, Ayers et al [28, pp. 17-18], there are ways to conduct a physical extraction namely, Hex Dumping and JTAG. Their work also explains that the Hex Dumping method is more of a non-invasive method and does not require taking apart the device. There exists software or tools that can be used to copy the storage of the device to a part of the memory [28, pp. 17-18]. We intend to make use of this non-invasive method because the smartphone that we will be using for the extraction will still be turned on when we work on it. We will make use of a TCP connection between the smartphone and our Windows 10 computer. We will then use dd as a means to acquire an image of the phone to analyze it in Sleuth Kit Autopsy.

3.1.4 Validation Test Analysis

In this subsection of our work, we will conduct a validation test on the data obtained from the automotive applications. In order for our extracted data to be forensically sound, we used 2 additional smartphones, a Samsung Galaxy A20e and iPhone XR, for these tests. We installed the free, yet popular navigation application, Waze, on the A20e and then recorded the screen

which displayed the route we took on the test drive from our starting point *(E. Vilde tee)* to the destination at Rimi Supermaket *(Sopruse pst).* The iPhone XR was used to record the vehicle's (Toyota Corolla Hybrid) dashboard in order to capture our speedometer readings. We then compare the values (Speed, GPS and timestamps) we obtained from the GoFar *default.Realm* file to the outputs recorded on Waze for the speed and GPS location. We used the GoFar application in this section because it shows most of the data; GPS and the route from the GoFar dashboard, and speed information from *default.Realm*. We aim to provide clarity to the reader when comparing the data between Waze, the vehicle's dashboard and the OBD-II application. Additionally, we correlate the speed captured by the iPhone recording to the speed recorded in the *default.Realm* file. A more detailed discussion of the *default.Realm* and its contents has been highlighted in Chapter 4 and Appendix 4 respectively.

We observed that Waze showed our starting point as *E.Vilde tee* which was similar to the one the GoFar application dashboard recorded. The GoFar map data shows our starting address at *E. Vilde tee 117* and triangulates this to a 500-meter radius. The Gofar application started our trip once the dongle established a Bluetooth connection with the car at 19:05 (07:05 PM). We actually started moving from the starting point at 19:08 (07:08 PM) as shown in the video recording of the Waze application[1]. The details in Figure 5 below, shows the starting points in both applications. The Waze application interface is on the image on the left while the next two images to the right of it is the GoFar interface.

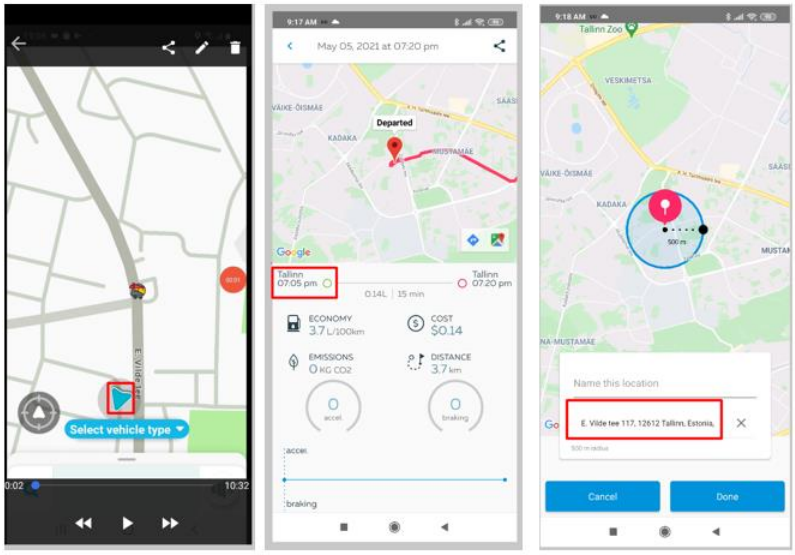


Figure 5. Departed address on Waze and GoFar applications

[1] https://youtu.be/x2eF-hbTlSI

Further, we also noted that our destination point, *Sopruse pst* was captured by both Waze and GoFar (showed *Sopruse pst 174*). As in the case like the starting point, GoFar states our address with the zip codes and gives it in a 500-meter radius. The arrival time in the background of the A20e phone that ran the Waze application (image on the left in Figure 6) showed 19:17 (07:17 PM). The GoFar dashboard showed 07:20 PM, which was the time we switched off the car's engine and severed the connection between the dongle and car. Figure 6 below shows the destination address and arrival times in both Waze (image on the left) and GoFar applications (images in the center and to the right).

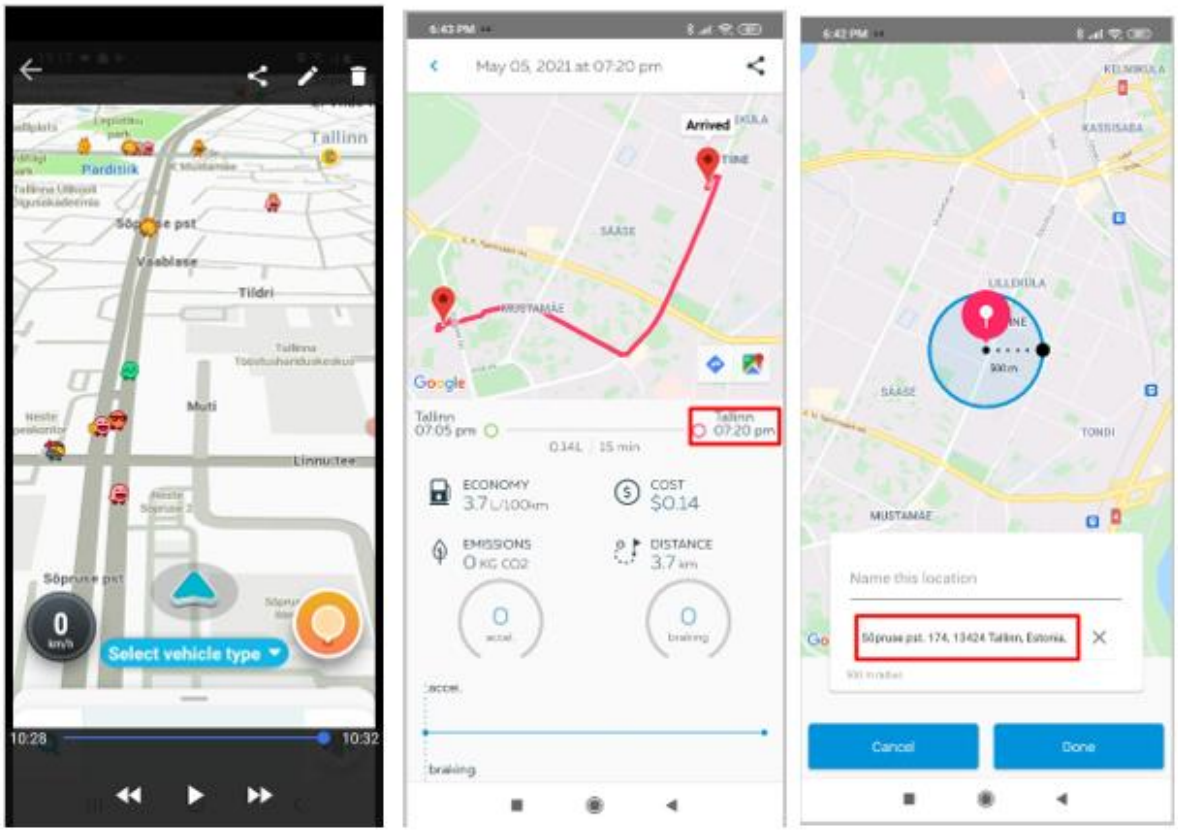


Figure 6. Arrival address on Waze and GoFar applications

Furthermore, we looked at the speed data captured by the Waze and GoFar application and correlate that to the Corolla's speedometer reading at a particular time (19:12 PM) as an example. We recorded [1]the speedometer of the Corolla on this trip as well. Both the Waze application and the vehicle's speedometer recorded 52 KM/h at this specific time. The GoFar application showed 51 KM/h at this time. As we have mentioned earlier in this thesis paper, there is a 1-2 second gap in the data recorded by the automotive applications. This is an area

[1] https://youtu.be/hKhciSXPZlc

we highlight that needs to be researched as an extension to our study. Forensic analysts getting an understanding of the real cause of the lag in logging the data by the maintenance applications can help them explain this in court to minimize any doubt on the extracted data. We do note that the data we have shown in the Figure 7 below indicates that the automotive applications do record and show the vehicle data although not exactly in "real-time" but the timing is commendable and further development may improve this gap of lagging. We were able to get the same speed data from Waze, the speedometer of the car and the GoFar logs for the times, 19:09, 19:13, and 19:14. The speeds we obtained across all three interfaces were 50 KM/h, 38 KM/h, and 50 KM/h respectively. However, the speed data we observed at 19:15 had a variation of 1 KM/h between the GoFar log entry and that of the vehicle's speedometer and Waze. Both Waze and the vehicle recorded speeds of 50 KM/h at this time, but the GoFar *default.Realm* file showed a speed of 49 KM/h. This is quite consistent with the 1-2 second lag in the data collection that we have highlighted as an area to be further researched as a future work. Appendix 5 shows additional snapshots of different times; 19:09, 19:13, 19:14 and 19:15.

Figure 7. Speed data of Waze, vehicle speedometer and GoFar Realm file

3.2 Experimental Setup

Our experiment was conducted using Volkswagen Golf, Mini Cooper, and Toyota Corolla vehicles. We drove within the city of Tallinn going between the areas of Mustamae and Kristiine. Most of the tests were conducted late at night usually after midnight in our effort to avoid traffic situations and also to have the flexibility to move at different speeds. The tests were done within four (4) timelines; February 25th, March 9th, March 12th, and March 16th. Due to logistical reasons, the delivery of each of the automotive applications varied hence we were able to start testing with the Zus Smart Vehicle Health Monitor mini. This was tested on the Golf. We subsequently tested the GoFar and Veepeak applications on the three (3) vehicles on dates mentioned in March 2021. One each test, we swapped the OBD-II dongles after one has been tested. We drove an average of 3.8 kilometers per application in each car during the test.

Figure 8. Volkswagen Golf used in our tests

Figure 9. GoFar OBD-II dongle connected to Volkswagen Golf's OBD-II port

3.2.1 Tools and Devices

In this section of our thesis, the tools and devices that were used for the experimentation are briefly presented. These include the vehicles used along with their make and models, the automotive applications used with their version numbers, the smartphone used, and also the computers along with their operating systems.

Operating System	Version	Host Machine
Windows 10 Home Single Language	64-bit (10.0, Build 19041)	Lenovo IdeaPad S540
Kali Linux	2020.4	VirtualBox 6.1

Table 1. Operating systems used in our work

We made use of a Windows 10 Home OS which is installed on a Lenovo IdeaPad S540. The Window 10 OS was chosen because of its compatibility with most of the android processes such as enabling the ADB as well as its ease of use. Kali Linux was preferred as well due to the availability of security tools, software, and commands such as dd which we used during the physical extraction phase of our data acquisition.

Tool	Version	OS	Used for
Android Debug Bridge (ADB)	1.0.41	Windows	Communicating and interacting with the android phone
Android Backup Extractor	V20210224105130	Windows	Packing the extracted backup file into a .tar
Apache Maven	3.6.3	Windows	Running the command to pack and unpack jar file
SQLite Browser	3.12.1	Windows	Reading database files obtained
Autopsy	4.16.0	Windows	Analyzing data obtained via physical extraction
Xiaomi Mi_Unlock Tool	4.5.813.51	Windows	Unlocking the bootloader of the Xiaomi Redmi 9 to “root” it
Magisk Manager	V20.4	Windows	Rooting the smartphone
BusyBox	1.32.1	Windows	Setting up a TCP connection between our computer and smartphone

Table 2. List of tools and software in our study

In terms of the tools and software that we used for the data acquisition and analysis; we used the ADB to aid us via the command line to communicate with the android smartphone. We also used the Android Backup Extractor which came with the Apache Maven, which we used to create the ADB backup from the phone to our computer and then extract it into a tarball. The SQLite browser was used to analyze the database files of the automotive applications that we obtained via the acquisition process. Furthermore, we used Xiaomi Mi Unlock Tool to aid us in unlocking the smartphone's bootloader in order to root the phone. The aforementioned tools were used during the logical extraction phase to obtain and analyze the data. Lastly, we made use of the Sleuth Kit Autopsy software to analyze for possible artifacts from the image files we collected from the physical extraction.

Make	**Model/Build**	**Version**
ZUS Smart Vehicle Health Monitor mini	ZUHMBKBTV	7.1.0_70102
Veepeak OBDCheck Bluetooth OBD2 Scanner	OBDCheck BLE+	1.75.9
GoFar	Model 3	2.4.17
Car Scanner	400765	1.76.5
Xiaomi	Redmi 9	MIUI 11.0.8.0 QJCEUXM
Samsung	Galaxy A20e	SM-A202F
iPhone	XR	iOS 14

Table 3. List of OBD-II devices and smartphone used

We used the devices listed in Table 3 in our experiment. The three automotive applications we used and their models or build as well as their versions have been indicated. We are not going to explain these OBD-II devices in detail here, but we have provided a description of how they are set up, and transmit data about the car to the smartphone in chapter 3.2.2. We also downloaded the Car Scanner application, which is one of the compatible OBD-II applications that work with the Veepeak OBDCheck Bluetooth OBD2 Scanner. We used the Car Scanner application because the Veepeak OBDCheck dongle does not come with its own software from the manufacturer. The device manual recommended using compatible software such as Torque,

Car Scanner, and OBD Auto Doctor that were available on Google Playstore as our smartphone is an android device.

Make	Model	Version
Volkswagen	Golf	2012
Toyota	Corolla	2020
Mini	Cooper	2018

Table 4. List of cars used in our experiments

The list of cars we used for our experimental setup has been listed in Table 4 above to include their models and versions. The Toyota Corolla and Mini Cooper were CityBee vehicles that we rented to use for our testing. The Volkswagen Golf vehicle is a personal vehicle of the thesis supervisor that we used in our experiments. We used three cars for this study so that our work will have more impact and also to ascertain whether we are able to obtain varying information from the automotive application in relation to the vehicle.

3.2.2 Communication between devices

The applications we used for our thesis have their respective OBD-II dongles that were used to connect to the vehicles' OBD-II port which receives information from the ECU via the CAN. The dongles use Bluetooth as the means of communication with the application installed on the smartphone. The Zus Smart Vehicle Monitor mini and GoFar applications have their own software that provides a dashboard to connect the dongles to the phone via their respective applications. The Veepeak OBDCheck Bluetooth OBD2 Scanner is compatible with the downloadable OBD-II scanner application, Car Scanner which we have highlighted in Table 3. We used the Car Scanner application to connect the Veepeak OBDCheck to the mobile device. The GoFar application saves data for each trip on the cloud, and this allows us to have access to the same dashboard information when we log into another smartphone with our login information. However, the GoFar application does not provide users with an interface to access the data saved on the cloud. This information was relayed by the device manufacturer when we attempted to get a login for a possible cloud data access and if possible, perform some cloud forensics. The Zus Smart Vehicle Monitor mini also requires logging in prior to adding the device to the dashboard. However, just like the GoFar application, there is no interface to log

in to the cloud to access the data. The login information as mentioned by the manufacturer is to protect the user's privilege to the application so that unauthorized access is prevented. Lastly, the Veepeak application also does not provide us with a platform to log in and access stored data on the cloud. We reached out to the Car Scanner application developers regarding possible cloud access, but we did not get a response back from the developers.

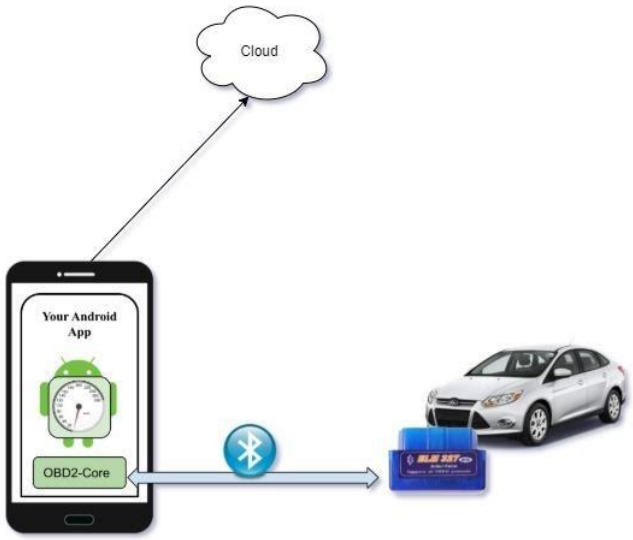


Figure 10. Sample OBD-II device communication diagram [32]

3.2.3. Process Flow for Data Collection

This section highlights the process of collecting the data for our study. The OBD-II applications for each of the automotive applications we used were installed on the Xiaomi Redmi Note 9 smartphone. We then connected the particular dongle to be tested to the car's OBD-II port. The vehicle's engine was then started. Next, we paired the automotive application on the smartphone via Bluetooth to the dongle that has been connected to the car. Once the devices established a connection, the car was driven around for an average of 3.8 kilometers per application on each vehicle to collect our data for analysis. We did an intermediate stopping, where the vehicle was parked and the engine was turned off for about two minutes, and then restarted again to continue the trip and collect data. We did this in order to find out whether there will be any data that relates to these intermediate stops. Lastly, the car was parked and the engine turned off once the trip was completed. The OBD-II dongle was then removed. These steps were repeated for all 3 applications. The GoFar and Veepeak OBDCheck Bluetooth OBD2 Scanner were tested on all 3 cars, however, the ZUS smart vehicle monitor mini had a

limitation and was able to connect to the Volkswagen Golf only. The manufacturers of the device advised that the device works better with a single-vehicle it connects to and will therefore have connectivity issues when trying it with multiple cars. This prevented us from testing the device on the Mini Cooper and Toyota Corolla. The diagram in Figure 11 below highlights the process for our data collection just described.

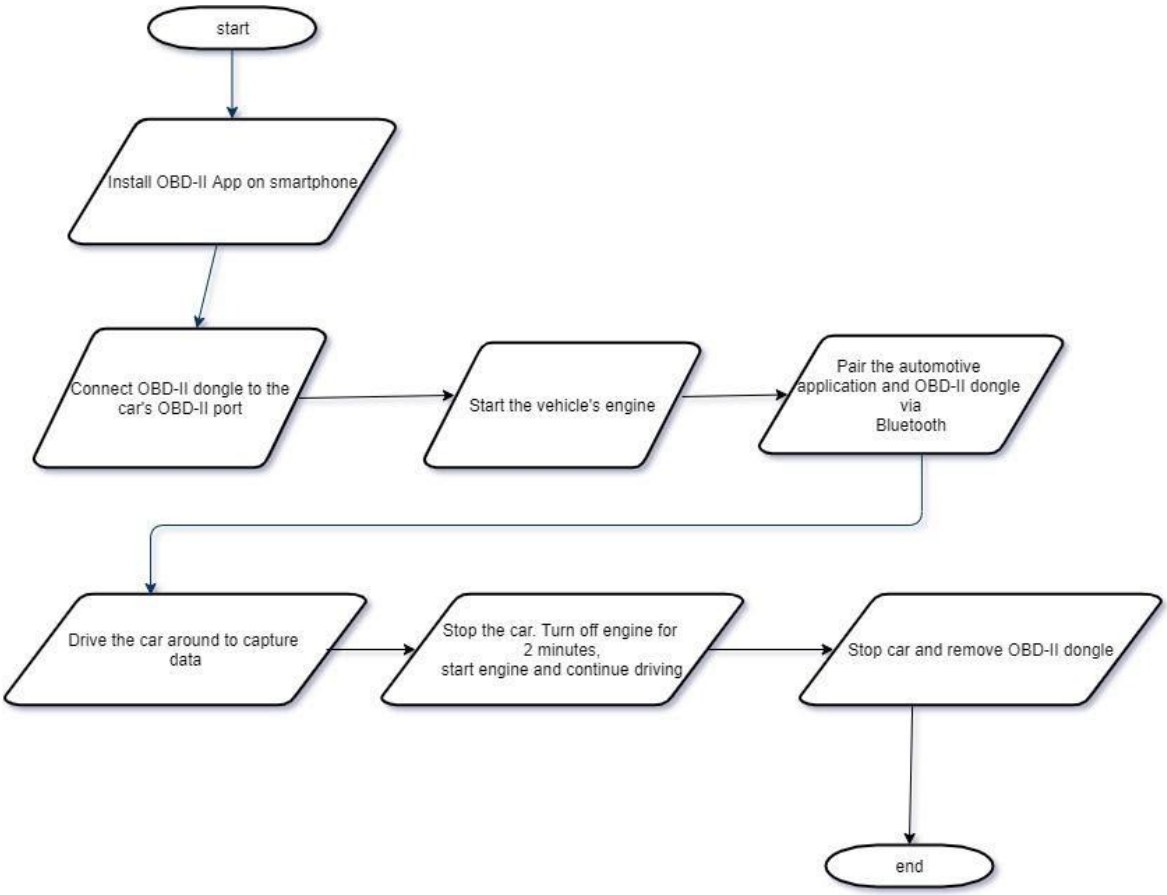


Figure 11. Process flow for the data collection

4. Results

In this chapter, the results of the data extracted through the mobile forensics process is discussed. The manual extraction process for the automotive applications and the findings are enumerated. The logical extraction when the smartphone is rooted and unrooted is discussed and the results are also outlined. Lastly, the results of the physical extraction conducted on the mobile phone is mentioned as well.

4.1 Data from Manual Extraction Process

In this section, we display the results of the data that were obtained from browsing each of the OBD-II application interfaces. We will present the data that meet the artifacts we aimed to find as discussed in chapter 1.3.1. Additionally, we have subcategorized the section into the respective applications and their corresponding findings.

4.1.1 Zus Smart Vehicle Health Monitor Mini

Firstly, we observed that obtaining data from the Zus application dashboard as soon as the OBD-II dongle lost its connection with the smartphone was not possible, however, this data is still stored in the log files which can be obtained using the logical extraction process. Therefore, an active connection is required to be able to navigate through the dashboard for information. This meant that we did the manual extraction while the vehicle's engine was turned off, but the dongle was still connected to the car's OBD-II port and there was an active Bluetooth connection. This is an important observation in terms of the forensic triage procedure as it would mean that depending on the OBD-II application, an investigator needs to ensure that the manual extraction is done on the smartphone while the dongle is not pulled out and there is close proximity between the OBD-II dongle and the smartphone. We were able to obtain vehicle speed, GPS coordinates from the start point to the endpoint. Furthermore, the Zus application also allowed us to obtain some vehicle health information such as the engine coolant temperature. The dashboard also showed the tire pressure as the vehicle moved. The screenshot in Figure 12 below depicts the dashboard interface with the corresponding data. Lastly, it is worth noting that this application was tested only on the Volkswagen Golf due to its limitation of working only on the first vehicle it connected with. Subsequent attempts to connect to the Toyota Corolla or Mini Cooper did not work.

Figure 12. Dashboard information from Zus application

4.1.2 GoFar

The presentation of data on the dashboard of the GoFar application was the same for all three cars we used for our work. In contrast to the Zus application, an active connection was not needed to browse through the dashboard to extract data manually. The homepage of the dashboard lists all vehicles that were tested with their respective values in total. The most recent vehicle tested showed first on the dashboard and proceeded in that order. The data we obtained about each specific car from using the GoFar application dashboard have been listed below. In addition, each trip's log file in .csv format can be exported by clicking on the "share" button, and then this file can be sent via email, Bluetooth, or saved on Google Drive. The specific screenshots for each car's data can be found in Appendix 1 of this document.

- total distance traveled in kilometers
- an estimate of the average gasoline consumption measured in kilometer per liter

- the average vehicle speed
- the total time traveled for the trip
- amount of carbon dioxide emissions in kilograms during the trip
- the estimated cost of gasoline used during the trip
- a map of the distance traveled including departed and arrived destinations

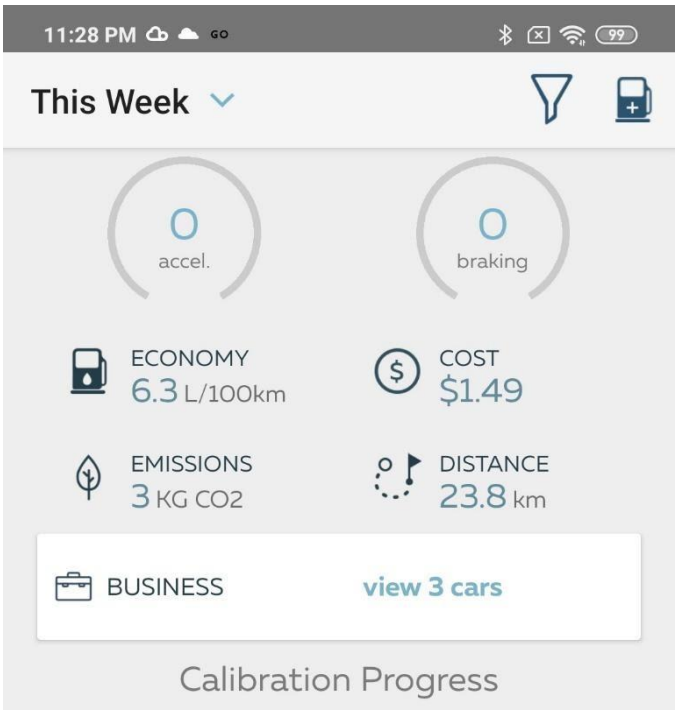


Figure 13. GoFar dashboard summary for all three cars

On the dashboard of the GoFar application as depicted in Figure 13 above, the total distance in kilometers of all the trips we took with the three cars is shown. The carbon dioxide (CO2) emissions are shown in kilograms. The estimated total cost of the fuel we consumed on the trips was captured as well as the fuel consumption per 100 kilometers was shown. The application also breaks down each individual trip in this manner.

4.1.3 Veepeak OBDCheck Bluetooth OBD2 Scanner

We used a compatible OBD-II scanner application called Car Scanner to manage the Veepeak application on the smartphone as the Veepeak dongle does not come up with its own dashboard management application. Similar to the GoFar application, we were able to extract information from this application without the need for an active connection between the dongle and the

smartphone. However, the Data recording, Statistics, and Current Car were the only accessible tabs that were available for us. The Data recording tab is where the data such as distance traveled, speed, fuel consumption, acceleration, and others regarding each trip per vehicle is stored. The recorded information is also stored by the date and time of the trip. A depiction of the contents of this tab is shown in Figure 14 below. As it can be seen in the figure, there are three recorded data that correspond with the cars (Golf, Corolla, and Mini Cooper) we used in our experiments. A click on any of the recorded data takes the user to the next interface where you enable which data you want to view and the values recorded are then displayed as shown in Figure 14. Other tabs required that an active connection was made between the dongle and the mobile phone. In our search for data from the dashboard, we found that the application listed the vehicle information that was tested based on the car's brand. For instance, the Toyota Corolla was a hybrid car, and the information that was shown under the "Data Recording" section listed additional information that was not present or tested in the Mini Cooper and Volkswagen Golf.

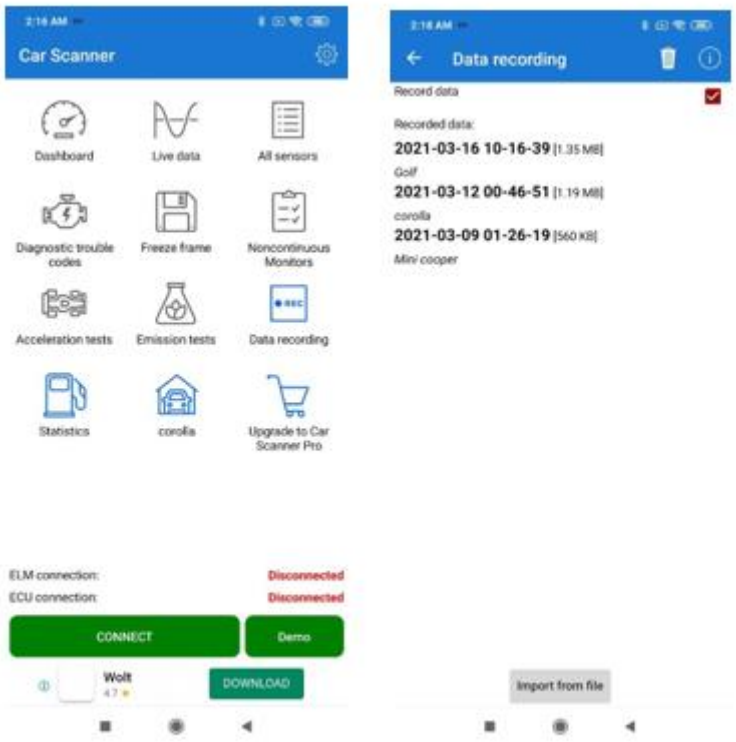


Figure 14. Data Recording tab showing logged files per brand of car

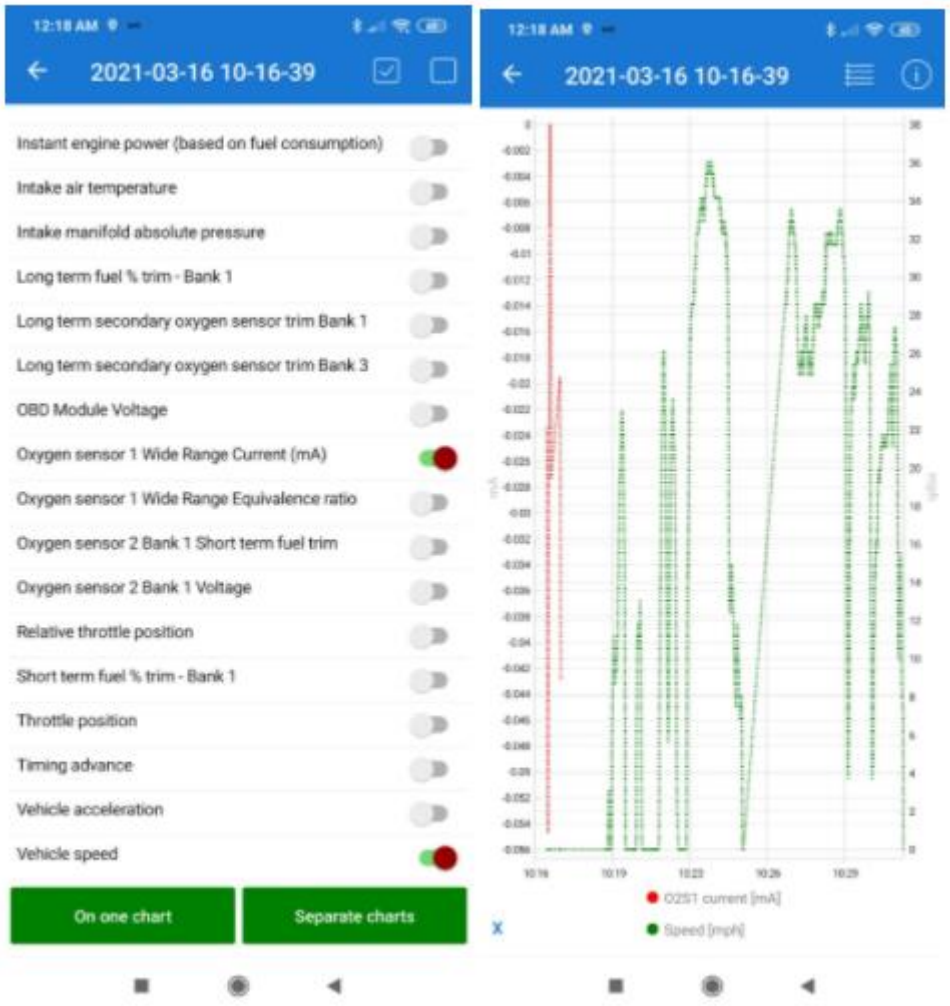


Figure 15. Oxygen sensor 1 and vehicle speed data from Veepeak OBDCheck

As it can be seen, the red dotted line shows the data recording for the oxygen sensor 1 while the green line shows the vehicle's speed during the trip. This information once enabled from the first picture on the left, displays the values for further analysis. The other options available can be selected to have their data displayed as well.

The Statistics tab provided vehicle speed and fuel consumption data for the Mini Cooper and Volkswagen Golf which have been depicted in tables 5 and 6 below. The data that we were interested in based on our thesis objective that could be obtained from the Statistics tab are average fuel consumption, average speed, distance traveled, and machine hours which is the time measured for how long the Veepeak has been connected to the car during that particular

trip and fuel used in liters. The Toyota Corolla displayed no such data when the Statistics tab was examined during this extraction phase. We were unable to find a reason for this lack of Statistics data by the Corolla upon further research, and we got no response from the device manufacturer when we inquired for some information regarding this. We suspect that the Toyota Corolla being a Hybrid and model 2020 may have some newer system configurations that hindered the presentation of this information. This area will need further research, which is not within the boundaries of our study.

Average fuel consumption (miles per gallon)	**Average speed (mph)**	**Distance traveled (miles)**	**Machine hours (hours)**	**Fuel used (liters)**
492.24	7.64	1.45	0.2	0.01

Table 5. Statistic tab data for Mini Cooper

Average fuel consumption (miles per gallon)	**Average speed (mph)**	**Distance traveled (miles)**	**Machine hours (hours)**	**Fuel used (liters)**
1586.02	10.38	1.46	0.1	0

Table 6. Statistic tab data for Volkswagen Golf

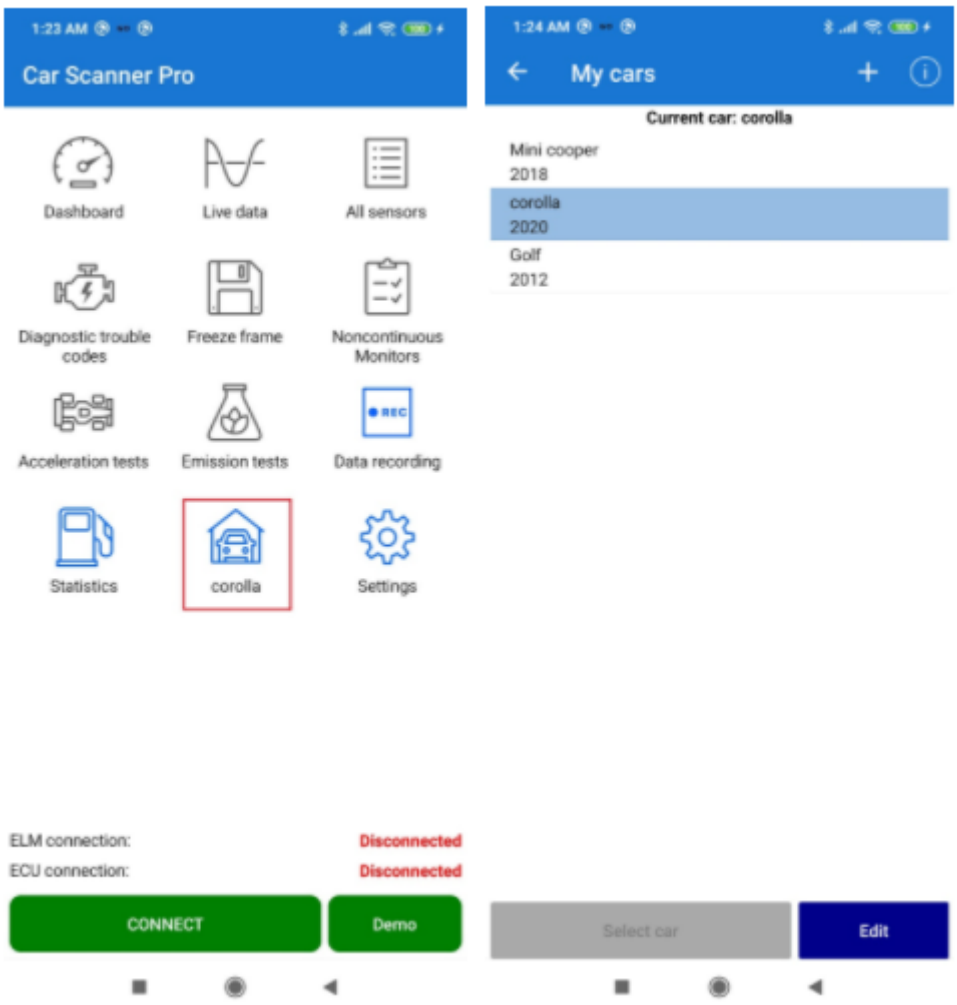


Figure 16. List of cars on the Car Scanner application of Veepeak OBDCheck

4.2 Data from Logical Extraction Process

In this section of our work, we present the data we were able to obtain during the logical extraction of the Xiaomi Redmi 9 phone. We focused on the application folders created on the android phone for the automotive applications. Furthermore, we present the results of our tests with the phone both unrooted and rooted.

4.2.1 Data from unrooted phone

The smartphone we used for our experiment was not rooted therefore in our logical extraction, we were extracting the data that is obtainable to us at the normal user level. We made use of ADB for this phase of our data acquisition. The use of ADB Backup for logical extraction to obtain forensic data from the android phone is considered as the go-to option in terms of the amount of data that can be obtained as well as the quality of the data [33]. Additionally, ADB Backup is an open-source tool that does not require us to purchase it. In their work [34], Lwin et al stated that the Android Backup can be used for logical extraction although they recommended the use of the Magnet Acquire tool for logical extractions. We were not able to obtain this tool for free and were asked to purchase it. We, therefore, proceeded with the use of ADB Backup. Firstly, we activated USB debugging on the phone to provide us with developer options on it. We then installed apache-maven version 3.6.3 as part of the Android Backup Extractor we obtained from the GitHub repository [35]. Next, the smartphone was connected via a USB cable to our Lenovo laptop and a connection was established between the two devices. We proceeded to conduct the backup from the Windows command prompt by typing the command:
C:\platform-tools> adb backup -shared -apk -all -system -f backup.ab

Once this command was entered, a prompt popped up on the phone to confirm a backup of the data on the phone. This then created a backup of the data on the phone into a backup.ab file in the location we extracted the data to. We then had the backup.ab file packed into a .tar file by issuing the command under the android-backup-extractor-master folder:
C:\android-backup-extractor-master> java -jar target/abe.jar unpack C:\platform-tools\backup.ab C:\platform-tools\backup.tar

Finally, we extracted the backup.tar file to obtain the data we acquired from the backup of the android phone. The table below shows the contents in automotive application folders we were able to recover.

Item	Folder/file Path	File Types	Application	Acquired Data
1	\apps\us.nonda.zus\ef\ZUS\Log	Log files (.txt)	Zus Smart Vehicle Health Monitor mini	Logs files containing GPS position, the details of the OBD-II dongle connected to the phone, and vehicle
2	\apps\us.nonda.zus\sp\FILE_NAME_VEHICLE	.xml file	Zus Smart Vehicle Health Monitor mini	an XML file that contains the VIN number of the vehicle
3	\apps\us.nonda.zus\sp\ezzy_saver_sp.xml	.xml	Zus Smart Vehicle Health Monitor mini	Maximum speed, maximum RPM, and Boolean value showing that the OBD recorded the values
4	\apps\com.ovz.carscanner\f	Log file (.txt)	Veepeak OBDCheck Bluetooth OBD2 Scanner	Log file generated by the application which has data on the connection details (Start date and time dongle connected to car, Vehicle name)
5	\apps\co.gofar.gofar\f	Log file (.txt)	GoFar	Log file showing the date and time of connection, mobile phone model, VehicleID, UserID, Android version on the phone
6	\apps\co.gofar.gofar\f\default	REALM file	GoFar	VIN, VehicleID, userID, email, firstname & lastname, GPS, speed, average speed, acceleration, braking, vehicle make & model

Table 7. List of extracted artifacts acquired from the logical extraction

We will provide some details on the data that were obtained for each OBD-II application that we were interested in and the relevance of the information to a forensic investigation or another form of investigation such as a traffic dispute.

- **Item 1**: The log file we extracted from the Zus application folder contained location data as well as the connection details of the OBD-II dongle's pairing with our android smartphone. We are interested in the GPS data in this log as it contains the initial location we started driving the car and then the final point where we stopped and disconnected the application from the car. The date when the log file was recorded is the name of the file and the time for each monitoring of the application is captured in this log file. This information could be useful in an investigation as a forensic investigator may check an OBD-II application folder from a smartphone of either a victim or a perpetrator of a crime to ascertain their location on a specific date and time. This information may be used to corroborate the evidence that the investigator may have already obtained. A section of this file is shown in Appendix 2.

- **Item 2**: The Zus application saved the vehicle VIN in the .xml file listed in Table 7. The VIN number for the sake of clarity is the vehicle identification number that is used to specifically identify the car. During the installation and setup of the Zus application, we did not provide this information when the application required us to provide the vehicle make and model. It was therefore interesting to find this information while checking for possible artifacts after the logical extraction. A lookup of the VIN on a free VIN checker online[1] shows some basic information about the vehicle to confirm the validity of the VIN. This data could be useful during an investigation ~~to~~ where the smartphone may provide additional information about the vehicle to link the user to the car. A snapshot of the details of the XML file containing the VIN is shown in Appendix 3.

- **Item 3:** This .xml file also found in the Zus application contained the maximum speed that the Volkswagen Golf went during our tests. We would like to state that this maximum speed information displayed on the dashboard for us (39 mph) during the manual extraction phase was less than the value of 51 mph we found here. It is crucial

[1] https://uk.vin-info.com/order-reports/WVWZZZ1KZCMG70522

to note that we observed a lag of about 1-2 seconds between the actual speed we were traveling and the information displayed on the dashboard. Monitoring the dashboard, we did not get this maximum speed of 51mph at the exact time we attained that speed. This slight delay in information being relayed to the dashboard as explained by the device manufacturers, is dependent on the computing power of the vehicle's ECU[1]. They state that it can take up to 1 second for the sent packet to be received by the OBD-II dongle which can cause that delay. Further checks on this issue showed that multiple users have this slight delay issue as well. We found the single entry of the maximum speed saved in the xml file. This information is important to take note so that an investigator should also be looking for such information using logical extraction as some automotive applications could be storing this useful artifact in .xml files or other formats. The file also shows a VehicleID number which the Zus application assigned to the vehicle, and this number is shown in the XML file that contained the information presented in Item 2. The maximum revolutions per minute (RPM), which is a measure of the speed in which the engine spun, was also recorded in this file. This information could be useful in a traffic investigation to confirm the maximum speed that the vehicle obtained, that is if an OBD-II dongle was connected to the car and paired with a smartphone. A screenshot of the XML file is shown in Appendix 3

- **Item 4:** The Veepeak OBDCheck application saved its connection details in this log file in a text format. This file contains information about the start date and time of the log which captures the trip. Additionally, detailed information about the functionalities of the vehicle was captured in this file. To simplify, the Mini Cooper's detailed information such as its adaptive headlights, navigation system, rain light sensor, parking brake, keyless ignition, and emergency response unit are shown to be present in the vehicle and working properly. Some of these functionalities we observed in the Cooper to be working were the adaptive headlights, keyless ignition, parking brake and the navigation system. For the remaining functionalities we did not encounter either rain or an emergency situation to be able to use these to ascertain whether they were working properly or not. This information could be useful in an insurance investigation where the functions of the car are being sought to determine the state of the car prior to an accident.

[1] https://nonda.zendesk.com/hc/en-us/articles/360046323211-Why-is-the-dashboard-data-delayed-

- **Item 5:** The GoFar application has a log file generated in the subfolder specified in Table 7, that stored the date and time the OBD-II dongle initiated connection with the smartphone for that trip. It also shows the make and model of the mobile phone that the application is installed on. The information regarding the smartphone that the log showed was *"androidModel":"Xiaomi M2004J19C"*. A check on this information reveals the exact model of our phone which was Xiaomi Redmi 9 (Lancelot). This information is useful for an investigator as it adds more credence to the information collected from the dashboard during the manual extraction process we discussed earlier in this paper, and pins that information to this phone specifically. This may help to reduce any doubt regarding the evidence collected earlier to solidify the case. We do however, should state that the log information showed the phone's Android version to be version; *"androidVersion":"10"* which also matched the Android version on the Xiaomi phone we used. This log file also contained VehicleID and UserID information, where unique identifiers the application tagged each vehicle that it connected with.

- **Item 6:** The REALM file we found in the GoFar application folder was rich with information. Using a REALM browser, we were able to read the contents of this file which contained the VIN for two (Mini Cooper and Toyota Corolla) of three cars we used after we ran a lookup of the numbers online[1]. We also found VehicleID and UserID which were information that we also found in the list of artifacts in Item 5. The GoFar application assigned each vehicle with a unique ID and a user ID and this two information also showed up in the log entries in the list of artifacts in Item 5. Furthermore, the REALM file stored the user information (email, firstname and lastname) that we provided during the initial setup of the application. We also found the country, locations we traveled with the cars, speed of the car that was recorded into this file every 2 seconds, and the average speed. The artifacts we obtained from this file could be crucial for an investigation. The user data such as email address and the name can help identify the user and provide some level of attribution. The speed information recorded every 2 seconds in this file provides a detailed log of the movement of the vehicle so for traffic or insurance investigations, this information could prove useful. Additionally, the application logged the GPS coordinates for each trip by storing the change in coordinates while the car moved. This information was being updated every

[1] https://www.autodna.com/vin-check

other second. We were also able to obtain the acceleration and braking information. Both are calculated based on the x and y axis acceleration (g-force data), speed and RPM. We obtained this information when we spoke to the developers of the application regarding to know about this data. The x and y axis acceleration data can also be found in this REALM file. The dashboard showed the headings for this information; however, it did not record any data. Further checks on this showed that the application must complete its calibration before this acceleration and braking information will be displayed on the dashboard. The information in the log file shows that the application logged this information both when we had the application was either calibrated or not. We found this information during the logical extraction. Lastly, we found details of a Subaru brand in the logs, and further search in the log revealed some information such as the GPS coordinates of Wollongong, a city in Australia. The application developers are in Australia as we ordered the device directly from their website. This would mean that the application was likely tested prior to being shipped to us with the information still staying in the logs. Appendix 4 shows the information in the REALM file.

4.2.2 Data from Rooted Phone

The phone we used for our experiment was by default unrooted, and so we needed to access files using the root privileges to ascertain whether any additional information in relation to the automotive applications could be extracted this way. We followed the steps outlined in this technical blog [36] to perform the rooting of our Xiaomi smartphone. We had to unlock the bootloader, restore the backed-up data, root the phone and finally proceed to extract the data from the phone with the root privileges.

- *Bootloader Unlocking*

  The first step in our quest to root the smartphone is to unlock the bootloader to provide us with an opportunity to extract possibly more data which could be useful in our study. It is essential to note that unlocking the bootloader of the phone will set it back to factory settings which will erase all data available on the phone. We therefore performed a backup of the data on our phone with the command below which we utilized earlier when we did the logical extraction on the unrooted phone:

  adb backup -shared -apk -all -system -f backup.ab

- *Rooting the phone*

  We downloaded the Redmi 9 firmware MIUI 11.0.8.0 which was the version used by our phone from the official Xiaomi site. We also downloaded the ADB & Fastboot driver and the Magisk Manager APK as mentioned in our guide [36]. We installed the ADB & Fastboot driver on our Windows 10 computer, and then downloaded the TWRP image for our smartphone's version. We also downloaded the *vbmeta.img* file and then booted into recovery mode to install the Magisk application to root the phone [37]. After the phone had been rooted, we proceed to obtain the */Android/data* folder which provided us with the subfolders including the automotive applications folders by utilizing the following ADB command:

  C:\platform-tools>adb pull "sdcard/Android/data"

  We observed that the data that was obtained from the automotive applications' folders were the same as the logical extraction that was done when the phone was not rooted. Therefore, the data we have listed in Table 7 were the same ones that we were able to obtain in this phase.

4.3 Data from Physical Extraction

As we have explained in chapter 3.1.3, we used the non-invasive way in this acquisition phase. We chose this method for our study because less specialized tools are needed to be able to obtain our data. As mentioned in [38], physical extraction can be conducted using the software method (using dd) or hardware namely Chip-off or JTAG. The Chip-off and JTAG techniques were recommended by the researchers as the options in the event that the phone is damaged and not turned on. The method we used for the physical extraction is useful since the smartphone was working, and we were able to connect the phone to our computer and run the dd commands. In the event that a suspect tries destroying the phone so that an investigator may not be able to collect any information from the phone, the chip-off method is recommended. This method is also recommended when all other means of extracting data from the smartphone we have discussed in this paper fails. As we already stated, this method is considered invasive and data may be altered or the device can be completely damaged when using this method, therefore the requisite expertise is needed to do so. Furthermore, the hardware methods are considered invasive which involves taking the phone apart to access the board [39] which we did not need to do in our case to obtain the data for further analysis.

In our case, rooting the Xiaomi Redmi 9 smartphone accorded us with the privileges to install BusyBox application, which we did from the Magisk Manager that was used to root the phone. We connected the phone to our computer for the transfer of the phone's image via a TCP connection. We then ran the *adb shell* command from our Windows command prompt to gain shell access to the phone then listed the partitions on our smartphone with the command *cat /proc/partitions* [40]. We then opened a second command prompt which served as our computer while the first served as our phone. We run the commands below to copy the physical disk of the phone, mmcblk0, as an image and save it on our Windows 10 computer. We also copied the dm-1 blocks which are partitions created after decryption of the phone.

- **Command prompt (as adb shell)**:

  130|galahad:/ # dd if=/dev/block/mmcblk0 | busybox nc -l -p 9999

- **Command prompt (as computer):**

  C:\platform-tools>adb forward tcp:9999 tcp:9999
  C:\platform-tools>ncat.exe 127.0.0.1 9999 > D:\Redmi9.dd

The actions above created an image of the mmcblk0 partitions which we stored on the D:\ drive of our computer. We then run the same set of commands but this time on dm-1 and also saved this on the D:\ partition of the Windows 10 computer:

- **Command prompt (as adb shell)**:

  130|galahad:/ # dd if=/dev/block/dm-1 | busybox nc -l -p 9999

- **Command prompt (as computer):**
  C:\platform-tools>adb forward tcp:9999 tcp:9999
  C:\platform-tools>ncat.exe 127.0.0.1 9999 > D:\dm-1.dd

We then proceeded to analyze these image files in the Sleuthkit Autopsy application on our Windows 10 computer. We added these ingest modules in Autopsy on our image files: File Type Identification, Central Repository, Embedded File Extractor, PhotoRec Carver, Exif Parser and Interesting Files Identifier. We analyzed the Redmi9.dd file which was about 61 GB, however, we were not able to obtain any information regarding the applications as the Xiaomi Redmi ran on Android 10, which comes by default encrypted. A research [39, p.431], states that versions of Android from 6.0 to 9.0 onwards come with encryption by default.

We subsequently loaded the dm-1 decrypted image that was created when we ran the TWRP during the rooting process of the phone under the same ingest modules. The /userdata provided us with information that included our 3 automotive which were the data we looked to analyze. The data captured was no different from the information obtained in the logical extraction process for these applications. The database files that were captured in the Zus and GoFar applications when examined in SQLite were empty and had no data to aid our investigation. The log files for the Veepeak and Zus applications were also captured here and the information was the same as the ones obtained during the logical extraction phase.

5. Discussion

This chapter of our paper outlines the discussion of our findings regarding the artifacts we were able to obtain during the acquisition phase of our study. We tabulate the data by categorizing them by applications, and the acquisition methods where this artifact was retrieved. We are going by this way in our discussion to highlight the triage procedure to guide investigators on what artifacts could be obtained from automotive applications and by which method that data can be extracted. From the list of artifacts listed in Table 8, we observed that it is possible to obtain artifacts such as vehicle speed, GPS or map information as well as fuel consumption from the manual acquisition method. The most important thing to note regarding the manual extraction on the phone when looking for artifacts from automotive applications is for an investigator to keep the OBD-II dongle connected to the vehicle's port. As we observed during the manual extraction process, the Zus application provided access to the dashboard so long as the dongle was still connected to the car's OBD-II port. The Veepeak application provided us access to only 3 tabs (Data Recording, Statistics and Current Car) when the device is not connected to the car. This mobile application, Veepeak, initiates a connection to the dongle by default each time it is accessed on the phone and this attempt is always logged as well. The GoFar application on the other hand does not require a connection to be maintained when browsing through the dashboard for artifacts. The GoFar interface provided us with the artifacts such as the map of the trip which included our starting point to the end point. It also logged the start time of our trip for each experiment we conducted as well as the end time. The map data shows the routes we took for all tests and these were recorded and saved on the dashboard for each trip. Table 8 below provides a summary of artifacts for each stage of the acquisition process.

Based on the data in Table 8, investigators who arrive at the scene of a crime or an incident involving a vehicle, should consider doing manual extraction as the first stage to quickly obtain some crucial information. This can assist them to decide at the scene whether they will need to take the phone to the laboratory for a deeper look for additional evidence. As we have seen from the manual extraction phase, the information stored by each automotive application was different, so performing this triage at the scene of the incident could provide initial leads or evidence that can help the case immensely. This will also help reduce backlogs at the laboratory since there will be no need to analyze the phone should enough evidence be obtained.

Extraction method	Application	Location	Relevant artifacts	Stage
Manual	Zus	Dashboard	Vehicle name, GPS coordinates and Top speed, Top RPM	1
	GoFar	Dashboard	Vehicle name, date and time of trip, Average speed, Fuel consumed, distance traveled and map of trip (GPS)	
	Veepeak	Dashboard	Vehicle name, speed, average speed, acceleration, engine RPM, distance traveled and fuel consumed	
Logical - Unrooted	Zus	\apps\us.nonda.zus\sp\FILE_NAME_VEHICLE	VIN number	2
		\apps\us.nonda.zus\sp\ezzy_saver_sp.xml	• Maximum speed • Maximum RPM • App connection date and time	
		\apps\us.nonda.zus\ef\ZUS\Log	• GPS data with complete addresses • Vehicle name • App connection date and time	
	Veepeak	\apps\com.ovz.carscanner\f\log	• Vehicle name • App connection date and time	
	GoFar	\apps\co.gofar.gofar\f\Logentries	• Phone model • Android version • App connection date/time	

		\apps\co.gofar.gofar\f\Default	• VIN • Email • Firstname & lastname • GPS • Speed • Average speed • Vehicle name • Acceleration • Braking	
Logical - Rooted	All apps	no change from Logical -Unrooted	same artifacts from logical - unrooted	3
Physical	All apps	no change from Logical -Unrooted	same artifacts from logical - unrooted	4

Table 8. Summary of relevant artifacts per OBD-II application

To elaborate further, the map information or GPS information that the automotive application records along with the date and time, could corroborate the alibi of an accused who was nowhere near the location of a crime at a particular time or possibly link them to the crime based on the information. It could as well lead investigators to know the movements of the person should the smartphone be seized for further forensic analysis when there is the need to do so and the legal basis to seize the phone for examination is duly adhered to. Furthermore, the fuel consumption log, although being a feature which the users of automotive applications utilize to calculate their fuel consumption costs, can be another vital artifact for an investigator. This fuel consumption data is recorded for each trip, and can be another information that backs up the route traveled by the vehicle for the specific trip that is being examined. For instance, a 4-kilometer trip recorded should have a corresponding average fuel consumption to ensure that the data can be used to back up the map data. This may help in minimizing any doubt regarding the credibility of the information displayed on the map. Lastly, the vehicle speed and acceleration information that we were able to obtain from the dashboard of the 3 applications we tested can be used to examine the driving pattern of the car at a particular moment. This information can be used to investigate for example traffic offenses where there is a dispute between the driver and law enforcement. We would like to caution though, that the information displayed on the dashboard lags slightly for about 1 to 2 seconds, and so the speed information captured via the dashboard may not be the exact reflection of the top speed that the vehicle went. We were able to find additional information about this in the subsequent extraction methods we undertook in the logical extraction phase.

Our observation during the analysis of the data we retrieved in the logical extraction phase with the phone being unrooted, showed that it is possible for an investigator to obtain key artifacts at this stage. The Zus and GoFar applications provided us with more artifacts when we performed the logical extraction with the smartphone unrooted and rooted as shown in Table 8. Also, as we have listed in the same table, we were able to obtain the Volkswagen Golf's VIN from an XML file that resides in the *us.nonda.zus\sp directory*. Likewise, we obtained the VIN for both the Cooper and Corolla from the GoFar log file which was saved as a REALM database file. These applications were able to capture this VIN information which we were able to use to verify the identity of the vehicles although we did not add this during the initial setups. The VIN information found in the Zus folder can be found in Appendix 3; VINs and details in the REALM file of the GoFar application are shown in Appendix 4. This information in our estimation is vital as it can be used to directly link the information that the automotive

application recorded on its dashboard to the specific vehicle. Another artifact that we found in Zus application in this acquisition method (logical) was the maximum speed that the vehicle attained during the trip and the corresponding RPM to match. This information was also extracted from an XML document under the same directory that the VIN information was obtained from; *\us.nonda.zus\sp*. The REALM file we extracted from the GoFar application also displayed the speed data of the vehicle as we reported earlier. This maximum speed information in our estimation can be used by law enforcement in a traffic dispute regarding over-speeding in a situation where there are no speed cameras to detect the violation, but the case is reliant on the observation of the traffic enforcer who noticed the over-speeding. We do not state emphatically that this may be an easy option for investigators, however, in a case where it is a serious legal case, our work provides investigators the option to look into the smartphone of the driver for possibly any automotive application folder and then retrieve this information as evidence that could be used in the case. This maximum speed data will not be seen by an investigator from the manual extraction when the vehicle has already stopped, since the dashboard information changes based on the actions of the vehicle. Furthermore, we were able to extract artifacts such as GPS information, vehicle name and the logged time for each entry in the log for the Zus application. The GPS information in the log file stored under the *\us.nonda.zus\ef\ZUS* directory was specific in terms of the address where we started our trip. It captured the complete address including the zip codes of our start and destination points. Similarly, the GoFar application provided us with such information in the REALM file in *\apps\co.gofar.gofar\f\Default*. The precision of the GPS information that is logged is a useful artifact for an investigator in terms of tracking the user to a particular location on that date in case there is an investigation or probe into the activities of the person on that date and time. Our work therefore provides investigators with another avenue to explore when forensically analyzing a smartphone for evidence. The Veepeak provided the vehicle name and the connection details (Start date and time dongle connected to car, Vehicle name, Bluetooth connection status) between the particular car and the OBD-II dongle. This information links the vehicle and its usage of the application, and this subsequently backs the information that the application provided us on the dashboard during the manual extraction phase. We did find a *.brc* file in the \apps\com.ovz.carscanner\f in the Veepeak application folder, and we had to use an online tool[1] to read the file, however, almost all (98%) as the data was encrypted on that file so we obtained garbled information which did not provide us with much information. The

[1] BRC File Extension - What is it? How to open a BRC file? (filext.com)

only information that was of any essence to our work was the vehicle name (Toyota) that we found. The Veepeak application therefore provided the least amount of data during this logical extraction method.

In terms of the logical extraction when we rooted the phone, the data we obtained were the same as the ones we got when the phone was not rooted. We performed the rooting of the phone for completeness in our work although we theorized that we possibly would end up with the same data as when the phone was unrooted. The subfolders in each of the 3 automotive applications we used for our experiments did not change. During the logical extraction with the phone unrooted, we were able to find 2 SQLite files in the GoFar application which had no data when we analyzed them in the SQLite browser application on our Windows 10 computer. Our work therefore shows that an investigator may not need to conduct a logical acquisition with the phone rooted when harvesting for forensic artifacts from an automotive application. In our case, we needed to root the phone to be able to proceed to the physical extraction phase since we were using the "dd" method to copy an image of the phone. Furthermore, the logical extraction process was still an important step in our research as we were able to obtain some more artifacts such as the VIN and maximum speed obtained by the vehicle on a particular trip. Additionally, the logical extraction method is a prudent method to acquire artifacts since some automotive applications require the OBD-II dongle to still be connected to the car in order to extract or browse through the mobile application for artifacts. As we can see from Tables 9-11, the logical extraction phase can present an investigator with interesting artifacts that have been saved by the application. Based on the log files in the Zus and GoFar applications, we recommend that investigators examine the smartphone at the forensic laboratory as more data could be extracted at the logical acquisition phase.

We also performed a physical extraction on the smartphone in order to find out if the automotive applications had any hidden data that the logical extraction process was not able to help us retrieve. We were unable to find any hidden information from the 3 applications when we analyzed the phone's disk image using SleuthKit Autopsy software. The application folders contained the subfolders we examined in the logical (root and unroot) phases. We can therefore suggest that an investigator may use the physical extraction to ascertain whether data is recoverable should the automotive application get deleted from the phone by the user. As we mentioned earlier, the physical extraction method we used was feasible as the smartphone was still working and so connecting it to our computer to copy the image using the BusyBox application via Magisk was possible.

To further aid investigators on what artifacts can be retrieved and from which stage of the acquisition phase, we present the information in Table 9, 10 and 11 where we have summarized the important extracted artifacts from our study per application per car. The artifacts collected for each application for all 3 cars show a slight variation in the artifacts that can be extracted for each OBD-II application. The Zus and GoFar applications provided us with the VIN which uniquely identifies the car; something the Veepeak applications did not provide. We were able to obtain maximum speed and RPM from both the Zus and Veepeak applications. The GPS coordinates for a trip were extractable from the Zus and GoFar applications but not the Veepeak application. We observed that the artifacts we were able to collect differed based on the OBD-II application, and what parameters that application checked in the car but not the car type. In other words, we obtained the same artifacts for the Golf and Mini Cooper cars, but as we reported earlier in our study, data such as average fuel consumption, average speed, distance traveled, and fuel used in the "Statistics" tab from the Veepeak application for the Toyota Corolla was nonexistent. This information can however still be obtained from the recorded file for the trip as we showed in Figure 14. Additionally, the information provided in Tables 9-11, where we show which acquisition phase each artifact was obtainable, adds credence to why we have set the manual extraction method as the first stage that an investigator upon arriving at the scene to look for artifacts in automotive applications. For the Zus application, we were able to obtain 5 artifacts from logical extraction as compared to 4 in manual extraction; GoFar provided us with the most data (VIN, email and user's name) in the logical extraction phase but the manual extraction also gave substantial data as we have enumerated in the tables below. On the other hand, we were able to get more artifacts from the manual extraction phase for Veepeak as shown in Table 9 - 11. We expected to extract more information from the logical extraction as manual acquisition generally yields the lesser amount of information for both phases of acquisition. Nonetheless, the results for the Veepeak applications show that it may not be necessary examine this at the forensic laboratory as the manual extraction can be conducted at the scene. Investigators will need to decrypt the *.brc* file located in the *\apps\com.ovz.carscanner\f* if there is a need to look for more information this file through logical acquisition. Another REALM file was extracted in the Zus application folder during the logical extraction, however, this file was encrypted and required us to provide a 128-bit hexadecimal key, which we did not have available hence our inability to access this file. Just like the GoFar REALM file, we believe that an investigator may reach out to the device manufacturer, and follow the right legal process to ask for this key in order to access more information in the file. We have deliberately excluded the information for both logical (rooted)

and physical extractions since the data obtained from these 2 stages were the same as what we retrieved from the logical (unrooted) phase.

Additionally, we would like to point out the possibility of performing cloud forensics to obtain data from automotive applications. As we mentioned in section 3.2.2, the OBD-II applications we used saved their data to the cloud, but did not provide us with an interface to login to retrieve or access this data. We believe that data is stored on the cloud as for instance, the GoFar application shows a prompt "Saving data to cloud" when the device is establishing a connection with the mobile phone. Additionally, the Zus application log has an entry in Chinese which translates as "Start upload server" and "Upload server successfully modified" and the information next to these heading was the trip information that included the GPS coordinates. Cloud forensics may be pertinent in the scenario where an investigator is not able to obtain information from the mobile due to it being damaged, or if the application has been deleted from the phone and the physical extraction is not able to retrieve the deleted application. An investigator may reach out to the device manufacturers and with the right legal warrants, seek any assistance in accessing the information on the cloud.

Volkswagen Golf			
Application	**Extracted Artifacts**	**Manual Extraction**	**Logical**
Zus	Vehicle name	✔	
	GPS coordinates	✔	✔
	App connection date/time		✔
	VIN		✔
	Maximum speed[1]	✔	✔
	Maximum RPM[2]	✔	✔
GoFar	Vehicle name	✔	✔

[1] This is also shown as Top Speed on the Zus dashboard
[2] This is also shown as Top RPM on the Zus dashboard

	GPS coordinates	✔	✔
	Average speed	✔	✔
	Distance traveled	✔	✔
	Trip date/time	✔	✔
	Phone model/OS details		✔
	Fuel consumed	✔	✔
	VIN		✔
	Email		✔
	Firstname /Lastname		✔
	Acceleration		✔
	Braking		✔
Veepeak	Vehicle name	✔	✔
	Speed	✔	
	Average Speed	✔	
	Acceleration	✔	
	Engine RPM	✔	
	Distance traveled	✔	
	Fuel consumed	✔	
	App connection date/time		✔

Table 9. Summary of relevant artifacts per extraction method for Volkswagen Golf

Mini Cooper			
Application	**Extracted Artifacts**	**Manual Extraction**	**Logical**
GoFar	Vehicle name	✔	✔
	GPS coordinates	✔	✔
	Average speed	✔	✔
	Distance traveled	✔	✔
	Trip date/time	✔	✔
	Phone model/OS details		✔
	Fuel consumed	✔	✔
	VIN		✔
	Email		✔
	Firstname /Lastname		✔
	Acceleration		✔
	Braking		✔
Veepeak	Vehicle name	✔	✔
	Speed	✔	
	Average Speed	✔	
	Acceleration	✔	
	Engine RPM	✔	
	Distance traveled	✔	
	Fuel consumed	✔	
	App connection date/time		✔

Table 10. Summary of relevant artifacts per extraction method for Mini Cooper

Toyota Corolla			
Application	**Extracted Artifacts**	**Manual Extraction**	**Logical**
GoFar	Vehicle name	✔	✔
	GPS coordinates	✔	✔
	Average speed	✔	✔
	Distance traveled	✔	✔
	Trip date/time	✔	✔
	Phone model and OS details		✔
	Fuel consumed	✔	✔
	VIN		✔
	Email		✔
	Firstname /Lastname		✔
Veepeak	Vehicle name	✔	✔
	Speed	✔	
	Average Speed	✔	
	Acceleration	✔	
	Engine RPM	✔	
	Distance traveled	✔	
	Fuel consumed	✔	
	App connection date/time		✔

Table 11. Summary of relevant artifacts per extraction method for Toyota Corolla

Finally, we present a guideline as outlined in Figure 17 below on how an investigator should analyze a smartphone to collect data from an automotive application. For the purpose of clarity, we have devised a scenario that relates to a traffic incident and an investigator has been called to the scene. We present this scenario to help explain the steps to follow. We assume that a vehicle using an automotive application moving on a secluded road at a speed unknown to the investigator crashes into another car upon a turn. Upon arriving at the scene, the investigator should check if the OBD-II dongle is still connected to the car's port. If there is a connection, a manual extraction can be done on the smartphone (Step 1), but in the scenario that the connection has been severed, the investigator can check if the application dashboard is still accessible (Step 2). It is important to note that based on our discussion on Tables 9-11, some OBD-II applications may not display the maximum speed, but will only show average speed on the dashboard and vice versa. If the dashboard can be accessed, then manual extraction can still be done. As we have highlighted earlier in this thesis, some automotive application dashboards (GoFar and Veepeak) still provide enough information even when they are disconnected from the vehicle. However, if the application provides no information due to the dongle being disconnected, the smartphone should be bagged and sent to the lab for further analysis (step 3). After manual extraction has been done and it is ascertained that enough evidence has been collected, the investigator documents the evidence and collects proofs (step 4) otherwise, the smartphone is then sent to the lab for detailed analysis (step 3). At the forensic lab, a logical extraction is then performed on the phone (step 5). In a situation where the investigator believes more evidence is needed, then a physical extraction is performed (step 6) to check for possible deleted information about the automotive application. In the event that the mobile phone has been damaged, the investigator can use other methods such as reaching out to the device manufacturers for assistance in accessing the data saved on the cloud.

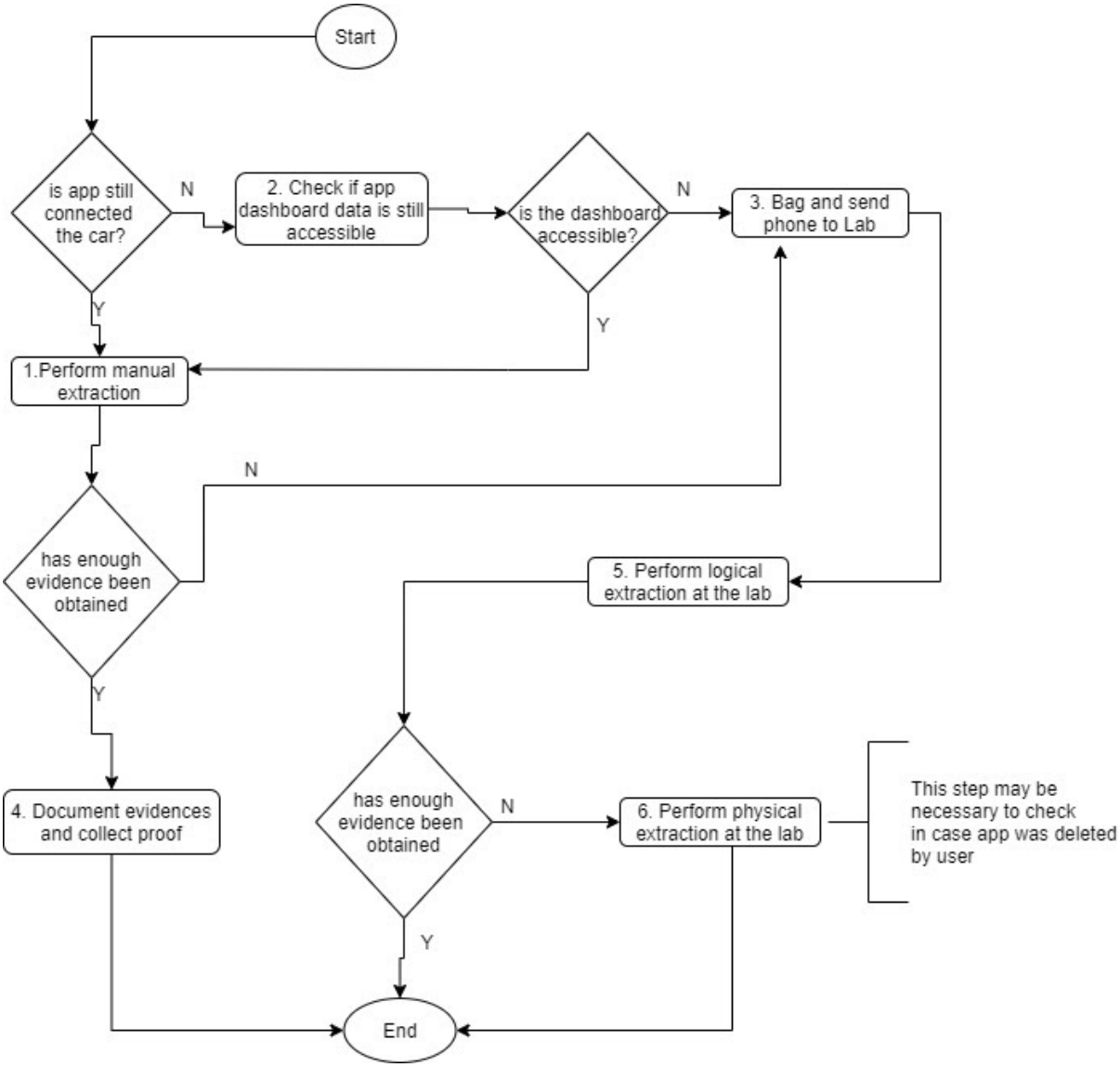


Figure 17. Sample procedure for data collection for a traffic incident

Our work can be seen to provide the steps that help the forensic analyst to make decisions at the scene of the incident. This triaging we have outlined can help minimize the need to send seized devices straight to the forensic laboratory for analysis and can prove useful in cases where time is of utmost importance. This is a move away from the traditional way of conducting forensic investigations as we are providing guidance for an investigator to possibly find evidence at the scene in relation to automotive applications. As the use of automotive applications are becoming prevalent, our work offers another angle to look for artifacts in the smartphone. As outlined in our related works in section 2.3, several studies have been conducted and frameworks have been developed to incorporate triage into forensics

investigations. [47] showed a plethora of models or frameworks that have been developed for evidence collection in digital forensics and most of them have incorporated triage. The authors themselves introduced a framework they called the Next Generation Digital Forensic Investigation Model (NGDFIM) which inculcated a triage stage at the scene of the crime and how the evidence collection or extraction should be done. Our proposed guideline shown in Figure 17 provides something similar, however, we have ventured into a kind of newer applications that the majority of car owners are patronizing.

5.1 Additional discussion

In this section, we briefly discuss some data integrity and legal considerations that could be taken into account in relation to the extraction methods we outlined in our work. It is a known fact that, forensic investigators collect artifacts that can be presented as valid evidence in the prosecution of a case in court. The integrity of such data is of crucial importance to a case as any tampering may render the evidence inadmissible. [48] stated that browsing through the smartphone manually for artifacts such as text messages, pictures and so on, is not recommended if the investigator is not forensically adept at examining the phone. Consequently, investigators should look into using tools that would allow them to collect artifacts without necessarily browsing through the phone during the manual extraction phase. Additionally, documenting all the steps while conducting the manual extract is advised as it will give validity to how the data was attained from the device [48].

Secondly, we presented a guideline that shows the steps to examine and collect artifacts from the smartphone based on one scenario, that is a traffic incident. However, there may be legal considerations that should be looked at when conducting for instance, the manual extraction upon arriving at the scene. There is the question of what the law is in regards to the seizure or collection of the smartphone by the investigator for immediate examination. In other words, is there a warrant to execute the search? We do not go in detail about the legal aspect of this, however, in our work we proposed our method by considering all things being equal and that there is a valid legal right for the investigator to collect the phone and look into it.

6. Conclusion and Future Work

Automotive applications are emerging software that come with OBD-II dongles to help monitor the health of a vehicle as well as track the fuel consumption of the car. As the market for these applications is projected to rise, we explored the relevance of the data that could be extracted from automotive applications to a forensic investigation. We used the mobile forensic techniques; manual, logical and physical extraction methods to extract and subsequently analyze the data. We observed the Zus and GoFar applications saved the VIN of the vehicle, data which the Veepeak applications did not provide. Furthermore, the Veepeak application did not provide GPS data unlike the Zus and GoFar applications. Overall, the GoFar application provided us with the most information in terms of numbers at both manual and logical acquisition phases.

We conclude that although the automotive applications recorded slightly varying data, the obtainable information remained the same for each application across the cars we tested them with. Secondly, the data such as maximum speed, GPS coordinates, VIN, etc. that can be extracted from automotive applications can be relevant to an investigator depending on the case at hand be it a criminal, insurance investigation or traffic incident. Additionally, based on our work we could see that by conducting a manual extraction on the phone at the crime scene, some useful information could be retrieved from the phone, however, depending on the importance of the case, an investigator may send the phone to the lab directly to conduct a more detailed analysis and retrieve information. Lastly, our study described a guideline that may reduce the workload around investigations of IoT devices such as automotive applications by following this triage process.

Future work

The variation in the data collected by each automotive application in our research showed that our work can further be extended as a future study. Firstly, multiple OBD-II applications can be tested to further verify the different data that can be obtained from these applications since these applications are being improved constantly and so some added functionalities may provide the opportunity to collect more data from them than we have done. Secondly, a kind of repository could be created for the most commonly used automotive applications, the kind of data that can be obtained from each and the stages of extraction method to be used for each of them to help guide investigators in the event that an investigation takes the course where evidence needs to be extracted from an automotive application. Finally, further research can be

conducted on how robust the automotive applications can be by clearing, deleting specific log files or deleting the application from the phone and check if this data is still recoverable from the phone.

References

[1] C. Jahn, “Distracted Driving”, Largest Distracted Driving Behavior Study, 2017. [Online]. [Available]: http://blog.zendrive.com/blog/distracted-driving/

[2] A. Chalimov, “CONNECTED CARS: TOP 5 IOT AUTOMOTIVE APPS AND HOW TO DEVELOP ONE”. Eastern Peak. 2020. [Online]. [Available]: https://easternpeak.com/blog/connected-cars-top-5-iot-automotive-apps-and-how-to-develop-one

[3] P. K. Reddy, Big data analytics: 5th International Conference, BDA 2017, Hyderabad, India, December 12-15, 2017: proceedings. Cham, Switzerland: Springer, 2017.

[4] N. A. Aziz, F. Mokhti and M. N. M. Nozri, "Mobile Device Forensics: Extracting and Analysing Data from an Android-Based Smartphone," 2015 Fourth International Conference on Cyber Security, Cyber Warfare, and Digital Forensic (CyberSec), Jakarta, 2015, pp. 123-128, doi:10.1109/CyberSec.2015.32

[5] D. Hintea, A. Sangins and R. Bird, "Forensic analysis of the telegram instant messenger application on android devices," 2018 17th European Conference on Cyber Warfare and Security (ECCWS), University of Oslo, Oslo, Norway, 2018, pp. 217-223. [Online]. [Available]: https://www.scopus.com/record/display.uri?eid=2-s2.0

[6] F. A. Awan, "Forensic examination of social networking applications on smartphones," 2015 Conference on Information Assurance and Cyber Security (CIACS), Rawalpindi, 2015, pp. 36-43, doi:10.1109/CIACS.2015.7395564.

[7] D. Jacobs, K. R. Choo, M. Kechadi and N. Le-Khac, "Volkswagen Car Entertainment System Forensics," 2017 IEEE Trustcom/BigDataSE/ICESS, Sydney, NSW, 2017, pp. 699-705, doi: 10.1109/Trustcom/BigDataSE/ICESS.2017.302.

[8] "ZUS Smart Vehicle Health Monitor Mini (Gen 4)" [Online]. [Available]: https://www.nonda.co/products/smart-vehicle-health-monitor-mini

[9] "GoFar Makes Any Car Smarter". GoFar.co [Online]. [Available]: https://www.gofar.co/

[10] J. Zaldivar, C. T. Calafate, J. C. Cano and P. Manzoni, "Providing accident detection in vehicular networks through OBD-II devices and Android-based smartphones," 2011 IEEE 36th Conference on Local Computer Networks, Bonn, Germany, 2011, pp. 813-819, doi: 10.1109/LCN.2011.6115556.

[11] T. Miller, “Which OBD2 Protocol is Supported by My Vehicle?,” OBD Solaris, 30-Mar-2019. [Accessed: 10-Apr-2021].

[12] A. X. A. Sim and B. Sitohang, "OBD-II standard car engine diagnostic software development," 2014 International Conference on Data and Software Engineering (ICODSE), Bandung, Indonesia, 2014, pp. 1-5, doi: 10.1109/ICODSE.2014.7062704.

[13] “OBD2 Explained - A Simple Intro (2021),” CSS Electronics, 2021. [Online]. Available: https://www.csselectronics.com/screen/page/simple-intro-obd2-explained/language/en. [Accessed: 10-Apr-2021].

[14] R. E. Overill, J.A.M. Silomon and K. A. Roscoe, “Triage template pipelines in digital forensic investigations,” Digital Investigation, vol. 10, no. 2, pp. 168–174, Mar. 2013.

[15] P. Domingues, M. Frade, L.M. Andrade, J.V. Silva. Digital forensic artifacts of the Your Phone application in Windows 10, Digital Investigation, September 2019. [Online]. [Available]: https://www.sciencedirect.com/science/article/pii/S1742287619301239

[16] K. Barmpatsalou, T. Cruz, E. Monteiro, P. Simoes. Current and Future Trends in Mobile Device Forensics: A Survey. 2018. ACM Comput. Surv. 51, 3, Article 46 (April 2018), 31 pages. [Online]. [Available]: https://doi.org/10.1145/3177847

[17] R. Ayers, S. Brothers, and W. Jansen. 2014. NIST Special Publication 800-101. Guidelines on Mobile Device Forensics: Revision 1. Technical Report SP 800-101. National Institute of Standards and Technology, Gaithersburg, MD. DOI: http://dx.doi.org/10.6028/NIST.SP.800-101r1

[18] G. Dorai, S. Aggarwal, N. Patel, C. Powell. VIDE - Vault App Identification and Extraction System for iOS Devices. Forensic Science International: Digital Investigation. 2020. [Online]. [Available]: https://www.sciencedirect.com/science/article/pii/S2666281720302560?via%3Dihub

[19] X. Zhang, I. Baggli, F. Breitinger. Breaking into the vault: Privacy, security and forensic analysis of Android vault applications. ScienceDirect. 2017. [Online]. [Available]: https://www.sciencedirect.com/science/article/pii/S0167404817301529

[20] A. Mahajan, M.S. Dahiya, H.P. Sanghvi. Forensic Analysis of Instant Messenger Applications on Android Devices. International Journal of Computer Applications. 68. 10.5120/11602-6965. [Online]. [Available]: https://arxiv.org/ftp/arxiv/papers/1304/1304.4915.pdf

[21] R. Das. Evidence Acquisition in Mobile Forensics. Infosec. 2017 [Online]. [Available]: https://resources.infosecinstitute.com/topic/evidence-acquisition-mobile-forensics-2/

[22] J.M.C. Gómez, J.R Gómez, J.C. Mondéjar, J.L.M. Martínez. Non-Volatile Memory Forensic Analysis in Windows 10 IoT Core. Entropy. 2019. [Online]. [Available]: https://www.mdpi.com/1099-4300/21/12/1141/htm

[23] A.K. Mandal, F. Panarotto, A. Cortesi, P. Ferrara, F. Spoto. Wiley Online Library. Static analysis of Android Auto infotainment and on-board diagnostics II apps. 2019. [Online]. [Available]: https://onlinelibrary.wiley.com/doi/full/10.1002/spe.2698

[24] C. J. Whelan, J. Sammons, B. McManus, T. W.Fenger, (2018). Retrieval of Infotainment System Artifacts from Vehicles Using iVe. Journal of Applied Digital Evidence, 1(1). Retrieved from https://mds.marshall.edu/jade/vol1/iss1/2

[25] J. Lacroix, Vehicular Infotainment Forensics: Collecting Data and Putting It into Perspective. 2017. [Online]. [Available]: https://ir.library.dc-uoit.ca/bitstream/10155/821/1/Lacroix_Jesse.pdf

[26] K. R. Choo, C. Esposito and A. Castiglione, "Evidence and Forensics in the Cloud: Challenges and Future Research Directions," in IEEE Cloud Computing, vol. 4, no. 3, pp. 14-19, 2017, doi: 10.1109/MCC.2017.39.

[27] S. Zawoad, R. Hasan and A. Skjellum, "OCF: An Open Cloud Forensics Model for Reliable Digital Forensics," 2015 IEEE 8th International Conference on Cloud Computing, New York, NY, 2015, pp. 437-444, doi: 10.1109/CLOUD.2015.65.

[28] R. Ayers, S. Brothers, and W. Jansen, “Guidelines on Mobile Device Forensics - NIST,” May-2014. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-101r1

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

[30] 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.

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

[32] “Welcome to Vatichub OBD2-Core.” [Online]. Available:https://github.com/vatichub/obd2-core.

[33] N. Y. P. Lukito, F. A. Yulianto and E. Jadied, "Comparison of data acquisition technique using logical extraction method on Unrooted Android Device," 2016 4th International Conference on Information and Communication Technology (ICoICT), Bandung, Indonesia, 2016, pp. 1-6, doi: 10.1109/ICoICT.2016.7571934.

[34] H. H. Lwin, W. P. Aung and K. K. Lin, "Comparative Analysis of Android Mobile Forensics Tools," 2020 IEEE Conference on Computer Applications(ICCA), Yangon, Myanmar, 2020, pp. 1-6, doi: 10.1109/ICCA49400.2020.9022838.

[35] N. Elenkov, Android Backup Extractor. 2021.[Online]. Available: https://github.com/nelenkov/android-backup-extractor

[36] A. Singh, “How to Root Xiaomi Redmi 9 and Unlock Bootloader (Guide),” YtechB, Jul-2020. [Online]. Available: https://www.ytechb.com/how-to-root-redmi-9-and-unlock-bootloader/

[37] U. Kaliki, “Cara Install TWRP MOD dan ROOT REDMI 9/REDMI 9 PRIME Lancelot,” 15-Nov-2020. [Online]. Available: https://www.youtube.com/watch?v=NuvTyrxWOR4.

[38] M. Boueiz, "Importance of rooting in an Android data acquisition," 2020 8th International Symposium on Digital Forensics and Security (ISDFS), Beirut, Lebanon, 2020, pp. 1-4, doi: 10.1109/ISDFS49300.2020.9116445.

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

[40] J. I. James, “Android Acquisition using ADB, root, ncat and DD,” Apr-2017. [Online]. Available: https://www.youtube.com/watch?v=KKkvkCgMeMA.

[41] B. Hitchcock, N.-A. Le-Khac, and M. Scanlon. Digital Investigation, “Tiered forensic methodology model for Digital Field Triage by non-digital evidence specialists,” in DFRWS 2016 Europe - Proceedings of the Third Annual DFRWS Europe, 2016, pp. S75–S85, doi: 10.1016/j.diin.2016.01.010.

[42] E. Casey, M. Ferraro, and L. Nguyen, “Investigation Delayed Is Justice Denied:Proposals for Expediting Forensic Examinations of Digital Evidence*,” Journal of Forensic Sciences, vol. 54, no. 6, Nov. 2009, doi: 10.1111/j.1556-4029.2009.01150.x

[43] M. K. Rogers, J. Goldman, R. Mislan, T. Wedge, and S. Debrota, “Computer Forensics Field Triage Process Model,” Journal of Digital Forensics, Security and Law, vol. 1, no. 2, Article 2. 2006, doi: 10.15394/jdfsl.2006.1004

[44] D. Kao, N. Wu and F. Tsai, "The Governance of Digital Forensic Investigation in Law Enforcement Agencies," 2019 21st International Conference on Advanced Communication Technology (ICACT), PyeongChang, Korea (South), 2019, pp. 61-65, doi: 10.23919/ICACT.2019.8701995.

[45] Horsman, G. (2020). The COLLECTORS ranking scale for “at-scene” digital device triage. *Journal of Forensic Sciences*, 179–189. doi:10.1111/1556-4029.14582

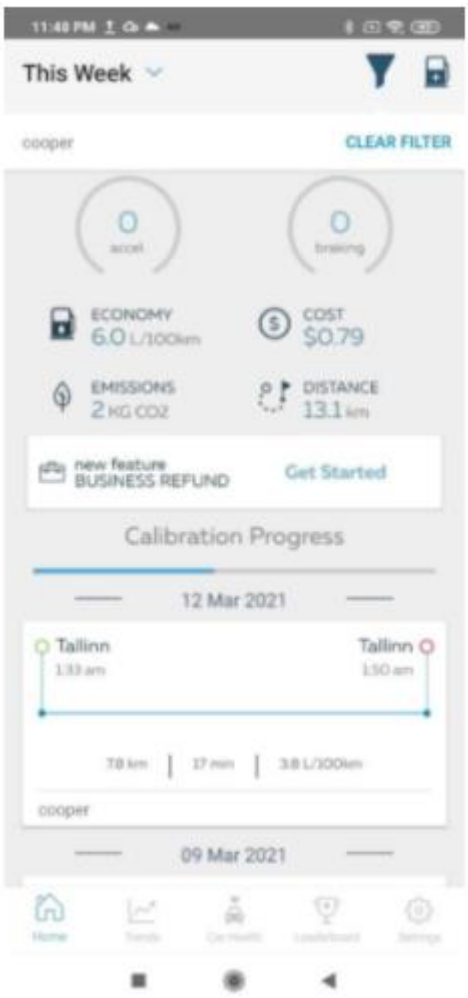
[46] Darren Quick, Kim-Kwang Raymond Choo, Darren Quick, Kim-Kwang Raymond Choo, Background and Literature Review, Big Digital Forensic Data, 10.1007/978-981-10-7763-0_2, (5-45), (2018).

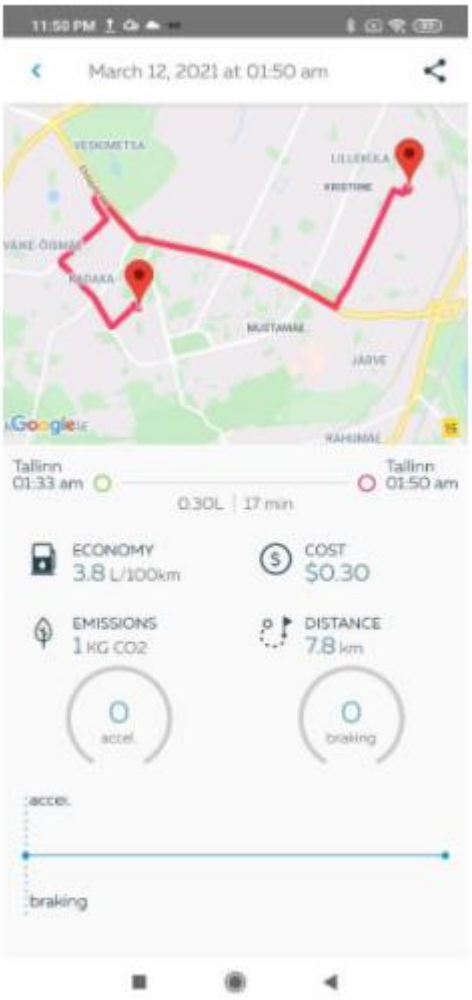
[47] Thakar, A. A., Kumar, K., & Patel, B. (2021). Next Generation Digital Forensic Investigation Model (NGDFIM) - Enhanced, Time Reducing and Comprehensive Framework. Journal of Physics: Conference Series 2020 International E- Conference on Data Analytics, Intelligent Systems and Information Security, ICDIIS 2020, 1767(1). doi:10.1088/1742-6596/1767/1/012054

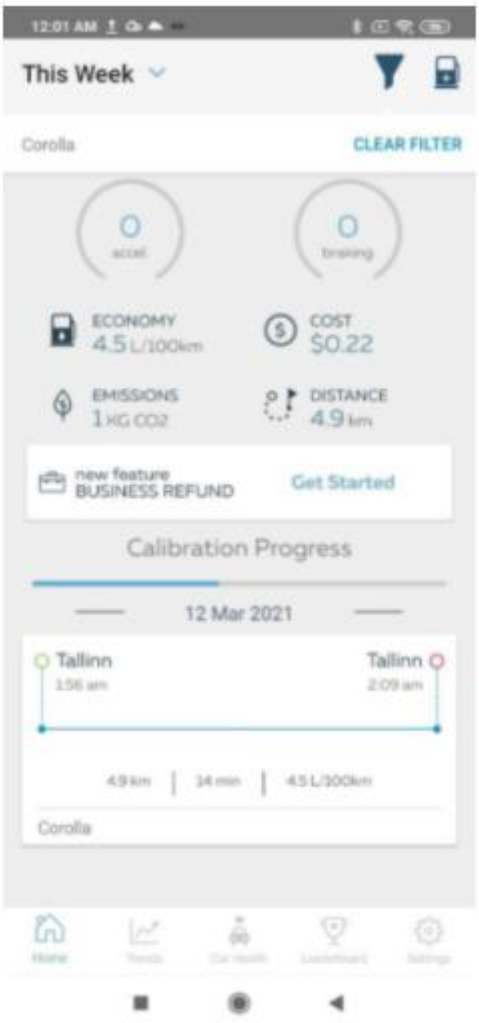
[48] R. P. Mislan, E. Casey, & G. C. Kessler, (2010). The growing need for on-scene triage of mobile devices. Digital Investigation, 6(3–4), 112–124, doi: 10.1016/j.diin.2010.03.001.

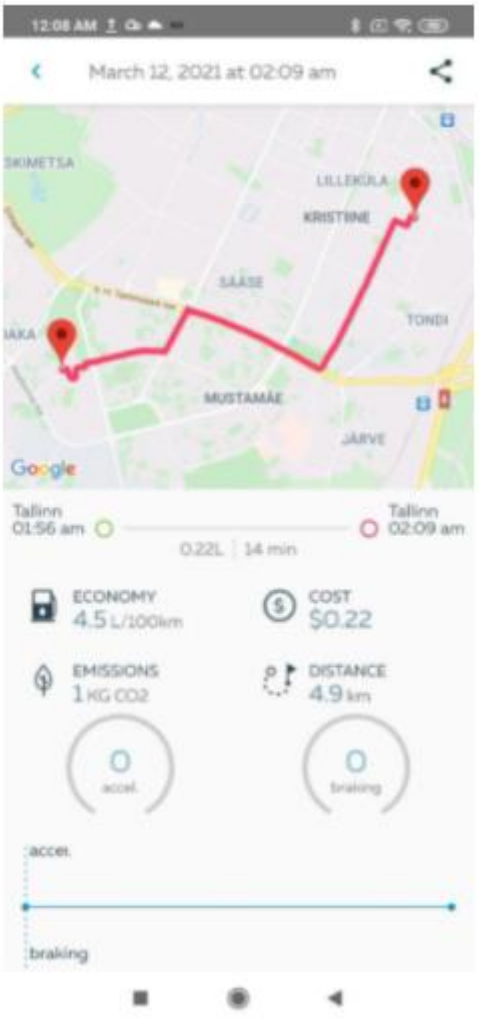
Appendices

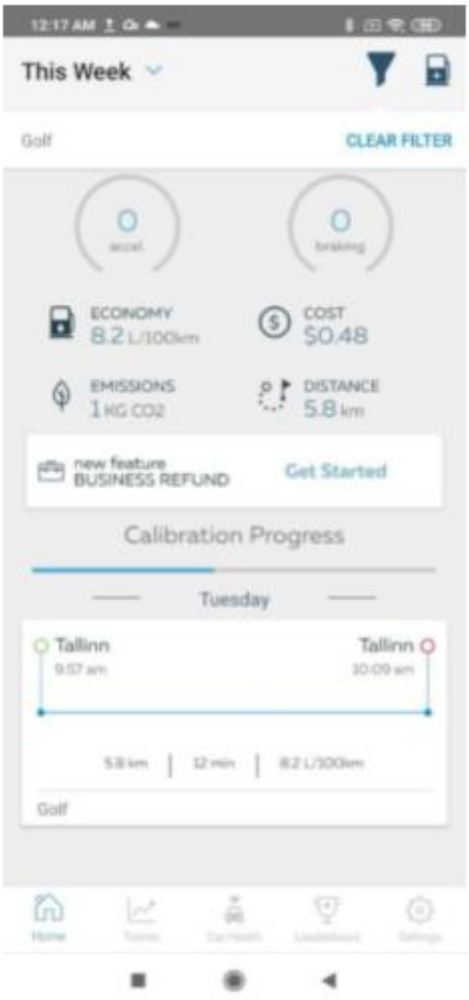
Appendix 1 – GoFar Dashboard for all three cars

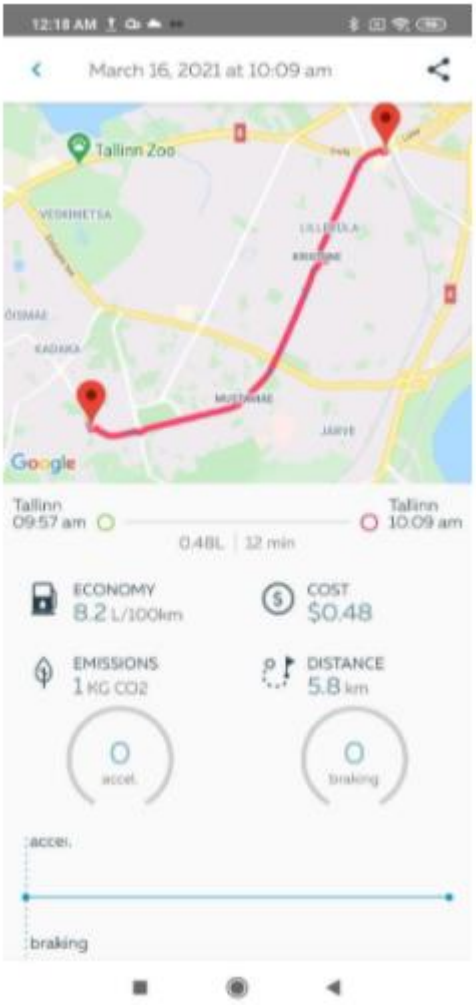





12:01 AM
This Week
Corolla
CLEAR FILTER
0
accel
0
braking
ECONOMY
4.5 L/100km
COST
$0.22
EMISSIONS
1 KG CO2
DISTANCE
4.9 km
new feature
BUSINESS REFUND
Get Started
Calibration Progress
12 Mar 2021
Tallinn
1:56 am
Tallinn
2:09 am
4.9 km | 14 min | 4.5 L/100km
Corolla


12:08 AM
March 12, 2021 at 02:09 am
SKIMETSA
LILLEKÜLA
KRISTIINE
SÄÄSE
TONDI
MUSTAMÄE
JÄRVE
Google
Tallinn
01:56 am
Tallinn
02:09 am
0.22L | 14 min
ECONOMY
4.5 L/100km
COST
$0.22
EMISSIONS
1 KG CO2
DISTANCE
4.9 km
0
accel.
0
braking
accel.
braking


12:17 AM
This Week
Golf
CLEAR FILTER
0
accel.
0
braking
ECONOMY
8.2 L/100km
COST
$0.48
EMISSIONS
1 KG CO2
DISTANCE
5.8 km
new feature
BUSINESS REFUND
Get Started
Calibration Progress
Tuesday
Tallinn
9:57 am
Tallinn
10:09 am
5.8 km | 12 min | 8.2 L/100km
Golf
Home


12:18 AM
March 16, 2021 at 10:09 am
Tallinn Zoo
Google
Tallinn
09:57 am
Tallinn
10:09 am
0.48L | 12 min
ECONOMY
8.2 L/100km
COST
$0.48
EMISSIONS
1 KG CO2
DISTANCE
5.8 km
0
accel.
0
braking
accel.
braking

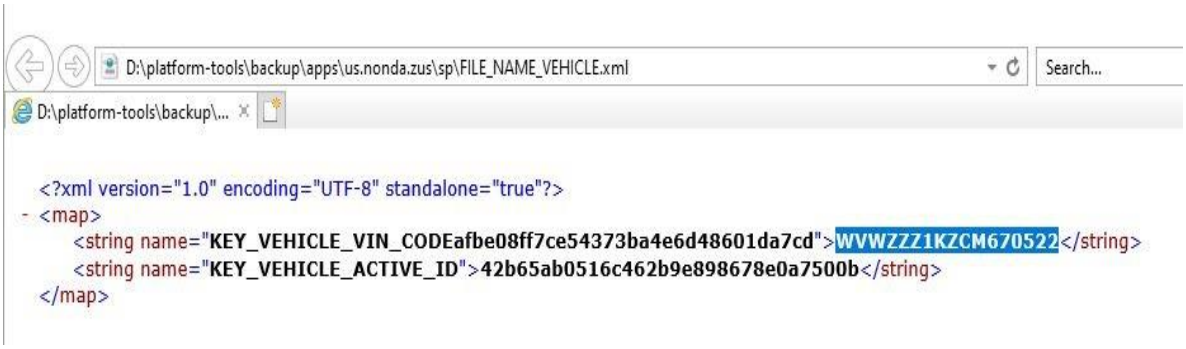
Appendix 2 - Zus Log file

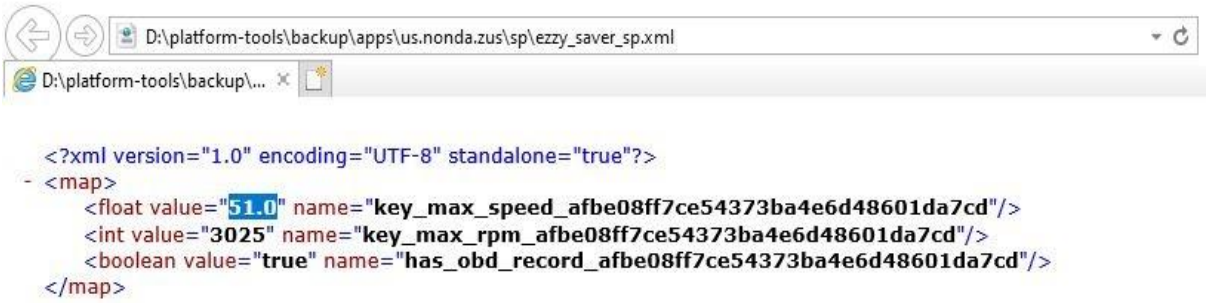
2021-02-25 11_14 - Notepad

File Edit Format View Help

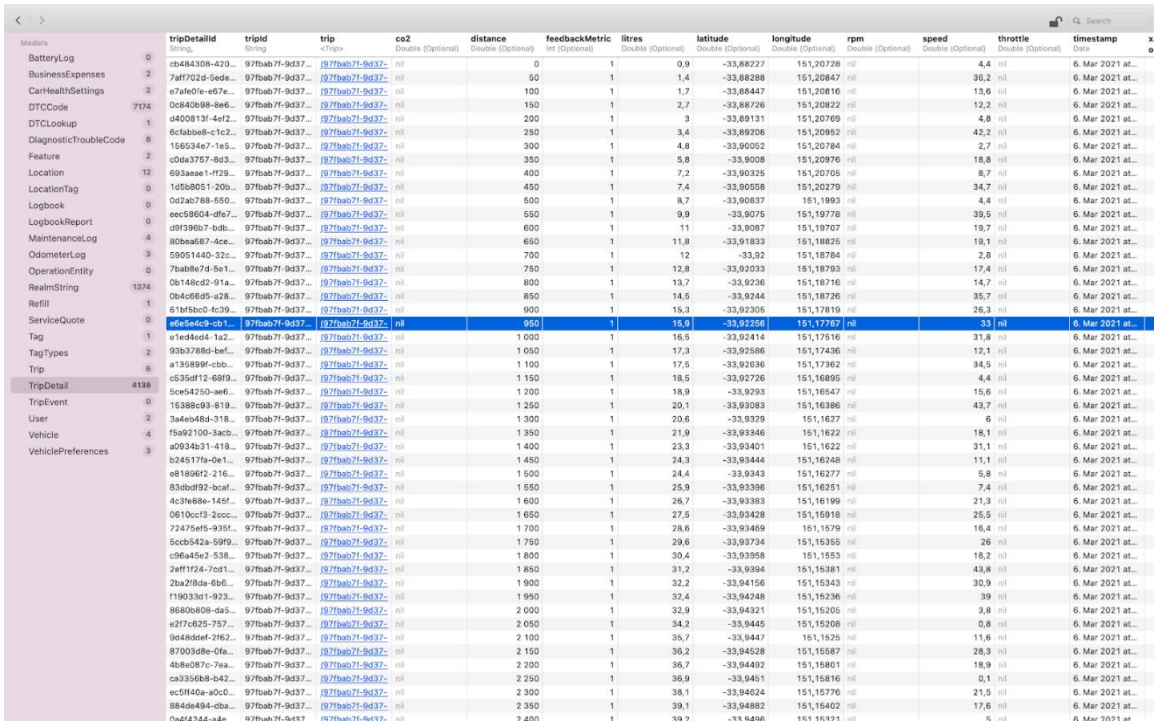
```
12:00:03        StartLocationName:Akadeemia tee 38, 12611 Tallinn, Estonia

12:00:03        EndLocationName:E. Vilde tee 121, 12612 Tallinn, Estonia

12:00:03        开始上传服务器=TripDO{localId='afbe08ff7ce54373ba4e6d48601da7cd1614245513483'id='null',
userId='null', uploaded='false', startLat=59.3983252, startLng=24.6629552, startedAt=1614245513483,
endLat=59.4033467, endLng=24.666425, endedAt=1614247128000, distance=5506.330078125, createdAt=0,
startLocationName=Akadeemia tee 38, 12611 Tallinn, Estonia, endLocationName=E. Vilde tee 121, 12612
Tallinn, Estonia, parkingFee=0.0, tolls=0.0, note=null, type=null, purpose=null}
```

Appendix 3 - Zus .xml files showing VIN and Maximum speed data



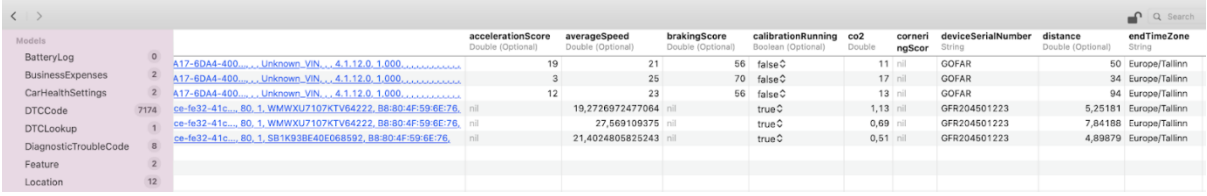


Appendix 4 - Contents of REALM file in GoFar

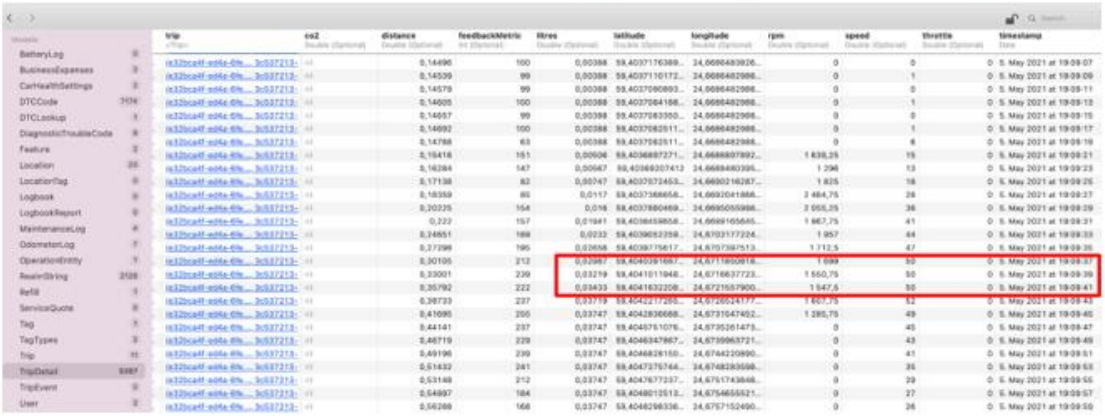


vehicleId String, Primary Key	userId String	obdConnection Details	obdProtocol Int (Optional)	vin String
28616686-dea3-4a90-a956-b1d80e285bb6	a3e064ce-fe32-41c1-b046-8da7fe28a69c	80	1	WMWXU7107KTV64222
5688EB55-9CCE-4027-9274-7A370FB56EA1	24121A17-6DA4-400D-8E5F-CC9218A1DEDD	nil	nil	Unknown_VIN
nil	nil	0	0	Unknown_VIN
0293cdb6-5d86-4a14-ace1-603dd697691d	a3e064ce-fe32-41c1-b046-8da7fe28a69c	80	1	SB1K93BE40E068592

Odo ate	displayName String	make String	model String	year Int (Optional)	engineCapacity Int (Optional)	fuelType String	fuelTrimMult Double (Optional)	transmission String
l at...	cooper	MINI	Cooper	2 018	2 000	Petrol / Gasoline	nil	Automatic
	Party Car	SUBARU	IMPREZA	1 991	7 000	Petrol	nil	Manual
	nil	nil	nil	0	0	nil	nil	nil
21 a...	Corolla	Toyota	Corolla	2 020	1 600	Petrol / Gasoline	nil	Automatic

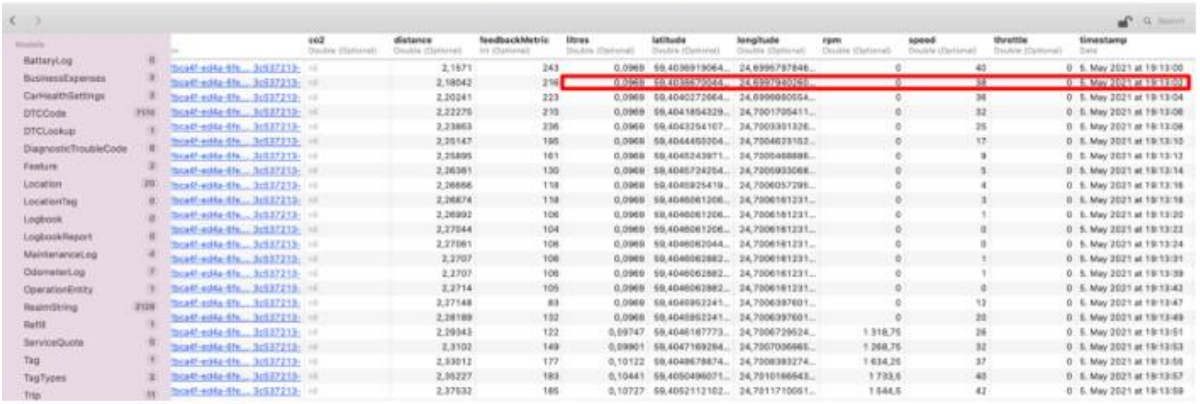


Appendix 5 – Additional screenshots from validation test analysis



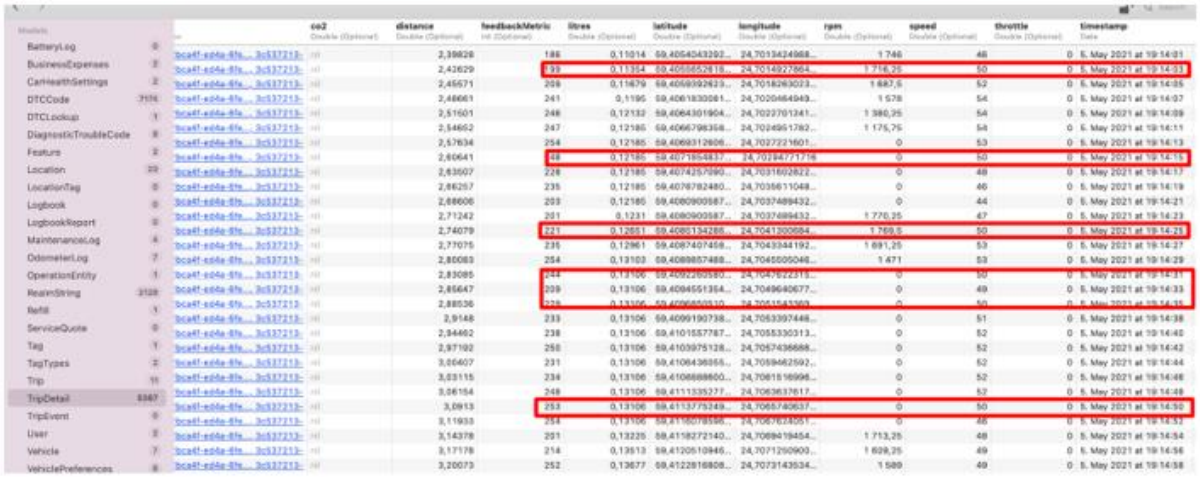

19:13
Kõmeedi
Liblika
Käbliku
Luige
Vindi
Nepi
Tedre
Rästa põik
Kao
Õõtuse
Räägu
Võrse
Püü
Nõmme põik
Muti
Tildri
Väablase
Sõpruse pst
Pardiliik
Tallinn
A. H. Tammsaare tee
38
km/h
Select vehicle type


Home
19:13
SKYPLUS


co2
distance
feedbackMetric
litres
latitude
longitude
rpm
speed
throttle
timestamp
BatteryLog
BusinessExpenses
CarHealthSettings
DTCCode
DTCLookup
DiagnosticTroubleCode
Feature
Location
LocationTag
Logbook
LogbookReport
MaintenanceLog
OdometerLog
OperationEntity
RealmString
Refill
ServiceQuote
Tag
TagTypes
Trip

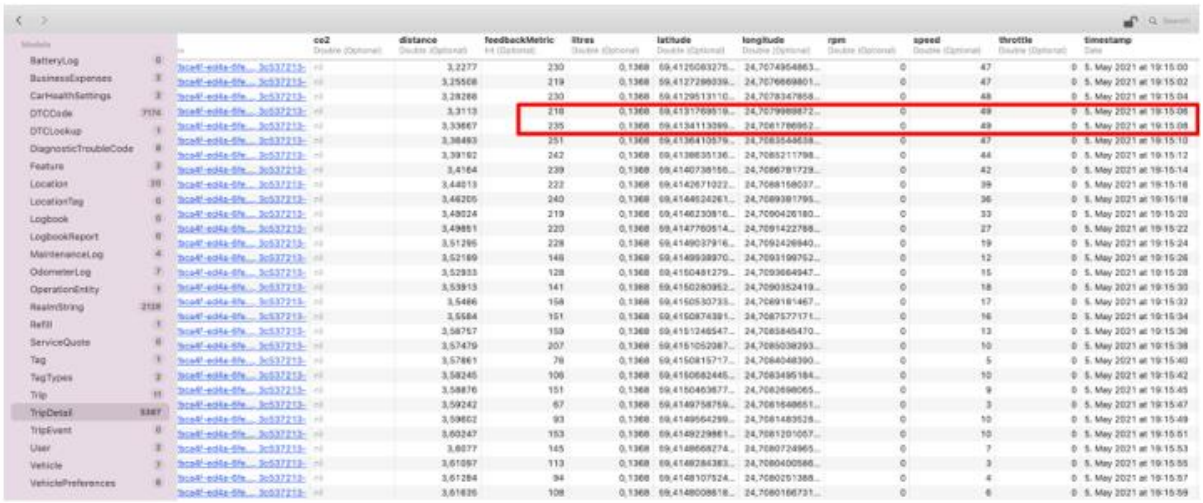

19:14
Tallinn
Vuti
Mõtuse
Varese
Sõpruse pst
Rästa
Räägu
Võrse
Võrsiku
Linnu tee
Nõrgi
Muti
Tildri
Vaablase
50
km/h
Select vehicle type


Home
SKYPLUS




19:15
Timuti
Kungla
Öle
Heina
Tehnika
Kevade
Mulla
Telliskivi
Wismari
Roopa
Paldiski mnt
Kullerkupu
Endla
Meleka
Haigru
Liblika
Lennus
Mooni
Lehe
Hüübi
Tedre
Kännu
Vuti
Sõpruse pst
Rästa
Mõtuse
Varese
Võrse
Võrsil
50
km/h
Select vehicle type


Home
19:15
FM station list
SKYPLUS


co2
distance
feedbackMetric
litres
latitude
longitude
rpm
speed
throttle
timestamp
BatteryLog
BusinessExpenses
CarHealthSettings
DTCCode
DTCLookup
DiagnosticTroubleCode
Feature
Location
LocationTag
Logbook
LogbookReport
MaintenanceLog
OdometerLog
OperationEntity
RealmString
Refill
ServiceQuote
Tag
TagTypes
Trip
TripDetail
TripEvent
User
Vehicle
VehiclePreferences