Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

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

DFRWS USA 2024 - Selected Papers from the 24th Annual Digital Forensics Research Conference USA

Hit and run: Forensic vehicle event reconstruction through driver-based cloud data from Progressive's snapshot application

Abdur Rahman Onik[a,b,*], Trevor T. Spinosa[a,b], Abdulla M. Asad[a,b], Ibrahim Baggili[a,b]

[a] Baggili(i) Truth (BiT) Lab, Center of Computation & Technology, Baton Rouge, LA, USA
[b] Division of Computer Science & Engineering, Louisiana State University, Baton Rouge, LA, USA

ARTICLE INFO

Keywords:
Driving insurance application (DIA)
Vehicle forensics
Digital forensics
Snapshot
Driving patterns
Cloud forensics

ABSTRACT

Driving Insurance Applications (DIAs) have emerged as a valuable resource in the ever-evolving digital landscape. Automobile owners are storing extensive data on driving behaviors and patterns. This study pioneers the forensic analysis of Progressive's Snapshot application, focusing on the extraction and potential forensic use of data that remains inaccessible through the mobile application's interface. In our approach we focused on four research questions: *How accurate is location and speed data collected by Progressive Snapshot?*, *What forensically relevant data can we extract from the Progressive Cloud that is unavailable to the user from the mobile application interface?*, *Can we employ anti-forensics techniques, specifically fake location data, to create false trip details?*, *Can we reconstruct a hit-and-run scenario from trip event details?* To answer these questions, we developed PyShot, a Python-based open-source tool, to extract data from the Progressive cloud. Our tests confirmed Snapshot's accuracy in recording speed and location. Despite efforts to fake the Global Positioning System (GPS) location, the cloud still maintained accurate records. PyShot revealed more detailed driving data, like dangerous maneuvers and distracted driving, compared to the mobile application. This study also explores the forensic reconstruction of hit-and-run incidents, using a mannequin and focusing on Progressive's server data. Analyzing event categories, geographical coordinates, and timestamps provides insights into the capabilities and constraints of this application in forensic investigations. The findings offer valuable insights into the forensic capability of data retained by DIAs, contributing to their potential use in forensic investigations.

1. Introduction

Progressive introduced Snapshot in 2008 as a plug-in device designed to gather car speed data. After each trip, the collected data is transmitted back to Progressive. Using this information, the system identifies events such as rapid acceleration and hard braking. Drivers who demonstrate safe driving habits are eligible for reduced insurance rates. By March 2014, Progressive had accumulated over 10 billion miles of driving data through Snapshot (Progressive Insurance 2014).

On September 2, 2015, Progressive launched the Snapshot mobile application for both iOS and Android operating systems. This software not only retained the features of the Snapshot plug-in device but also offered drivers a personalized trip summary. This summary includes a map of a driver's journey, the percentage of phone usage during the trip, and an overall rating of their driving performance (Progressive Insurance 2015).

With the rapidly increasing integration of Internet of Things (IoT) devices into our daily lives, the data they collect has raised numerous privacy concerns (Arabo et al., 2012). For example, Snapshot generates a map after each trip, displaying details such as the routes taken, driving behavior, and the timing of each trip. In the event of a data breach, the user's private lifestyle and habits could be exposed, posing a significant privacy risk.

At the time of writing, Progressive is considered one of the largest insurance companies. The company's market share grew from 13.8 % in 2021 to 14.1 % in 2022 (Rudden 2023). In 2022, Progressive recorded $51.1 billion of net premiums written which is a 10 % increase growth over the prior period (Progressive Corporation 2023). Given the company's market presence, it is relevant to understand how its data may be used in a forensic setting.

To that end, our goal in this work is to answer the following research questions: **RQ1:** *How accurate is location and speed data collected by Progressive Snapshot?* **RQ2:** *What forensically relevant data can we extract from the Progressive Cloud that is unavailable to the user from the mobile*

* Corresponding author. Baggili(i) Truth (BiT) Lab, Center of Computation & Technology, Baton Rouge, LA, USA.
E-mail addresses: aonik1@lsu.edu (A.R. Onik), tspino2@lsu.edu (T.T. Spinosa), aasad1@lsu.edu (A.M. Asad), baggili@gmail.com (I. Baggili).

https://doi.org/10.1016/j.fsidi.2024.301762

Available online 5 July 2024

application interface? **RQ3:** *Can we employ anti-forensics techniques, specifically fake location data, to create false trip details?* **RQ4:** *Can we reconstruct a hit-and-run scenario from trip event details?*

Our contributions are as follows:

- We introduced a new data source for digital forensic investigations by leveraging cloud-based data obtained from the driving insurance application, Progressive's Snapshot. This work introduces a novel data source that enhances the accuracy and comprehensiveness of vehicle event reconstructions.
- We carried out an initial forensic examination of Progressive's cloud infrastructure, focusing on how it communicates to uncover and utilize undocumented and concealed Application Program Interfaces (API). This technique enabled us to collect operational data without the need for direct interaction with the car or mobile device.
- We developed PySHot, an open-source application built using Python that uses the APIs we identified. This tool facilitates secure and authenticated interaction with Progressive's cloud infrastructure, allowing for the extraction of artifacts while preserving their forensic integrity. PySHot is now publicly accessible on GitHub.[1]
- We simulated a hit-and-run case study to demonstrate the forensic capabilities of our tool. This study illustrated how PySHot can create a chronological sequence of events that occur during a journey and produce a comprehensive map that offers insights beyond what is available through the Snapshot application.

The structure of this paper is as follows. Section 2 provides ongoing research on vehicle forensics and other related works. In section 3, we present the procedure we followed. We present the artifacts that we retrieved as APIs in Section 4. We introduced the Python application in Section 5. Section 6 evaluates and discusses the results obtained followed by a case study in Section 7. In Section 8, we present the limitations of our work. Finally, the work is concluded in Section 9.

2. Related work

As GPS technology undergoes rapid advancements, precise location data holds growing significance in forensic investigations and criminal cases (Berman et al., 2015). This section will delve into related research conducted on other GPS devices and will closely examine the methodologies employed in data acquisition.

2.1. GPS and Mapping forensics

Numerous open-source tools now enable investigators to extract digital location data. In a recent study by Pace et al. (2023), an application was developed to retrieve memory data from devices controlling a Tile, a Bluetooth tracker primarily used for locating items like backpacks and car keys. Forensic analysis of the Tile ecosystem exposes sensitive user data, including geolocation coordinates. The authors introduced an open-source tool to assist investigators in efficiently parsing and mapping forensically relevant geolocation artifacts.

GPS tools have undergone rigorous testing in numerous studies to verify the accuracy of the acquired location data for forensic investigations. In a study conducted by Listi et al. (2007), materials were dispersed in open areas, and the GPS data proved accurate within half a meter when logging the location data of the entities. Given the advancements in GPS tools since this experiment, it is evident that location data obtained through these technologies is sufficiently precise for cases related to vehicles.

Moore et al. (2017) delved into the transition from traditional navigation devices to smartphones, specifically investigating popular mapping applications like Google Maps, Apple Maps, Waze, MapQuest, Bing, and Scout on Android and iOS. The study revealed that smartphones store substantial user navigation data, including addresses and latitudevlongitude points. A key outcome was the development of a forensic tool capable of parsing data from these applications, and creating maps from the acquired location information.

Additional experiments have been conducted on location-sharing applications. Bays and Karabiyik (2019) specifically addressed the prevalence of emerging third-party applications facilitating user location data sharing. The authors focused on scrutinizing *Life360* and *iSharing* and identified their recoverable GPS artifacts.

2.2. Vehicle Forensics

The importance of digital data in vehicles cannot be overstated. Many crimes, including hit-and-run incidents, suspect escape routes, and traffic violations, are directly related to vehicle use.

Feng et al. (2019) implemented a non-invasive approach to gather forensic data originating from autonomous vehicles in smart city environments. The method they used involves establishing a Transport Layer Security handshake to form secure communication between the vehicle and the infrastructure.

A study by Mahr et al. (2022) delved into digital evidence produced by Android Auto and Apple CarPlay. It introduced an open-source tool as a proof-of-concept to assist investigators in automatically extracting relevant artifacts from these applications. Investigating crashes requires an understanding of how applications store data, providing essential details about the incident's circumstances. This includes information such as the last application displayed on the head unit, the layout of the user's home display screen, and other evidence that indicates the use of mobile device features while driving.

Le-Khac et al. (2020) conducted a case study involving the collection and analysis of forensic data from the entertainment system of a Volkswagen car. The study explores potential hardware and software solutions for acquiring forensic artifacts from such vehicles. Additionally, the research delves into the analysis of mobile data traffic from Audi, Volkswagen, and BMW cars. Ebbers et al. (2021) analyzed Audi, BMW, Ford, Mercedes, Opel, Seat, Tesla, and Volkswagen's applications. The trips, refueling processes, parking positions, and vehicle access were successfully reconstructed.

Many vehicles, especially in recent years, have become digitized and use electronic means to handle a plethora of functions, ranging from navigation and entertainment to engine diagnostics and safety features. However, vehicle data is not internationally standardized, making it difficult to perform analysis (Kopencova and Rak 2020). Thus, performing forensic analysis on other GPS applications that are not associated directly with the vehicle can be simpler and more effective.

Our research contributes to this field by being the first, to our knowledge, to focus on cloud data from usage-based insurance applications. We demonstrate how to reconstruct event data from these applications during forensic investigations of accidents or crime scenes, thereby adding a new dimension to vehicle forensics.

2.3. Network application analysis

Forensic data can be acquired through network traffic. Progressive Snapshot, like many applications, transmits sensitive data over the network. Onik et al. (2024) intercepted and utilized this network data for forensic analysis. The study introduced PyRoomba, an application designed to extract and display specific details related to the crime scene from the network traffic data, which may not be directly accessible through the Roomba application.

In their analysis, Hassenfeldt et al. (2019) utilized web-scraping techniques to extract artifacts from health and fitness web applications. The study revealed the recovery of extensive data, raising significant privacy concerns. The authors introduced an open-source tool to aggregate forensic data from the analyzed applications.

[1] https://github.com/npseudo1806/pyShot/tree/master.

Karpisek et al. (2015) performed network forensic analysis on the popular messaging application *Whatsapp*. The authors were able to decrypt the network data and find artifacts related to the newly added call feature while also creating an open-source tool to visualize the communication between callers. They employed similar techniques to those used in our analysis of the Progressive Snapshot application.

Walnycky et al. (2015) forensically acquired and analyzed data stored on devices and network traffic from 20 popular instant messaging applications for Android. The study reveals that 16 out of the 20 applications tested allowed for the reconstruction of some or the entire message content, highlighting potential shortcomings in the security and privacy measures of these apps. While this poses concerns for user privacy, it presents opportunities for digital forensic practitioners in evidence collection. The research identifies specific features within these instant messaging applications that leave traces, showcasing the ability to reconstruct or partially reconstruct suspect data through both network and device forensics.

Yarramreddy et al. (2018) focused on the forensic analysis of immersive Virtual Reality (VR) systems and their associated social applications. A key focus of the study is the detailed analysis of network traffic and its implications for forensic discoveries. Particularly, the research highlights the vulnerability of certain applications, like Bigscreen, to potential Man-in-the-Middle (MitM) attacks due to unencrypted communications. This vulnerability exposes the application to security risks, emphasizing the importance of network analysis in identifying potential threats and safeguarding user information.

In their research, Denton et al. (2017) reversed General Electric's protocol for the GE Fanuc Series 90–30 Programmable Logic Controller (PLC). They unveiled the Service Request Transport Protocol (SRTP), previously hidden for security. The study also introduced a direct communication software tool for the PLC, eliminating the need for an intermediate server. While the tool allows non-intrusive register reading, it exposes potential register manipulation, enabling actions like shutting down the PLC or altering processes.

While there is an abundance of past research on the forensic analysis of mobile applications, no prior studies have specifically addressed the forensic analysis of driver-based insurance applications and their effectiveness in reconstructing vehicle events.

3. Methodology

Our methodology involved four phases: preparation, identification of cloud APIs, tool creation, and validation.

3.1. Preparation

The preparation phase for our analysis involved obtaining the necessary hardware and installing the required software on the devices (Table 1). We first acquired an Android smartphone (Samsung Galaxy S10+) and the official Progressive Insurance mobile application, which includes the Snapshot feature, was downloaded from the Google Play Store.

In our study of the Snapshot application's HTTPS traffic, we employed the *apk-mitm* tool, a Command Line Interface (CLI) application tool to prepare Android APK files for HTTPS inspection. This tool enabled the modification of the Progressive application by decoding the APK, adjusting its Network Security Configuration to accept user-added certificates, disabling certificate pinning mechanisms, and then re-encoding and signing the altered APK. This process enabled us to install a modified version of the Progressive Insurance application on our test phone, which was vital for our HTTPS traffic analysis (See Fig. 1a).

To monitor the application's HTTPS traffic, we established a network connection between our test phone and a Windows PC via the mobile hotspot feature, ensuring both devices operated on the same network. Subsequently, we installed *mitmproxy* on the Windows PC and the previously modified Progressive Snapshot application on our test phone. After successful installation and user authentication within the application, we commenced our analysis of the application's network traffic. The integration of the *mitmproxy* certificate into both the Android phone and Windows PC was a critical step, allowing us to effectively monitor and analyze the encrypted traffic between the mobile application and its servers. This setup provided a comprehensive view of the application's network interactions.

Table 1
List of apparatus.

Hardware/ Software	Use	Company	Software/ Model Version
Galaxy s10+	Progressive Android Application	Samsung	Android 13.0.0
iPhone 6s	Progressive ioS Application	Apple	iOS 15.7.1
Galaxy Tab A7 Lite	Global Positioning System (GPS) Logger	Samsung	Android 13.0.0
Windows PC	MITM Web	Microsoft Corporation	Windows 10.0.22621
Progressive Application	Application analysis	Progressive	Version-3.92
APK-MITM Tool	Modification of Android Package Kit (APK) files to enable Hypertext Transfer Protocol Secure (HTTPS) inspection	Open Source	v1.2.1
MITMProxy Tool	An interactive HTTPS proxy tool for intercepting and analyzing network traffic	Open Source	mitmproxy 10.1.0
GPS Logger	Android app to track location and speed	BasicAirData	Version- 129
Fake GPS Location Spoofer	Android application to fake location data	IncorporateApps	Version 5.8.1

3.2. Identification of undocumented APIs

In the next stage of our analysis, we employed *mitmweb*, the web interface of the *mitmproxy* tool, to observe and analyze the HTTPS communication between the Snapshot and its server. This step was important because the traffic was Transport Layer Security (TLS) encrypted, and a MitM approach was necessary to decrypt and examine the data. Our examination of the network traffic aimed to identify all API requests. We identified an API as significant if its JavaScript Object Notation (JSON) response included details like timestamps, user metadata, coordinates, or any related information. We carefully recorded all identified APIs requests (See Fig. 1b), including their headers, query parameters, and responses. These interactions were stored as "Flows" - a sequentially arranged list of network requests and responses that can be accessed and analyzed through the *mitmweb* interface. Subsequently, we used these insights about the APIs to create a forensic tool named PyShot (See Fig. 1c).

3.3. Validation

To validate the PyShot tool, controlled experimental drives were conducted in a spacious, empty parking area using a car and a mobile phone. These experiments aimed to establish ground truth data and ensure safety and control. The testing comprised three trips: the first represented standard driving behavior for comparison, while the second and third trips simulated phone usage while driving, including receiving calls and executing specific maneuvers like turns without braking and hard braking. Following each trip, we compared the map produced by PyShot against the actual driving ground truth data. Furthermore, we conducted an additional analysis comparing the map generated by the

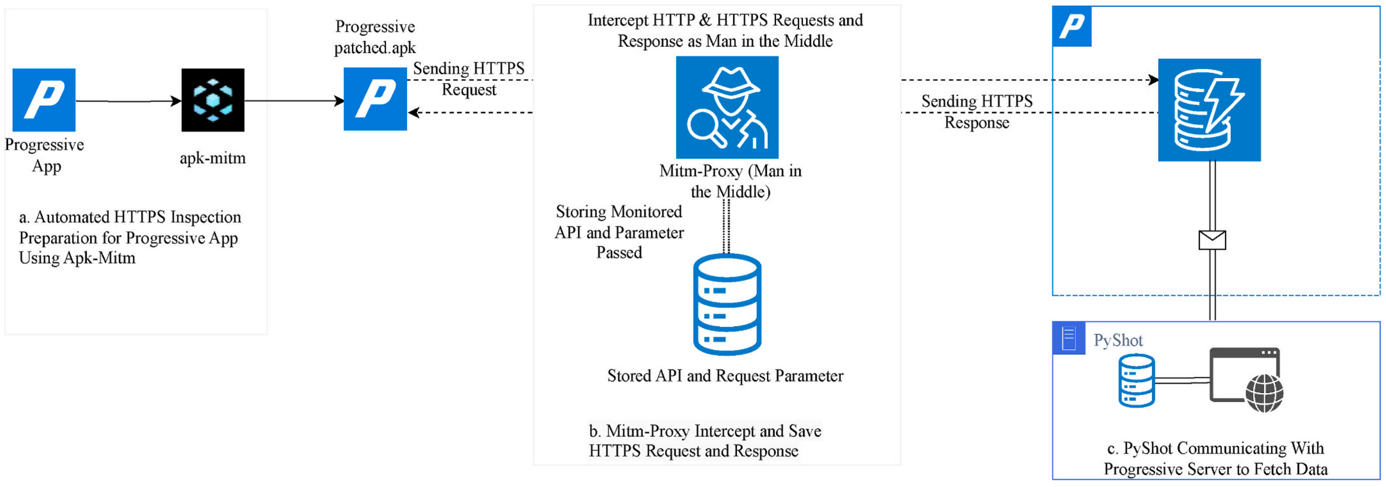

Progressive App
apk-mitm
a. Automated HTTPS Inspection Preparation for Progressive App Using Apk-Mitm
Progressive patched.apk
Sending HTTPS Request
Intercept HTTP & HTTPS Requests and Response as Man in the Middle
Mitm-Proxy (Man in the Middle)
Storing Monitored API and Parameter Passed
Stored API and Request Parameter
b. Mitm-Proxy Intercept and Save HTTPS Request and Response
Sending HTTPS Response
PyShot
c. PyShot Communicating With Progressive Server to Fetch Data

Fig. 1. Network traffic analysis of progressive application (Onik et al., 2024).

PyShot application with that produced by the Snapshot application, which offers fewer details than our application.

3.4. PyShot: A forensic tool

We developed PyShot, a forensic web application created using Python 3.11, designed to extract detailed driving data from the Progressive cloud via undocumented APIs without requiring direct interaction with the device. PyShot is developed not only to collect comprehensive trip information such as start and end coordinates and all trips conducted by an individual but to also capture specific incidents during each trip that are not available in the Snapshot application, including hard brakes, dangerous maneuvers, phone usages, calls, and distractions. The PyShot application documents these events with precise timestamps and locations, providing a thorough understanding of driving behavior and patterns which is of forensic value.

4. Undocumented Snapshot APIs

In analyzing the Progressive Snapshot application, we used `mitmweb` to scrutinize its internet traffic. The application predominantly utilizes secure, encrypted connections for data transfer, and the returned data is typically in JSON format. Subsequently, we retrieved specific data from the Progressive cloud server by accessing the APIs with valid user credentials. The Progressive cloud service, like other cloud services, employs specific undisclosed APIs for data transfer. Our study is the first to unveil these previously undocumented APIs, revealing valuable insights into system settings and user activities through the analysis of the returned data.

4.1. Login API

In our analysis, the Progressive API login mechanism revealed a structured process. A POST request is made to the Login API (see Table 2 row 1) for user authentication. Essential headers for authorization include `X-PGRIdentifier`, `User-Agent`, `api_key`, `Host`, supporting the identification and security of the request. The request body, formatted in JSON, carries login credentials. The response, also in JSON format, is integral for forensic analysis. It includes an `accessToken` and links for subsequent API interactions. This token is key for accessing user-specific resources and actions within the Progressive ecosystem.

4.2. Telematics registration API

We observed that the application requires mobile phone registration using a phone number to activate and access the Snapshot feature. Notably, each session requires phone number verification to access Snapshot data, enhancing security measures. Our investigation successfully identified the Registration API, which is instrumental in initiating a request to Progressive's server for an OTP delivery to the registered phone number. The API accepts the phone number, labeled as registrationCode, in the JSON request body. Subsequently, an OTP is sent to the provided number. Critical headers for authorization during this API interaction include `api_key`, `Host`, `User-Agent`, `Connection`, and `Content-Type`. These headers are necessary for successful communication and authentication with the server, and their forensic examination provides insights into the security protocols and data transmission methods of the Progressive mobile application.

Table 2
Snapshot endpoint APIs and forensic relevance.

No.	API Category	Undocumented APIs	Artifacts
1	Login	https://api.progressive.com/onlineaccount/v1/accesstoken	Access token is retrieved to use as authorization for other APIs
2	Telematics Registrations	https://telematics.api.progressive.com/TelematicsRegistrations/Telematics/MobileDevices/MobileRegistrations/v1/challenge	Phone number entered to get a One Time Password (OTP)
3	Telematics Authentication	https://telematics.api.progressive.com/TelematicsRegistrations/Telematics/MobileDevices/MobileRegistrations/v1/authenticate	Phone number and OTP sent to server for verification
4	Device and Driver Authentication	https://api-snapshot.cens.io/ubi3/v1/companies/2212aa38-0ada-40a3-ace8-fd1096302d8d/oauth	Device id, Driver id and Access token is retrieved. These are used as Json request body in the Driver Id and Trip details APIs
5	Trip Details	https://api-snapshot.cens.io/mobile/v3/driver/driver_id/trips/start/start_time/end/end_time	Information regarding Driver's trip details and coordinates to reconstruct the map and timeline

4.3. Telematics authentication API

After obtaining a code from the Telematics registration, the Telematics authentication API was engaged to submit the OTP along with the mobile number. This POST request was securely authenticated using key headers: api_key, Host, User-Agent, Connection, and Content-Type. The response from this request yielded critical data elements: `apiToken`, `firstname`, `mobileParticipantId`, and `vehicleList`. The `apiToken` plays a pivotal role in accessing detailed driver information and subsequent trip data. Each user is uniquely identified by the `mobileParticipantId`, which is utilized in both device and driver identification APIs as `user_id`. The forensic significance of these data elements lies in their ability to trace user activity and authenticate device interactions within the Progressive framework.

4.4. Device and driver authentication API

Utilizing the same authorization headers in 4.3, a JSON request is sent containing device and application details, including the `access_token` (previously obtained `apiToken` from Telematics Authentication API) and `user_id` (the mobileParticipantId). The APIs URL includes a Progressive company ID - 2212aa38-0ada-40a3-ace8-fd1096302d8d, which will remain unchanged as it is a unique identifier for Progressive Insurance, and the response yields vital forensic data such as `device_id`, `driver_id`, and a new `access_token`. This `access_token`, essential for retrieving trip details, has a limited validity (86,400 s) and is accompanied by a `refresh_token`. The `driver_id` obtained is also necessary for accessing detailed trip information in 4.5.

4.5. Trip details API

The Trip Details API, accessible with `driver_id`, `start_time`, `end_time` (in Unix Timestamp), and `access_token` from section 4.4, delivers detailed trip metrics. The response encompasses a wide array of data, including trip timestamps, locations with coordinates, battery levels, adjusted start points, distance in meters, and braking distance. Additionally, it covers usage and call durations, trip ID, an event list, transit mode, corrections, exit door details, processing timestamp, operator mode, vehicle association, total events, "dings", data validity, distraction counts and durations, focus ratings, professional driving predictions, distracted driving ratings, and phone usage percentage.

The provided data is significant for mapping trips and understanding each driver's pattern. Key to this analysis is the `Events` attribute, which records specific incidents during the trip such as distractions, speeding incidents, dangerous maneuvers, and hard braking with precise timestamps, locations, and event types. If the Snapshot application finds no events for a length of time, the application will periodically store location and speed data and label it as Valid Period. This detailed data aids in constructing accurate driver profiles and recognizing behavior patterns for performance assessment.

5. PyShot: A DIA forensic tool

PyShot is a forensic tool, built using Python 3.11, designed to enhance the capabilities of the standard Progressive Snapshot application. It provides comprehensive trip details with a focus on forensic accuracy, ensuring data integrity and secure access. The following are the key features of PyShot:

- **Telemetrics Registration and OTP Authentication:** PyShot utilizes a robust security framework similar to the Progressive Insurance application, requiring investigators to authenticate via OTP sent to their mobile number. This process ensures secure and user-specific access to trip information.
- **Local Database Management:** Upon authentication, PyShot creates a local database for each user, organizing trip data by the driver's name and the latest timestamp. This approach minimizes reliance on the Progressive server and ensures uninterrupted access to trip details.
- **Comprehensive Trip History:** PyShot captures detailed logs of all trips, including start and end points, precise times, and the percentage of phone usage. It also records critical driving events like hard braking, phone usage, dangerous maneuvers, distractions, and speeding, pinpointing their exact locations and times.
- **Comprehensive Timeline Analysis:** The tool provides a detailed timeline of the driver's presence, helping to determine an individual's precise location during specific events.
- **Map Visualization:** PyShot uses the Google Maps API for its map visualization capabilities, enabling users to not only mark and color-code various events on the map but also to switch between Map and Satellite views. This flexibility aids the user in correlating events with precise times and locations, providing a comprehensive perspective that is particularly valuable in forensic investigations such as hit-and-run incidents. Fig. 2 presents a visual representation of the PyShot interface.

6. Evaluation and discussion

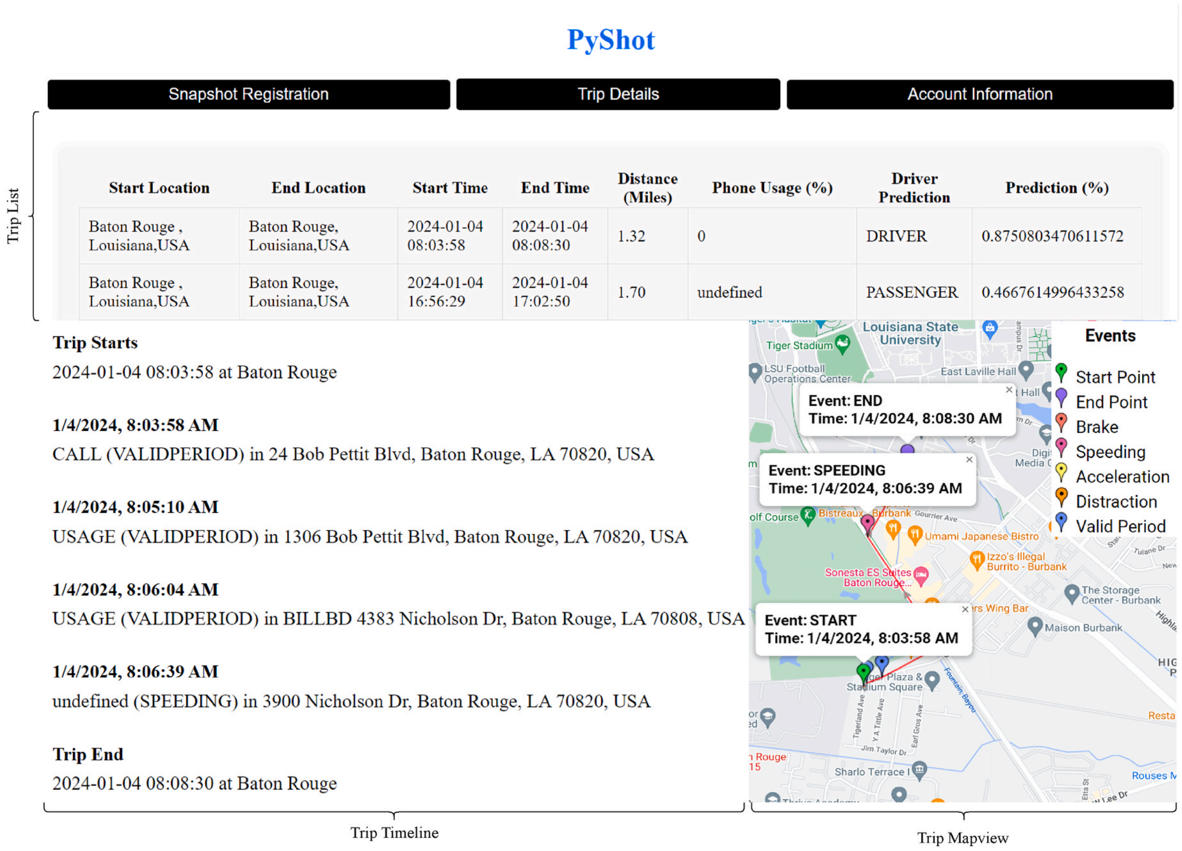
In this section, we present a series of experiments conducted that assist in the understanding of the forensic capabilities of Snapshot and the Progressive cloud data accessed via PyShot. Our analysis was structured around four research questions (RQs). We examined location and speed data accuracy, looked at the forensic relevance of cloud data retrieved from undocumented APIs, assessed the application's resilience against anti-forensic techniques such as location spoofing, and examined the feasibility of reconstructing specific driving incidents for forensic analysis.

*6.1. **RQ1:** location data and speed accuracy analysis*

To evaluate the accuracy of geolocation and speed data, an investigation was carried out by collecting data during a trip using the Snapshot application, which stores its data in the cloud. This data was subsequently retrieved from the cloud using PyShot for analysis. The comparison involved data collected simultaneously by the Snapshot application on an Android phone and a GPS Logger application installed on a Samsung Galaxy Tablet. Both devices were placed side by side during the trip to ensure similar environmental conditions and minimize discrepancies due to device placement, although slight variations in location coordinates were anticipated.

Two graphical visualizations were generated from the dataset collected using PyShot from Progressive's cloud and a GPS Logger application. The first graph, Fig. 3a, illustrates the geographical distribution of recorded locations, showing a close alignment between the GPS Logger and data points collected by the Snapshot application along the travel route. The second graph, Fig. 3b, presents the analysis of vehicle speed over time, highlighting the Snapshot application's capacity to track speed trends that are consistent with the more frequently updating GPS Logger. The GPS Logger application records location and speed data every second, while Snapshot records every few seconds, labeled "VALIDPERIOD." The analysis showed that even though the Snapshot application didn't collect data as often as the GPS Logger, which records data every second, it still demonstrated proficiency at tracking the vehicle. Even with less frequent updates, the Snapshot application was able to correctly capture important driving actions like speeding up, braking hard, and making sharp turns. This means the Snapshot application can provide valuable information for investigations, despite rounding off speed numbers to the nearest whole number (for example, changing 7.19 mph to 7 mph).

From a forensic perspective, consistency and accuracy in documenting location and speed are important for reconstructing events or behaviors in investigations. The Snapshot application's ability to capture critical events despite lower sampling rates and its consistency with the GPS Logger data indicate its potential utility in digital forensic investigations.



PyShot

Snapshot Registration | Trip Details | Account Information

Trip List

Start Location	End Location	Start Time	End Time	Distance (Miles)	Phone Usage (%)	Driver Prediction	Prediction (%)
Baton Rouge , Louisiana,USA	Baton Rouge, Louisiana,USA	2024-01-04 08:03:58	2024-01-04 08:08:30	1.32	0	DRIVER	0.8750803470611572
Baton Rouge , Louisiana,USA	Baton Rouge, Louisiana,USA	2024-01-04 16:56:29	2024-01-04 17:02:50	1.70	undefined	PASSENGER	0.4667614996433258

Trip Starts
2024-01-04 08:03:58 at Baton Rouge

1/4/2024, 8:03:58 AM
CALL (VALIDPERIOD) in 24 Bob Pettit Blvd, Baton Rouge, LA 70820, USA

1/4/2024, 8:05:10 AM
USAGE (VALIDPERIOD) in 1306 Bob Pettit Blvd, Baton Rouge, LA 70820, USA

1/4/2024, 8:06:04 AM
USAGE (VALIDPERIOD) in BILLBD 4383 Nicholson Dr, Baton Rouge, LA 70808, USA

1/4/2024, 8:06:39 AM
undefined (SPEEDING) in 3900 Nicholson Dr, Baton Rouge, LA 70820, USA

Trip End
2024-01-04 08:08:30 at Baton Rouge

Trip Timeline

Event: END
Time: 1/4/2024, 8:08:30 AM

Event: SPEEDING
Time: 1/4/2024, 8:06:39 AM

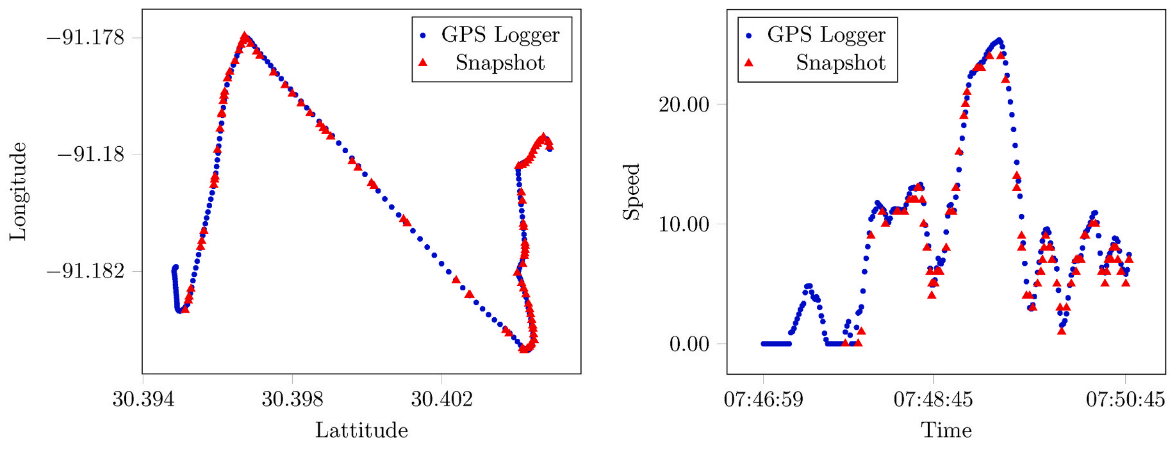
Event: START
Time: 1/4/2024, 8:03:58 AM

Events

- Start Point
- End Point
- Brake
- Speeding
- Acceleration
- Distraction
- Valid Period

Louisiana State University
Tiger Stadium
LSU Football Operations Center
East Laville Hall
Umami Japanese Bistro
Izzo's Illegal Burrito - Burbank
Sonesta ES Suites Baton Rouge
The Storage Center - Burbank
Maison Burbank
Stadium Square
Sharlo Terrace
Trip Mapview

Fig. 2. Visualization of the main features of the PyShot application.



(a) Geo-spatial distribution of recorded locations: a comparative visualization from GPS logger and Snapshot application data

(b) Comparative analysis of vehicle speed over time: GPS logger application versus Snapshot application data visualization

Fig. 3. Comparative analysis of vehicle geo-spatial positioning and speed: Insights from GPS logger and snapshot application data.

6.2. *RQ2: Forensic data from Progressive Cloud beyond Mobile application access*

In our analysis, we examined various event types extracted from cloud data using our PyShot application. This involved a detailed comparison of trip-related event types between PyShot and the Snapshot applications for both Android and iOS platforms. A notable observation was regarding the iOS version of Snapshot, which displayed minimal information, limited to only the start and end points of a journey. In contrast, the Android version of Snapshot provided slightly more detail, including braking and acceleration points along with corresponding timestamps. However, PyShot demonstrated superior performance in this aspect. PyShot was able to show a wide array of data points, including information on start and end points, hard braking, acceleration, dangerous maneuvers, and distracted driving, all of which are recorded and stored on Progressive's servers. Table 3 presents the type of event details provided by the iOS, Android, and PyShot applications showing PyShot outperforms both applications in showing event details.

6.3. *RQ3: **investigating** anti forensic techniques- location spoofing*

(a) On the left (1) is a map from the Snapshot application showing a

Table 3
Evidence of Trip Events: iOS v.s. Android v.s. PyShot

Evidence Type	iOS	Android	PyShot
Start Point	✓	✓	✓
End Point	✓	✓	✓
Hard Brakes	×	✓	✓
Accelerations	×	✓	✓
Speeding	×	×	✓
Distractions	×	×	✓
Phone Usage & Calls	×	✓	✓

trip with incorrect start and end points in Texas (indicated by a red dot) due to a fake GPS signal. On the right (2) is a zoomed-in view of the same trip showing several valid location points within Baton Rouge that the Snapshot application documents periodically. The trip data shows zero events, no captured speed, and a distance of zero miles.

(b) This is a map from the PyShot application displaying a journey with accurate starting and ending locations, despite interference from a counterfeit GPS signal. Map revealing correct location markers in Baton Rouge, meticulously recorded by PyShot at intervals. The map shows several incidents, captured speed, showcasing PyShot's capability to neutralize strategies designed to tamper with forensic examinations.

To test Progressive Snapshot's resilience to location spoofing, we used the Fake GPS Location Spoof Android application to simulate a false location on the same phone as Snapshot. After initializing our location to a false point and conducting a drive to collect trip data via Snapshot, the results were interesting. The Snapshot android application displayed anomalous visuals, indicating both the start and end points at the spoofed location, yet ambiguously pointing toward the actual location without precision (see Fig. 4a). Additionally, it failed to display events, mileage traveled, or any other information. PyShot, which accesses Progressive's cloud data, accurately depicted the real route, start and end points of the trip, speed, including the detection of an acceleration event depicted in Fig. 4b.

From a digital forensic perspective, this experiment highlights a critical consideration: while the superficial data visualizations in the Snapshot application may mislead users into believing they have successfully deceived the application, the underlying data integrity remains uncompromised. Progressive's cloud-based analytics, as evidenced by the PyShot findings, retain accurate trip records despite attempts at location spoofing.

6.4. *Transit Mode*

Snapshot allows users to adjust the transit mode for each trip. For example, if the user is a passenger, they can switch the transit mode to "passenger" to avoid impacting their insurance rate. Notably, the Snapshot application excels in accurately predicting whether a user drove the vehicle. For every trip, the application predicts the transit mode and provides an accuracy value, indicating its confidence in the prediction. Interestingly, our analysis revealed that the application's predictions were consistently correct, even when dealing with factors like a left-side passenger in a right-hand steering vehicle. Although the exact mechanism behind these predictions is unclear, the data holds significance in forensic settings. Therefore, PyShot gives the transit mode along with the prediction accuracy as seen in Fig. 2.

7. RQ4: case study: hit and run

This case study presents an analysis of a simulated hit-and-run incident, utilizing Progressive's Snapshot cloud data. The objective was to demonstrate the capabilities of PyShot in reconstructing vehicular incidents, particularly hit-and-run cases.

To simulate a hit-and-run scenario, a mannequin was positioned at specific coordinates (30.406, −91.180) in a parking lot. A vehicle equipped with Progressive's Snapshot application was driven (see Fig. 5a), initiating a collision with the mannequin followed by a hard brake and immediate acceleration away from the incident. The scene was documented with photographs and video recordings.[2] Progressive's cloud server provided the telemetries data, including GPS coordinates, hard brakes, and acceleration timestamps and coordinates.

At precisely 2024-01-28 at 18:12:06, Snapshot's cloud data indicates a marked braking event at the coordinates 30.406135, −91.180137, marked (A) in Fig. 5b. This location aligns with the position of the mannequin. The brake data recorded a change in speed from approximately 7 mph to 0 mph in a short time frame, suggesting a forceful and abrupt braking action as it was also mentioned in the cloud data. After a hard brake, the Snapshot application recorded periodic location captures as the car fled, capturing coordinates at 30.4060101, −91.1796551. This location, marked (B) in Fig. 5b, indicates the car's position during its escape. Following the periodic location capture, the data captures a rapid acceleration at 2024-01-28 at 18:12:24 at coordinates 30.405424, −91.179565, marked (C) in Fig. 5b. This acceleration is characterized by an increase in speed from 0 mph to a peak of 17 mph, indicative of the driver's intent to leave the scene quickly.

By mapping the sequence of GPS coordinates and speed data logged by the Progressive cloud, an outline of the vehicle's path was reconstructed using PyShot in Fig. 5b. The vehicle's movement from the initial point of impact through the braking and subsequent acceleration was traced, providing a visual representation of the incident's timeline and the driver's maneuvers.

The use of exact location data, time records, and information on how fast the vehicle was going made it possible to piece together the incident step by step. The analysis provided insight into the duration of hard braking by the car, whether it stopped moving at any point, and the rate at which it accelerated when leaving the scene. This detailed account of when and where events occurred offers a clear understanding of how the hit-and-run incident unfolded, aligning with what is observable in the video evidence.

8. Limitations

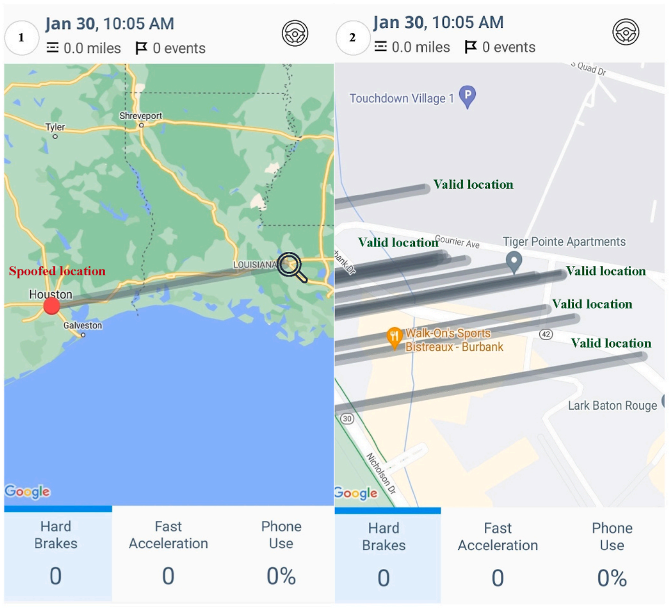
While our study offers a systematic approach to utilizing Progressive's Snapshot data for digital forensics, it acknowledges potential areas for enhancement. Our focus on Progressive, despite its significant market share, limits the scope of our investigation to a single provider's DIA. Including DIAs from several insurance companies would offer a richer, more diverse dataset for forensic analysis. Furthermore, the reliance on the current API structure of the Snapshot application introduces a potential limitation; any future updates could disrupt PyShot's functionality. If the Snapshot application is updated with new APIs in the future, PyShot's connection to the server will be severed. However, Updating PyShot to work with new API changes in Snapshot's system would be straightforward, making this a minor issue. Additionally, our testing with previous versions of the Snapshot application confirmed that the API structure has remained consistent.

9. Conclusions

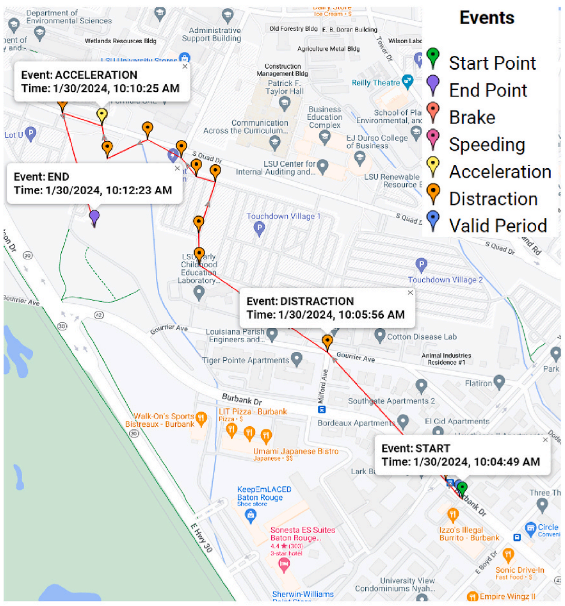
In this study, we addressed four research questions (RQs), demonstrating the effectiveness of our developed forensic tool, PyShot, in enhancing digital forensic investigations using DIA data. We have demonstrated that PyShot excels in extracting driving data from Progressive's cloud storage, offering a more detailed and accurate representation of driving behaviors and events compared to the native Snapshot application available on both Android and iOS platforms.

Our work revealed that PyShot could effectively capture a wide array of event types, including braking, acceleration, dangerous maneuvers,

[2] https://drive.google.com/file/d/14EXVu3feBcyQm6GcTHpX6Hw1vmCVZ1Wv/view?usp=sharing.

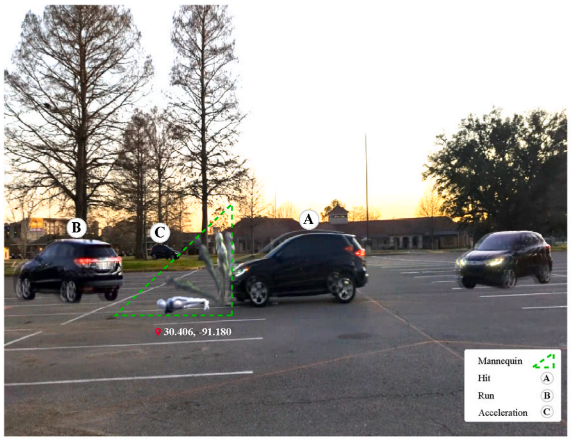


(a) On the left (1) is a map from the Snapshot application showing a trip with incorrect start and end points in Texas (indicated by a red dot) due to a fake GPS signal. On the right (2) is a zoomed-in view of the same trip showing several valid location points within Baton Rouge that the Snapshot application documents periodically. The trip data shows zero events, no captured speed, and a distance of zero miles.

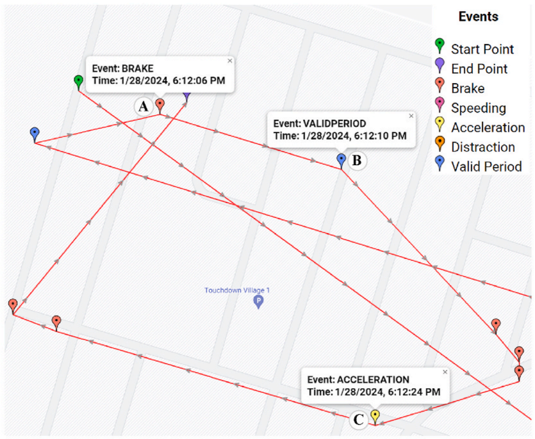


(b) This is a map from the PyShot application displaying a journey with accurate starting and ending locations, despite interference from a counterfeit GPS signal. Map revealing correct location markers in Baton Rouge, meticulously recorded by PyShot at intervals. The map shows several incidents, captured speed, showcasing PyShot's capability to neutralize strategies designed to tamper with forensic examinations.

Fig. 4. Comparison of snapshot android application and PyShot application map on a trip using fake GPS anti-forensic technique.



(a) This image depicts a hit-and-run scenario in which the perpetrator did a hard brake (A) and then accelerated quickly from the crime scene (C). Snapshot also captured the driver's location in a Valid Period shortly after the collision (B).



(b) This map shows the exact timestamps and coordinates of the events (A), (B), and (C) that occurred in the hit-and-run. The PyShot map details give a clear representation of the crime scene that occurred.

Fig. 5. Comparison of crime scene and map of crime scene created by PyShot

and distracted driving—details that are crucial for forensic analyses but are only partially available or absent in the native Snapshot application. Our study explored how the Snapshot application handled location spoofing. We found that although the Snapshot application could show incorrect locations due to spoofing, the data in Progressive's cloud was still accurate. PyShot could show the real driving routes and events despite spoofing, proving its worth in keeping data authentic and reliable for forensic use. The findings from our research establish the value of driving data from insurance applications like Progressive Snapshot in the field of digital forensics.

References

Arabo, A., Brown, I., El-Moussa, F., 2012. Privacy in the age of mobility and smart devices in smart homes. In: 2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing', pp. 819–826.
Bays, J., Karabiyik, U., 2019. Forensic analysis of third party location applications in android and ios. In: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)'.
Berman, K.J., Glisson, W.B., Glisson, L.M., 2015. Investigating the impact of global positioning system evidence. In: 2015 48th Hawaii International Conference on System Sciences', pp. 5234–5243.
Denton, G., Karpisek, F., Breitinger, F., Baggili, I., 2017. Leveraging the srtp protocol for over-the-network memory acquisition of a ge fanuc series 90-30. Digit. Invest. 22, S26–S38.
Ebbers, S., Ising, F., Saatjohann, C., Schinzel, S., 2021. Grand theft app: digital forensics of vehicle assistant apps. In: Proceedings of the 16th International Conference on Availability, Reliability and Security', pp. 1–6.

Feng, X., Dawam, E.S., Li, D., 2019. Autonomous vehicles' forensics in smart cities. In: 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (*SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pp. 1688–1694. https://api.semanticscholar.org/CorpusID:215738962.

Hassenfeldt, C., Baig, S., Baggili, I., Zhang, X., 2019. Map My Murder: A Digital Forensic Study of Mobile Health and Fitness Applications. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3339252.3340515.

Karpisek, F., Baggili, I., Breitinger, F., 2015. Whatsapp network forensics: Decrypting and understanding the whatsapp call signaling messages. Digit. Invest. 15, 110–118. Special Issue: Big Data and Intelligent Data Analysis. https://www.sciencedirect.com/science/article/pii/S1742287615000985.

Kopencova, D., Rak, R., 2020. Issues of vehicle digital forensics. In: 2020 XII International Science-Technical Conference AUTOMOTIVE SAFETY', pp. 1–6.

Le-Khac, N.-A., Jacobs, D., Nijhoff, J., Bertens, K., Choo, K.-K.R., 2020. Smart vehicle forensics: challenges and case study. Future Generat. Comput. Syst. 109, 500–510. https://www.sciencedirect.com/science/article/pii/S0167739X17322422.

Listi, G.A., Manhein, M.H., Leitner, M., 2007. Use of the global positioning system in the field recovery of scattered human remains. J. Forensic Sci. 52 (1), 11–15.

Mahr, A., Serafin, R., Grajeda, C., Baggili, I., 2022. Auto-parser: android auto and apple carplay forensics. In: Gladyshev, P., Goel, S., James, J., Markowsky, G., Johnson, D. (Eds.), Digital Forensics and Cyber Crime'. Springer International Publishing, Cham, pp. 52–71.

Moore, J., Baggili, I., Breitinger, F., 2017. Find me if you can: mobile gps mapping applications forensic analysis & snavp the open source, modular, extensible parser. J. Digital Forensics, Security Law 12. https://commons.erau.edu/jdfsl/vol12/iss1/7.

Onik, A.R., Alsmadi, R., Baggili, I., Webb, A.M., 2024. So fresh, so clean: cloud forensic analysis of the amazon irobot roomba vacuum. Forensic Sci. Int.: Digit. Invest. 48, 301686.

Pace, L.R., Salmon, L.A., Bowen, C.J., Baggili, I., Richard, G.G., 2023. Every step you take, i'll be tracking you: forensic analysis of the tile tracker application. Forensic Sci. Int.: Digit. Invest. 45, 301559. https://www.sciencedirect.com/science/article/pii/S2666281723000689.

Progressive Corporation, 2023. Progressive 2023 Shareholders' Report: Third Quarter. https://progressive2023rd.q4web.com/files/doc_financials/2023/q3/interactive/. (Accessed 2 April 2024).

Progressive Insurance, 2014. Progressive Snapshot Reaches 10 Billion Mile Mark. https://progressive.mediaroom.com/2014-03-20-Progressive-Snapshot-reaches-10-billion-mile-mark. (Accessed 2 April 2024).

Progressive Insurance, 2015. Progressive® Insurance Launches Pilot Snapshot Mobile App. https://progressive.mediaroom.com/2015-09-02-Progressive-R-Insurance-Launches-Pilot-Snapshot-Mobile-App. (Accessed 2 April 2024).

Rudden, J., 2023. Top u.S. Private Passenger Auto Insurance Writers by Market Share. Statista. https://www.statista.com/statistics/186523/top-us-private-passenger-auto-insurance-writers-by-market-share. (Accessed 2 April 2024).

Walnycky, D., Baggili, I., Marrington, A., Moore, J., Breitinger, F., 2015. Network and device forensic analysis of android social-messaging applications. Digit. Invest. 14, S77–S84. The Proceedings of the Fifteenth Annual DFRWS Conference.

Yarramreddy, A., Gromkowski, P., Baggili, I., 2018. Forensic analysis of immersive virtual reality social applications: a primary account. In: 2018 IEEE Security and Privacy Workshops (SPW), pp. 186–196.