future internet

MDPI

Article

The Digital Footprints on the Run: A Forensic Examination of Android Running Workout Applications

Fabian Nunes [1], Patrício Domingues [1,2,*] and Miguel Frade [1,3]

[1] School of Technology and Management, Polytechnic Institute of Leiria, Morro do Lena—Alto do Vieiro, 2411-901 Leiria, Portugal; miguel.frade@ipleiria.pt (M.F.)
[2] Instituto de Telecomunicações, Morro do Lena—Alto do Vieiro, 2411-901 Leiria, Portugal
[3] Computer Science and Communication Research Centre (CIIC), Polytechnic of Leiria, Morro do Lena—Alto do Vieiro, 2411-901 Leiria, Portugal
* Correspondence: patricio.domingues@ipleiria.pt

Abstract: This study applies a forensic examination to six distinct Android fitness applications centered around monitoring running activities. The applications are Adidas Running, MapMyWalk, Nike Run Club, Pumatrac, Runkeeper and Strava. Specifically, we perform a post mortem analysis of each application to find and document artifacts such as timelines and Global Positioning System (GPS) coordinates of running workouts that could prove helpful in digital forensic investigations. First, we focused on the Nike Run Club application and used the gained knowledge to analyze the other applications, taking advantage of their similarity. We began by creating a test environment and using each application during a fixed period. This procedure allowed us to gather testing data, and, to ensure access to all data generated by the apps, we used a rooted Android smartphone. For the forensic analysis, we examined the data stored by the smartphone application and documented the forensic artifacts found. To ease forensic data processing, we created several Python modules for the well-known Android Logs Events And Protobuf Parser (ALEAPP) digital forensic framework. These modules process the data sources, creating reports with the primary digital artifacts, which include the workout activities and related GPS data.

Keywords: Adidas Running; Android; DFIR; MapMyWalk; Nike Run Club; Pumatrac; Asics Runkeeper; Strava; ALEAPP

Citation: Nunes, F.; Domingues, P.; Frade, M. The Digital Footprints on the Run: A Forensic Examination of Android Running Workout Applications. *Future Internet* **2024**, *16*, 304. https://doi.org/10.3390/fi16090304

Academic Editors: Filipe Portela and Dino Giuli

Received: 25 July 2024
Revised: 11 August 2024
Accepted: 19 August 2024
Published: 23 August 2024

1. Introduction

Nowadays, smartphones and smart wearables are becoming more prevalent and powerful. One of the areas that saw considerable growth with mobile devices is the fitness industry Silva et al. [1], Yeoh et al. [2]. Wearable devices such as smartbands are oriented for fitness purposes as they pack a significant set of sensors, such as heart rate monitor, step counter, Oxygen Saturation (SpO_2), and sleep monitoring, to name just a few. The functionalities of wearables often depend on and are augmented by smartphones, like, for example, a smartband paired with a specially developed mobile application, communicating through Bluetooth Low Energy (BLE). This companion application interfaces between the device and the smartphone, providing services such as data visualization, geolocalization, access to the Internet for cloud storage and firmware updates. Another crucial role of companion applications is to serve as a bridge between the wearable device and its connected smartphone, facilitating the transfer of fitness data from the wearable to the phone. Subsequently, the phone uploads these data to the application's cloud server. Frequently, a companion application is specifically designed by a fitness brand, making it exclusively compatible with wearables produced by that brand. An example is Garmin Connect (https://play.google.com/store/apps/details?id=com.garmin.android.apps.connectmobile (accessed on 8 August 2024)), which is the companion application for Garmin devices such

as smartbands and smartwatches. Another example is the Fitbit application, which interacts with Fitbit fitness devices. Furthermore, these applications extend their reach and functionality by providing APIs that can be used by independent fitness applications to send/receive data from the associated companion application. For example, through the proper Application Programming Interface (API), the diet application MyFitnessPal (https://play.google.com/store/apps/details?id=com.myfitnesspal.android (accessed on 8 August 2024)) can receive the number of burnt calories from companion applications, providing in return the amount of ingested calories by the users. The Strava (https://play.google.com/store/apps/details?id=com.strava (accessed on 8 August 2024)) application is another example, as it can synchronize data from a running activity tracked by a Garmin smartband.

As they often couple precise GPS-based locations with date/time, fitness applications can provide valuable digital forensic data, allowing one to locate the whereabouts of the device bearer at a given date and time [3]. Additionally, if other metrics are available, such as heart rate, it is possible to infer the activity state of the device bearer: idle, normal or high. This might prove of paramount importance in criminal and fraud investigations [4].

An actual illustration of the benefits of smartwatch data in assisting with a criminal investigation is evident in a murder case. The Fitbit data of the victim played a crucial role in securing the husband's conviction, as the Fitbit timeline demonstrated movements within the house and the distance travelled, conclusively placing the husband at the crime scene and refuting his alibi [5]. In 2015, the police used GPS coordinates and step pace, which were stored in a Garmin smartwatch, to help bring charges to a man for a double homicide, correlating the time/date of the homicides with the coordinates stored in the smartwatch to draw the suspect's escape route [6]. Another publicly known case is the use of Strava in running mode to identify and convict a man of hitting and injuring a cyclist in Virginia, USA [7]. A final example is a case where the fitness application Apple Health data were used to convict a man of rape and murder: the police apprehended the man's iPhone, extracted the data and correlated them with the time and location of the murder [8]. Therefore, fitness applications can provide valuable data in digital forensics as they store a wealth of private information related to the user in specific companion applications such as Garmin Connect [9]. They can store health data, GPS data, sleep data and more. However, a type of application that is also interesting to analyze is running applications. The focus of these applications is related to outdoor running and other types of physical activities. Usually, these applications also function as fitness-tracking social networks, where one connects with other users and shares their running activities, sometimes to compare performances. The difference between these and companion applications is that they do not require the user to wear a smartwatch or smartband, as it is sufficient to use a smartphone with GPS and step counter capabilities. As running applications store data such as GPS coordinates and the date/time of run workouts, they can be instrumental in digital forensic analysis. Because of all their benefits, fitness applications have grown in numbers and users in recent years. In 2019, the major app stores featured over 350,000 healthcare and fitness applications, resulting in an annual download count of 3.7 billion [10]. This prevalence increases the likelihood of their relevance in a criminal investigation.

The high volume makes these types of applications a target for cybercriminals intending to steal valuable private data related to the applications' users. Since these applications handle private data, security and privacy should be their top priority, considering the number of regulations and legislation that they must follow. However, this is not always the case. Various studies have shown failures and shortcomings in the security and privacy of these applications. Scott et al. [11] studied fitness applications, showing that most stored data are in plain text and do not encrypt their communication. These facts made health and fitness apps a target of attacks that could cause the leak of millions of users' data.

Another issue is the poor privacy features implemented in some of these applications, which could be exploited by users with malicious intent, such as stalking another person by studying their GPS activities. A famous example is the Strava application, where a man

used it to stalk his ex-partner [12]. In 2022, it was revealed that Strava harbored a security vulnerability capable of enabling user tracking, even for the ones who had set the strictest privacy settings allowed by the application. The failure could be exploited by uploading fake running segments, allowing malicious users to learn the identities and past routes of Strava's users in the area [13]. In recent years, applications claim to have significantly evolved in security and privacy concerns by adding various privacy features, often due to discovered failures. However, some of these problems remain, and many applications still store a large amount of sensitive data, making them vulnerable to data leaks.

In this work, we conduct a forensic analysis of six running applications. As demonstrated later, Android applications were selected due to their widespread popularity, indicated by download numbers in the millions. All running applications except Strava are linked to well-known and popular sportswear brands. Strava, while not affiliated with a sportswear brand, is a highly regarded sports application. It allows users to record workouts and share their achievements on social networks, and it is favored by top professional runners and cyclists, contributing significantly to its popularity Couture [14].

The main contributions of this work are (*i*) the analysis and findings of forensic artifacts in post mortem scenarios for the Android version of the studied applications; (*ii*) the development of 12 modules for the framework Android Logs Events And Protobuf Parser (ALEAPP) to ease the extraction of forensic artifacts and creation of reports for further analysis; (*iii*) two new functionalities incorporated into ALEAPP: the capability to handle Flexible and Interoperable data Transfer (FIT) files and the introduction of a timeline plugin.

The remainder of this paper is organized as follows. Section 2 discusses related work while Section 3 describes the materials and methods of this study. Section 4 studies the *Nike Run Club* application, explaining the process and the main analysis tools. In Section 5, we focus on the peculiar analysis of *Strava*, as the results and artifacts differ from the other applications. In Section 6, we present the methodology gathered from the *Nike Run Club* and apply it to the other studied applications. Section 7 focuses on our open-source Python 3 modules developed for ALEAPP and the added `timeline` feature. Finally, Section 8 concludes the paper.

2. Related Work

We begin by defining two important concepts in digital forensics—*forensic artifacts* and *data extraction*—before examining related work. *Forensic artifacts* are data elements that provide reliable evidence to support or refute hypotheses about user activities, system operations or network communications on digital devices. *Data extraction*, within the context of digital forensics, involves retrieving data from digital devices in a sound manner, maintaining the integrity and authenticity of data, a crucial step in the preservation phase of digital investigations [3].

Forensic research on fitness applications is nothing new. However, most of the studies focus on companion applications such as Fitbit (https://play.google.com/store/apps/details?id=com.fitbit.FitbitMobile (accessed on 8 August 2024)) and Garmin Connect (https://play.google.com/store/apps/details?id=com.garmin.android.apps.connectmobile (accessed on 8 August 2024)), analyzing the application with the coupled fitness tracker. Next, we review the main works involving the digital forensic analysis of fitness applications.

Scott et al. [11] examined 20 health apps, primarily concentrating on security and privacy aspects but not targeting specific apps under our scope. While the analysis may lack relevance due to subsequent updates received by the applications, their methods remain pertinent for current post mortem analysis.

Hassenfeldt et al. [4] analyzed nine Android fitness apps. Some of these applications—Runkeeper (from Asics), Strava and Runtastic (now designated *Adidas Running*)—are also the focus of our work. The authors created their testing environment by collecting data and extracting them from the smartphone to the forensic computer through Android Debug Bridge (ADB) and the commercial XRY (https://www.msab.com/product/xry-extract

(accessed on 8 August 2024)) tool. In their analysis, they found (*i*) account data, (*ii*) personal information and (*iii*) GPS data. The authors also developed a Python tool to extract potential artifacts from the Extensible Markup Language (XML) and database files of the applications' private directory.

Sinha et al. [15] performed forensic analysis of six fitness applications with an interesting focus on the application **Nike Training Club**, from the same developers as **Nike Run Club**, and **MapMyFitness**, whose developers also developed **MapMyWalk**, two applications studied in our research. The data found for each application are quite similar: (*i*) user profile data, (*ii*) health and exercise data, (*iii*) captured device data and (*iv*) GPS data.

Hutchinson et al. [16] performed a forensic analysis of three companion applications using different devices: **Amazon Halo**, **Garmin Connect** and **Mobvoi**. Their comprehensive analysis unveiled the following forensic artifacts: (*i*) health data; (*ii*) profile information; (*iii*) phone notifications; (*iv*) exercise data; (*v*) GPS data; (*vi*) steps data. The in-depth study enabled us to grasp how fellow analysts establish their data collection and analysis environments.

Donaire-Calleja et al. [17] studied the forensic analysis of wearable devices, specifically smartwatches. The paper focuses on the challenges that forensic analysts face in this area, such as the lack of standardized procedures and the use of private communication protocols.

In prior research [9], we analyzed the Garmin Connect application paired with the Garmin Vivosmart 4 smartband. Our developed open-source tools can extract several artifacts, such as (*i*) daily summary data; (*ii*) GPS data; (*iii*) response cache data; (*iv*) network logs; (*v*) Facebook API tokens; (*vi*) device synchronization cache; and (*vii*) SpO_2 reading charts. These results motivated us to delve deeper into Android applications, focusing on applications to monitor running activities.

Several papers study wearable and smartphone accuracy in physical activity measurements such as step count and walking distances [18–20]. van Zandwijk and Boztas [21] reported that the iPhone has a low 2% error in step count reporting but can diverge up to 40% for walking distance measurement. Goh et al. [22] documented that smartphones overestimated step counts when assessed on a 3-day free-living condition, with precision affected by factors such as walking style and phone-wearing location. Precision assessment is beyond the scope of this work.

To summarize this section, Table 1 shows a comparison between our work and the ones presented above, focusing on the different goals of the analysis, applications targets and major findings.

Table 1. Comparative analysis of forensic research on fitness applications.

Study	Applications/Devices Analyzed	Focus of Analysis	Key Forensic Artifacts Found
Scott et al. [11]	20 health apps (general, not app-specific)	Security and privacy analysis	Security flaws, privacy concerns
Hassenfeldt et al. [4]	Android fitness apps (Runkeeper, Strava, Runtastic/Adidas Running)	Forensic analysis of fitness apps	Account data, personal info, GPS data
Sinha et al. [15]	Fitness apps (Nike Training Club, Nike Run Club, MapMyFitness, MapMyWalk)	Forensic analysis of fitness apps	User profile data, health/exercise data, device data, GPS data
Hutchinson et al. [16]	Companion apps (Amazon Halo, Garmin Connect, Mobvoi)	Forensic analysis of companion apps	Health data, profile info, notifications, exercise data, GPS data, steps data
Donaire-Calleja et al. [17]	Wearables (smartwatches)	Forensic challenges with wearables	N/A—Focus on challenges, not specific artifacts
Nunes et al. [9]	Garmin Connect with Garmin Vivosmart 4	Forensic analysis of fitness companion apps	Daily summary, GPS data, response cache, network logs, Facebook API tokens, sync cache, SpO_2 charts
van Zandwijk and Boztas [18]	Wearables, smartphones	Accuracy in physical activity measurements	Step count error rates, distance measurement discrepancies
Goh et al. [22]	Smartphones (general)	Validation of smartphone activity measurement	Overestimated steps, affected by walking style, phone location
Our Work, 2024	6 running apps (Strava, Adidas Running, Nike Run Club, etc.)	Forensic analysis of Android running apps	Post mortem artifacts, 12 ALEAPP modules, FIT file handling, timeline plugin

3. Materials and Methods

This section outlines the materials employed in this research, encompassing both hardware and software, and subsequently details the process of populating the device with data and conducting the analysis.

3.1. Hardware

We employed a Samsung A40 smartphone running Android 11 (API 30) as our primary tool for app analysis. These fitness apps typically interact with wearable devices like smartbands. Our objective was to gather GPS, speed and heart rate data. However, smartphone sensors alone cannot provide heart rate data. Hence, we utilized both the smartphone and a Garmin Vivosmart 4 smartband. It is worth noting that the Garmin Connect app is the native companion for this smartband. Nevertheless, utilising this smartband to gather data for other applications is also possible. Table 2 details the hardware and OS versions used.

Table 2. List of devices used in the study and their respective OS versions.

Device	OS Version
Vivosmart 4	V5.40
Samsung A40	Android 11 (API 30)

Retrieving data from devices without root access can be a challenging task due to the security measures implemented by manufacturers. These measures restrict data access, particularly to the private directories of applications, where most digital forensic artifacts are stored. As a result, forensic experts frequently find themselves resorting to exploits or alternative methods to gain access to data on non-rooted devices [23]. This approach is time-consuming and does not guarantee success, potentially leading to limitations in the amount or type of data that can be acquired or, in the worst-case scenario, data loss. Given the intricate and device-specific nature of this process, obtaining data from non-rooted devices falls outside the scope of this study. However, it is worth noting that, in some digital forensic scenarios, obtaining root access to the companion smartphone may not be feasible or suitable. Our chosen method is not a standard method used in real-world investigations. However, it can be applied in research, eliminating the need for commercial tools or other methods, such as complete forensic copies.

3.2. Software

We utilized the most recent versions of each examined application, which were accessible on the Google Play Store at the start of our research. Table 3 provides data gathered from the Play Store for each application, including the version utilized, the developer's information, the total download count and the release date. Our analysis is divided into (*i*) post mortem analysis and (*ii*) module creation. For each part, we used different tools that we succinctly describe next.

Table 3. List of analyzed applications, corresponding versions and Google Play stats.

Application	Developer	Version	# of Downloads	Release Date
Adidas Running	Adidas Runtastic	13.6	50M+	14 March 2023
MapMyWalk	MapMyFitness, Inc.	23.5.2	10M+	21 March 2023
Nike Run Club	Nike, Inc.	4.21.0	10M+	2 March 2023
Pumatrac	PUMA SE	4.19.9	1M+	18 September 2022
Runkeeper	ASICS Digital, Inc.	14.3	10M+	17 March 2023
Strava	Strava Inc.	299.19	50M+	22 March 2023

3.2.1. Post Mortem Software

We employed different tools for our forensic analysis to extract data. Among these tools, we utilized the ADB (https://developer.android.com/studio/command-line/adb

(accessed on 8 August 2024)), which provides access and command-line interaction with a connected mobile device. ADB is the standard approach for data extraction from Android mobile devices. However, it does have specific requirements, such as the device being in debugging mode and authorized when the system shows a dialogue asking whether to accept an RSA key that allows debugging through the connected computer. While these conditions are ideal, they may not always be feasible in real-world scenarios.

Additionally, we created a set of Python 3 scripts as part of our automation efforts for various analysis tasks. These scripts included one specifically designed for automating data acquisition. We used the PyCharm (https://www.jetbrains.com/pycharm/ (accessed on 8 August 2024)) Integrated Development Environment (IDE) to develop these scripts.

Analyzing the contents of an Android application's database is a pivotal aspect of a forensic investigation. We employed the widely recognized open-source tool *DB Browser for SQLite* (https://sqlitebrowser.org/ (accessed on 8 August 2024)) to facilitate this process. To better understand individual databases and their table relationships, we utilized *schemacrawler* (https://www.schemacrawler.com/weak-associations.html (accessed on 8 August 2024)) to generate database diagrams. These diagrams were further enhanced using *DBDiagram.io* (https://dbdiagram.io/home (accessed on 8 August 2024)).

3.2.2. Module Development

We resorted solely to standard programming tools for developing the ALEAPP modules, namely the above-referenced Pycharm as IDE. Table 4 lists all tools used for this project, their versions and their usage.

Table 4. Software tools.

Tool Name	Version	Usage
ADB	33.0.1	Data access
ALEAPP	3.1.6	Framework to generate report
DBDiagram.io	online tool	Create database diagrams
DB Browser for SQLite	3.12.2	Database analysis and queries
Magisk	24.3	Root the device
Odin	3.14.4	Flash the patched boot image
Pycharm	2022.3	Python Development
Python	3.10	Module and script development
schemacrawler	16.19.5	Generate database diagrams

This method uses (*i*) a rooted smartphone device and (*ii*) open-source tools. Although police forces and forensic practitioners most likely have access to commercial tools that automate most of the work presented here, our approach gives researchers an insight into how to obtain the data manually, thereby enabling a deeper understanding of the Android operating system that can be applied to study other applications.

The rooting process has various steps. The two main ones are the installation and configuration of **Magisk** (https://github.com/topjohnwu/Magisk (accessed on 8 August 2024)) and **Odin** (https://odindownload.com/ (accessed on 8 August 2024)). Magisk is an open-source software suite developed by the XDA community to customize Android, supporting devices with versions higher than Android 6.0. Magisk achieves device rooting through a boot image patching method, providing comprehensive root access to the Android OS without making any alterations or modifications to the `/system` partition. Subsequently, we employed Odin, a flashing tool designed for Samsung smartphones and tablets, to install the patched `ROM`. We followed the guides from both Magisk (https://topjohnwu.github.io/Magisk/install.html (accessed on 8 August 2024)) and Odin (https://www.droidwin.com/root-samsung-magisk-odin/ (accessed on 8 August 2024)). This procedure varies depending on the device's brand, model and Android version.

3.3. Method

Applications monitoring running activities have a growing user base and popularity [24]. In this paper, we study six popular applications: *Adidas Running* (https://play.google.com/store/apps/details?id=com.runtastic.android (accessed on 8 August 2024)), *Asics Runkeeper* (https://play.google.com/store/apps/details?id=com.fitnesskeeper.runkeeper.pro (accessed on 8 August 2024)), *MapMyWalk* (https://play.google.com/store/apps/details?id=com.mapmywalk.android2 (accessed on 8 August 2024)), *Nike Run Club* (https://play.google.com/store/apps/details?id=com.nike.plusgps (accessed on 8 August 2024)), *Pumatrac* (https://play.google.com/store/apps/details?id=com.pumapumatrac (accessed on 8 August 2024)), and *Strava* (https://play.google.com/store/apps/details?id=com.strava (accessed on 8 August 2024)). The criteria for choosing these applications were (*i*) having millions of downloads; (*ii*) recording running activities with GPS coordinates; (*iii*) being free or providing a freemium model; (*iv*) having the ability to execute without an always-on Internet connection; (*v*) having availability for the Android platform. Note that hundreds of applications could fit these criteria. We selected the most used and reviewed applications, rounding our selection to these six. As stated earlier, Table 3 summarizes the main stats of the studied applications from the Google Play Store.

We initiated by configuring a rooted smartphone with the requisite applications installed to gather ample data for our research. Subsequently, we established user accounts and meticulously documented identical outdoor activities within each application. This methodology facilitated the accumulation of data, enabling us to conduct a comparative analysis of the variations in data across different applications. To thoroughly examine each application, we undertook a month-long data-gathering process, during which we monitored a varied set of 20 workouts. These workouts were evenly distributed between indoor running and outdoor walks, spanning a total distance of 3 km and lasting over 30 min each. Furthermore, we conducted these activities across various locations within the city to capture a broad spectrum of data. Additionally, we thoroughly explored the distinctive features of each application, augmenting them with additional information and integrating them with companion apps and other relevant platforms.

In our analysis, we adhered to the methods outlined in NIST's publication 800-86, which incorporates forensic techniques and incident response plans [25]. In short, the analyst is advised to guide the investigation with the following steps in a post mortem analysis: (*i*) identify all relevant features of the application and use them to generate data; (*ii*) collect data using forensic tools or manual methods from a rooted phone or emulator; (*iii*) preserve the data in a secure location and make sure that the data are not altered or tampered with; (*iv*) examine the data using forensic or non-forensic tools and methods; (*v*) analyze the respective findings; (*vi*) report and present the results of the investigation.

As the applications are similar and the data-gathering processes were equal, we decided to employ the same analysis method for each application as depicted in Figure 1.

The analyst proceeds through these steps: (*i*) extract both public and private data (access to private data requires root access or the use of a forensic tool able to bypass Android security); (*ii*) analyze the public directory, knowing that, in most cases, it will not yield meaningful results; (*iii*) find and analyze the main database of the application in the private directory; (*iv*) analyze the Shared Preferences folder to obtain user's data that might not be found within the databases.

We document and display the complete method for the application **Nike Run Club** and implement it on the rest of the applications to extract and analyze their artifacts.

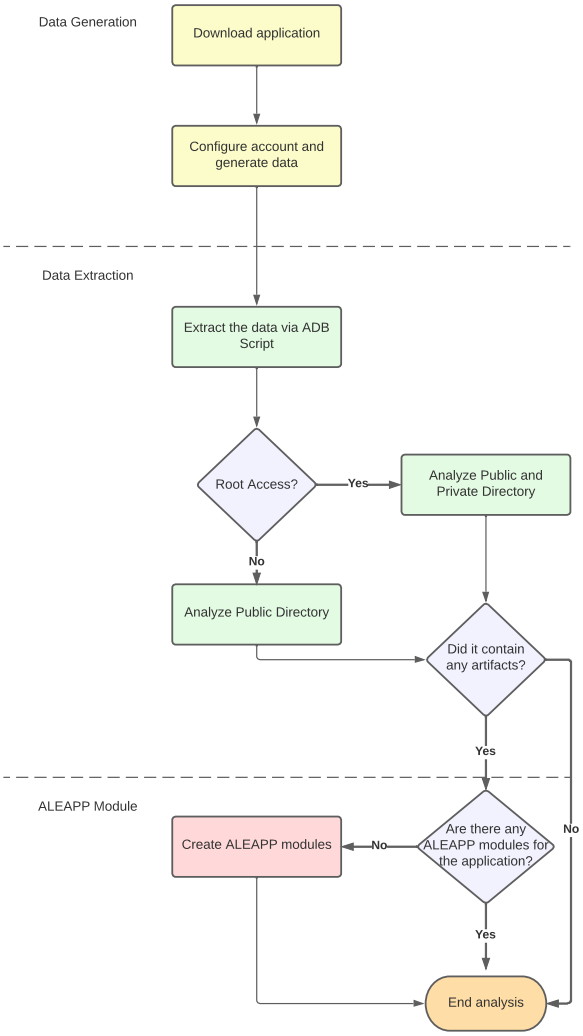


Figure 1. Analysis method.

4. Post Mortem Analysis

This section presents the post mortem analysis of each application. This kind of analysis focuses on extracting all data generated by the applications (after one month of usage) and analyzing them in a dedicated computer. After gathering the data, we followed a structured approach to retrieve the information from the device.

By default, application-generated data are stored in the internal storage, which remains private to the application and inaccessible to other apps or users unless root access is granted. As implied by its name, internal storage is a suitable location for storing application data that users do not directly interact with, such as database files, application logs and related data. Also, Android devices support a shared "external storage" space where developers can save files. Files saved to external storage are accessible and modifiable by the user when they enable USB mass storage to transfer files to a computer. Forensic practitioners aim to extract data stored in both the internal and, if available, external storage systems. The extraction process can be accomplished using commercial forensic tools, like Cellebrite (https://cellebrite.com/ (accessed on 8 August 2024)). However, because our device was rooted, we utilized ADB. ADB, or Android Debug Bridge, is a command-line tool for communicating with Android devices connected to a computer via USB or wirelessly (since Android 11). ADB facilitates various device actions, such as app installation and debugging, and provides access to a Unix shell for executing commands on the device (https://developer.android.com/studio/command-line/adb (accessed on 8 August 2024)). The procedure for extracting data using ADB is the same across all studied applications:

1. Access the device via ADB;
2. Navigate to the public or private directory (requires `sudo` privileges);
3. Locate the application folder;
4. Archive and compress the folder, storing it in external storage;
5. Transfer the archived data from the device to the analyst's workstation.

The following steps were undertaken to initiate the extraction process using ADB. Initially, we extracted data from the public folder located at

```
/storage/emulated/0/Android/data/<package name>
```

Subsequently, the most relevant part was extracting the application's private folder. However, unlike the public directory, accessing the private directory requires root access. In a rooted device, the user must only execute the **su** command to enter the privileged mode and access the private directory, which resides in the following path:

```
/data/data/<package name>
```

The binaries of the installed application can also be important in studying their behavior. Their location is

```
/data/app/<package name>
```

Since Android 8, app folders have been named using a random string in base64 to enhance privacy and security, making it more difficult for unauthorized users or malicious applications to access sensitive app data [26]. To identify the correct path, the following command was executed:

```
adb shell pm path <package name>
```

This command outputs the path to the application's APK file location. That file can then be extracted using ADB with the corresponding path. Extracting data from all locations can be time-consuming. Therefore, we developed a Python script named `ADB-Extractor` to automate the process. The script allows users to select what data they want to extract (public directory, private directory or application binary file) from the chosen device (Android emulator or physical device). The usage of the script is as follows:

```
python3 acquisition.py -a <package name> -d [emulator | physical] -t [public | private | apk]
```

A graphical interface is also available, where users can select the device, what to extract and where to save it. Additionally, users can select the desired package name from a list of all installed applications. Figure 2 displays the tool's graphical interface.

Maintaining forensic integrity is critical to the credibility and admissibility of digital evidence. To ensure that digital evidence remains authentic, unaltered and verifiable from the point of collection through to its presentation in legal or investigative contexts, investigators compute the hash of extracted files prior to analysis to guarantee that the data are not tampered [27]. Our tool calculates the `SHA256` hash for each extracted file and directory, storing the results in an output file for subsequent integrity verification.

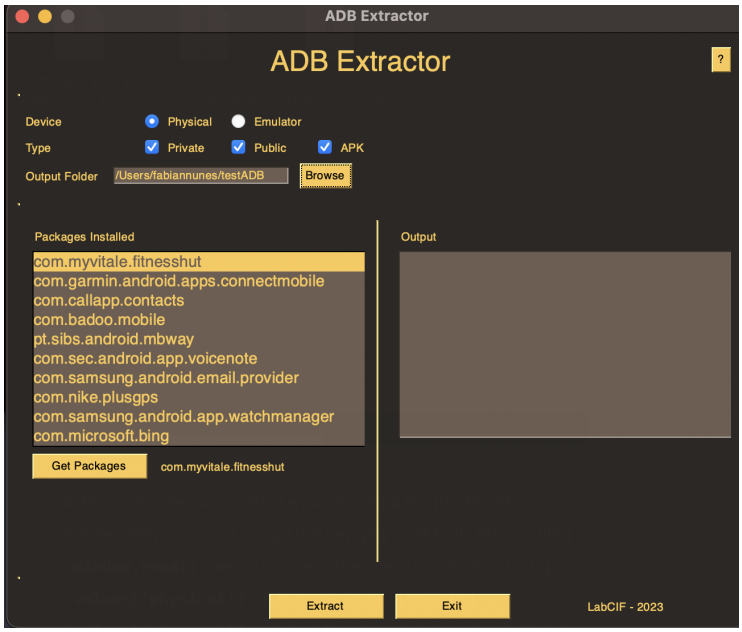


Figure 2. Graphical interface of ADB extractor.

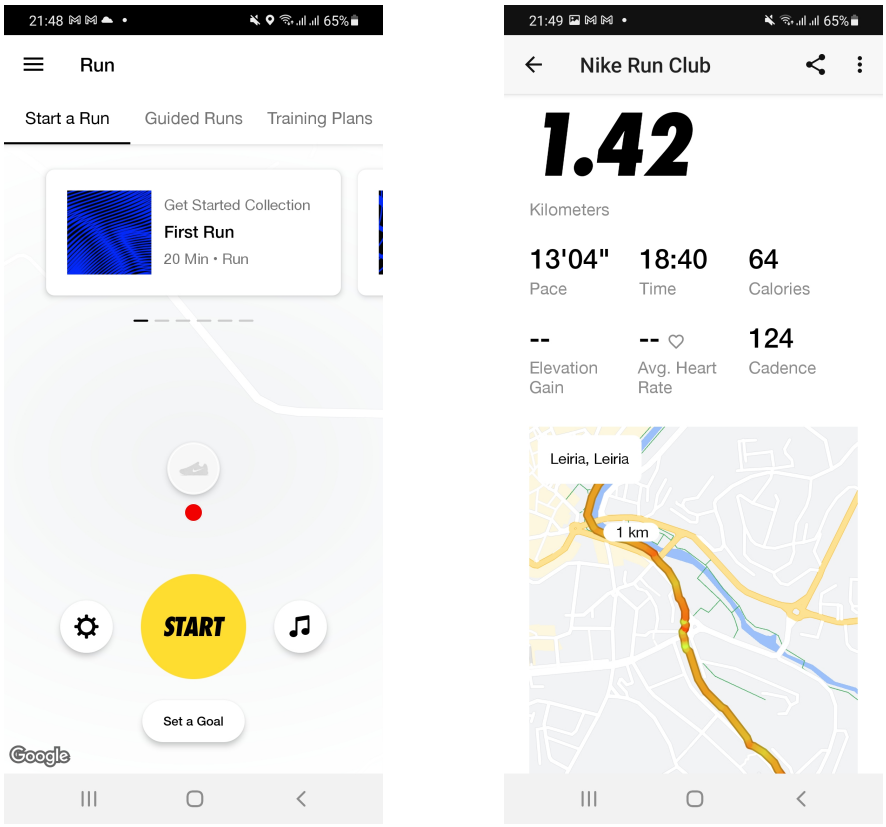
As mentioned previously, our study examines six distinct run-tracking applications. They all have the same goal—to record running routes, times and performances—and thus exhibit many similarities. To prevent redundancy, we will provide an in-depth analysis of one of the applications, specifically *Nike Run Club*, which is the most feature-rich among the set. Our methodology for analyzing *Nike Run Club* is then applied to the remaining fitness applications.

4.1. Nike Run Club

Nike Run Club has 10M+ downloads in Google Play Store (see Table 3) and focuses on preparing athletes for competitions by mixing virtual personal training with a gamification system and social network. Users can connect with friends through the club feature, which acts as a small community. Within this community, they can engage in challenges, compete with each other and collectively work toward enhancing their overall fitness levels.

Nike Run Club lacks a built-in authentication method. During login request or account creation, the application opens a webview for the `URL` https://accounts.nike.com/ (accessed on 8 August 2024), where the browser will display the authentication form. This fact means that the authentication is performed in the browser and is exposed to possible website vulnerabilities, which is vital for future forensic dynamic analysis. The registration form asks for (*i*) the email (where the verification code is sent); (*ii*) password; (*iii*) name; and (*iv*) birth date. After successfully creating an account, the user returns to the application, where the application asks for (*v*) gender, (*vi*) height and (*vii*) weight.

Upon login, the authenticated user lands on the main dashboard, featuring a live Google Maps display of their current location, shown in Figure 3a. Here, users can track their activity, with metrics such as distance, calories burned and duration showcased alongside the activity's route on the map. This run-tracking dashboard is standard, complemented by extras like training plans and audio-guided runs accessed by swiping. The app includes a side menu with options like an activity summary screen detailing stats and a log of past activities. Users can delve into individual activities for detailed statistics like pace, duration, calories burned, heart rate and GPS route, as shown in Figure 3b. Although all studied running applications are unique, we identified similar points, such as the activity screen and the challenges page. This fact is essential from a forensic standpoint because it helps us to identify the possible data commonly stored in all of them.



(a) **(b)**

Figure 3. Comparison between two screens. (**a**) Main screen of Nike Run Club. (**b**) Nike Run Club activity summary.

Data Synchronization

As standard in modern applications, Nike Run Club can synchronize data between different devices. After analyzing its main database, as will be explained in Section 4.4, we discovered that the application can store, in its local databases, the data from the devices that are synchronizing in the same account.

In addition, the Nike Run Club application can import activities from the cloud of companion applications. We had previously used the Nike Run application on an iOS device, and all activities from Apple's Healthkit were seemingly imported into the Android database. To test this venue further for collecting data, we installed the application on an Android emulator, resorting to Android Studio. After installing the application and logging into the Nike Run account, we extracted the application's private directory. Again, all previous activities were stored in the databases. This finding is relevant from a forensic standpoint since digital practitioners are not bound to using a rooted device to access data. Indeed, with access to the account credentials, one can install the application on an emulator and collect all data. Moreover, suppose that the Nike Run Club account is connected to a companion application such as Garmin Connect or Fitbit. In that case, one can potentially access activities recorded by devices running under iOS, resorting to Healthkit.

4.2. Android Permissions

Upon accessing the Play Store, one can observe the permissions that the application requests during installation: Play Store → About this app → permissions (at the bottom of the page) → view details. Table 5 displays the permissions requested by Nike Run Club, which include several high-level permissions. It is important to note that these permissions are typical for this category of applications, and the other apps in our study also request similar permissions.

Table 5. Nike Run Club permissions.

Permission	Function
Wi-Fi connection information	View Wi-Fi connections
Device ID & call information	Read phone status and identity
Camera	Take pictures and videos
Contacts	Read your contacts
	Find accounts on the device
Storage	Read the contents of your USB storage
	Modify or delete the contents of your USB storage
Photos/Media/Files	Read the contents of your USB storage
	Modify or delete the contents of your USB storage
Wearable sensors/Activity data	Body sensors (like heart rate monitors)
Identity	Add or remove accounts
	Read your own contact card
	Find accounts on the device
Phone	Read phone status and identity
Microphone	Record audio
Location	Approximate location (network-based)
	Precise location (GPS and network-based)
Other	Read sync statistics and receive data from Internet
	Download files without notification
	View network connections
	Change network connectivity
	Pair with Bluetooth devices
	Create accounts and set passwords
	Toggle sync on and off
	Control vibration
	Read sync settings
	Google Play licence check
	Prevent device from sleeping
	Access Bluetooth settings
	Run at startup
	Full network access
	Use accounts on the device

4.3. Extraction of data

We used ADB to extract both public and private data from each application. This process can be time-consuming since it requires finding the data, compressing them, storing them in the phone's public storage and then pulling them from the device to the forensic practitioner's computer. Therefore, we created a Python 3 script to automate the process with the following syntax:

```
python3 acquisition.py <package_name> -d <emulator|physical> -t <private|public|apk>
```

The `acquisition.py` script runs on Linux, Mac OS and Windows systems. The script is open-source and hosted on GitHub (https://github.com/labcif/ADB-Extractor (accessed on 8 August 2024)).

Next, we analyze the data generated by the Nike Run Club app. We start with the so-called public data and then analyze the private data.

4.3.1. Public Data

The directory structure of the public data is shown in Figure 4, which holds 2 subdirectories and 11 files. However, as expected, the data in these directories do not hold much forensic value, as all stored files are cache files related to maps. Nonetheless, there was one specific application whose public directory held relevant forensic data: **Strava**. This will be discussed later on.

```
/sdcard/Android/data/com.nike.plusgps/
                    └── cache/
                        └── debug/
```

Figure 4. Directories inside the public folder of Nike Running Club.

4.3.2. Private Data

Applications typically store most data in their private folders, which holds for the applications under examination in this study. Unfortunately, accessing private data requires root privileges, meaning that such data can only be retrieved when the mobile device is rooted. Consequently, using a rooted device was essential for conducting this study.

The private directory structure, up to two sub-levels, is depicted in Figure 5. This structure is notably more extensive than the public directory, encompassing 182 subdirectories and 411 files.

```
/data/data/com.nike.plusgps/
├── cache/
│   ├── apiCache/
│   ├── com.google.android.gms.maps.volley/
│   ├── imageCache/
│   └── messageImages/
├── code_cache/
├── databases/
├── files/
│   ├── analytics-storage/
│   ├── com.nike.persistence/
│   ├── dropship/
│   └── phenotype/
├── no_backup/
│   ├── com.urbanairship.databases/
│   ├── dropship/
│   └── urlmanager/
└── shared_prefs/
```

Figure 5. Directories inside the private folder of Nike Run Club.

Examining all these files can be a daunting task. However, obtaining a clear idea of where the most crucial information is stored is feasible by leveraging prior knowledge of the Android operating system and understanding how private data are organized [28]. The `files/` folder typically contains cache or temporary files and log files. In the case of the Nike Run Club, there are no forensic relevant data in this folder. The `database/` folder usually holds the most meaningful forensic artifacts, as it holds databases, usually SQLite 3 ones. Applications such as Nike Run Club have many `SQLite` databases, yet most of the relevant data are stored in just a few databases. The remaining databases are usually related to several services, such as Google `APIs`, integration with other applications of the same developer or even features related to premium features.

The `shared_prefs/` is another folder where relevant forensic data can be found. It usually holds several XML files that store key-value data. It is common for applications to store data such as user account credentials and other information related to hardware and interactions with authentication services, such as Facebook and Google. This folder can also hold API keys if developers are not careful enough to protect them. Next, we focus on the databases of the Nike Run Club application.

4.4. Databases

The Nike Run Club application keeps 15 different SQLite 3 databases in its private directory. The most relevant ones, from a digital forensic perspective, are (*i*) `com.nike.nrc.room.database`, which acts as the main database, and (*ii*) `ns_inbox.db`, which holds the application notifications.

4.4.1. com.nike.nrc.room.database

The database `com.nike.nrc.room.database` is the core storage of the application, holding 53 tables. The main elements stored by the database are (*i*) user activities; (*ii*) user weight; (*iii*) training plans; (*iv*) challenges; (*v*) audio runs; and (*vi*) achievements.

User Activities

Nike Run Club splits data related to the activities into eight different tables. The main table `activity` stores the basic information, such as start and end time. Each subsequent

table is connected to this table from where the data came: directly from the application or from a companion application. An activity can have multiple records related to it in other tables. Table 6 explains the different tables and their forensic value. Indeed, the application stores a large amount of information related to an activity. We could extract information like the timespan of the run, duration, distance, heart rate, calories burned and more by analyzing the data. The database diagram, which highlights relationships among the tables listed in Table 6, is shown in Figure 6. This diagram was carried out with the aid of `schemacrawler` and `DBDiagram.io`, the first to create a base diagram enhanced with the second tool. Since the diagram is quite large, we only highlight relationships between tables in Figure 6 related to a fraction of the database. A more detailed version of the diagram and the code used to generate it in `DBDiagram.io` is available in a GitHub repository (https://github.com/labcif/Running-Databases (accessed on 8 August 2024)). This repository contains the diagrams and code for each diagram presented in this study.

Table 6. Most important tables of `com.nike.nrc.room.database`.

Table	Forensic Value	Description
`activity`	fair	Base data of an activity (duration and source)
`activity_metric_group`	low	Metrics used in the activity
`activity_moment`	high	Updates during activity (start, pause and stop)
`activity_polyline`	high	Stores the polyline of the activity
`activity_raw_metric`	high	Stores the metrics for specific timestamps
`activity_summary`	high	Stores the summary data of the activity
`activity_tag`	fair	Stores tag data of the activity (title, type, location, etc.)
`fullpower_activity_link`	low	Stores various ids related to the activity

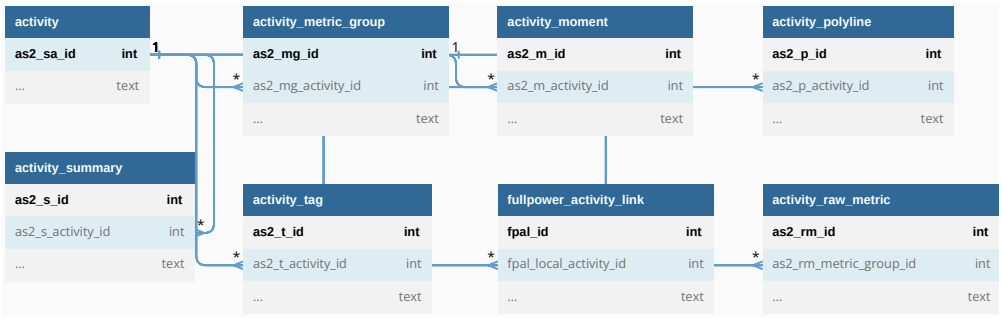


Figure 6. Simplified diagram of the `com.nike.nrc.room.database`.

One of the most relevant artifacts found is the activities' GPS coordinates. Nike Run Club stores these coordinates in a unique polyline format in the table `activity_polyline`. `Polyline` is a string of characters that encode coordinates used by Google Maps to draw the route line on the map [29] as shown in Figure 3b.

The coordinates bear crucial forensic significance as they precisely identify the user's whereabouts during a specific timeframe. According to Google's documentation, reversing a polyline back into grouped coordinates is feasible. Utilizing the Python library **polyline** (https://pypi.org/project/polyline/ (accessed on 8 August 2024)), we devised a Python script to decode Google's polylines into coordinates and store them in an `XLSX` file. This file hosts the activity's coordinates. Additionally, employing the **geopy** library (https://pypi.org/project/geopy/ (accessed on 8 August 2024)), our script enriches these coordinates with additional details such as the corresponding **road**, **city**, **postcode** and **country**. This file aims to facilitate the analysis of GPS coordinates and streamline the identification of potential locations of interest. Subsequently, utilizing these coordinates, we generate a file showcasing the user's traveled route as exemplified in Figure 7. The script allows for the export of this file in either `HTML` or Keyhole Markup Language (KML). The **Polyline2GPS** (https://github.com/labcif/Polyline2GPS (accessed on 8 August 2024))

script is available as a standalone tool after the functions described were initially developed for the ALEAPP framework.

Figure 7. Map with a route overlay from a decoded polyline.

Another forensic relevant table of the `com.nike.nrc.room.database` database is `activity_raw_metric`. It stores specific actions during the activity, such as the user pausing, resuming or ending the activity. The table stores the action that occurred and timestamps it. This can be useful if the need arises to accurately determine an activity's sequence of events. For instance, cross-referencing data allows one to detect when a user stopped walking for a given period and when the stop happened.

4.4.2. ns_inbox.db

The `ns_inbox.db` database holds only one relevant table that stores notifications received through the applications inbox. These messages could be the application's news or notifications from other users. This database stores the notifications' contents and the date/time that the message was sent. Note, however, that Nike Run Club does not have an integrated message system; hence, all messages are system-generated. Thus, no user messages are stored in this database.

5. Strava

The Strava application notably diverged from the others. In contrast to the other applications, no artifacts were discovered within Strava's private directory. This divergence can be attributed to the exposure of certain Strava features that could be misused in inferring locations. These issues received substantial media attention, as reported by sources like the BBC News [30] and Gritten [13]. As a result, Strava took significant measures to bolster its security, which likely led to the absence of artifacts in its private directories. Therefore, this absence of meaningful data in private directories is understandable. Our surprise came from a set of FIT files in the public directory. These files were stored in a subfolder called `files` that was the only subfolder inside the public directory. FIT is a binary file format developed by Garmin to store and share fitness data. FIT's capacity to encompass diverse fitness metrics, including workout details, heart rate and GPS location, has led to widespread adoption by fitness applications and trackers [31]. It is widely used by fitness enthusiasts, athletes and trainers to track and analyze their performance and progress over time [32].

We developed the Python 3 script `decode.py` (https://github.com/labcif/FIT2GPS (accessed on 8 August 2024)) to ease the processing of FIT files. It resorts to the `fitdecode`

Python library to decode FIT files, extracting GPS coordinates and producing as output, just like our polyline script, either an HTML file with a map or a KML compressed file (KMZ) file.

6. Remaining Applications

To preserve space, we grouped the remaining applications into a single section, as the process used for Nike Run Club could be applied to all of them.

6.1. Public Directory

Except for Strava, as detailed in Section 5, the public directory of the analyzed applications did not yield any relevant data. Except for a few audio files linked to workout instructions within the ASICS Runkeeper application, the publicly accessible files closely resembled those found in Nike Run Club. These files predominantly comprised cache data related to GPS maps.

6.2. Private Directory

Next, we analyze the directories under the private directory, focusing on databases and XML files where relevant artifacts exist.

6.3. Databases

As before, the primary artifacts came from the application's databases. In all four applications, we found data related to activities and, in some cases, user information. Each application handled activities differently, especially the GPS coordinates of the run, as we next explore.

6.3.1. Adidas Running

The Adidas Running application has several databases, with two having meaningful forensic data: `db` and `user.db`. The former has 30 tables, although most are empty. Table `session` is the largest—90 fields—and the most relevant one, as it records all activity data, namely distance, duration and calories, to name just a few. The GPS coordinates are stored in polyline format, just as in Nike Run Club. The `user.db` database keeps user-related data, such as height, weight, email and which companion applications are connected. Knowing which companion applications are connected to the account can prove helpful in a digital forensic investigation.

6.3.2. MapMyWalk

`MapMyWalk` has a database for each feature, such as workout plans, activities and user data. The `workout.db` database stores the user's activities, mainly in the `timeSeries` table. The application periodically collects date/time, distance, heart rate and GPS coordinates, yielding many data records. In a forensic analysis, this requires the aggregation of data to interpret them correctly. The application also stores account data in the `mmdk_user` database. It stores email, name, location, birthdate, height, weight, etc., in the `user_entity` table.

6.3.3. Pumatrac

Pumatrac has two databases in its database folder: `pumatract-db` and `OneSignal.db`. Only the former is forensically relevant. It holds 27 tables, with only a select few ones keeping meaningful forensic data: (*i*) `calendar`; (*ii*) `completed_exercises`; (*iii*) `completed_workouts`; (*iv*) `positions`; (*v*) `heart_beat`. All records are linked to a user account through the `users` table. Activity data such as timestamps, duration, calories burned, distance and GPS location are kept in the `completed_exercises` table. A simplified diagram of the `pumatract-db` database is given in Figure 8.

Figure 8. Simplified diagram of the `pumatrac-db`.

6.3.4. Runkeeper

Although the ASICS Runkeeper application has a few database files, all essential data are kept in the `RunKeeper.sqlite` database, with 53 tables. Although most tables remain empty, the database stores information about activities realized, challenges, training plans and user weight. The activity information is stored in the `trips` table, holding timestamps, distance, calories and duration. GPS coordinates are stored in the `points` table, recording each point as a pair of text-based latitude and longitude values, the altitude, time spent in that spot, the current speed and the distance from the last recorded point. Figure 9 portrays the `RunKeeper.sqlite` database, focusing on the forensic artifacts.

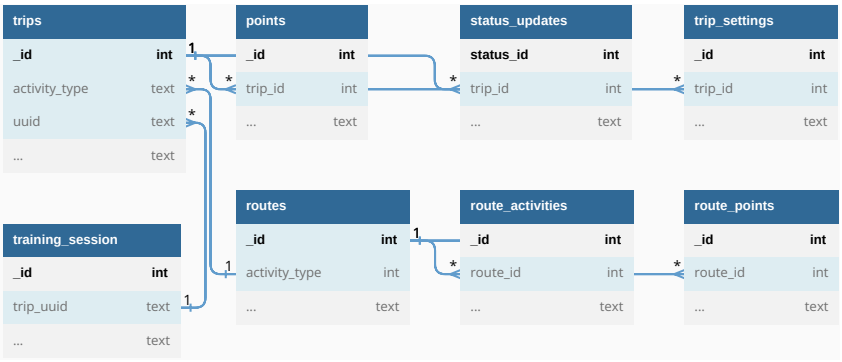


Figure 9. Simplified diagram of the `RunKeeper.sqlite`.

6.4. Shared_Prefs

The `shared_prefs` folder holds XML files storing key-value data. Often, there are user-account-related data. Surprisingly, only the ASICS Runkeeper application keeps content in the `shared_prefs` folder, namely the file `com.fitnesskeeper.runkeeper.pro_preferences.xml`. This file stores user account data, such as email, height and weight.

6.5. Synopsis

Utilizing identical analysis methods employed for the `NikeRunClub` application, we extracted the main digital forensic artifacts from each application. The sole application necessitating adaptation was `Strava`. The scripts devised to parse the distinct data sources for each application are summarized in Table 7.

The most significant digital evidence found in the examined applications consists of timestamp and location data pairs, with GPS coordinates linked to specific dates and times. These data points enable us to determine, with GPS accuracy, the movements and locations of the smartphone's primary user over a specific period. The data can be used to confirm/refute an alibi, identify patterns of behavior and reconstruct crime scenes. For example, an individual under investigation might state that he/she never has been in a given place, just to be refuted by the forensic artifacts found in the running application that he/she has used. Obviously, the contrary can also happen, with the forensic data attesting his/her alibi. It is important to note that while such data can provide relevant leads, they should be used in conjunction with other evidence to build a solid case. Obviously, it is

the responsibility of the forensic team to respect and properly address the legal and ethical requirements when handling such sensitive information. These requirements depend on the jurisdiction under which they are serving. Table 8 summarizes the main digital forensic artifacts of the six studied applications.

Table 7. List of modules developed for ALEAPP and the sources employed for parsing data. Except for Strava, the folders for all other data sources are located as subfolders within Android's `/data/data/`.

Application	Module	Data source
Adidas Running	`adidasActivities.py`	`com.runtastic.android/databases/db`
	`adidasGoals.py`	`com.runtastic.android/databases/goals`
	`adidasUsers.py`	`com.runtastic.android/databases/user.db`
MapMyWalk	`MMWActivities.py`	`com.mapmywalk.android2/databases/workout.db`
	`MMWUsers.py`	`com.mapmywalk.android2/databases/mmdk_user`
Nike Run Club	`NikeAMoments.py`	`com.nike.plusgps/databases/com.nike.nrc.room`
	`NikeActivities.py`	`com.nike.plusgps/databases/com.nike.nrc.room`
	`NikeNotifications.py`	`com.nike.plusgps/databases/ns_inbox.db`
	`NikePolyline.py`	`com.nike.plusgps/databases/com.nike.nrc.room`
Pumatrac	`PumaActivities.py`	`com.pumapumatrac/databases/pumatrac-db`
	`PumaUsers.py`	`com.pumapumatrac/databases/pumatrac-db`
Runkeeper	`RunkeeperActivities.py`	`com.fitnesskeeper.runkeeper.pro/databases/` ↪ `RunKeeper.sqlite`
	`RunkeeperUser.py`	`com.fitnesskeeper.runkeeper.pro/shared_prefs/` ↪ `com.fitnesskeeper.runkeeper.pro_preferences`
Strava	`StravaGPS.py`	`/sdcard/Android/data/com.strava/files/*`

Table 8. Main digital forensics artifacts per application.

Application	Databases	Digital Forensic Artifacts
Adidas Running	db, user.db	GPS (polyline), duration, distance, calories, user's data (height, weight, email,...)
MapMyWalk	workout.db	GPS, date/time, distance, heart rate, user's data (height, weight, email,...)
Nike Run Club	com.nike.nrc.room.database ns_inbox.db	GPS (polyline), duration, distance, heart rate, notifications
Pumatrac	pumatract-db	GPS, timestamps, duration, calories, distance
Runkeeper	RunKeeper.sqlite	GPS (text-based latitude and longitude), timestamps, duration, distance, calories, user's data in XML file.
Strava	None.	FIT files (public directory)

7. ALEAPP Modules

To ease and automate the analysis of the six studied run-tracking apps for digital forensic practitioners, we provide several modules for the ALEAPP forensic framework created by Alex Brignoni (https://github.com/abrignoni/ALEAPP (accessed on 8 August 2024)). ALEAPP is an open-source Python-based tool created specifically for digital forensics investigations of Android devices. It is used to parse and analyze various types of data extracted from Android devices, including logs, events and protocol buffer (Protobuf) files, a common data format used in mobile applications. By automating some of the repetitive tasks in Android forensics, ALEAPP allows digital forensics examiners to save time. Figure 10 shows the graphical user interface of ALEAPP.

ALEAPP was conceptualized with scalability and modularity in mind. Developers can seamlessly integrate their contributions into the framework by creating a single Python 3 file, which acts as an ALEAPP module. An ALEAPP module is a single Python 3 script (also known as a plugin) within the ALEAPP framework, created to process and analyze a particular type of forensic artifact found on Android devices. Data parsing occurs from previously extracted Android files, with the results being presented in the `HTML` report generated by ALEAPP. An ALEAPP module reads the specified file, extracts relevant data and presents them to the user. Although this is a primary function, additional features have been incorporated into ALEAPP to improve its reporting features.

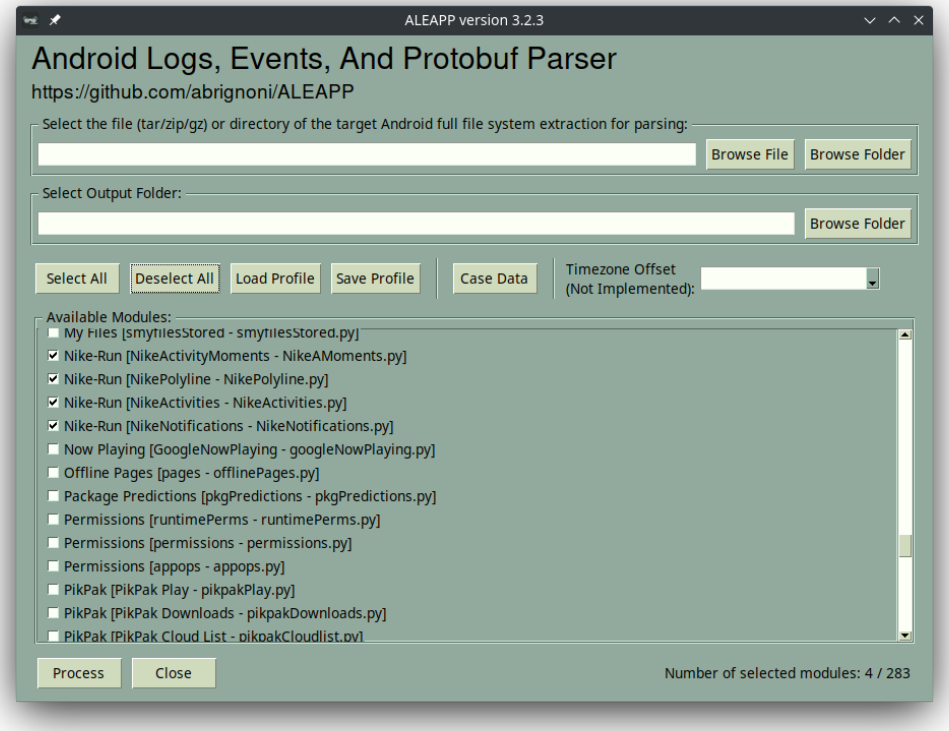


Figure 10. ALEAPP interface.

The decision to target ALEAPP stems from its ease of use, the platform's open-source nature and its popularity and transparent integration in the also-open-source `Autopsy` forensic software [33].

To deal with the forensic artifacts of the six applications, we developed 14 new modules for ALEAPP and a new ALEAPP's feature that we named **timeline**. Table 7 lists all 14 created scripts that have been added to ALEAPP's GitHub repository (https://github.com/abrignoni/ALEAPP/).

7.1. Timeline

Our addition to the core of ALEAPP was the creation of a timeline plugin. The plugin displays, in a timeline format, events in chronological order. This plugin was developed using the open-source components `timeline.js` (https://github.com/squarechip/timeline (accessed on 8 August 2024)). The plugin is utilized to showcase data from the `activity_moment` table of the `NikeRunClub` database. This table stores specific moments for each activity, such as pausing, stopping and the completion of each additional kilometre. These data are read by the special `NikeAMoments.py` module, which resorts to the `timeline` plugin to render the run activity, as shown in Figure 11.

Note that any ALEAPP module can use the `timeline` plugin, as it was designed to be application-agnostic. In Listing 1, we exemplify how it is possible to generate a timeline using this feature.

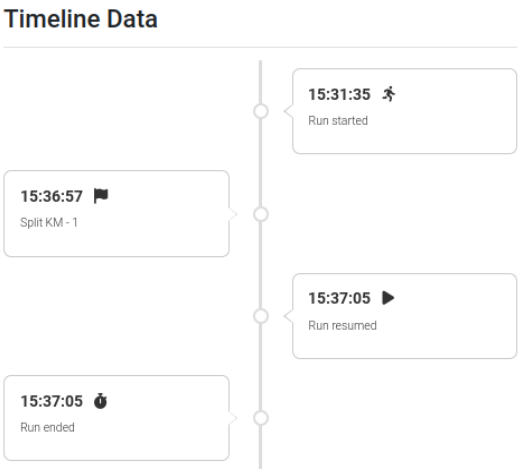


Figure 11. ALEAPP timeline.

Listing 1. Code snippet for implementing the timeline feature on ALEAPP.

```
                    ...
    timeline_array = [
        {'time': '15:22:00', 'text': 'Run Started', 'type': 'fa-solid fa-person-running'},
        {'time': '15:32:00', 'text': 'Run Stopped', 'type': 'fa-solid fa-stopwatchg'},
    ]
    # The type is the class name of an icon of font awesome. You can use any icon you want
     ↪  from there or use another library. However, FA is already included in ALEAPP
    report.add_section_heading("Timeline Data")
    #id of the timeline
    id = timeId
    #generate timeline
    report.add_timeline(id, timeline_array)
    #add script to present timeline
    report.add_timeline_script()
    report.end_artifact_report()
                    ...
```

7.2. Specific ALEAPP Modules for Running Applications

Although it is possible to extract all data from an application using a single Python 3 script for each module, the ALEAPP community suggests developing separate modules for each parsed data source. This strategy simplifies module maintenance because the analyzed running applications often share common features and stored data. The individual modules created for parsing each application's data demonstrate high similarity.

The **Activities** modules (e.g., `MMWActivities.py`) are responsible for extracting and presenting information regarding running activities. These modules handle data such as (*i*) the duration of the run; (*ii*) calories burned; and (*iii*) distance travelled. Additionally, they utilize a functionality previously developed by us for ALEAPP to display the user's route on an OpenStreetMap. The same approach applies to the **Users** modules (e.g., `adidasUsers.py`). These modules process data related to user accounts, including (*i*) gender, (*ii*) email, (*iii*) height and (*iv*) weight.

The module `NikePolyline.py` contains the visual representation of each activity stored by the application shown in OpenStreetMap. The reason for placing it in a separate module stems from the fact that the activities module already contained a sizable amount of data, and the aim was to streamline it and reduce bloat. The module `NikeNotification.py` contains the notifications that the application sends.

The module `adidasGoals.py` extracts data related to the user-defined goals inside Adidas Running. Lastly, although the output delivered by the `StravaGPS.py` module is similar to the Activities modules, its approach is entirely different as it extracts data from FIT files.

8. Conclusions

In this work, we developed a post mortem forensic analysis of six popular fitness applications for Android focusing on outdoor running: *Adidas Running*, *MapMyWalk*, *Nike Run Club*, *Pumatrac*, *Runkeeper* and `Strava`. We resorted to a rooted Android smartphone and a Garmin Vivosmart 4 smart band to generate, collect and analyze data in search of meaningful forensic artifacts.

As all applications were similar in nature and structure, we focused on one application—`NikeRunClub`—and then used the process and knowledge learned to adapt the analysis methodology to the other applications.

The `NikeRunClub` application stores forensically meaningful data—activities, GPS coordinates and account data—in SQLite 3 databases in its private directory, mainly in the `databases` directory. Related work has already demonstrated the high value of accessing data from applications that track running workouts as they frequently merge locations with date/time. These can yield critical proof for refuting or, on the contrary, confirming someone's alibi.

We analyzed the remaining applications based on `NikeRunClub` and, by using the same methods, we acquired similar information in their respective files. The main findings are as follows:

- The applications do not store data in their public directory. The exception is Strava.
- Workout activities are stored in SQLite 3 databases, with the most significant forensic data being GPS coordinates, timestamps and duration. Strava is again the exception, resorting to the FIT file format to store workouts.
- The format for GPS coordinates depends on the applications, with some encoding geolocation coordinates using the *polyline* format and others relying on a text-based pair of latitude/longitude values.
- User's account data are often stored in a database, although they can also be in XML format in the `shared_prefs` folder.

To decode FIT files, we developed the `decode_FIT` Python script. It extracts activity data and formats them for rendering on an OpenStreetMap map or generates a KMZ file for visualization in Google Earth. Additionally, we created a set of modules for the ALEAPP framework to parse application data and generate a case report for streamlined data interpretation. We further enhanced the ALEAPP report with a new timeline feature, previously incorporated into the ALEAPP repository. As ALEAPP is integrated with the Windows version of the Autopsy forensic software, our modules will also be seamlessly incorporated into this forensic application.

As future work, we plan to assess the privacy and security features of the applications, analyzing the data collected and sent to their respective cloud servers. For this purpose, one needs to resort to dynamic analysis techniques of mobile applications, intercepting and analyzing the HTTPS traffic between applications and their cloud servers, to decode and map their APIs [34]. Another task for future work is to assess the performance of our methodologies, looking for possible optimization. Finally, we also plan to analyze the iOS version of the studied applications and see if the same or even more data can be obtained so that we can adapt our ALEAPP modules to the iOS Logs, Events, And Plists Parser (iLEAPP) framework focused on, which is similar to ALEAPP but focused on iOS applications.

Author Contributions: Conceptualization, F.N., P.D. and M.F.; software, F.N.; Validation, P.D. and M.F.; writing: F.N., P.D. and M.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Fundação para a Ciência e a Tecnologia grant number UIDB 04524/2020 and by European Union grant number UIDB/EEA 50008/2020.

Data Availability Statement: Data are not available due to privacy concerns.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ADB	Android Debug Bridge
ALEAPP	Android Logs Events And Protobuf Parser
API	Application Programming Interface
BLE	Bluetooth Low Energy
FIT	Flexible and Interoperable data Transfer
GPS	Global Positioning System
IDE	Integrated Development Environment
iLEAPP	iOS Logs, Events And Plists Parser
KML	Keyhole Markup Language
KMZ	KML compressed file
SpO_2	Oxygen Saturation
XML	Extensible Markup Language

References

1. Silva, A.G.; Simões, P.; Queirós, A.; P Rocha, N.; Rodrigues, M. Effectiveness of Mobile Applications Running on Smartphones to Promote Physical Activity: A Systematic Review with Meta-Analysis. *Int. J. Environ. Res. Public Health* **2020**, *17*, 2251. [CrossRef] [PubMed]
2. Yeoh, R.; Kim, H.K.; Kang, H.; Lin, Y.A.; Ho, A.D.; Ho, K.F. What Determines Intentions to Use Mobile Fitness Apps? The Independent and Joint Influence of Social Norms. *Int. J. Hum.–Comput. Interact.* **2024**, *40*, 121–130. [CrossRef]
3. Reiber, L. *Mobile Forensic Investigations A Guide to Evidence Collection, Analysis, and Presentation*, 2nd ed.; McGraw-Hill Education: New York, NY, USA, 2019; p. 560.
4. Hassenfeldt, C.; Baig, S.; Baggili, I.; Zhang, X. Map My Murder! A digital forensic study of mobile health and fitness applications. In Proceedings of the 14th International Conference on Availability, Reliability and Security, Canterbury, UK, 26–29 August 2019. [CrossRef]
5. Watts, A. Police Use Murdered Woman's Fitbit Movements to Charge Her Husband—CNN. 2017. Available online: https://edition.cnn.com/2017/04/25/us/fitbit-womans-death-investigation-trnd/index.html (accessed on 8 August 2024).
6. Ganjoo, S. GPS Data from Garmin Smartwatch Helps Police Catch a Man Convicted of Two Murders. 2019. Available online: https://www.indiatoday.in/technology/news/story/how-a-garmin-smartwatch-helped-police-catch-a-man-convicted-of-two-murders-1435570-2019-01-21 (accessed on 8 August 2024).
7. Moyer, J. Police Used a Fitness App to Find a Man Accused of Knocking a Bicyclist to the Ground in Virginia—The Washington Post. 2018. Available online: https://www.washingtonpost.com/local/public-safety/police-used-a-fitness-app-to-find-a-man-accused-of-knocking-a-bicyclist-to-the-ground-in-virginia/2018/05/18/0a4ac6f8-5ab6-11e8-a3d1-b39671d2371e_story.html (accessed on 8 August 2024).
8. Cole, S. Apple Health Data Is Being Used as Evidence in a Rape and Murder Investigation. Available online: https://www.vice.com/en/article/43q7qq/apple-health-data-is-being-used-as-evidence-in-a-rape-and-murder-investigation-germany (accessed on 8 August 2024).
9. Nunes, F.; Domingues, P.; Frade, M. Post-mortem digital forensic analysis of the Garmin Connect application for Android. *Forensic Sci. Int. Digit. Investig.* **2023**, *47*, 301624. [CrossRef]
10. Byambasuren, O.; Beller, E.; Glasziou, P. Current Knowledge and Adoption of Mobile Health Apps Among Australian General Practitioners: Survey Study. *JMIR Mhealth Uhealth* **2019**, *7*, e13199. [CrossRef] [PubMed]
11. Scott, K.; Richards, D.; Adhikari, R. A review and comparative analysis of security risks and safety measures of mobile health apps. *Australas. J. Inf. Syst.* **2015**, *19*, 1–18. [CrossRef]
12. Wylie, K. Stalked via Strava: 'Heartbroken' Man Refused to Believe Romance Was Over. 2022. Available online: https://www.pressandjournal.co.uk/fp/news/crime-courts/5170075/heartbroken-boyfriend-stalked-woman/ (accessed on 8 August 2024).
13. Gritten, D. Strava App Flaw Revealed Runs of Israeli Officials at Secret Bases—BBC News. 2022. Available online: https://www.bbc.com/news/world-middle-east-61879383 (accessed on 8 August 2024).
14. Couture, J. Reflections from the 'Strava-sphere': Kudos, community, and (self-)surveillance on a social network for athletes. *Qual. Res. Sport. Exerc. Health* **2021**, *13*, 184–200. [CrossRef]
15. Sinha, R.; Sihag, V.; Choudhary, G.; Vardhan, M.; Singh, P. Forensic Analysis of Fitness Applications on Android. *Commun. Comput. Inf. Sci.* **2022**, *1544*, 222–235. [CrossRef]
16. Hutchinson, S.; Mirza, M.M.; West, N.; Karabiyik, U.; Rogers, M.K.; Mukherjee, T.; Aggarwal, S.; Chung, H.; Pettus-Davis, C. Investigating Wearable Fitness Applications: Data Privacy and Digital Forensics Analysis on Android. *Appl. Sci.* **2022**, *12*, 9747. [CrossRef]
17. Donaire-Calleja, P.; Robles-Gómez, A.; Tobarra, L.; Pastor-Vargas, R. Forensic Analysis Laboratory for Sport Devices: A Practical Use Case. *Electronics* **2023**, *12*, 2710. [CrossRef]

18. van Zandwijk, J.P.; Boztas, A. Digital traces and physical activities: Opportunities, challenges and pitfalls. *Sci. Justice* **2023**, *63*, 369–375. [CrossRef] [PubMed]
19. van Zandwijk, J.P.; Boztas, A. The phone reveals your motion: Digital traces of walking, driving and other movements on iPhones. *Forensic Sci. Int. Digit. Investig.* **2021**, *37*, 301170. [CrossRef]
20. Jennings, L.; Sorell, M.; Espinosa, H.G. Interpreting the location data extracted from the Apple Health database. *Forensic Sci. Int. Digit. Investig.* **2023**, *44*, 301504. [CrossRef]
21. van Zandwijk, J.P.; Boztas, A. The iPhone Health App from a forensic perspective: Can steps and distances registered during walking and running be used as digital evidence? *Digit. Investig.* **2019**, *28*, S126–S133. [CrossRef]
22. Goh, C.M.J.L.; Wang, N.X.; Müller, A.M.; Yap, R.; Edney, S.; Müller-Riemenschneider, F. Validation of Smartphones and Different Low-Cost Activity Trackers for Step Counting Under Free-Living Conditions. *J. Meas. Phys. Behav.* **2023**, *6*, 79–87. [CrossRef]
23. Fukami, A.; Stoykova, R.; Geradts, Z. A new model for forensic data extraction from encrypted mobile devices. *Forensic Sci. Int. Digit. Investig.* **2021**, *38*, 301169. [CrossRef]
24. Business Research Insights. Running Apps Market Size, Trend, Growth and Overview 2023 to 2030. 2023. Available online: https://www.businessresearchinsights.com/market-reports/running-apps-market-103263 (accessed on 8 August 2024).
25. Kent, K.; Chevalier, S.; Grance, T.; Dang, H. Special Publication 800-86 Guide to Integrating Forensic Techniques into Incident Response Recommendations of the National Institute of Standards and Technology 2006. Available online: https://csrc.nist.gov/pubs/sp/800/86/final (accessed on 8 August 2024).
26. Developers, A. Android 8.0 Behavior Changes—Android Developers. 2023. Available online: https://developer.android.com/about/versions/oreo/android-8.0-changes#security-all (accessed on 8 August 2024).
27. Muraina, I.; Alobaedy, M.; Ibrahim, H. A Framework for Preserving Data Integrity during Mobile Device Forensic in Open Source Software Environment. In Proceedings of the Free and Open Source Software Conference (FOSSC), Muscat, Oman, 14–15 February 2017; pp. 22–26.
28. Skulkin, O.; Tindall, D.; Tamma, R. *Learning Android Forensics: Analyze Android Devices with the Latest Forensic Tools and Techniques*, 2nd ed.; Packt Publishing: Birmingham, UK, 2018.
29. Google. Encoded Polyline Algorithm Format. 2023. Available online: https://developers.google.com/maps/documentation/utilities/polylinealgorithm (accessed on 8 August 2024).
30. BBC News. Fitness App Strava Lights Up Staff at Military Bases. 2018. Available online: https://www.bbc.com/news/technology-42853072 (accessed on 8 August 2024).
31. Song, K.; Oh, D. Bike computer forensics: An efficient and robust method for FIT file recovery. *Forensic Sci. Int. Digit. Investig.* **2023**, *46*, 301606. [CrossRef]
32. Mehmood, N.Q.; Culmone, R. An ANT+ Protocol Based Health Care System. In Proceedings of the 29th International Conference on Advanced Information Networking and Applications Workshops, WAINA 2015, Gwangju, Republic of Korea, 24–27 March 2015; pp. 193–198. [CrossRef]
33. Wu, T.; Breitinger, F.; O'Shaughnessy, S. Digital forensic tools: Recent advances and enhancing the status quo. *Forensic Sci. Int. Digit. Investig.* **2020**, *34*, 300999. [CrossRef]
34. Brown, J.; Onik, A.R.; Baggili, I. Blue Skies from (X's) Pain: A Digital Forensic Analysis of Threads and Bluesky. In Proceedings of the 19th International Conference on Availability, Reliability and Security, ARES '24, Vienna, Austria, 30 July 2024. [CrossRef]