8-2019

Map My Murder: A Digital Forensic Study of Mobile Health and Fitness Applications

Courtney Hassenfeldt
University of New Haven

Shabana Baig
University of New Haven

Ibrahim Baggili
University of New Haven, ibaggili@newhaven.edu

Xiaolu Zhang
University of Texas at San Antonio

Publisher Citation

Hassenfeldt, C., Baig, S., Baggili, I., & Zhang, X. (2019, August). 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, ARES '19 (Article No. 42). ACM.

Comments

Dr. Baggili was appointed to the University of New Haven's Elder Family Endowed Chair in 2015.

This is the authors' version of the paper published in ARES '19 Proceedings of the 14th International Conference on Availability, Reliability and Security by ACM. ISBN: 978-1-4503-7164-3 The paper of record can be found at http://dx.doi.org/10.1145/3339252.3340515

Map My Murder! A Digital Forensic Study of Mobile Health and Fitness Applications

Courtney Hassenfeldt
University of New Haven
chass1@unh.newhaven.edu

Shabana Baig
University of New Haven
sbaig1@unh.newhaven.edu

Ibrahim Baggili
University of New Haven
ibaggili@newhaven.edu

Xiaolu Zhang
University of Texas at San Antonio
xiaolu.zhang@utsa.edu

ABSTRACT

The ongoing popularity of health and fitness applications catalyzes the need for exploring forensic artifacts produced by them. Sensitive Personal Identifiable Information (PII) is requested by the applications during account creation. Augmenting that with ongoing user activities, such as the user's walking paths, could potentially create exculpatory or inculpatory digital evidence. We conducted extensive manual analysis and explored forensic artifacts produced by (n = 13) popular Android mobile health and fitness applications. We also developed and implemented a tool that aided in the timely acquisition and identification of artifacts from the examined applications. Additionally, our work explored the type of data that may be collected from health and fitness web platforms, and Web Scraping mechanisms for data aggregation. The results clearly show that numerous artifacts may be recoverable, and that the tested web platforms pose serious privacy threats.

KEYWORDS

Forensics, Artifacts, Applications, Health, Fitness.

ACM Reference Format:
Courtney Hassenfeldt, Shabana Baig, Ibrahim Baggili, and Xiaolu Zhang. 2019. 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 (ARES 2019) (ARES '19), August 26–29, 2019, Canterbury, United Kingdom.* UK, August 26-29, 2019, 12 pages. https://doi.org/10.1145/3339252.3340515

1 INTRODUCTION

Mobile/smart devices are being used as powerful self-health monitoring tools. This movement is considered to have more impact on health care than immobile health facilities [31]. Due to the advancement of mobile health, there are now ample applications and wearable technologies that address not just our health and fitness concerns, but treatments as well. The number of mobile health applications has increased enormously for both consumers and professionals. As of 2017, more than 325,000 'mhealth' applications were available, Android being the leading platform[1]. The mobile health industry is expected to hold a market share of almost $37 billion by 2019 and $60 billion by 2020[2].

However, the widespread use of mobile fitness/health technologies is just another catalyst for user data aggregation [5]. The large amount of sensitive personally identifiable data stored by mobile health and fitness applications poses an extensive threat to our privacy. Yet, for digital sleuths, these user-data-rich applications are a treasure chest of digital evidence. In fact, evidence retrieved from such applications has already helped in solving real cases[3].

Our work focused on the collection and analysis of forensic artifacts retrieved from widely used mobile health and fitness applications. The aim was to outline a template for digital forensic investigators to follow when examining an Android device with installed health and fitness application(s). In total, (n = 13) Android health and fitness applications were analyzed. A single application had 50 million downloads at the time of this writing, illustrating the ubiquity of such software. To date, there has not been a comprehensive forensic analysis of these applications, and thus our contributions are as follows:

- Our work has produced a comprehensive collection of Android health and fitness application digital forensic artifacts, which are shared on the Artifact Genome Project [18].
- Our work provides a novel method for triaging data potentially stored on a mobile device.
- Our devised method is also implemented in a tool that aids in that automation of the acquisition and analysis of evidence retrieved from health and fitness applications on Android. [4]
- As an auxiliary contribution, our work provides insight into privacy findings related to the amount of user data made public by some of the health and fitness applications.

The rest of this paper includes related work in Section 2. Section 3 shares the employed methodology. We then discuss the experimental results and findings in Section 4, followed by the discussion of our findings in Section 5. Lastly in Section 6, we conclude and discuss future work.

ARES '19, August 26–29, 2019, Canterbury, United Kingdom
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-7164-3/19/08…$15.00
https://doi.org/10.1145/3339252.3340515

[1] https://research2guidance.com/325000-mobile-health-apps-available-in-2017/
[2] https://www.statista.com/topics/2263/mhealth/
[3] http://www.newser.com/story/268299/iphone-app-foils-husband-who-murdered-wife.html
[4] The tool may be downloaded from https://www.unhcfreg.com/datasetsandtools.

2 RELATED WORK

Mobile forensics and application analysis are not new topics. As early as the first generation of smartphones, Bader et al., [9] conducted the first forensic analysis on the iPhone 3Gs. That early work aimed at finding significant evidence such as e-mail messages, text and multimedia from the logical backup of the phone. In 2011, research investigating 3rd party mobile applications on the iPhone was conducted [28]. The research focused on analyzing built-in application data stored in files with standardized formats like databases, JSON and XML.

Android and iOS now occupy the majority of the mobile market and this is reflected in mobile forensics research. While data privacy has become paramount, digital forensic researchers continue to seek solutions for extracting digital evidence from the changing mobile landscape.

In Section 2.1, we share work that improved data acquisition and application analysis on smart phones. In Section 2.2, we discuss case studies that focused on sharing forensic solutions for mobile devices or applications.

2.1 Data Acquisition Approaches

Starting from the first generation of mobile devices, data acquisition has been continuously studied. A typical manner for acquiring logical data from an Android device is to utilize the Android Debug Bridge (ADB) client since Android has a built-in ADB server [23, 27]. Although this approach is still wildly in use, it requires the investigator to gain physical device access by unlocking the screen to activate the ADB server. Therefore, an acquisition method based on changing the custom recovery image has also been studied [39]. Based on this approach commercial tools such as Cellebrite[5] and XRY[6] are able to pull physical forensic images of devices as well.

Other research has focused on using hardware acquisition approaches. For example, Universal Synchronous Receiver-Transmitter (UART), a fancy name for a serial port, and Joint Test Action Group (JTAG), a reserved parallel port for system debugging, are the most commonly used physical ports for examiners to acquire firmware/file systems physically from FLASH memory chips on the a smartphone's Printed Circuit Board (PCB) [12, 15]. These approaches have also been applied to other devices.

Work conducted by [29] utilized UART to pull a vehicle's data from its automotive system. Similarly, work by [36] performed data acquisition on Global Positioning System (GPS) navigation systems through UART. Other than accessing the PCB physically, the last resort for physical acquisition is Chip-off. In Chip-off, investigators disassemble the flash memory chip from the PCB and read the data by using dedicated chip readers [16].

2.2 Application Analysis

Mobile application analysis is an evolving topic. Work has primarily focused on executable analysis and user data analysis.

In [14], researchers presented a decompiler that converts executable files back to source code. This enabled digital investigators to reverse engineer applications as well as understand operational mechanisms of the application(s). For large scale application analysis, [41] focused on in-memory analysis of large application datasets. AndroParse, an extendable feature dataset of 67,703 benign and 46,683 malicious Android applications [33] has also been created by researchers.

In the application analysis domain, work has focused on the manual analysis of social applications for Android/iOS [6, 30, 37]. Other work has focused on single widely used mobile applications, such as WhatsApp [7, 26], and WeChat [38, 42]. Researchers have also analyzed cryptowallets [21], and application valuts [40].

Other relevant work by [20] studied medical information from both Android and iOS applications associated with a physical medical device. Related work has also focused on wearable devices such as fitness trackers [25] and smart watches [10, 13].

Lastly, it is important to note that there is some previous work in the area of health/fitness application analysis. [8] examined 40 popular Android health applications in 2015. In their work, user credentials, location and more personal information was discovered. Since 2015, some of the applications examined by [8] are no longer in use or have been significantly updated (e.g. MyFitnessPal was examined in version 3.6.1 and its current version is 18.9.2). Most importantly, this past work did not introduce an automation tool to aid in the future forensic analysis of such health data found on mobile platforms. This work is discussed more in Section 5.

3 METHODOLOGY

In this study, (n = 13) mobile health and fitness applications (See Table 2) were selected by searching the Google Play Store with the following search terms "fitness" and "health." From these preliminary results, we took into account the number of downloads for each result.

In the first phase, we set up the experimental environment which included downloading the health applications on to a hypothetical suspect's Android device, generating user data through regular daily usage, installing digital forensic tools on our examiner workstation and setting up the network (more details can be found in Sec. 3.1).

In the second phase, we utilized the tools installed for acquiring data from the hypothetical suspect's phone (See Sec. 3.2). To analyze the data generated by the health and fitness applications, we explored the data manually and theorized a new approach that can be potentially used for application analysis (See Section 3.3). This approach will assist digital investigators in triaging the structure of the mobile data and validating their findings afterwards.

Finally, we constructed the theorized approach as a python tool to automate the process of data analysis and artifact collection (See Sec. 3.4), and concluded with the use of a web scraper on two different websites (Strava and MyFitnessPal).

3.1 Apparatus

Table 1 shows the hardware and software used in this work. Considering that results may vary on different Android devices, we used two Android devices (a Nexus tablet and a ZTE phone) with different Android OS versions for our testing.

A Windows 10 and a Mac OSX computers were employed as forensic workstations. On the Windows machine we installed forensic tools for data acquisition and network traffic analysis. The Mac

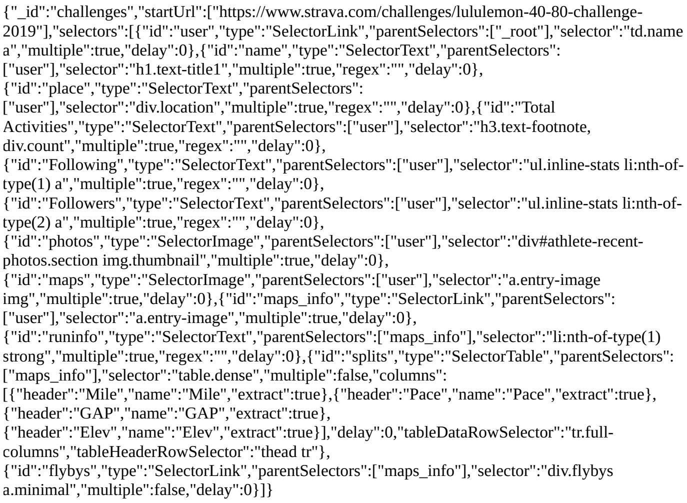
[5] https://www.cellebrite.com/en/home/
[6] https://www.msab.com/products/

Table 1: Apparatus

Hardware/software	Version	Use
Examination PC	Windows 10	Run .XRY, Wireshark, Network Miner, Netwitness Investigator, and Hotspot
Computer	MacOS High Sierra	To analyze data and run python script
XRY	7.4.0	To acquire the data from Android devices
DB Browser for SQLite	3.10.1	To analyze db files
ADB	1.0.40	To pull data from Android device
Wireshark	2.4.1	To capture and analyze network traffic
Netwitness Investigator	10.6	To analyze network traffic
Network Miner	2.3.2	To analyze network traffic
Nexus 7 Tablet	Android 6.0.1	To test applications on
ZTE (Model Z557BL)	Android 7.1.1	To test applications on
Web Scraper (Google Extension)	0.3.8	To extract data from Strava and MapMyFitness accounts

computer was used for application data analysis. We tested our proposed tool on the acquired data.

A WiFi hotspot was created on the Windows workstation, which the Nexus device connected to. Wireshark was used to capture the network traffic when the applications were used. The network traffic for each application was saved in individual pcap files, which were then analyzed using Wireshark, Network Miner, and Netwitness Investigator.



```
{"_id":"challenges","startUrl":["https://www.strava.com/challenges/lululemon-40-80-challenge-
2019"],"selectors":[{"id":"user","type":"SelectorLink","parentSelectors":["_root"],"selector":"td.name
a","multiple":true,"delay":0},{"id":"name","type":"SelectorText","parentSelectors":
["user"],"selector":"h1.text-title1","multiple":true,"regex":"","delay":0},
{"id":"place","type":"SelectorText","parentSelectors":
["user"],"selector":"div.location","multiple":true,"regex":"","delay":0},{"id":"Total
Activities","type":"SelectorText","parentSelectors":["user"],"selector":"h3.text-footnote,
div.count","multiple":true,"regex":"","delay":0},
{"id":"Following","type":"SelectorText","parentSelectors":["user"],"selector":"ul.inline-stats li:nth-of-
type(1) a","multiple":true,"regex":"","delay":0},
{"id":"Followers","type":"SelectorText","parentSelectors":["user"],"selector":"ul.inline-stats li:nth-of-
type(2) a","multiple":true,"regex":"","delay":0},
{"id":"photos","type":"SelectorImage","parentSelectors":["user"],"selector":"div#athlete-recent-
photos.section img.thumbnail","multiple":true,"delay":0},
{"id":"maps","type":"SelectorImage","parentSelectors":["user"],"selector":"a.entry-image
img","multiple":true,"delay":0},{"id":"maps_info","type":"SelectorLink","parentSelectors":
["user"],"selector":"a.entry-image","multiple":true,"delay":0},
{"id":"runinfo","type":"SelectorText","parentSelectors":["maps_info"],"selector":"li:nth-of-type(1)
strong","multiple":true,"regex":"","delay":0},{"id":"splits","type":"SelectorTable","parentSelectors":
["maps_info"],"selector":"table.dense","multiple":false,"columns":
[{"header":"Mile","name":"Mile","extract":true},{"header":"Pace","name":"Pace","extract":true},
{"header":"GAP","name":"GAP","extract":true},
{"header":"Elev","name":"Elev","extract":true}],"delay":0,"tableDataRowSelector":"tr.full-
columns","tableHeaderRowSelector":"thead tr"},
{"id":"flybys","type":"SelectorLink","parentSelectors":["maps_info"],"selector":"div.flybys
a.minimal","multiple":false,"delay":0}]}
```

Figure 1: Sitemap for Web Scraper

3.2 Data Acquisition

After the application accounts were created and the fitness activities were collected, we completed the data acquisition process using two approaches. We utilized XRY toolkit for physical acquisition and the Android Debug Bridge (ADB) for logical acquisition. XRY backed up everything on the device and saved the extracted data to the workstation. The command `adb pull` then acquired the logical files and folders from the smart phones. Afterwards, the data was thoroughly examined and findings were iteratively recorded.

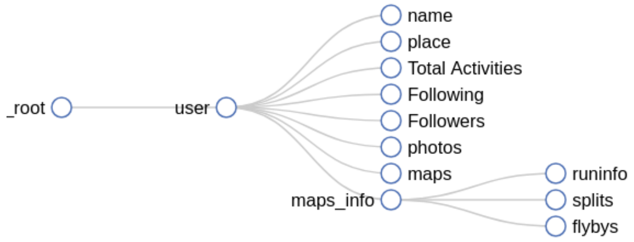


Figure 2: Sitemap Graph

3.3 Data Analysis

Before performing manual analysis, there is an opportunity for investigators to utilize Open Source InTelligence (OSINT) to gain insight into what could be potentially retrieved from a mobile application, or to simply profile a suspect. Applications may also have a web interface, and suspects may attempt misinformation tactics by engaging with a web platform after a device has been legally seized. Therefore, as a primary step to our manual acquisition and analysis, we explored the web application for Strava as a proof of concept.

First, we logged into the Strava web platform with the user account created in Phase 1 and employed a Google extension called *Web Scraper* to scrape user data. The scrapped data was stored in a *Sitemap* (a layered Json file). The structure of the data can be visualized in a *Sitemap* layered graph. Creating a new Sitemap required a structured listing of pages, links and texts to be scraped from the site. Once a Sitemap is created, it can be imported when needed. For example, Figure 1 shows a sample sitemap and Figure 2 is its corresponding graph. We mapped the Strava site, and executed the scraper for 10 minutes to explore the kind of public data that may be downloaded about the users.

3.4 Tool Creation

The purpose for constructing this tool was to expedite the time for acquiring and locating forensically relevant data. Investigators face an increased need for timely evidence collection and examination. We implemented a Python tool that (1) extracts data from the Android device, and (2) searches through the extracted files while locating forensic artifacts. The high-level algorithm employed is shown in Algorithm 1.

The constructed tool allows for any Android device to be connected to a computer and, using the Android Debug Bridge (adb), a command line tool that allows communication with Android devices, fosters the data extraction process. The data is searched through according to file type. Here, the three files types focused on were image, XML, and SQLite database files.

When dealing with image files, images are copied and stored into a directory. An output file is also created to store information on the files found. For image files, the image name and the path to the location of the image is written to the output file.

Algorithm 1 Automation algorithm

```
Requirements: Connected Android device and ADB installed.
Input: (optional) inputFile with keywords
Output: Mobile application artifacts
if inputFile then
    keywords ← getContent(inputFile)
else
    keywords ← DEFAULTKEYWORDS
end
dataPath ← getExtractedDataPath()                    ▷ Path to device's data
hashDict ← computeHash(dataPath)                     ▷ For verification
hashDict ← saveHash(hashDict)
fileList ← getAllFiles(dataPath)
fileTypes ← ['XML', 'DB', 'IMAGE']
sortedList ← sortFilesByType(fileList, fileTypes)
for file in sortedList do
    if fileType is XML then
        if hasKeywords(file) then
            outputFile.append(getName(file), keyword,   linenumber)
    else if fileType is DB then
        if hasKeywords(file) then
            outputFile.append(getName(file), keyword, table,
            rownumber)
    else if fileType is IMG then
        saveImg(file) to artifactDir
        outputFile.write(getName(file))
    val ← compareHash(computeHash(file), hashDict(file))
    checkHash(val)                         ▷ Check if hash of the file is unchanged
end
```

With XML and database files, each file is opened and user-defined keywords are searched for. Keywords were generated using known-artifacts. If a match to one of the keywords in an XML file is found, then the path of the file, the keyword that was found, the line number where the keyword was found, and the line that the keyword was found on is printed to the output file. If it is a database file, then the file path, keyword found, and the row that the keyword was found in is printed to the output file.

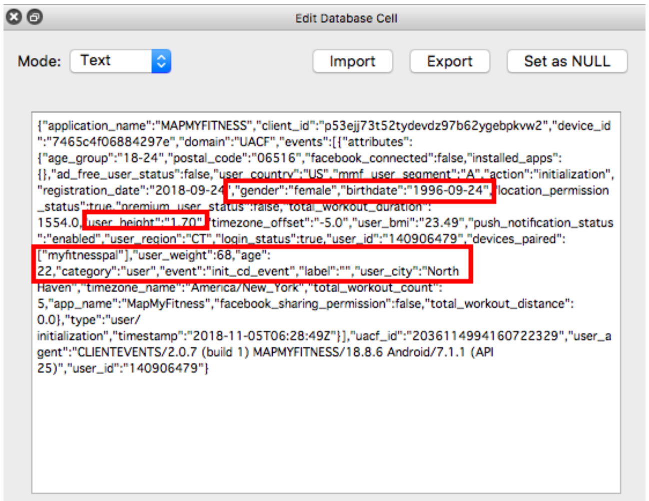
As the admissibility of any type of evidence is important, a copy of the extracted data is made (Copy 2) and hashes are generated for future comparison. The original extraction (Copy 1) is stored to be verified against at any given point in time to show that data has not been tampered with. Any data that is found to be important is copied to an output file outside of the data itself while not altering the original file. Once the search process is complete, the tool will then re-hash the files and compare them against the original. This is done to prove that data has remained untouched after the search process ends its execution.

4 EXPERIMENTAL RESULTS

Extensive application artifacts were found on the devices. All artifacts can be downloaded from the Artifact Genome Project (AGP) at https://agp.newhaven.edu [19]. Some of the more intriguing data is outlined in the following sections. Section 4.1 elaborates on the personal data that was supplied to the applications by the user. Section 4.2 shows the different types of user health data that was found stored on the device. Section 4.3 presents the applications that stored location data from activities completed by the user, followed by Section 4.4 which covers password information found stored on the device and then Section 4.5, which covers interesting findings. Lastly, Section 4.6 summarizes information found when employing the web scraper on the Strava platform. A comprehensive report of the artifacts is shared in Tables 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17 in the Appendix.

4.1 Personal Data

The type of user data requested by tested applications is shown in Table 2. Much of this requested personal data can be found stored in database and XML files. Health and fitness applications tend to request data such as name, e-mail, age, gender, birthday, zip code, height, weight, and blood pressure. All of this data is especially sensitive and can be extremely helpful with identifying the user. This data helps to not only identify the user by name, but also their physical characteristics. An example of this is shown in Figure 3, which illustrates the extraction of gender, height, weight, age, and city from MapMyFitness. Note that the MapMyRun, MapMyFitness, MapMyRide, MapMyWalk and MapMyHike all use a unified account.



```
{"application_name":"MAPMYFITNESS","client_id":"p53ejj73t52tydevdz97b62ygebpkvw2","device_id
":"7465c4f06884297e","domain":"UACF","events":[{"attributes":
{"age_group":"18-24","postal_code":"06516","facebook_connected":false,"installed_apps":
{},"ad_free_user_status":false,"user_country":"US","mmf_user_segment":"A","action":"initialization",
"registration_date":"2018-09-24","gender":"female","birthdate":"1996-09-24","location_permission
_status":true,"premium_user_status":false,"total_workout_duration":
1554.0,"user_height":"1.70","timezone_offset":"-5.0","user_bmi":"23.49","push_notification_status
":"enabled","user_region":"CT","login_status":true,"user_id":"140906479","devices_paired":
["myfitnesspal"],"user_weight":68,"age":
22,"category":"user","event":"init_cd_event","label":"","user_city":"North
Haven","timezone_name":"America/New_York","total_workout_count":
5,"app_name":"MapMyFitness","facebook_sharing_permission":false,"total_workout_distance":
0.0},"type":"user/
initialization","timestamp":"2018-11-05T06:28:49Z"}],"uacf_id":"2036114994160722329","user_a
gent":"CLIENTEVENTS/2.0.7 (build 1) MAPMYFITNESS/18.8.6 Android/7.1.1 (API
25)","user_id":"140906479"}
```

Figure 3: MapMyFitness User Information in Database

4.2 User Health Data

The Fitness Tracker application calls for an excessive amount of personal information from the user to completely fill out the profile. Of interest is the request for the following data: ethnicity, blood type, pregnant (yes or no), lactating (yes or no), waist measurement, hip measurement, neck measurement, and metabolism specifications. All of this data was found stored in a log file on the phone in the application directory with the timestamp of when it was added. Each of these different characteristics are attached to a separate date, that way one is able to see when the data was changed and how current it is. This is especially interesting because the application allowed for an upgraded *DNA doctor Connect* feature. The feature allows users to add their raw genetic data from their personalized DNA information from sites such as 23andMe and AncestryDNA.

Table 2: Requested Data at Account Creation

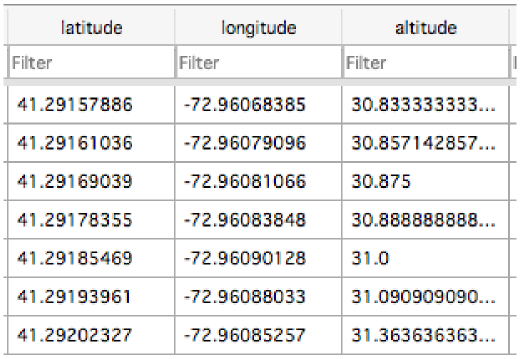
Data Requested	Name	Birthday	Sex	Height	Weight	Email	Location
MapMyFitness (18.8.1)	✗	✗	✗	✗	✗	✗	✗
RunKeeper (9.2.1)	✗	✗	✗			✗	✗
Strava (64.0.0)	✗	✗	✗	✗			
MyFitnessPal (18.9.2)		✗	✗	✗	✗	✗	✗
Runtastic (8.9.2)	✗	✗	✗	✗	✗	✗	
Health Infinity (1.3.7)	✗	✗	✗	✗	✗		
Fitness Tracker (2.1.0)	✗		✗	✗	✗	✗	
Nike Training (5.16.0)	✗	✗	✗			✗	✗
JEFIT (10.11)		✗	✗	✗	✗	✗	

There is also a feature called *Medical Log* which allowed for the user to store health data from doctor visits.

4.3 Geographical Locations

The applications analyzed allowed the user to record their fitness activities as they were being completed. The different types of activities that were available for selection depended on the application being used. Some of those activities include run, walk, jog, swim, bike ride, etc.

To be able to record such data, the applications requested the user allow access to their GPS. When searching through the extracted data, we were able to find SQLite database files that stored data such as longitude, latitude, and elevation during the fitness activity. The date and time stamps for each of these activities, start and end time is also stored. An example can be seen in Figure 4. These applications included Runtastic and MapMyFitness.

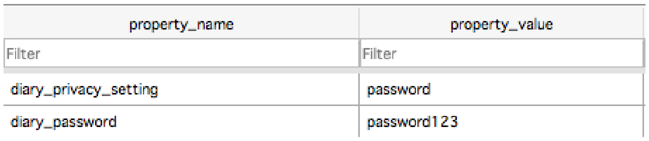


latitude	longitude	altitude
Filter	Filter	Filter
41.29157886	-72.96068385	30.833333333...
41.29161036	-72.96079096	30.857142857...
41.29169039	-72.96081066	30.875
41.29178355	-72.96083848	30.888888888...
41.29185469	-72.96090128	31.0
41.29193961	-72.96088033	31.090909090...
41.29202327	-72.96085257	31.363636363...

Figure 4: Location Database Example

4.4 Password Data

The MyFitnessPal application stored a couple different passwords in clear text. For instance, the user is allowed to set a diary password, making the diary view able only by those with the password. The password is stored in a database file in clear text as shown in Figure 5. This is particularly interesting when one considers that Virginia Tech University and Dashlane analysts carried out a study and "after examining a database of over 28 million users and their 61 million passwords, they have uncovered an alarming figure: 52% of the users studied have the same passwords (or very similar and easily hackable ones) for different services" [1]. This shows that this information can be helpful for investigators with not only this application, but other digital accounts of the same user. MyFitnessPal also allows for the user to set a pin on the application itself so no information is viewable until that pin is entered. This pin is also stored in clear text in the same database file. JEFIT, on the other hand, stores their username and the MD5 hash of the user's password in an XML file.

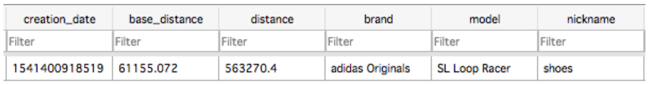


property_name	property_value
Filter	Filter
diary_privacy_setting	password
diary_password	password123

Figure 5: Diary Password Stored in Clear Text in Database File

4.5 Other Data

With fitness and health applications there is a vast amount of stored data that may not be considered at first. In some cases, this information can be beneficial to investigators. Take shoe tracking for example. Some applications, such as RunKeeper and Runtastic, give users the option to include the specific shoes they wear, as well as which activity they wear them on, as a way to track the amount of distance traveled with the specific pair of shoes. Shoe prints may be benifitial in an investigation, and knowing the type of shoe worn by a suspect may indeed prove useful. This data is stored in a database file as shown in Figure 6.

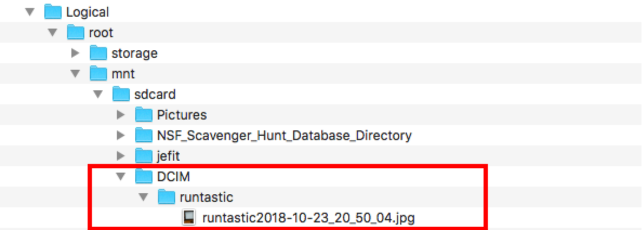


creation_date	base_distance	distance	brand	model	nickname
Filter	Filter	Filter	Filter	Filter	Filter
1541400918519	61155.072	563270.4	adidas Originals	SL Loop Racer	shoes

Figure 6: RunKeeper Shoe Information in Database

Another interesting feature allowed by some applications is the ability to upload photos to correlate with a specific activity or post. These are then stored on the device and in some cases, such as in Figure 7, they are stored with a timestamp. This timestamp correlates with when it was uploaded to the application.

The RunKeeper application also allows for users to select what are called *sub feels.* These sub feels are selected by the user after an activity is completed and are used as a way to track any issues they



```
▼ Logical
  ▼ root
    ▶ storage
    ▼ mnt
      ▼ sdcard
        ▶ Pictures
        ▶ NSF_Scavenger_Hunt_Database_Directory
        ▶ jefit
        ▼ DCIM
          ▼ runtastic
              runtastic2018-10-23_20_50_04.jpg
```

Figure 7: Runtastic Photo Folder

had during the activity. An example of this could be an injury or bad weather. If any of these are selected by the user, they are stored in a database file and reference the trip id as seen in Figure 8. The trip id correlates with the specific activity that was recorded. A piece of information unique to the Health Infinity application is that the user could track their medicine. They can input the medicine name, dosage, time to take, start date, length of time to take the medicine, and comments.

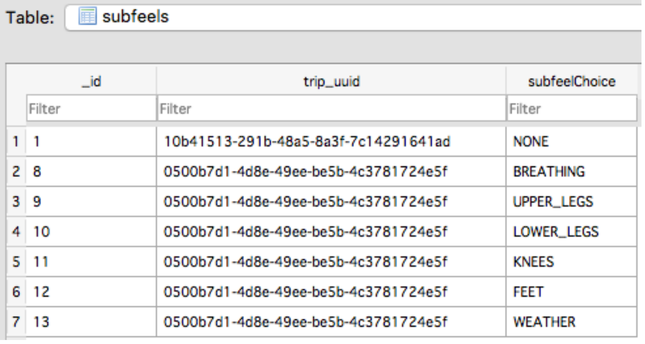


Table: subfeels

	_id	trip_uuid	subfeelChoice
	Filter	Filter	Filter
1	1	10b41513-291b-48a5-8a3f-7c14291641ad	NONE
2	8	0500b7d1-4d8e-49ee-be5b-4c3781724e5f	BREATHING
3	9	0500b7d1-4d8e-49ee-be5b-4c3781724e5f	UPPER_LEGS
4	10	0500b7d1-4d8e-49ee-be5b-4c3781724e5f	LOWER_LEGS
5	11	0500b7d1-4d8e-49ee-be5b-4c3781724e5f	KNEES
6	12	0500b7d1-4d8e-49ee-be5b-4c3781724e5f	FEET
7	13	0500b7d1-4d8e-49ee-be5b-4c3781724e5f	WEATHER

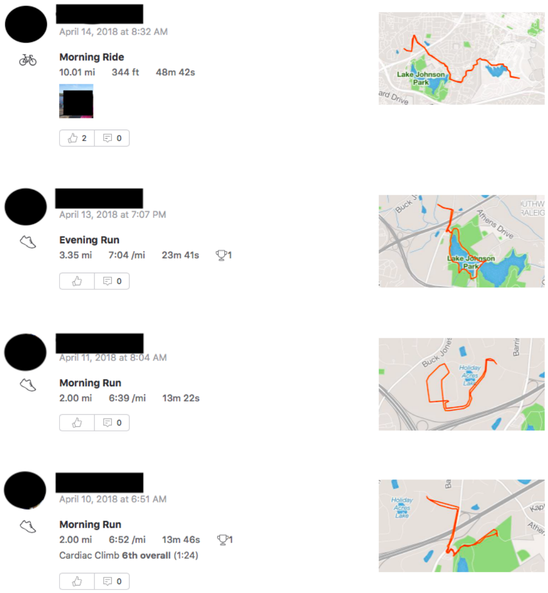
Figure 8: RunKeeper Subfeels in Database

4.6 Auxiliary Finding

As a side finding, when we tried to utilize a Web Scraper to gather data from Strava’s website, we found that through the Leaderboard page, we were able to access other users’ profiles regardless of whether the profile was set to public or private. For both public and private profiles, we were able to extract significant amount of information, such as name, achievements, and photos.

To show how much information we can gather from this website, we analyzed the CSV file that contained our scraped data and summarized it in Table 3. Table 3 shows the summary of number of entries found for the mentioned fields. Some of the field values were null for some users because some of their information was set to private.

Additionally, most of the users shared maps that show routes that they took on a certain day for a certain workout, and they were able to attach photos of that workout as well, as shown in Figure 9. We inspected the element and analyzed the network tab for traffic. We found an XMLHttpRequest (XHR), which had coordinates for every move of the user, as shown in Figure 10. With this information, we were able to know a person’s approximate position at a certain time. Moreover, the maps web page also provided a link to flybys. Flyby is a specific tool which compares distance and time of different users for the specific activity. This is another potential way of getting a list of Strava users. Regardless, the amount of data that one can retrieve about other users using a simple Web Parser is significant. The data that was found from the Strava mobile application can be seen in Table 13.



[redacted] [redacted]
April 14, 2018 at 8:32 AM
Morning Ride
10.01 mi 344 ft 48m 42s
[redacted]
2 0

[redacted] [redacted]
April 13, 2018 at 7:07 PM
Evening Run
3.35 mi 7:04 /mi 23m 41s 🏆1
0

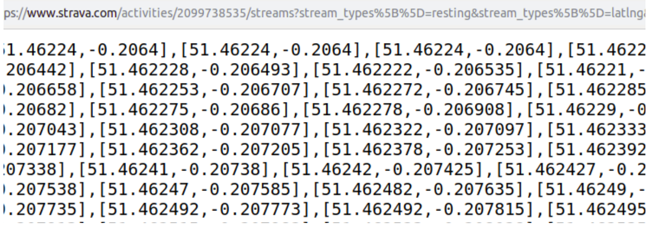
[redacted] [redacted]
April 11, 2018 at 8:04 AM
Morning Run
2.00 mi 6:39 /mi 13m 22s
0

[redacted] [redacted]
April 10, 2018 at 6:51 AM
Morning Run
2.00 mi 6:52 /mi 13m 46s 🏆1
Cardiac Climb **6th overall** (1:24)
0

Figure 9: A profile on Strava

Field	Number of Entries Found
Users	362
Places	18
Following	17
Followers	17
PhotoSources	82
MapSources	76
MapInfo	97
RunInfo	97
FlybysLinks	97

Table 3: Web Scraper Output for Strava



```
ps://www.strava.com/activities/2099738535/streams?stream_types%5B%5D=resting&stream_types%5B%5D=latlng
1.46224,-0.2064],[51.46224,-0.2064],[51.46224,-0.2064],[51.4622
206442],[51.462228,-0.206493],[51.462222,-0.206535],[51.46221,-
.206658],[51.462253,-0.206707],[51.462272,-0.206745],[51.462285
.20682],[51.462275,-0.20686],[51.462278,-0.206908],[51.46229,-0
.207043],[51.462308,-0.207077],[51.462322,-0.207097],[51.462333
.207177],[51.462362,-0.207205],[51.462378,-0.207253],[51.462392
07338],[51.46241,-0.20738],[51.46242,-0.207425],[51.462427,-0.2
.207538],[51.46247,-0.207585],[51.462482,-0.207635],[51.46249,-
.207735],[51.462492,-0.207773],[51.462492,-0.207815],[51.462495
```

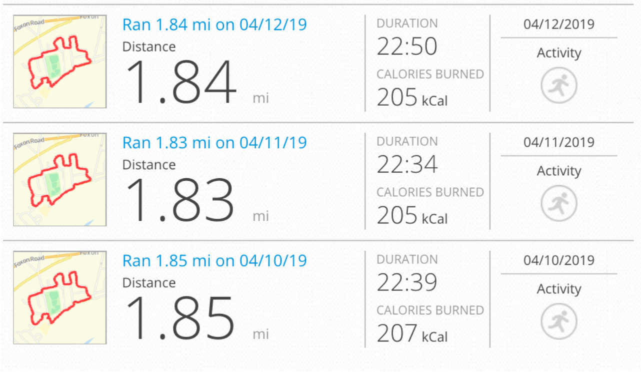
Figure 10: User Coordinates

Under Armour [7], a footware company, owns six applications to track fitness activities. These include MapMyRun, MapMyFitness, MapMyRide, MapMyWalk, MyFitnessPal, and MapMyHike. All of these applications sync together and allow for a unified account. When logging into MapMy applications, the number of miles tracked by all users since the start of 2019 is provided. When we logged in on April 12th, 2019 the amount of miles was 178,756,688.

When looking at the MapMyFitness website it was discovered that a user, once logged in, is able to search for routes that were created near the location they selected. We experimented with the aforementioned finding by using the Web Scraper to collect data about the routes created near the University of New Haven. Table 4 shows a summary of the assembled data. Moreover, users are able to search all the activities or narrow them down based on run, bike ride, walk, etc. Then, once a specific route is selected from the list, they are able to see who created it, and can go to that user's profile. If the person has a profile with public postings, anyone with an account can view their posts. While looking at this, we were able to observe that some users had consistent patterns with their workouts and that some users took the same path at the same time every day. Additionally, it was observed that some users were shown to be starting and ending their workout at their house. For some users we were able to find their social media accounts based on their names and locations of their fitness account. On their public account, one also has access to their friends. An example of workouts posted by a public user account can be seen in Figure 11. The artifacts that were found on the MapMy applications can be seen in Tables 5, 6, 7, 8, and 17.

Field	Number of Entries Found
`Users`	22
`Places`	22
Total Workouts	22
Workout Description	11
Distance covered in miles	22
Map Sources	11

Table 4: Web Scraper Output for MapMyFitness



Ran 1.84 mi on 04/12/19
Distance
1.84 mi
DURATION
22:50
CALORIES BURNED
205 kCal
04/12/2019
Activity

Ran 1.83 mi on 04/11/19
Distance
1.83 mi
DURATION
22:34
CALORIES BURNED
205 kCal
04/11/2019
Activity

Ran 1.85 mi on 04/10/19
Distance
1.85 mi
DURATION
22:39
CALORIES BURNED
207 kCal
04/10/2019
Activity

Figure 11: MapMyFitness User Profile

5 DISCUSSION

Similar research was conducted in 2015 where 40 mobile health applications were analyzed and three were discussed in detail. Our work provided more granular results, and examined OSINT methods using web platforms as well. Prior work was tested using Android 5.0.1 compared to our use of Android 6.0.1 and 7.1.1. Two of the applications they discussed in detail, MyFitnessPal and Runkeeper, were also analyzed here. In terms of the Runkeeper application, older research used version 5.4 whereas our research used version 9.2.1. They focused only on one database file in detail, that being RunKeeper.sqlite. The findings for this file line up with what is stored in the most current version that we examined. We also present the artifacts for this file, as well as other database and XML files as shown in Table 9. Similarly, the MyFitnessPal version used in older work was version is 3.6.1 and the application is now up to version 18.9.2 which was used in our work. The file they discuss in detail for this application is MyFitnessPal.db. While much of the same data is still recorded in this database file, as shown in Table 14, one noticeable difference is that the hash of the user's password is no longer stored, however, as discussed in Section 4.4, they still store the diary password and pin for the application in clear text. While this paper's focus was to provide a generalized idea of mobile fitness applications and the categories of data that they store on devices, our main focus was a detailed account of artifacts found stored due to these applications, as well as a novel, automated mechanism to collect them. We did also examine privacy concerns related to users allowing their data to be public.

In section 4.3, the geographical data found stored on the devices was explained. This type of information has proven helpful to investigators in the past, such as in the murder cases of Jessica Patel and Maria Ladenburger [2, 17]. These cases were able to use the data stored on smartphone health applications to track the activities of a suspect, as well as their location. Now, with the increasing amount of applications to track health and fitness activities and locations are more likely to be used as evidence in cases. These applications also allow for wearable devices to be connected. These wearable devices, such as Fitbits, have become increasingly popular, and the trend of using them for sources of digital evidence will surely continue, such as in the murder case of Karen Navarra [22], where the data from her Fitbit was used as evidence. Our work, therefore, has practical utility.

The data stored pertaining to shoes is beneficial for investigators when the feature is utilized by the user. As seen in Section 4.5, some applications allow for their users to track the exact shoes that they own and the activities they complete with those shoes. Systems are able to track the brand and model, along with the amount of distance the shoe has been worn for. If the user completes a fitness activity, they are able to link their shoe with that activity so it would help show that the device was at this location, at this time, and according to the user input, these shoes were the ones being worn. This is especially interesting when considering that shoe prints have been helpful in the past in terms of building evidence against a suspect. For instance in a 2013 case, a shoe print left at the scene of a murder was used as evidence to convict a suspect [32]. More recently, there was a case involving data from use of the Strava

[7] https://www.underarmour.com/en-us/ua-record

application that helped to place the suspect near the location of the crime at the time of the incident [34].

This is not the first time that the data from Strava user's had public implications. Last year, the application released their heat map that ultimately revealed military bases that were in war zones [24]. There was also an article written that referenced how easy it was for someone to stalk their boyfriend on Strava, and ultimately other females he ran with. The author stating, "I could see where she lived, where she drank beer and got coffee. I knew how many calories she burned working out, and how often. I knew when and where and with whom she spent time" [11].

Occurrences such as those led to the need for section 4.6, where we discussed the utilization of a Web Scraper to show the amount of data that can be collected by anyone with a user account strictly from the data that other users leave public. To show how easily this could lead to unfavorable implications, we used the information found on the site from public profiles to find out more about the users. To show the severity of allowing so much information to be public, we started with finding a user based off the name and current city they provided. With this information, we were able to find some of their corresponding social media accounts such as Facebook and Pinterest. To be able to verify in some cases that this was in fact the right account, the profile photo provided for the MapMy account was matched to their other accounts. Some of these users' other social media profiles were also public and included photos of their children. For one user, we were able to find their baby registry that included the husband's name, who we were also able to find information on based off their spouse's public accounts. We were also able to find their voting registry that included their current and past addresses. From there we used the address listed on their voter's registry to show that the start and end of their path was in fact where they lived.

This is concerning, especially when looking at the fact that "statistics show that 76 percent of female homicide victims are stalked before they are murdered". As well as the fact that Betsy Carlson, a professional dealing with domestic violence cases stated, "Social media has become more of a tactic used by stalkers, and because people can create a false identity or use GPS, on-line sources, and email they don't even have to present and these tools have emboldened stalkers". [35].

6 CONCLUSION AND FUTURE WORK

Having an idea of the types of artifacts that can be found in fitness applications will provide investigators a better lead. The same concept can be applied to Web platforms, due to the fact that they show all the information one can find on a user that is registered with a Strava and MapMyFitness account, and possibly other fitness websites. With the help of a tool such as the one referenced in this paper, investigators can efficiently find specific pieces of information they are looking for.

Future work should focus on health and fitness application analysis on iOS, which makes up 44.3% of mobile users [4]. Another possible area for future work would be to focus on wearable devices. These have grown in popularity with devices such as Apple Watches and FitBits. In 2016, the number of connected wearable devices was 325 million worldwide [3].

In addition, a Sitemaps management system for the Web Scraper warrants future work. This could help examiners create Sitemaps for different websites as needed enabling reuse of such maps. Lastly, data fusion and visualization approaches for correlating data between health and fitness apps, and other devices would be of paramount importance, to expedite future investigations.

ACKNOWLEDGMENTS

The authors would like to thank Justin Grannis, Laura Sanchez and Cinthya Grajeda-Mendez for their review and support of the paper and Justin's help with the automation algorithm.

REFERENCES

[1] 52% of users reuse their passwords. *Panda Security*, 2018. https://www.pandasecurity.com/mediacenter/security/password-reuse/.

[2] Apple health data used in murder trial. *BBC News*, 2018. https://www.bbc.com/news/technology-42663297.

[3] 10 notable facts about wearable technology. *Medium*, 2019. https://medium.com/@TechTalker/10-notable-facts-about-wearable-technology-c01c21070324.

[4] Subscriber share held by smartphone operating systems in the united states from 2012 to 2018. *Statista*, 2019. https://www.statista.com/statistics/266572/market-share-held-by-smartphone-platforms-in-the-united-states/.

[5] Linda Ackerman. Mobile health and fitness applications and information privacy. *Privacy Rights Clearinghouse, San Diego, CA*, 2013.

[6] Noora Al Mutawa, Ibrahim Baggili, and Andrew Marrington. Forensic analysis of social networking applications on mobile devices. *Digital Investigation*, 9:S24–S33, 2012.

[7] Cosimo Anglano. Forensic analysis of whatsapp messenger on android smartphones. *Digital Investigation*, 11(3):201–213, 2014.

[8] Abdullah Azfar, Kim-Kwang Raymond Choo, and Lin Liu. Forensic taxonomy of popular android mhealth apps. *21st Americas Conference on Information Systems*, 2015.

[9] Mona Bader and Ibrahim Baggili. iphone 3gs forensics: Logical analysis using apple itunes backup utility. 2010.

[10] Ibrahim Baggili, Jeff Oduro, Kyle Anthony, Frank Breitinger, and Glenn McGee. Watch what you wear: preliminary forensic analysis of smart watches. In *2015 10th International Conference on Availability, Reliability and Security*, pages 303–311. IEEE, 2015.

[11] Elizabeth Barber. What happens when you stalk your boyfriend on strava. *WIRED*, 2018. https://www.wired.com/story/strava-love-surveillance/.

[12] MF Breeuwsma. Forensic imaging of embedded systems using jtag (boundary-scan). *digital investigation*, 3(1):32–42, 2006.

[13] Quang Do, Ben Martini, and Kim-Kwang Raymond Choo. Is the data on your wearable device secure? an android wear smartwatch case study. *Software: Practice and Experience*, 47(3):391–403, 2017.

[14] William Enck, Damien Octeau, Patrick D McDaniel, and Swarat Chaudhuri. A study of android application security. In *USENIX security symposium*, volume 2, page 2, 2011.

[15] Junbin Fang, Zoe Jiang, Kam-Pui Chow, Siu-Ming Yiu, Lucas Hui, Gang Zhou, Mengfei He, and Yanbin Tang. Forensic analysis of pirated chinese shanzhai mobile phones. In *IFIP International Conference on Digital Forensics*, pages 129–142. Springer, 2012.

[16] Aya Fukami, Saugata Ghose, Yixin Luo, Yu Cai, and Onur Mutlu. Improving the reliability of chip-off forensic analysis of nand flash memory devices. *Digital Investigation*, 20:S1–S11, 2017.

[17] Jenn Gidman. iphone app foils husband who murdered wife. *Newser*, 2018. http://www.newser.com/story/268299/iphone-app-foils-husband-who-murdered-wife.html.

[18] Cinthya Grajeda, Laura Sanchez, Ibrahim Baggili, Devon Clark, and Frank Breitinger. Experience constructing the artifact genome project (agp): Managing the domain's knowledge one artifact at a time. *Digital Investigation*, 26:S47–S58, 2018.

[19] Cinthya Grajeda, Laura Sanchez, Ibrahim Baggili, Devon Clark, and Frank Breitinger. Experience constructing the artifact genome project (agp): Managing the domain's knowledge one artifact at a time. *Digital Investigation*, 26:S47–S58, 2018.

[20] George Grispos, William Bradley Glisson, and Peter Cooper. A bleeding digital heart: Identifying residual data generation from smartphone applications interacting with medical devices, 2019.

[21] Trevor Haigh, Frank Breitinger, and Ibrahim Baggili. If i had a million cryptos: Cryptowallet application analysis and a trojan proof-of-concept. In *International Conference on Digital Forensics and Cyber Crime*, pages 45–65. Springer, 2018.

[22] Christine Hauser. Police use fitbit data to charge 90-year-old man in stepdaughterâĂŹs killing. *NYTimes*, 2018. https://www.nytimes.com/2018/10/03/us/fitbit-murder-arrest.html.

[23] Andrew Hoog. *Android forensics: investigation, analysis and mobile security for Google Android.* Elsevier, 2011.

[24] Bart Jansen. Strava fitness tracking map reveals military bases, movements in war zones. *USA Today*, 2018. https://www.usatoday.com/story/news/world/2018/01/29/strava-war-zones/1073975001/.

[25] Serim Kang, Soram Kim, and Jongsung Kim. Forensic analysis for iot fitness trackers and its application. *Peer-to-Peer Networking and Applications*, pages 1–10, 2018.

[26] Filip Karpisek, Ibrahim Baggili, and Frank Breitinger. Whatsapp network forensics: Decrypting and understanding the whatsapp call signaling messages. *Digital Investigation*, 15:110–118, 2015.

[27] Jeff Lessard and Gary C Kessler. Android Forensics : Simplifying Cell Phone Examinations. *Small Scale Digital Device Forensics Journal*, 4(1):1–12, 2010.

[28] Alex Levinson, Bill Stackpole, and Daryl Johnson. Third party application forensics on apple mobile devices. In *2011 44th Hawaii International Conference on System Sciences*, pages 1–9. IEEE, 2011.

[29] Hafizah Mansor, Konstantinos Markantonakis, Raja Naeem Akram, Keith Mayes, and Iakovos Gurulian. Log your car: The non-invasive vehicle forensics. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 974–982. IEEE, 2016.

[30] Farhood Norouzizadeh Dezfouli, Ali Dehghantanha, Brett Eterovic-Soric, and Kim-Kwang Raymond Choo. Investigating social networking applications on smartphones detecting facebook, twitter, linkedin and google+ artefacts on android and ios platforms. *Australian journal of forensic sciences*, 48(4):469–488, 2016.

[31] Michael Rucker. Mobile health apps and technology. *verywellhealth*, 2018. https://www.verywellhealth.com/mobile-health-4014014.

[32] Mark Russell. Shoe fits in 'frenzied killing' case. *The Age*, 2013. https://www.theage.com.au/national/victoria/shoe-fits-in-frenzied-killing-case-20130628-2p2c4.html.

[33] Robert Schmicker, Frank Breitinger, and Ibrahim Baggili. Androparse-an android feature extraction framework and dataset. In *International Conference on Digital Forensics and Cyber Crime*, pages 66–88. Springer, 2018.

[34] Brent Schrotenboer. Kellen winslow trial day 5: How bike location data incriminated him. *USA Today*, 2019. https://www.usatoday.com/story/sports/2019/05/28/kellen-winslow-trial-cycling-clues-hurt-ex-nfl-star-criminal-case/1267080001/.

[35] David Sharos. Social media has 'emboldened' stalkers, officials say. *The Chicago Tribune*, 2018. https://www.chicagotribune.com/suburbs/aurora-beacon-news/ct-abn-stalking-st-0118-20180117-story.html.

[36] Onno Van Eijk and Mark Roeloffs. Forensic acquisition and analysis of the random access memory of tomtom gps navigation systems. *Digital Investigation*, 6(3-4):179–188, 2010.

[37] Daniel Walnycky, Ibrahim Baggili, Andrew Marrington, Jason Moore, and Frank Breitinger. Network and device forensic analysis of android social-messaging applications. *Digital Investigation*, 14:S77–S84, 2015.

[38] Songyang Wu, Yong Zhang, Xupeng Wang, Xiong Xiong, and Lin Du. Forensic analysis of wechat on android smartphones. *Digital investigation*, 21:3–10, 2017.

[39] Seung Jei Yang, Jung Ho Choi, Ki Bom Kim, and Taejoo Chang. New acquisition method based on firmware update protocols for android smartphones. *Digital Investigation*, 14:S68–S76, 2015.

[40] Xiaolu Zhang, Ibrahim Baggili, and Frank Breitinger. Breaking into the vault: Privacy, security and forensic analysis of android vault applications. *Computers & Security*, 70:516–531, 2017.

[41] Xiaolu Zhang, Frank Breitinger, and Ibrahim Baggili. Rapid android parser for investigating dex files (rapid). *Digital Investigation*, 17:28–39, 2016.

[42] Fan Zhou, Yitao Yang, Zhaokun Ding, and Guozi Sun. Dump and analysis of android volatile memory on wechat. In *2015 IEEE International Conference on Communications (ICC)*, pages 7151–7156. IEEE, 2015.

7 APPENDIX

Table 5: MapMyRun Artifacts

File Name	Information Found
io-uacf-client-events.db	clientID, deviceID, duraton, requestID, timestamp, domain, user agent, app version
workout.db	workouts altitude, latitude, longitude, date, workoutID, start time, end time, distance, calories, note, steps, weight, heart rate, timestamp
com.amplitude.api	userID, OS, OS version, device, language, gender, age, registration date, weight, height, birthday, city
mmdk_user.db	userID, username, email, first name, last name, display name, birthday, gender, height, weight, timezone, date joined, last login, country, region, locality, link to profile image, language
com.google.android.gms.appid.xml	app version, timestamp created, token
com.google.maps.api.android.lib6.drd.PREFERENCES_FILE.xml	sessionID, legal country
cSPrefs.xml	last measurement timestamp, last sessiontimestamp, amount of runs, active user session count, first install ID
currentUserId.xml	current UserID
deviceLocationChecker.xml	last time location was checked
deviceLocationManager.xml	country, zip code, city
eventLogHarness_prefs.xml	last work out ID
fcm_.xml	FCM (firebased client messaging) clientID, registrationID, enabled or not
mfpApiPrefs.xml	API access token
session.xml	appID, clientKeyInfo, clientID, creation timestamps, clientTokenInfo, access token, server key info, user info map, domainUserID, email, display name, full name, userID, user creation date
notification-inbox.db	notifications including userID, engagementID, time created, deleted or not, state (unread, pending, etc.), expiration date, notification info
log_2019-03-18_18/20/47.txt	log of application events

Table 6: MapMyRide Artifacts

File Name	Information Found
io-uacf-client-events.db	clientID, deviceID, duraton, requestID, timestamp, domain, user agent, app version
workout.db	workouts altitude, latitude, longitude, date, workoutID, start time, end time, distance, calories, note, steps, weight, heart rate, timestamp
com.amplitude.api	userID, OS, OS version, device, language, gender, age, registration date, weight, height, birthday, city
mmdk_user.db	userID, username, email, first name, last name, display name, birthday, gender, height, weight, timezone, date joined, last login, country, region, locality, link to profile image, language
com.google.android.gms.appid.xml	app version, timestamp created, token
com.google.maps.api.android.lib6.drd.PREFERENCES_FILE.xml	sessionID, legal country
cSPrefs.xml	last measurement timestamp, last sessiontimestamp, amount of runs, active user session count, first install ID
currentUserId.xml	current UserID
deviceLocationChecker.xml	last time location was checked
deviceLocationManager.xml	country, zip code, city
eventLogHarness_prefs.xml	last work out ID
fcm_.xml	FCM (firebased client messaging) clientID, registrationID, enabled or not
mfpApiPrefs.xml	API access token
session.xml	appID, clientKeyInfo, clientID, creation timestamps, clientTokenInfo, access token, server key info, user info map, domainUserID, email, display name, full name, userID, user creation date
notification-inbox.db	notifications including userID, engagementID, time created, deleted or not, state (unread, pending, etc.), expiration date, notification info
log_2019-03-18_18/20/47.txt	log of application events

Table 7: MapMyWalk Artifacts

File Name	Information Found
io-uacf-client-events.db	clientID, deviceID, duraton, requestID, timestamp, domain, user agent, app version
workout.db	workouts altitude, latitude, longitude, date, workoutID, start time, end time, distance, calories, note, steps, weight, heart rate, timestamp
com.amplitude.api	userID, OS, OS version, device, language, gender, age, registration date, weight, height, birthday, city
mmdk_user.db	userID, username, email, first name, last name, display name, birthday, gender, height, weight, timezone, date joined, last login, country, region, locality, link to profile image, language
com.google.android.gms.appid.xml	app version, timestamp created, token
com.google.maps.api.android.lib6.drd.PREFERENCES_FILE.xml	sessionID, legal country
cSPrefs.xml	last measurement timestamp, last sessiontimestamp, amount of runs, active user session count, first install ID
currentUserId.xml	current UserID
deviceLocationChecker.xml	last time location was checked
deviceLocationManager.xml	country, zip code, city
eventLogHarness_prefs.xml	last work out ID
fcm_.xml	FCM (firebased client messaging) clientID, registrationID, enabled or not
mfpApiPrefs.xml	API access token
session.xml	appID, clientKeyInfo, clientID, creation timestamps, clientTokenInfo, access token, server key info, user info map, domainUserID, email, display name, full name, userID, user creation date
notification-inbox.db	notifications including userID, engagementID, time created, deleted or not, state (unread, pending, etc.), expiration date, notification info
log_2019-03-18_18/20/47.txt	log of application events

Table 8: MapMyHike Artifacts

File Name	Information Found
io-uacf-client-events.db	clientID, deviceID, duraton, requestID, timestamp, domain, user agent, app version
workout.db	workouts altitude, latitude, longitude, date, workoutID, start time, end time, distance, calories, note, steps, weight, heart rate, timestamp
com.amplitude.api	userID, OS, OS version, device, language, gender, age, registration date, weight, height, birthday, city
mmdk_user.db	userID, username, email, first name, last name, display name, birthday, gender, height, weight, timezone, date joined, last login, country, region, locality, link to profile image, language
com.google.android.gms.appid.xml	app version, timestamp created, token
com.google.maps.api.android.lib6.drd.PREFERENCES_FILE.xml	sessionID, legal country
cSPrefs.xml	last measurement timestamp, last sessiontimestamp, amount of runs, active user session count, first install ID
currentUserId.xml	current UserID
deviceLocationChecker.xml	last time location was checked
deviceLocationManager.xml	country, zip code, city
eventLogHarness_prefs.xml	last work out ID
fcm_.xml	FCM (firebased client messaging) clientID, registrationID, enabled or not
mfpApiPrefs.xml	API access token
session.xml	appID, clientKeyInfo, clientID, creation timestamps, clientTokenInfo, access token, server key info, user info map, domainUserID, email, display name, full name, userID, user creation date
notification-inbox.db	notifications including userID, engagementID, time created, deleted or not, state (unread, pending, etc.), expiration date, notification info
log_2019-03-18_18/20/47.txt	log of application events

Table 9: RunKeeper Artifacts

File Name	Information Found
EventLog.sqlite	last activity location, system language, event name, page accessed, and user ID
RunKepper.sqlite	challenge id, description, name, rewards, social share component, trigger ID, start date, end date
admob.xml	last weight sync, userID, app language, birthday, whether activity tracker is connected, logged out or not, creation time, name, lifetime total distance, email, profile pic URL, twitter, facebook info (if applicable), user weight, height, country
com.google.android.gms.measurement.prefs.xml	time active, active start, pause, and upload time
com.google.maps.api.android.lib6.drd.PREFERENCES_FILE.xml	sessionID and legal Country
com.localytics.android.f77bd0ebf3c2fb361afd41bc7cc211855f2b50bcd9a5b18b17cc63b3dfa1888.a nalytics.sql	api key, userID, created time, registrationID, registration version, customerID

Table 10: Runtastic Artifacts

File Name	Information Found
DCIM/runtastic	folder of photos uploaded to app with timestamps stored on SD card
files/Pictures	same photos as above, stored in different location with timestamps
apptimize.db	age, gender, app install date, device specifications, app key
db	shoe information, connected devices, longitude, latitude, altitude, distance, speed, friends, photos uploaded (local location, URL, timestamp), heart rate with timestamp, activity session info (userID, serverSessionID, distance, start time, end time, date, avg speed, max speed, note, pulse, longitude, latitude, elevation, shoeID)
google_analytics_v4.db	userID, appID, app version
RKStorage.db	userID, first name, last name, access token, profile URL, avatar URL
com.google.android.gms.analytics.pref.xml	first run, monitoring start timestamp
com.google.android.gms.measurement.pref.xml	first open time, time open, start time, pause time, upload time
com.google.android.gms.signin.xml	email, display name, google sign in account, tokenID, ID
com.newrelic.android.agent.v1_com.runtastic.android.xml	deviceID, version code, data token, app version, app build, device model, app token
com.pushwoosh.registration.xml	appID, userID, registrationID
com.runtastic.android_preferences.xml	first name, last name, avatar URL, height, weight, birthdate, first app start timestamp, country, gender

Table 11: Fitness Tracker Artifacts

File Name	Information Found
000003.log	height, weight, waist measurement, hip measurement, neck measurement, gender, ethnicity, blood type, pregnant (yes or no), lactating (yes or no)
google_analytics_v4.db	customerID, app version
com.google.android.gms.analytics.prefs.xml	first run time, monitoring start timestamp

Table 12: JEFIT Artifacts

File Name	Information Found
/jefit/PPictures/Thumbnails/	folder stores thumbnails of uploaded photos
/jefit/PPictures/Fullsize/	folder stores same photos as above, but full size
/sdcard/jefit/PPictures/Fullsize/	same photos as above, but full size saved on sdcard
/sdcard/jefit/PPictures/Thumbnails/	same photos as above, but thumbnails saved on sdcard
com.google.android.gms.analytics.pref.xml	first run, monitoring start timestamp
com.google.android.gms.measurement.pref.xml	first open time, time open, start time, pause time, upload time
JEFITPreferences.xml	first install timestamp, MD5 hash of password, username, userID
data.db	exerciseLogs(timestamp, date, type, log), photos (photoID, time taken, edit time), profile(height, weight, log time, my date, edit time), setting (gender, date of birth)

Table 13: Strava Artifacts

File Name	Information Found
files/	uploaded photos in files directory
branch_referral_shared_pref.xml	user profile URL, last update time, app version, identityID
com.google.android.gms.analytics.pref.xml	first run, monitoring start timestamp
com.google.android.gms.measurement.pref.xml	first open time, time open, start time, pause time, upload time
com.strava.preference.userPreferences.xml	last activity type, last activity change timestamp
strava.db	first name, last name, email, gender, birthdate,
com.strava_preferences.xml	athleteID key, user access token
com.strava.preference.userPreferences.xml	account type, google fit linked (true or false), gender, last activity type
com.strava.xml	installed timestamp, registration key, last update, userID, projectID
apptimize.db	app key, device model, app version code, app install date, app first run date, system language
analytics-android-ibhm3Jtwz1rZ4EQemYsxTqglKoMlVSSC.xml	name, email, api key, userID

Table 14: MyFitnessPal Artifacts

File Name	Information Found
myfitnesspal.db	Exercise notes, food notes, images, measurements, user info (email, username, weight), user properties (email, timezone, gender, date of birth, country, zip code, weight, height, diary password), users (which users were logged into the app)
notification-inbox.db	notifications with timestamps and current state of notification
device_id.xml	encryptedID, deviceID
login-shared-preferences.xml	email, username
global_settings_preferences.xml	app version, last logged in user, current user logged in, if pin is required for app
session.xml	email, userID, display name, gender, birthday, profile picture URL, height, weight, IDtoken
geo-location.xml	last country code, last locale code

Table 15: Nike Training Artifacts

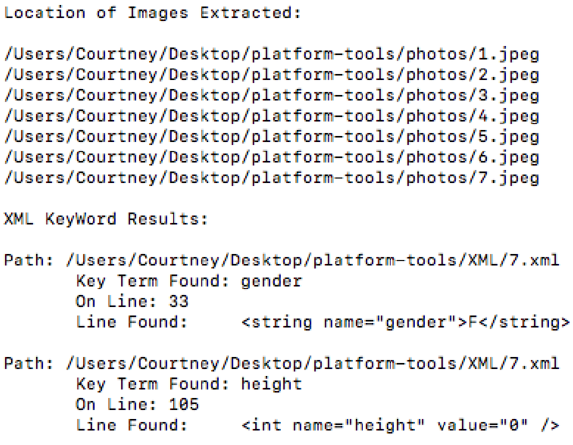
File Name	Information Found
com.nike.ntc.database.user	activity detail (type, start time, end time, duration, location), timezone
com.newrelic.android.agent.v1_com.nike.ntc.xml	deviceId, data token, accountID, app version, device model, version code
com.nike.ntc_preferences.xml	current account name
com.nike.ntc.config.ntc_prefs.xml	gender, age
com.nike.ntc.shared.ntc_prefs.xml	user hash
NRAnalyticAttributeStore.xml	userID
ns_common.db	first name, last name, user name, birthdate, gender, email, registration date, country, height, weight
ns_feed2.db	userID, username, first name, last name, last updated timestamp, post information
ns_inbox.db	inbox information(message, timestamp, notificationID, identity hash

Table 16: Health Infinity Artifacts

File Name	Information Found
`HealthAndFitness.db`	medicine pill tracker(last modified timestamp, start date timestamp, number of days on medicine, comments), profile details(display name), activity (activity details), user details (name, gender, birthday, height, weight)
`com.droidinfinity.healthplus_preferences.xml`	height, weight, age, birthday, email, gender
`google_analytics_v4.db`	customerID, appID, app version
`analytics-android-ibhm3Jtwz1rZ4EQemYsxTqglKo MlVSSC.xml`	name, email, api key, userID
`com.google.android.gms.measurement.prefs.xml`	time active, start time, pause time, last upload, previous os version

Table 17: MapMyFitnessArtifacts

File Name	Information Found
`io-uacf-client-events.db`	clientID, deviceID, duraton, requestID, timestamp, domain, user agent, app version
`workout.db`	workouts altitude, latitude, longitude, date, workoutID, start time, end time, distance, calories, note, steps, weight, heart rate, timestamp
`com.amplitude.api`	userID, OS, OS version, device, language, gender, age, registration date, weight, height, birthday, city
`mmdk_user.db`	userID, username, email, first name, last name, display name, birthday, gender, height, weight, timezone, date joined, last login, country, region, locality, link to profile image, language
`com.google.android.gms.appid.xml`	app version, timestamp created, token
`com.google.maps.api.android.lib6.drd.PREFERENCES_FILE.xml`	sessionID, legal country
`cSPrefs.xml`	last measurement timestamp, last sessiontimestamp, amount of runs, active user session count, first install ID
`currentUserId.xml`	current UserID
`deviceLocationChecker.xml`	last time location was checked
`deviceLocationManager.xml`	country, zip code, city
`eventLogHarness_prefs.xml`	last work out ID
`fcm_.xml`	FCM (firebased client messaging) clientID, registrationID, enabled or not
`mfpApiPrefs.xml`	API access token
`session.xml`	appID, clientKeyInfo, clientID, creation timestamps, clientTokenInfo, access token, server key info, user info map, domainUserID, email, display name, full name, userID, user creation date
`notification-inbox.db`	notifications including userID, engagementID, time created, deleted or not, state (unread, pending, etc.), expiration date, notification info
`log_2019-03-18_18/20/47.txt`	log of application events



```
Location of Images Extracted:

/Users/Courtney/Desktop/platform-tools/photos/1.jpeg
/Users/Courtney/Desktop/platform-tools/photos/2.jpeg
/Users/Courtney/Desktop/platform-tools/photos/3.jpeg
/Users/Courtney/Desktop/platform-tools/photos/4.jpeg
/Users/Courtney/Desktop/platform-tools/photos/5.jpeg
/Users/Courtney/Desktop/platform-tools/photos/6.jpeg
/Users/Courtney/Desktop/platform-tools/photos/7.jpeg

XML KeyWord Results:

Path: /Users/Courtney/Desktop/platform-tools/XML/7.xml
       Key Term Found: gender
       On Line: 33
       Line Found:     <string name="gender">F</string>

Path: /Users/Courtney/Desktop/platform-tools/XML/7.xml
       Key Term Found: height
       On Line: 105
       Line Found:     <int name="height" value="0" />
```

Figure 12: Sample output from python script