Android Forensic Data Analyzer (AFDA): An Opensource Tool to Automatize Event Correlation Analysis on Android Devices

Dimitrios Kasiaras
University of the Aegean
Samos, Greece

Thomas Zafeiropoulos
University of the Aegean
Samos, Greece

Nathan Clarke
Plymouth University
Plymouth, UK

Georgios Kambourakis
University of the Aegean
Samos, Greece

Abstract

Forensic analysis on mobile devices in general and smartphones in particular is on the rise. Naturally, this is because these devices are more than ever used by criminals of all kinds to perform a variety of offensive actions. The mushrooming of mobile services and the way people use their smartphones in their daily activities results in a plethora of valuable and private data stored in the device, which of course can be extremely helpful towards resolving a criminal case. The automatic or semi-automatic correlation of end-user events as recorded in the mobile device can be of great value to the investigator in their struggle to resolve a case. Unfortunately, existing forensic tools targeted to Android lack of such a functionality. To fill this gap, we propose the Android Forensic Data Analyzer (AFDA), a tool that is able to gather end-user's data stored in critical system areas and then inter-correlate them in terms of a time and location-based series of events. We argue that this type of analysis not only saves time and effort from an investigator's viewpoint but also can reveal the interrelationship between artifacts providing a more robust and comprehensive approach.

1. Introduction

During the last few years we have witnessed a rapid evolution in the technology of mobile devices, where the emergence of the so-called smartphones has revolutionized the way we work, travel and keep in touch with others. Indeed, a recent survey shows that 97% of adult Americans own at least one mobile phone, with more than half of them using smartphones [1]. From all the mobile Operating Systems (OS) available in the market today, the Android platform seems to possess a significant and increasing market share. Considering smartphone capabilities and the services that they can offer, it is inevitable that such devices could be exploited in criminal activities. Specifically, smartphones amass and broadcast personal information and are frequently used in e-transactions [2]. This fact makes them a very enticing target for various kinds of aggressors to attack and steal information for financial gain. On the other hand, criminals always relied on mobile devices to exercise their daily operations. Therefore, smartphones are sure to be a powerful tool for them because of the variety of services they offer. As a result, from a forensic investigator viewpoint, these devices store valuable information for solving a case. In fact, Mobile Forensics (MF) is on the rise receiving more attention from researchers and practitioners. A smartphone can be the instrument, the target or just be the source of valuable information related to a crime. So, few would disagree that unlike personal computers, smartphones can eventually reveal additional information and provide stronger evidence about a user's behavior. This is due to the sophisticated hardware these devices embed; for example, high-resolution cameras and GPS interfaces that can be utilized for taking photos or acquiring location-based services (LBS).

Considering all the above, the need for advanced forensic tools that focus on smartphones and can help examiners to solve a case in a timely and efficient manner is deemed imperative. Even though existing MF tools have been greatly enhanced, as discussed in section II, to our knowledge, none of them is able to produce a comprehensive analysis about the hidden evidence contained in the device without significant out-of-tool effort from the investigator. This means that the great majority of tools occupy themselves with the acquisition of raw evidence rather than their subsequent analysis. So, ultimately a great deal of the analysis of the collected evidence needs to be undertaken manually.

Our contribution: To contribute to the aforementioned need, in this paper, we propose a novel forensic tool for the Android platform that concentrates on users data and thus advocates a more passive role for the examiner. A key contribution of the proposed tool is that of the automatic or at least semi-automatic inter-correlation of user's data pertaining to the different events that have been logged by the device in both the file system and application layer. This can significantly assist an investigator by highlighting the relationship between the various artifacts, rather than relying upon the investigator to manually identity them. It may also reveal latent information related to the particular case at hand.

The rest of paper is structured as follows. The next section addresses previous work. Section 3 details on our

proposal, namely the Android Forensic Data Analyzer (AFDA). The latter is evaluated through some case studies in Section 4. The last section concludes and provides pointers to future work.

2. State of the Art

2.1. Prior Art within Android Forensics

Considering the latest developments in the area of MF, it is for sure that one has to cope with several challenges that require further research. In fact, the complexity of the problem is mainly due to (a) the fast-evolving pace of the mobile technology, (b) the mushrooming of mobile services offered to the end-user, and (c) the non-convergence between the various mobile platforms and the absence of mature specifications and practices in the field. So far, the majority of research in the MF ecosystem concentrates on techniques destined to the acquisition of evidence and the maintenance of the file system's integrity. There is only a limited number of studies that cope directly with methods for the analysis and presentation of the collected evidence to the investigator. The most promising of them are briefly analyzed in the following paragraphs.

Work by Lessard reports on the acquisition procedure from an HTC Hero device and the examination of the evidence using two commercial MF tools [3]. The authors draw their attention to the acquisition methods that can be used, and the evidence that can be examined by existing MF tools. However, there is no reference to the way that the collected evidence will be displayed to the investigator. Similar work elaborates on two main MF acquisition methods (physical - logical), by showing the advantages and the disadvantages of each one as well as the importance of preserving the integrity of the data [4]. In the same context, the work by Vidas introduces a data collection methodology, in which data integrity is sustained [5]. The proposed methodology capitalizes on the recovery mode partition of the Android platform for acquiring an image of the device's file system. Moreover, other authors build upon the previous methodology by presenting a simple GUI tool written in C++ that simplifies the procedure of the extraction of evidence [6].

More recently, the work by Mahajan examines two Instant Messaging (IM) apps, namely Viber, and What's Up, using the Universal Forensics Extraction Device (UFED) physical analyzer and subsequently describes the valuable information that can be retrieved from such an app [7] [8]. Furthermore, Andriotis et al. examine four scenarios based on real crime cases that have been committed by utilizing wireless connections [9]. They focus upon what evidence can be found in app databases and log files. However, as the authors correctly point out, log files are not always reliable as they depend on the time and the usage prior to the seizure and what's happening because of the internal mechanism that manages the log files.

A couple of further works given by Maus and Kramer concentrate on the analysis of geo-data [10] [11]. In the former study, the authors describe some of the difficulties that arise during the examination of user's geo-data, and propose an approach to perform a comprehensive analysis of such data contained in smartphones. In the latter, the authors present a tool that dynamically searches for geo-data generated in the mobile device by the corresponding apps. On the other hand, their tool merely presents static data as it does not attempt to offer any correlation of geo and non-geo tagged artefacts. Nevertheless, as explained further down in section III, a closer analysis of geo-data could lead to a richer set of clues, such as the comparison of user's routes spread across different dates, which could thus unveil unusual behaviors regarding the user of the device.

There are also a number of studies that focus on the admissibility, organization and presentation of evidence in a chronological order. However, these studies are not specific to MF, but rather applicable to computer forensics in general. Buchholz el al. present a timeline editor (Zeitline) that is designed as an extensible tool capable of using various app data sources to automatically collect and process system events [12]. Although this tool provides some useful functionality, it does not offer a graphical timeline. Other work presents a tool that collects the timestamps from a hard disk of a personal computer and visualizes them in a graphical timeline [13]. Whereas this tool does includes a graphical timeline view, it is destined for desktop machines, thus it cannot be used for smartphone devices where the file system has an entirely different structure. Perhaps, the most relevant research to ours is that given by Jin [14]. The author implements a tool for the Android platform that produces a graphical timeline representing the activity of each application of interest. On the other hand, this software only takes as input the time that each application was used and not the information related to events, e.g., the message communicated in an SMS transaction or the caller and callee IDs in a phone call. So, this timeline can be very useful for revealing unusual behavior of apps that may indicate the presence of malware.

2.2. Analysis of Forensic Tools

An examination of the currently available open source MF tools developed for Android reveals that the majority of them have limited functionality and sometimes are quite complicated even for the experienced examiner to use. Perhaps, the most well-known tools in this category are the so-called Autopsy – Sleuth Kit and OSAF (Open source Android Forensics) [15] [16]. On the other hand, there are tools such as the ViaExtract or Oxygen Forensics that do not analyze an image file, but install an agent in the device in order to retrieve valuable data [17] [18]. This however results in altering the integrity of the file system, which in turn creates admissibility problems.

Another important aspect here is that some MF tools target specific apps such as the browsing history (database)

and not only legacy artefacts (e.g., call, SMS) found in mobile phones. ADEL is an open source tool that uses ADB shell to retrieve such sorts of data (contacts, calls/SMS/MMS, browser history, media, IM accounts, e-mail, contacts, accounts, and so on) [19]. It can also create a map with user's geographical locations as logged by the device. A notable limitation of the tool is related to the incompatibility issues caused by newer Android versions, and the one that require the device to be rooted in order to get access to the file system. An interesting commercial tool, IEF (Internet Evidence Finder) also targets specific applications, including both pre-installed and installed applications by the user (Facebook, Instagram, etc.). This is very important because broad swaths of users employ these apps and therefore the corresponding databases may contain valuable data for solving a case. The tool is also able to create a timeline of events that have occurred in the device, including received calls and e-mails [20].

A succinct analysis of the aforementioned tools along with their key characteristics is included in Table 1. It should be noted that the tools contained in the table are open source or commercial in trial version mode. For testing the tools, two rooted Android devices were utilised. The first was a Samsung Galaxy Nexus with EXT4 file system running Android version 4.3, and the other was a Sony Ericsson XPERIA Neo V MT11i with YAFFS2 file system running Android 4.0.4. As observed from the table, although several commercial tools have started to develop aggregation mechanisms designed for better analysis of the evidence (frequently based on popular applications, as that of social networks) there are no dedicated and comprehensive mechanisms for the correlation and visualization of evidence stemming from different sources.

Table 1. Forensic Tools Main Characteristics

	Main Feature	Plus	Minus	OS	Android Focus	**Open**
Autopsy	Search engine	YAFFS 2 support	Struggling to find evidence	Win, Mac, Linux	No	Yes
ADEL	Exports db	Map with Locations	Works till v.2.X	Just Pyhon	Yes	Yes
OSAF	Forensic, malware	For best results requires familiarity		VM	Yes	Yes
Via Extract	Exports artefacts	Quick	Alters file system	VM	Yes	No
Oxygen	Evidence long list	Detailed report, Timeline	Alters file system	Win, Mac	Yes	No
IEF	Graphical TimeLine	Filtering evidences	Moderate filtering	Win, Mac	Yes	No

3. The AFDA System

As previously highlighted, most studies on Android forensics concentrate on finding better ways to acquire the data from the device rather than on how to present the data to the investigator in a simple and intelligible fashion. Additionally, most research that refers to "a timeline of events" has been conducted for digital forensics in general, and thus they are not specific to the Android platform. To the best of our knowledge, no research focuses upon the inter-correlation of events pertaining to a users' activity. This means that the investigator needs to undertake this work by manually examining the corresponding log files.

Therefore, taking into account the existing forensic tools and related literature there is a need for advanced MF tools that will target specific applications and be able to display the artifacts independent of the app in a comprehendible and usable manner. However, this should be achieved without removing the investigator's ability to perform manual searches and navigate through the file system of a given image file. This is because fully automatic procedures may miss a considerable mass of evidence.

To respond to this need, AFDA was designed and implemented - an open source MF tool designed for the Android platform. The tool specifically targets the user's data partition of the Android platform. This partition contains all the artifacts generated by the installed apps. The interface of the tool has been designed to be user friendly and as intuitive as possible, developing a similar design to that of well-known commercial tools. Furthermore, AFDA provides the investigator with an automated mechanism for basic artifact examination and also a separate one for performing correlation analysis among the artifacts. Finally, AFDA offers the ability for a manual examination of the file system structure in order for the investigator to be able to locate specific information depending on the case. The tool has been implemented in JAVA for the Linux platform and does not require the mobile device to be rooted. Linux was selected because AFDA uses UNIX shell commands for conducting the extraction of the artifacts from the image file, and mainly uses the well-known shell command "mount" for mounting the image on the host machine. For the latter we used an Ubuntu OS v. 13.10.

Furthermore, AFDA requires the alteration of the "sudoer" file of the Linux OS. Usually, this file is located in /etc/ folder and contains the rules that users must follow when using the sudo command. Specifically, this file must be changed so that the system does not require from the user their password when they run the sudo command. In fact, this modification is necessary due to the fact that the sudo command is called inside the Java code and currently there is

no effective way to pass the password as a variable to the command.

To sum up, the tool requires a binary image file to be extracted from the Android device under investigation. This refers to a bit-by-bit copy of a certain partition from the device memory. Actually, Android uses several partitions to organize files and folders on the device, including /boot, /system, /recovery, /userdata, /cache, /misc, /sdcard, and others. Currently, AFDA is able to only examine Android devices that contain EXT file system. This is because Linux cannot mount YAFFS and YAFFS2 file systems by default. In order for the investigator to examine such a file system they would have to modify the Linux kernel to support it. For the interested reader, a comprehensive guide to help understand YAFFS2 and recompile the Linux kernel aiming to mount YAFFS file system is described in Regan [2009]. Also, it is to be noted that the partition AFDA uses is /userdata.

3.1. AFDA GUI

Few would argue that one of the main requirements for any software tool pertains to its usability. That is, the user must not struggle with a complicated interface to launch specific functionalities. Furthermore, the functionalities of any MF tool should be as automated as possible, thus assuring optimized performance and straightforward use.

As illustrated in Fig. 1, the AFDA GUI is divided in five areas of interest that help toward the best analysis of a case. The first one is the menu bar that contains all the functionalities offered by the tool. Next is the "structure view" that displays the file system structure of the partition containing the user's data. The investigator is able to navigate around this area by selecting among the included data. There is also a "content display" area that presents the contents of the selected file/folder. The "processing details" area is used to present information about the running processes and other pieces of data regarding the currently selected items in the structure area. Finally, the "common data" area includes all the artifacts that are extracted during the acquisition procedure.

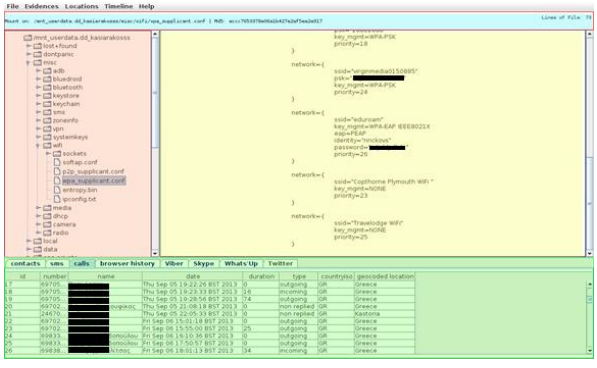

Figure 1. GUI of AFDA

3.2. AFDA core Functionalities

The core functionality of AFDA comprises of five distinct phases. That is, case creation, mounting of an image, extraction of artifacts, generation of the hash file, and generation of reports. Whilst there is no novelty here, these common functionalities to any MF tool are necessary for conducting a constructive and acceptable forensic investigation, from which the advanced correlation functionality is built upon.

The creation of a new case includes the construction of a log file that records the processing details and the general information about the case and the mounting of the image is needed for the analysis of the user data partition. Once complete, the structure of the file system is displayed in the "structure area" and the analyst is able to locate the files of interest depending on the case at hand. After the mounting procedure takes place, the investigator can process, extract and analyze app-level data. The extraction functionality aims at specific apps that may offer special forensic value. During this procedure, the app-related artifacts are extracted from the image file and stored in a local database, namely "commonData.db". Also, as depicted in Fig. 2, the artifacts are displayed in the "common data" area. These pieces of data will be also used at a later stage for performing the advanced analysis.

As it is well-known, each Android application stores its data in corresponding databases. However, it is not to be taken for granted (especially for new apps) that the investigator knows details about how this is done (i.e., the name of a table the data is stored in, the names of the columns of the table, what type of data are contained, etc). Hence, usually a reconnaissance stage to examine the way the data is stored in the device is required. Currently, this examination should be performed manually Based on publicly available data and facts about the apps most people use or have download from the Android market, the following apps were chosen to be included in AUDA tool: Contacts, SMS, Telephone calls, Browsing history, Viber, Skype, What's Up, and Twitter. Notably, however, the tool has been developed to make this extendable to meet the needs of future apps.

Generally, the data acquisition process triggered by AFDA unfolds in the following way. First, AFDA connects to the appropriate database in the mounted image file. After a connection has established, the program executes some SQL SELECT statements to retrieve the required information. Next, it connects to the local database and executes some INSERT SQL statements for storing the retrieved information to the appropriate tables. Finally, the tool fetches the artifacts that were stored in the local database and displays them in the "common data" area using a different tab per app.

The extraction procedure uses different functions for the extraction of artifacts per application, so in case of an error, the extraction procedure will continue unattended for the rest of them. This approach also makes it simple to include

additional functions for the extraction of artifacts for any new application. Legacy hashing is used for assuring the integrity of the file. As illustrated in Fig. 3, during this procedure a separate file is generated that contains all the original filenames along with the file path from the mounted image and the corresponding MD5 hash value. The final basic functionality of AFDA is the generation of the corresponding report. This is a comma-delimited file (csv) that contains the details of the case, the procedures that were conducted (e.g., the title of the case, a unique id based upon a timestamp, the path of the mounted image, etc) and the artifacts that were extracted from the examined apps. An example of such a examination is given in Fig. 4. Such a report can be used for the presentation of forensic analyses within court.

As already mentioned, the investigator is able to use the "content display" area for viewing the contents of files, and analyze them manually depending on the case. However, before processing the files for viewing, there exist two restrictions. The first is related to the size of the file and the other with its type. As for the size, some files are too large for the tool to process thus an external software should be used. Fortunately, it seems that the files that need such a special treatment are usually not important for forensic purposes. On the other hand, the restriction that has to do with the file format stems from the fact that while normal files, including text documents and images can be displayed, a filter needs to be used for preventing others to be processed for viewing purposes. This category refers mainly to .apk and other binary data files. This typically does not present an issue as they hold little value from a forensic point of view. Of course, there are also some other types of files that have been intentionally rejected for viewing, including those of SQLite databases. Such files cannot be directly viewed inside AFDA, requiring an external program to parse them (e.g., an SQLite Editor).

Hashes.md5 ×

```
cac2fccfc193c155a3b98d483025ebdf  /mnt_userdata.dd_makos/misc/adb/adb_keys
e8096e7d73219de97ef4326b59e4305c  /mnt_userdata.dd_makos/misc/bluedroid/bt_config.xml
e8096e7d73219de97ef4326b59e4305c  /mnt_userdata.dd_makos/misc/bluedroid/bt_config.old
4352d88a78aa39750bf70cd6f27bcaa5  /mnt_userdata.dd_makos/misc/keystore/.metadata
e4b37605f094fc5265c2c35596d6aa69  /mnt_userdata.dd_makos/misc/keychain/serial_blacklist.txt
ee82cb284c51b674256e37513f61e4cd  /mnt_userdata.dd_makos/misc/keychain/pubkey_blacklist.txt
eccbc87e4b5ce2fe28308fd9f2a7baf3  /mnt_userdata.dd_makos/misc/sms/metadata/version
d3d69c061fa455b22a6d4a7237ede77e  /mnt_userdata.dd_makos/misc/sms/codes
9b53ad3fb5c3c06ff8c95bca8c1cd62e  /mnt_userdata.dd_makos/misc/wifi/softap.conf
548628cde1e9fdb960391cbf0a7a4f2c  /mnt_userdata.dd_makos/misc/wifi/p2p_supplicant.conf
eccc7653378e06a1b427e2af5ee2e917  /mnt_userdata.dd_makos/misc/wifi/wpa_supplicant.conf
```

Figure 2. Hash File

	A	B	C	D	E	
774	480	1381790035697	[redacted]	[redacted]	306979777189	A ok [redacted]
775	481	1381790308056	[redacted]1	[redacted]4	306979777189	[redacted]
776						
777	Twitter IDS were Used from Device					
778	id	Twitter ID				
779	1	[redacted]45				
780	2	2[redacted]8				
781						
782	Twitter Data					
783	id	User ID	latitude	longitude	created	content
784	1	[redacted]	[redacted]	[redacted]	1386066658000	E[redacted]
785	2	[redacted]	[redacted]	[redacted]	1386071151000	To[redacted]
786	3	[redacted]	null	null	1386061436000	[redacted]o
787	4	1[redacted]	null	null	1386063292000	[redacted]

Figure 3. csv File Sample Report

3.3. Advanced Functionality of AFDA

The advanced inter-correlation functionality that AFDA offers comprises of two modules, namely the Graphical Timeline representation, and the Geodata Chronology.

3.3.1. Graphical Timeline. The graphical timeline of events is designed to provide the investigator with a succinct view of the occurred events in chronological order. To do so, the module uses the timestamps stored in the database of the corresponding app. This functionality allows for better understanding of when (and how) the owner of the device used the application. Most apps in the Android platform realize the current time (timestamp) in milliseconds. Also, a timestamps is based on the local time zone. For example, if a timestamp is to be created in Greece at 7:00 am, it will be different from that generated in UK at 7:00 am on the same date. So, after a timestamp is retrieved it is interpreted into the time zone of the local machine. So if an event occurred in Greece but the case is investigated in UK the investigator will have the events in their local time zone and consequently they have to assign them to the Greek time zone manually.

This timeline of events can be created for two different ranges of time. First, it can be outputted for a single day where the investigator can obtain a concise view of the events and be informed about the duration of each one of them. The second option, described in Fig. 5, is for one month where a general view of the events can be concluded and specific behavior patterns can be spotted related to the application(s) of interest. As observed from Fig. 6, the graphical timeline represents each application using a different axis and color and each event is displayed with a line where its thicknesses is proportional to the duration of the event. For the graphical representation of the events related to the time it occurred, the Google visualization JavaScript API was used. This API provides a chart gallery, which provides a plethora of charts designed to accommodate a variety of data visualization needs. These charts are based on pure HTML5/SVG technology (adopting VML for old IE versions), so no plugins are required. All of them are interactive, and most of them are pan-able and zoom-able as well.

During the construction of a timeline two other informational objects are created. The first one is a table that contains the contacts of other persons the user interacted with more frequently, while the second is another table summarizing the events displayed in the graphical timeline, but with more information. An example of this situation is presented in Figs. 7 and 8. The investigator can use these two tables in conjunction for achieving the best results.

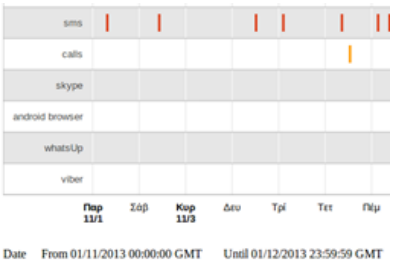

Figure 4. Sample Timeline per App

The analyst is also able to select which apps will be displayed and enter search terms for filtering the messages corresponding to events. Moreover, they are able to search for information by specifying the contacts of persons that may have used the app. All these functionalities can be used in tandem to reveal correlations among the events generated by different apps as well as possible interrelationships between the persons who use them. The first time the investigator selects to create the graphical timeline a new database is created (normalized.db). This database contains all the artifacts that were extracted during the acquisition process. This is important because there is a need for the events to be in a specific format so as to be used in the graphical timeline and the geo-data chronology views. For instance, pieces of data that are stored in a table of a given database that the app of interest is using, e.g., location (latitude, longitude), are not always in the expected format. So, these data need to be normalized in the default format prior to be used by AFDA.

3.3.2. Geodata Chronology. As previously highlighted, most Android apps store information about the occurring events in a corresponding database. As known, such events frequently include information about the geographical location of the device at that point of time. AFDA capitalizes upon this fact and provides an option that enables the investigator to automatically seek for such information, create the (geo) route of the user's device, and present it in chronological order. As shown in Fig. 9, for each position on the map, information about the corresponding event is available for the investigator to evaluate. This functionality is of great help to an investigator for not only perceiving the positions of a suspect along with the associative events, but also for revealing any unusual user behavior. This results because some points on the map may expose considerable differences from others included in the user's normal routine (i.e., via the comparison of routes across different dates).

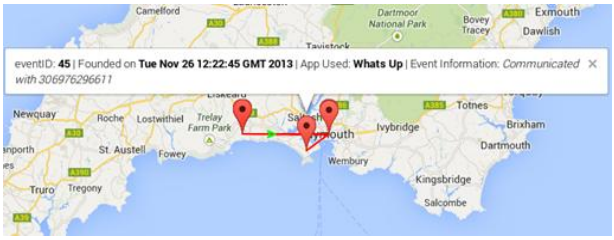

Figure 5. A snapshot of user's route with the associated events

Although many applications may access the user's location, this piece of data is not always stored in the corresponding app's database. Typical apps that do so are Twitter, Viber, and What's Up. Moreover, while a lot of events may have occurred, only a small fraction of them are directly associated with geo-location data (event and geolocation in the same record). So, AFDA performs a correlation of events having no available location, with others that do in an effort to reveal possible locations or proximities about where these events have taken place. For example, let's assume a Skype call that occurred at a given time. This event is stored in the corresponding database, but by default there is no information about its location. However, there is another event approximately at the same time, say, a Viber call, which does hold location information. Hence, it is easily deduced that the Skype call took place at about the same geographical position to that of the Viber call. More specifically, as observed from Fig. 10, the algorithm searches all app databases for events (event A) that include both geo-data and the associated timestamp. Then, given a threshold (time t), it searches for events (event B) stored in app databases that only include the timestamp of the event. In a final stage, the software attempts to correlate between events A and B that are found to have nearby timestamps ((time x of event A, location) ∩ (time x ± threshold t of Event B, no location available)) in an effort to reveal the location of event B. Naturally, this procedure heavily depends on the threshold the forensic analyst provides but as the tool enables the investigator to vary the time value, this is left to the investigator to determine.

In order to create a map containing the location where an event has occurred, Google Maps JavaScript API v3 has been used. The results are exported to a HTML file which holds the corresponding locations as points on the map. This well-known API was utilized because it offers a lot of functionality and a plethora of options that can be really helpful to both the implementer as well as the forensic analyst. The limitation however is that the tool requires Internet access.

The central database ("commonData.db") contains a variety of data and events. To collect only those corresponding to locations, AFDA searches the aforementioned database and transfers location data to another called "location.db". After that, the records of the latter database are added in chronological order to the map, all connected by a red timeline. For showing the chronological sequence of the events included in the map, a

green arrow points from one location to another. As some of the events may happened on the exact same point on the map, the markers that point toward the location of an event can be moved. However, the green arrow that shows the direction is not affected. For every point that is included in the map, the forensic analyst is able to click on the marker and retrieve information about the event. Also, because the forensic analyst may not be interested in viewing all the locations that are stored in the "locations.db", and instead target only to a specific period of time, AFDA is able to adjust the map accordingly.

4. Evaluation

A preliminary evaluation of AFDA was conducted with the aim to assess its usability and effectiveness as compared to other well-known forensic tools.

4.1. EnCase

EnCase is one of the most popular forensic tools [21]. However, it is used for general digital forensics and not specifically for smartphones. Although this tool is able to examine an Android image, the investigator has to struggle to locate the traces that they need. This happens because EnCase does not focus specifically on the installed apps and the evidence that they may bear, but merely on the file system. Moreover, EnCase can be used directly on the device. That is, by installing an agent it makes possible to retrieve specific information about the mobile device. On the downside, this information is limited and does not include app artifacts. During the examination of an image some EnCase filters helped us to concentrate on specific time intervals. However, the presented information was enormous and it was very hard to find specific events in a timely manner. Finally, EnCase does not provide any module to visualize the locations visited by the user within a window of time. While EnCase offers the ability to navigate the file structure, it does not include any automated procedure for the representation of artifacts that may be forensically significant.

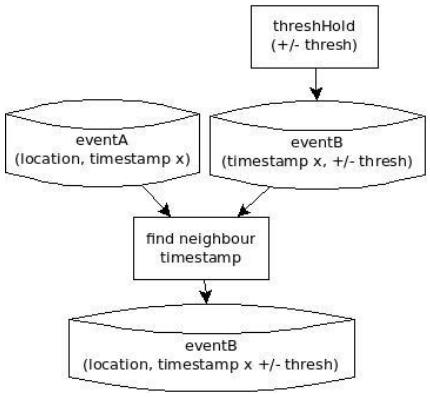

Figure 6. The Threshold Procedure

4.2. Autopsy

Autopsy is one of the most popular open source tools for digital forensics, including many interesting functionalities, such as file structure navigation and filtering, in its arsenal. During the analysis of an image, the most valuable characteristic of Autopsy was its search engine, as it can traverse the unallocated space to find deleted files. Although this tool offers great functionalities and a plethora of options it does not enable one to search for specific apps and artifacts. Furthermore, it does not include any module for the graphical representation of the evidence. The newer Autopsy ver. 3.0.9 for Windows does incorporate a beta procedure to retrieve the user's timeline, but this is limited to the events per se, without including any details about them. One can conclude that while this tool offers great potential to conduct manual analysis, it still needs significant improvement regarding the representation of artifacts to the investigator.

4.3. Internet Evidence Finder

Internet Evidence Finder (IEF) is perhaps the most similar tool to AFDA. This tool targets specific applications and artifacts that can be extracted from the installed apps. Further, it offers a dynamic app finder that searches for additional apps (databases) besides the ones defined by default. However, after trying it repeatedly with different images it was not able to locate and retrieve any valuable artifacts from the device (although there were some). Also, this app provides a well-designed timeline view along with filtering functionalities and a geo-data one that includes the geographical coordinates per event. Despite these modules can contribute towards a comprehensive analysis of a given image, it seems that the investigator is only able to analyze the artifacts that were extracted from specific applications and they do not have the ability to navigate the file structure of the image. Summarizing, we can say that IEF provides effective mechanisms for the examination of specific artifacts; however, some improvements are needed to ease the work done by the forensic analyst. Most importantly, it should include the ability of navigation through the image and to perform correlation among the various artifacts.

4.4. AFDA

As already pointed out, AFDA has been designed with usability in mind. So, it includes automated mechanisms for the representation of app data to the investigator in a clear and straightforward fashion. It also provides several options for the analysis of the file system of a given image. In our opinion, the graphical timeline and the geo-data chronology views can greatly assist in the analysis of artifacts produced by the various apps. Bearing in mind the discussion of section III, it is clear that the investigator is not only able to obtain a luminous view of the possible relationships between the various artifacts, but also can reveal obscured correlations among the events. This is achieved without

struggling with a vast amount of data. Similarly, the geo-data chronology view allows the investigator to easily perceive the route followed by the user of the device. To sum up, the examination of the image with AFDA is a much simpler task due to the automated mechanisms it offers. However, this is not limited to app data as the structure of the file system is provided for examination as well.

5. Conclusions & Future Work

Developments on mobile technology and forensic research and practice are more or less meant to go hand in hand [22]. This is because the advancements in mobile hardware and services are sure to facilitate and therefore increase criminal activities. So far, research on MF concentrates mainly on the acquisition methods and the selection of data that may be of forensic value. Little have been completed focusing upon optimizing the way the various artifacts are presented to an investigator. Motivated by this fact, in this paper, we present AFDA, an MF tool that incorporates some advanced features targeting to automatize and streamline the presentation of artifacts that present forensic value to the investigator. Also, AFDA aims to improve the exposure of hidden information of forensic value by taking advantage of the correlations among the various events recorded by the same or different apps. An evaluation of the tool showed that AFDA includes many useful mechanisms that can aid the investigator to analyze a given image more effectively and more efficiently.

However, it is acknowledged that the tool needs further improvements and additional functionality in order to be considered an integrated forensic tool. Future research is to make the tool compatible with several image formats and mobile platforms [14]. Furthermore, AFDA has to be evaluated from an investigator's point of view. Given the huge explosion in the number of mobile apps, a dynamic procedure for finding app databases and locating relevant forensic data is required. Finally, an option that will not only return the events that are correlated with a keyword, but also those that are in the vicinity of that time or geo-location would be much appreciated. This would allow for a better comprehension of the analyzed data.

Availability

AFDA source code is available on GitHub: https://github.com/msildigitalrage/project.

6. References

[1] Cell Phone Statistics,http://www.accuconference.com/blog/Cell-Phone-Statistics.aspx, 2014 (accessed 2014).

[2] Goel, A., Tyagi, A., &Agarwal, A. (2012).Smartphone Forensic Investigation Process Model. International Journal of Computer Science & Security (IJCSS),6(5), 322-341.

[3] Lessard, J., & Kessler, G. (2010). Android Forensics: Simplifying Cell Phone Examinations.

[4] Abalenkovs, D., Bondarenko, P., Pathapati, V. K., Nordbø, A., Piatkivskyi, D., Rekdal, J. E., & Ruthven, P. B. (2012). Mobile Forensics: Comparison of extraction and analyzing methods of iOS and Android.

[5] Vidas, T., Zhang, C., &Christin, N. (2011).Toward a general collection methodology for Android devices. Digital investigation, 8, S14-S24.

[6] Son, N., Lee, Y., Kim, D., James, J. I., Lee, S., & Lee, K. (2013).A study of user data integrity during acquisition of Android devices. Digital Investigation,10, S3-S11.

[7] Mahajan, A., Dahiya, M. S., &Sanghvi, H. P. (2013).Forensic Analysis of Instant Messenger Applications on Android Devices. arXiv preprint arXiv:1304.4915.

[8] UFED, "UFED Physical Analyzer", http://www.cellebrite.com/, 2013

[9] Andriotis, P., Oikonomou, G. C., &Tryfonas, T. (2012, December).Forensic analysis of wireless networking evidence of Android smartphones.In WIFS (pp. 109-114).

[10] Maus, S., Höfken, H., &Schuba, M. (2011, June). Forensic Analysis of Geodata in Android Smartphones.In International Conference on Cybercrime, Security and Digital Forensics, http://www.schuba. fh-aachen. de/papers/11-cyberforensics. pdf.

[11] Kramer, J. A. (2013). DroidSpotter: A Forensic Tool for Android Location Data Collection and Analysis.

[12] Buchholz, F. P., & Falk, C. (2005, August). Design and Implementation of Zeitline: a Forensic Timeline Editor. In DFRWS.

[13] Olsson, J., &Boldt, M. (2009).Computer forensic timeline visualization tool.digital investigation, 6, S78-S87.

[14] Jin, Y. (2013). Timeline analysis for Android-based systems (Doctoral dissertation, MS thesis, Technical University of Denmark, DTU Compute, E-mail: compute@ compute. dtu. dk, Matematiktorvet, Building 303-B, DK-2800 Kgs. Lyngby, Denmark. DTU supervisor: Robin Sharp, robs@ dtu. dk, DTU Compute).

[15] Sleuthkit – Autopsy, "Autopsy Digital Forensic Platform", http://www.sleuthkit.org/autopsy/, 2013

[16] OSAF, "OSAF community", http://osaf-community.org/, 2013

[17] ViaExtract, "Android Forensics Software", Available at: https://viaforensics.com/products/viaextract/, 2013

[18] Oxygen Forensics (2013): "Oxygen Forensics Suite", Available at: http://www.oxygen-forensic.com/en/

[19] Spreitzenbarth, M. (2013). Dissecting the Droid: Forensic Analysis of Android and Its Malicious Applications (Doctoral dissertation, Erlangen, UniversitätErlangen-Nürnberg, Diss., 2013).

[20] Magnet Forensics, "Internet Evidence Finder", http://www.magnetforensics.com/software/internet-evidence-

finder/ief-advanced/, 2013Nichols, D., &Twidale, M. (2003).The usability of open source software. First Monday, 8(1).

[21] EnCase, "EnCase Forensic Tool", http://guidancesoftware.com/, 2014.

[22] Barmpatsalou, K., Damopoulos, D., Kambourakis, G., & Katos, V. (2013). A critical review of 7 years of Mobile Device Forensics. Digital Investigation, 10(4), 323-349.

