

Electronic Theses and Dissertations, 2004-2019

2015

Providing Context to the Clues: Recovery and Reliability of Location Data from Android Devices

Connie Bell
University of Central Florida

 Part of the [Computer Sciences Commons](#), [Engineering Commons](#), and the [Forensic Science and Technology Commons](#)

Find similar works at: <https://stars.library.ucf.edu/etd>

University of Central Florida Libraries <http://library.ucf.edu>

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations, 2004-2019 by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation

Bell, Connie, "Providing Context to the Clues: Recovery and Reliability of Location Data from Android Devices" (2015). *Electronic Theses and Dissertations, 2004-2019*. 1354.

<https://stars.library.ucf.edu/etd/1354>

PROVIDING CONTEXT TO THE CLUES:
RECOVERY AND RELIABILITY OF LOCATION DATA
FROM ANDROID DEVICES

by

CONNIE BELL
B.S. University of Florida, 2004

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2015

Major Professor: Sheau-Dong Lang

© 2015 Connie Bell

ABSTRACT

Mobile device data continues to increase in significance in both civil and criminal investigations. Location data is often of particular interest. To date, research has established that the devices are location aware, incorporate a variety of resources to obtain location information, and cache the information in various ways. However, a review of the existing research suggests varying degrees of reliability of any such recovered location data. In an effort to clarify the issue, this project offers case studies of multiple Android mobile devices utilized in controlled conditions with known settings and applications in documented locations. The study uses data recovered from test devices to corroborate previously identified accuracy trends noted in research involving live-tracked devices, and it further offers detailed analysis strategies for the recovery of location data from devices themselves. A methodology for reviewing device data for possible artifacts that may allow an examiner to evaluate location data reliability is also presented. This paper also addresses emerging trends in device security and cloud storage, which may have significant implications for future mobile device location data recovery and analysis. Discussion of recovered cloud data introduces a distinct and potentially significant resource for investigators, and the paper addresses the cloud resources' advantages and limitations.

This work is dedicated to my beloved Granddaddy. Every one of your “Seven Pillars” has supported me in this effort. Thank you for a legacy of love, an example of incredible integrity, and a lifetime of cherished memories. Rest with the peace you have so truly earned.

ACKNOWLEDGMENTS

Special thanks to Theresa Adams and Donna Wallace for their support and assistance with this project, and with so much more.

Additional thanks to Mike Baute, Korey Diener, Ron Serber, and Jonathan Gajda for the opportunity to utilize the latest and greatest in data recovery technology.

Finally, I cannot overstate the gratitude and love I have for my family. Thank you all for the constant influx of joy and encouragement you are. You really are my inspiration, and you are so, so treasured!

TABLE OF CONTENTS

LIST OF FIGURES	ix
LIST OF TABLES	xii
LIST OF ACRONYMS (or) ABBREVIATIONS	xiv
CHAPTER ONE: INTRODUCTION.....	1
CHAPTER TWO: LITERATURE REVIEW.....	3
Proliferation and Utilization of Mobile Devices.....	3
Device Capabilities and Location Awareness	6
Existing Research on Location Data Recovery and Reliability.....	14
Accuracy of Device Location Services	14
Recovery of Location Data from Devices	23
Use of Mobile Device Location Data in Criminal Cases	26
Emerging Issues	29
CHAPTER THREE: METHODOLOGY.....	33
CHAPTER FOUR: FINDINGS.....	37
Extraction Issues	37
Cloud Data Recovery	39
Recovery of Device Location Data.....	42

App and File Review	42
Identifying Location-Permitted Apps	44
SQLite Database Analysis – Fitness App Example	49
Chat and Location-Sharing Apps	52
Leisure and Navigation Apps.....	56
Browser, Weather Apps, and Games.....	60
App Trends.....	64
Metadata and Logs	66
Accuracy Evaluation	72
Geo-Tagged Photos	72
Cloud Data Limitations	75
General Trends	82
Location Data Recovery and Evaluation Strategy	90
Limitations and Future Research	94
CHAPTER FIVE: CONCLUSION.....	97
APPENDIX A: TEST SESSION WORKSHEET	101
APPENDIX B: APPS AND SETTINGS INFORMATION	103
APPENDIX C: TEST SESSIONS	105
APPENDIX D: ANALYSIS TOOLS	108

APPENDIX E: POSSIBLE KEYWORD SEARCH TERMS.....	110
APPENDIX F: SQLITE QUERIES USED.....	112
REFERENCES	116

LIST OF FIGURES

Figure 1 - Cell Tower Localization Errors by Environment	19
Figure 2 - Relative Error Range by Resource Type.....	22
Figure 3 - Cloud Analyzer Location History Options (30-day range).....	39
Figure 4 - RunKeeper Permissions from “Packages.xml” with “ACCESS_FINE_LOCATION”	45
Figure 5 - RunKeeper.sqlite database, <i>trips</i> table.....	49
Figure 6 - RunKeeper.sqlite database, <i>points</i> table.....	50
Figure 7 - RunKeeper.sqlite database, query converting timestamps and combining content from <i>trips</i> and <i>points</i> tables.....	50
Figure 8 - RunKeeper.sqlite database, query results.....	51
Figure 9 - Contents of the Locate My Friends app's "messaging.db" database as viewed within Physical Analyzer (coordinates rounded to nearest degree).....	54
Figure 10 - Contents of the Locate My Friends app's "messaging.db" database as viewed within SQLite Studio (more precise coordinates)	54
Figure 11 - Contents of the Locate My Friends app's "360LocationDB" database, SQLite query results. High precision coordinates noted, but less precise than those associated with messages found in "messaging.db."	55
Figure 12 - Content of the Field Trip app's "lastLocation.xml" file	57
Figure 13 - Content of the Field Trip app's "lastNotification.xml" file	58
Figure 14 - Navigation request from Waze app's "waze_log.txt" text file.....	60
Figure 15 - Content of web browser's "https_www.google.com_0.localstorage" database, showing search terms, location coordinates, and timestamps	61

Figure 16 - Content of AccuWeather app's "accuwx_locations" file.....	63
Figure 17 - Content of AccuWeather app's "accuwx_geocoder_cache" file.....	63
Figure 18 - Content of Words With Friends app's "iad.dat" file, showing latitude and longitude values and connection type.....	64
Figure 19 - Sample entry from "dumpLogsDatabase" file.....	67
Figure 20 - Contents of the QuizUp game's "mixpanel" database showing WLAN network state with timestamp.....	68
Figure 21 - Content of the "ContextLog_0.db" database, filtered to show Waze app activity	71
Figure 22 - Geo-tagged photo example latitude and longitude accuracy check.....	73
Figure 23- Content of the "googlesettings.db" database	75
Figure 24 - Content of the "accounts.db" database.....	76
Figure 25 - Cloud Analyzer (left) and RunKeeper (right) location data for same timeframe	78
Figure 26 - Google user account interface content management.....	79
Figure 27 - Option to save timeframe as KML file via Google user account web interface	80
Figure 28 - Comparison of cached coordinates for same timeframe from RunKeeper (purple), Google user location history (yellow), and Cloud Analyzer (cyan)	81
Figure 29 - Actual versus Cloud location with cellular service only (error of over 1.5 kilometers)	83
Figure 30 - Actual, Device, & Cloud locations with all sensors active (errors from 5 to 45 meters).....	84
Figure 31 - Map My Walk device (green) versus cloud (red) locations	85
Figure 32 - Location Tracker device data (yellow) versus cloud data (magenta) - no GNSS	86

Figure 33 - Location Tracker device data (purple) versus cloud data (green) - with GNSS.....86

Figure 34 - Device versus actual versus cloud locations in urban environment without GNSS ...87

Figure 35 - Device (purple) versus cloud (green) locations in urban environment with GNSS ...88

Figure 36 - Google Maps directions artifacts recovered via Strings command utility from
"gmm_storage.db" database91

Figure 37 - Timestamp from Google Maps "gmm_storage.db" database directions search record
.....92

LIST OF TABLES

Table 1 - Android Application Location Requests	12
Table 2 - Factors Affecting Cell Tower Activity.....	17
Table 3 - Strategies for Recovery of Location Data from Android Devices.....	25
Table 4 - Test Device Information.....	33
Table 5 - Average Cloud Location Frequency	41
Table 6 - Databases of Interest	47
Table 7 - Selection of content recovered from Viber app's "viber_messages" database as reported by Internet Evidence Finder	52
Table 8 - Selection of content recovered from Facebook Messenger app's "threads_db2" database as reported by CelleBrite Physical Analyzer	53
Table 9 - Content of Foursquare's "fsq.db" database, extracted via SQLite query.....	56
Table 10- Destination data extracted from the Waze app's "user.db" database, <i>RECENTS</i> and <i>PLACES</i> tables, using SQLite query.....	58
Table 11 - Turn-by-turn directions recovered from Waze app's "tts.db" database, using SQLite query.....	59
Table 12 - OneWeather app's "oneweather.db" cached locations, from <i>geocodes</i> table, showing device location history	62
Table 13 - AccuWeather app's "accu_forecast.db" database content showing stored user-configured location, not actual device location	62
Table 14 - Examples of connection types logged in the Amazon app's "event" database that are not consistent with test session documentation	69

Table 15 - Excerpt of the Google Mobile Services' "herrevad" database showing WLAN network connection details.....	70
Table 16 - LG VS870 geo-tagged photos, average accuracy	74
Table 17 - OnePlus One A0001 geo-tagged photos, average accuracy	74
Table 18 - Recovery and evaluation strategies	93
Table 19 - Apps and Settings Information	104
Table 20 - Test Session Information.....	106
Table 21 - Analysis tools used in this study	109
Table 22 - Suggested keywords for recovery of metadata, application code, or sensor activity	111
Table 23 - SQLite query details	113

LIST OF ACRONYMS (or) ABBREVIATIONS

ADB	Android Debugging Bridge
API	Application Programming Interface
BLOB	Binary Large Object
BSSID	Basic Service Set Identifier
CDMA	Code Division Multiple Access
FCC	Federal Communications Commission
GLONASS	Globalnaya Navigazionnaya Sputnikovaya Sistema, or Global Navigation Satellite System
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GREP	Globally search a Regular Expression and Print
GSM	Global System for Mobile
ISP	In-System Programming
JPG	Joint Photographic Experts Group
JTAG	Joint Test Action Group
MAC	Media Access Control
OS	Operating System
P.A.	Physical Analyzer
SIM	Subscriber Identity Module
SQLite	Structured Query Language Lite
SSID	Service Set Identifier
UTC	Coordinated Universal Time
WLAN	Wireless Local Area Network
XML	Extensible Markup Language

CHAPTER ONE: INTRODUCTION

Mobile devices have become pervasive throughout modern society and everyday life. As the devices have increased in proliferation, they have likewise improved in capabilities. They essentially function as pocket-sized computers, with full operating systems and the ability to install and run additional applications. Their hardware features have expanded to move beyond the transmission of voice and text content to include multimedia, internet browsing and streaming, location awareness, and navigation functionalities. This enrichment of features and capabilities has further entrenched the mobile device into everyday life for many, whether it be checking email or searching for a restaurant nearby.

Leaving aside the sociological implications of this heightened connectivity and convenience phenomenon, the frequency and intimate nature of use by the mobile device owner make them rich sources of data about an individual's interests, activity, relationships, and communication. Naturally, this makes the mobile device a particularly valuable source of evidence in both criminal and civil investigations. Because users interact so often and so personally with their devices, their content may be of interest even in less obvious cases, where the incident does not directly involve the use of the device but the device may still hold information that could inform the investigation or adjudication of the case.

In particular, the location-awareness of mobile devices means they could store or generate historical location data that may document the device's, and by extension, the user's, whereabouts during a particular timeframe of interest in the case. Such data, if recovered and identified, could corroborate the accounts of witnesses in the case or implicate a suspect or even assist in the recovery of a missing person. Of course, much depends on the reliability of any such

cached location information, as well as an examiner's ability to recover and interpret it from a particular device.

This paper will elaborate on the investigative role of mobile device location data and review existing research pertaining to the capabilities and data recovery efforts specifically involving Android devices and location information. The study focuses on criminal investigations in particular, but the concepts and methodologies presented would apply in civil contexts, as well. It further aims to address the complex nature of such recovery and interpretation by utilizing test devices in controlled conditions and examining the data extracted from the devices for evaluation of accuracy via comparison with the documented location history. It also examines the emerging role of cloud resources and presents test device cloud data retrieved for this study. Informed by the test findings, a general strategy for analysis of Android devices for location data will be presented, as will a discussion of limitations and recommendations for future research.

CHAPTER TWO: LITERATURE REVIEW

Proliferation and Utilization of Mobile Devices

According to the Pew Research Center (2014), a majority of American adults (58%) own a smartphone, and 29% of them identify their device as a necessity which they cannot imagine doing without. A separate report focusing on Internet users found that 80% of online adults possess a smartphone, spending an average of 1.85 hours per day online via their mobile device. Android-based devices comprise the largest market share at 54% (Mander, 2014), while also boasting high growth rates in emerging markets and a large app base of over 200,000 downloadable, third-party apps (Maus, Hofken, & Schuba, 2011). Android is an open-source mobile device operating system developed and maintained by Google.

Empirically, it is clear that smartphones have become an entrenched and intimate part of daily life in the modern world. Pew (2014) reports that 44% of cell phone owners have slept with their phone next to their bed. Just over a third of U.S. households utilize a cell phone in lieu of a landline, and individuals carry their phones with them everywhere and use them in a myriad of ways. As a society, we have come to rely on these devices not just for communication, but also to get directions, coordinate schedules, and even make purchases (Wells, 2014). Smartphones have clearly evolved into more complex and powerful tools. Such developments prompted the Washington State Senate Judiciary Committee to ask in 2012, “Have [mobile devices] also effectively become tracking devices (State of Washington Senate Judiciary Committee, 2012)?”

Certainly, there are those who would see the advantages of this, at least in some circumstances. Senator Charles Schumer has called for the Federal Communications Commission (FCC) to implement improvement plans for emergency call tracking of cell phones, noting that 70% of 911 calls now come from a cell phone (Fox, 2015). He recounted an incident from his state of New York, in which an elderly woman called 911 from a cell phone after having a stroke. Because her speech was slurred due to the stroke, she was unable to provide her address to the dispatcher. An address was obtained based on which cell tower(s) she was connected to during the call. The address proved inaccurate, however, and it was 8 hours before responders managed to find her. She died the next day (Schumer, 2015).

The big U.S. carriers (Verizon, AT&T, Sprint, and T-Mobile) pledged their commitment to a goal of providing precise location data to 911 dispatchers for 40% of cell phone 911 calls within two years, and 80% within six years. The FCC suggests this specific location information should be accurate within 50 meters horizontally and 3 meters vertically. The proposed method for achieving this level of accuracy involves the incorporation of nearby, static Bluetooth and WLAN-enabled devices, such as smoke detectors or wireless routers, which will be logged with a precise location in a special emergency services database. Carriers also plan to ensure that handset WLAN and Bluetooth functionality can be enabled remotely in an emergency, if they are disabled (Selyukh, 2014).

Other aims involving smartphones as tracking devices are less consequential, perhaps, but may be just as sensitive. Marketing companies, such as Path Intelligence (PI), claim to be able to detect phones entering their client's business, recording the frequency and duration of their visits, as well as the typical routes they take from business to business. PI markets this

technology to shopping malls. However, this obviously involves the collection of shoppers' location information, likely without their knowledge or consent, and then using that data for profit (Michael, 2013).

On the more altruistic end of the spectrum, some researchers have focused on the potential use of smartphones as a low-cost tracking device for Alzheimer patients. The devices often combine GNSS (Global Navigation Satellite System) functionality with internet access, so providers or caregivers could use a web-based application to monitor current or past location information for the patients (Paiva & Abreu, 2012). So the concept of using smartphones to monitor users' locations has been embraced from multiple perspectives.

Even smartphone users themselves often use their devices as tracking devices, as a convenience. A study published in 2010 of almost 50,000 Android apps showed that 40% of the apps utilize the device location (Maus et al., 2011). Apps like Gas Buddy, Yelp, Waze, and Back Country Navigator allow users to find products or services nearby, navigate to them, and even use their devices as GPS guides in remote places (Reisinger, 2013). Pew (2014) reported that 74% of smartphone owners had used their device to get directions, recommendations, or some other location-based information, with 12% saying they had used their device to "check in" at a particular location or to share their whereabouts with friends/family. Some apps market this location sharing feature as a public safety measure, allowing parents to monitor their children's locations.

As another interested party on public safety matters, law enforcement also explores the use of mobile devices for tracking purposes. Given the intimate nature of the devices and the fact that over 6 billion cell phone subscriptions exist in the world (Wells, 2014), law enforcement

interest in such functionality is unsurprising. In discussing the usage of cell phones as tracking devices by law enforcement, Wells also noted that cell phone carriers informed Congress that they collectively received 1.3 million requests for customer information in 2011. This involved both real-time tracking and historical information on cell site connections obtained from the carrier. Both of these practices will be explored in greater detail later in the paper, in terms of how they are implemented as well as existing research on their reliability.

These descriptions of smartphone roles in law enforcement, emergency response, marketing, user convenience, and even healthcare emphasize their ubiquity and the depth to which they have permeated modern life. Their internet connectivity and location awareness have been powerful contributors to these developments. The next section explores the technological features that enable these functions.

Device Capabilities and Location Awareness

Indeed, mobile devices come equipped with a substantial arsenal of hardware sensors and transmitters to facilitate location awareness, augmented by web-based services operated by various private and public entities. This section of the paper will examine these resources and their role in sustaining location-based services in Android devices.

The State of Washington Senate Judiciary Committee (2012) identified four key ways mobile devices may determine a user's location: GNSS, cellular network information, WLAN access points, and users themselves. A review of the first three of these resources will demonstrate their relative strengths and weaknesses, followed by a discussion of the services that supplement the hardware-based abilities of the device.

Before evaluating the utility of GNSS, more detail on its components and operation is warranted. GNSS is a relatively newer term used to encompass all potential satellite systems a device may use, since the activation of the Russian Global Navigation Satellite System (GLONASS) toppled the exclusivity previously held by the U.S.-operated Global Positioning System (GPS). Other satellite systems operated by other entities or nations are emerging, but thus far, these are the two commonly used in U.S. devices (Last, 2015). Throughout this paper, when the term “GNSS” is used, it refers to GPS and GLONASS. Where a cited study strictly refers only to “GPS,” the term GPS will be used.

The GPS network includes 27 operational satellites and has been operating for over 30 years (State of Washington Senate Judiciary Committee, 2012). Alternatively, GLONASS utilizes 23 operational satellites as of January 2012 (Cai, 2013). Both operate in similar fashion, via trilateration. The device receives signals from as many satellites as it can, then measures the distance between each satellite based on the time each satellite’s signal takes to travel to the device, in order to determine its precise location. Devices do not actively transmit their location (Last, 2015). Since both systems require line-of-sight paths between at least three satellites and the terrestrial devices utilizing them, they are subject to some limitations. Specifically, it can take a significant amount of time to get signals from enough satellites, and signals may be degraded or blocked entirely if there are atmospheric interferences or solar activity, topographical obstructions, or if the device is being used indoors or underground (Michael, 2013). In short, the big factor in the accuracy or utility of GNSS services is the surroundings (Last, 2015). Additional constraints are more fundamental: sensors are not present in all cell phones (though certainly in most if not all smartphones); the process requires a lot of power, draining the device

battery quickly; due to the power concerns, the sensor(s) are not typically enabled by default and must be activated by the user. On the plus side, when GNSS services are used successfully, the data is typically extremely accurate and often includes additional detail on speed, altitude, and direction of travel, or bearing (Lifchitz, 2010).

Cellular network strategies typically present the opposite traits. All cellular phones possess the necessary transceiver, so additional hardware is not required. Furthermore, the function is typically enabled by default and consumes significantly less battery power, provided cellular network coverage is adequate. Cellular coverage continues to expand and improve in the United States, with over 280,000 cell sites in use as of June 2012. The tradeoff usually involves accuracy. In rural areas, where cell site towers are sometimes miles apart, accuracy tends to be less than in urban areas, which have higher tower densities (Wells, 2014). Other factors affecting the reliability of cellular network location data will be detailed later in the paper.

Since every cell tower is uniquely identified and carriers maintain location information on each of their towers, the towers with which a phone communicates provide the basis for tracking its location. When a phone is on and the cellular service is enabled, the phone will attempt to connect to a tower as often as every 7 seconds (State of Washington Senate Judiciary Committee, 2012). Phones will often communicate with multiple towers simultaneously or within very short amounts of time. In these instances, a more precise location for the device may be obtained via triangulation. This process may be activated and tracked remotely from the network side, unlike GNSS, in a process known as “pinging” (Lifchitz, 2010). So in an emergency, for example, responders can utilize carrier resources to activate the device’s E911 system, which will then use the cellular network pinging process or even activate the phone’s

GNSS service. The device can then obtain precise GNSS coordinates, if available, and then transmit them to emergency services via the carrier's network to help guide responders to the right place (Daniel, 2014).

The last of the hardware-based tools involves WLAN networks. This is actually a sort of hybrid approach, because it involves the use of network-based geolocation services. The technique involves the detection of nearby WLAN access points, along with their relative signal strengths (Brouwers & Woehrle, 2012), followed by a query of a remotely stored database containing location information associated with the particular access point's media access control (MAC) address (State of Washington Senate Judiciary Committee, 2012). Theoretically, it could circumvent this remote query process if the MAC address-location information were stored locally on the device, as some third-party apps billed as "WLAN hotspot finders" purport to do. In either case, the benefits of the WLAN method typically involve lower power consumption, no performance detriment indoors, and decent accuracy. However, WLAN connectivity must also be enabled by the user, relies on the presence of access points (or other nearby hardware tracked in the queried database, such as cell towers, Bluetooth devices like smoke detectors, etc. [24]), and may not be as accurate as GNSS location results (Lifchitz, 2010).

The discussion of WLAN-based methods introduced the extended functionality offered by remotely operated services. A key provider of the aforementioned remote location lookup services is Google itself, which maintains a database supported by the "anonymous" collection of geographic data for cell tower or WLAN routers to which its users are connected. By opting to use Google's Location Services, users agree to contribute to this database (4RENSIKER, 2012).

The Google location database is thus vast and constantly updated, receiving data from Google's cars (street view and automated) and of course, Android phones (Lifchitz, 2010).

Android devices have built-in application programming interfaces (APIs) for location services developed and maintained by Google. Both stock Android apps and third-party apps use these services (Davydov, 2011). The location API may use any of the previously described resources to obtain device location, depending on the availability of the resource, as well as the parameters of the particular app. Apps may be programmed to request the current device location, receive updates on the location, look up addresses from detected device latitude and longitude (this process is known as reverse geocoding), or perform geofencing or activity recognition ("Making your app," 2015). Geofencing involves the caching of location history of the device or recognition of a particular location, in order to provide additional context-based functionality, such as reminding a user of some task when they arrive home (Maus et al., 2011).

Google's application development tutorials offer some insight into the functionality of their own location services API, the fused location provider. Specifically, app developers must not only code for the location functions described above, but they must also ensure their app contains the appropriate permissions to do so. Location permissions come in two flavors: coarse and fine, with fine being the more precise of the two. The permission level and the details of the location request determine the accuracy of the information obtained via Google's location services ("Making your app," 2015). The 2010 study of Android apps showed that of the 50,000 reviewed, 25% of the apps requested permission to the device coarse location, and 15% requested permission to the fine location. Examples of such apps include Facebook, Foursquare, and Twitter, as well as more obvious tools like mapping and navigation apps (Maus et al., 2011).

Indeed, the Android development tutorial offers an enticing pitch to developers to include location-based features in their app development tutorial (“Making your app,” 2015), noting the following:

If your app can continuously track location, it can deliver more relevant information to the user. For example, if your app helps the user find their way while walking or driving, or if your app tracks the location of assets, it needs to get the location of the device at regular intervals. As well as the geographical location (latitude and longitude), you may want to give the user further information such as the bearing (horizontal direction of travel), altitude, or velocity of the device. (p. 5)

This statement offers further insight into the capabilities and strategies of the device. It suggests that location information may be routinely updated at configurable intervals, with the update rate having reliability implications. It also implies that GNSS technology may be used, with its references to latitude and longitude, bearing, altitude, and velocity. Furthermore, it suggests some good application types to use for studying this topic, specifically navigation and tracking apps, in addition to those apps already mentioned.

The tutorial goes on to lay out how developers code for this type of functionality. First, developers must ensure their applications have the proper permissions, as discussed previously. Coarse permissions are described as yielding locations with maximum accuracy within approximately one city block, while fine permissions are needed for functions requiring greater precision (“Making your app,” 2015). Permissions are disclosed to the user when they attempt to download or update an app from the Google Play market, and they must accept these in order for the download to proceed (State of Washington Senate Judiciary Committee, 2012). Later, we

will examine how this arrangement figures into issues of user awareness and consent in terms of device location capabilities and monitoring.

Having secured the proper permissions, developers must then program their app to request the device location. This may be done a single time, to request the device’s last known location, or it may be configured to receive regular location updates. The parameters of any of these location requests will affect the accuracy of the information, dictating the update interval and the priority level. The table below summarizes the different request types, as described in the Android developer tutorial (“Making your app,” 2015).

Table 1 - Android Application Location Requests

<i>Request Type</i>	<i>Description</i>
Balanced	Considered coarse-level (max accuracy within 100 meters) Likely to use WLAN and cell towers to obtain device location, depending on availability
High accuracy	Most precise When used with fast update interval of 5 seconds, this request type can return information accurate within a few feet More likely to use GNSS Appropriate for mapping/navigation apps
Low power	Consumes less power City-level precision, accurate to within 10 kilometers
No power	Receive updates when available as other apps request location updates Accuracy dependent on permissions/request details of other application(s)

The varying options for the location requests themselves reflect the inherent inverse relationship between accuracy prioritization and battery life, as well as the developers’ desire to balance the two. High accuracy requires high power consumption. There would also seem to be some correlation between these request parameter types and the app permission details. Indeed, in order for the high accuracy parameter to be implemented, an app must have permission to the

device fine location (“Making your app,” 2015). This implies that applications without such permissions may be less reliable sources of location data, when examining artifacts recovered from the device itself. This hypothesis helps shape the strategy of this study and will figure in the review of study findings.

The Android tutorial essentially states that the fused location provider updates apps periodically with the best available location, and that the accuracy of the information depends on the active sensors (i.e., cellular, WLAN, GNSS), the location permissions, and the options in the app’s location request (“Making your app,” 2015). Thus, beyond identifying app types that may be of interest, the tutorial’s description of the fused location provider’s features might also suggest strategies for analyzing the data associated with those apps. For example, if applications log timestamped metadata about their activity, a review of this data could provide some insight into how reliable any associated, cached location information may be, by documenting its location request activity and details. This study’s methodology will include a search for such metadata.

Clearly, the devices have many resources and many purposes involving location information. To date, much of the research involving the location-awareness features of Android phones has focused on the accuracy of tracking a live device as well as the review of historical cell site data obtained from cell phone carriers for a particular user. However, from a forensics perspective, many crimes are not reported or known until well after the incident, leaving investigators with a device seized later from an identified suspect and rendering live tracking moot. Furthermore, historical cell site data may not be accessible, or as we shall see in

subsequent sections, entirely reliable. Thus, the impetus and motivation for this study involves the recovery and reliability evaluation of the data recovered from the device itself.

Existing Research on Location Data Recovery and Reliability

Maus et al. (2011) noted that in general, locations derived by smartphones are generally quite accurate, in most cases. Empirically, this would make sense, since companies want users to enjoy and rely on their products, but it of course implies that the accuracy varies under different conditions. This section of the paper reviews the existing research on accuracy associated with device location services, as well as the recovery and reliability of the data recovered from the devices themselves.

Accuracy of Device Location Services

Much prior study has targeted the live tracking of devices, rather than data recovered from the devices after the incident. Even though this study focuses on the analysis of data recovered from the devices themselves, these works still inform the study methodology and shape hypotheses. Part of this work's aims involve an assessment of the accuracy trends noted in live tracking with regard to location resources such as cellular data versus GNSS, for example. If these trends do hold, it will place a higher priority on developing strategies for reviewing device data to determine which resources were active when the recovered data was cached, or evaluating if such a determination is even feasible at all.

In an excerpt from his upcoming book, *Cellular Location Evidence for Legal Professionals*, Larry Daniel (2014) states that two fundamental options exist for tracking a

device in real-time: activate the device GPS and request information via pings from the carrier side or triangulate the location using the cellular network. Daniel describes GPS as accurate to within 50 feet, while noting that triangulation demands three reference points and can err by up to several thousand feet. Each of these techniques has its own set of benefits and limitations.

As noted earlier, GNSS-based services drain battery power quickly, require time and line of sight with several satellites to fix the device location, and can be subject to interference by atmospheric conditions, buildings and trees, or radio activity/jamming. On the other hand, the U.S. government reports that the civilian GPS service offers accuracy within 7.8 meters with a 95% confidence level, making it an exceptionally accurate resource (Michael, 2013). Given that this service is now being used in conjunction with the Russian GLONASS system in newer devices, the effective accuracy may be even greater. Plus, as Professor David Last (2015) notes, GNSS services also often log additional metadata beyond just latitude and longitude information, including quality (accuracy) information, altitude, speed, and bearing. This additional metadata, if cached on the device, may be of use in evaluating the reliability of the recovered location history. This idea will be incorporated into this study's evaluation of test results.

Last (2015) also points out that an examiner can always check if the recovered coordinates make sense, lining up with travel routes or roads, for example. Furthermore, the precision of the coordinates and altitude may also offer some insight. For example, a decimal GPS coordinate with hundredths-level precision (two places after the decimal) is accurate to within 1,111.1 meters, but a coordinate with ten thousandths-level precision (four places after the decimal) can pinpoint locations within 11.1 meters. Too many places after the decimal point, however, may indicate that a particular coordinate is dubious or fabricated, because consumer-

level technology in mobile devices is not typically capable of resolving locations to that level of precision (Bairstow, 2015). Since altitude is typically less accurate than latitude and longitude values, if the altitude seems quite accurate, the latitude and longitude data should be very reliable. Also, since one of the known GNSS issues involves atmospheric interference, an examiner can check for unusual solar activity at the time in question, as solar activity is well-tracked and documented by multiple entities.

This latest point highlights another way GNSS and cellular-based location features differ, in that at least a number of GNSS limitations can be evaluated after the fact, such as the solar activity or the topography of the region in question. However, a few major caveats apply to the use of cellular tower connections to locate a device, whether in a live tracking situation or in an effort to reconstruct location traces from historical cell site records obtained from the carrier.

The primary issue arises from the assumption that a device will connect to the tower(s) nearest its current location. This assumption is problematic for a number of reasons. For multiple reasons, cell tower coverage varies and may not be known for a given time. Phones will connect to the tower with *the best signal at the time*, but since multiple factors affect cellular signals, that tower may not necessarily be the closest (Daniel, 2014). Some of the factors involve the tower infrastructure and operation, the local environment, and even the devices themselves (Wells, 2014). The table below summarizes the various factors affecting phone-cell tower interaction.

Table 2 - Factors Affecting Cell Tower Activity

Cellular Network/Operations	Environmental	Device
Number of cell sites	Weather	Wattage output
Ongoing maintenance or repairs	Topography (hills, trees, etc.)	Broadband capability (age of phone)
Height of tower	Urban structures (buildings, signs, etc.)	Indoor/outdoor at time of use
Height of tower above sea level	Time of day	
Wattage output		
Range of coverage		
Number of antennas per tower		
Direction and height of antennas		
Call traffic via each antenna at time of connection		
Interference from other towers or radio signals		

In cases of live tracking, these complicating factors may be rendered moot when the device is actually located, but in cases involving historical cell site data, Daniel (2014) asserts that because of these issues, it is not possible to know the coverage area of a cell tower at the time of a particular event, nor is it possible to recreate exactly the conditions under which the event occurred, in terms of the cell coverage and operation. Furthermore, even if carriers maintain the location of each of their towers, one cannot know from the carrier's records if the recorded tower was actually the one *closest* to the phone at the time. For example, even if a tower is closest, it may be inundated with heavy call traffic during peak activity times, forcing a phone to connect to a tower further away. These issues have significant implications for the use of these historical carrier records in the adjudication of criminal cases, as will be discussed in the next section of the paper.

In terms of the accuracy of the cellular-based location derivations, varying results have been obtained. By working off just the single tower connected to and referring to carrier

maintained locations associated with that tower, accuracy levels within 100 to 3000 meters have been assessed. Triangulation techniques, available when a device communicates with multiple towers simultaneously or in rapid succession, have produced accuracy levels as precise as 25 meters (Maus et al., 2011). How are these assessments performed, though?

Yang et al. (2010) highlight the technique of wardriving, wherein “a vehicle drives within the target area recording signals emanating from nearby cell towers (or WLAN access points) and the locations these signals were received at.” They then use various mathematical strategies to infer the location of the various towers and access points detected during the collection phase. By comparing these inferred locations to the known locations of the towers in question, they evaluate the accuracy levels.

Yang et al. (2010) performed their own wardriving effort around Los Angeles, an area of roughly 1396 square kilometers covered by 54 cell towers at the time. The team collected data, measuring signal strengths every two seconds as they drove around the area for a period of two months in 2009. Having gathered their data, they used techniques to infer locations of the towers they detected and compared the results to the known tower locations they had documented. They then developed their own supplemental mathematical strategies to refine those results, in an attempt to establish a way of improving the reliability of location data inferred from cellular tower interactions. The figure below provides a visual representation of the improvements to accuracy they were able to achieve via their innovative post-processing, as well as a quick comparison of the variations in accuracy among different environments, namely rural, residential, or urban areas (Yang et al., 2010).

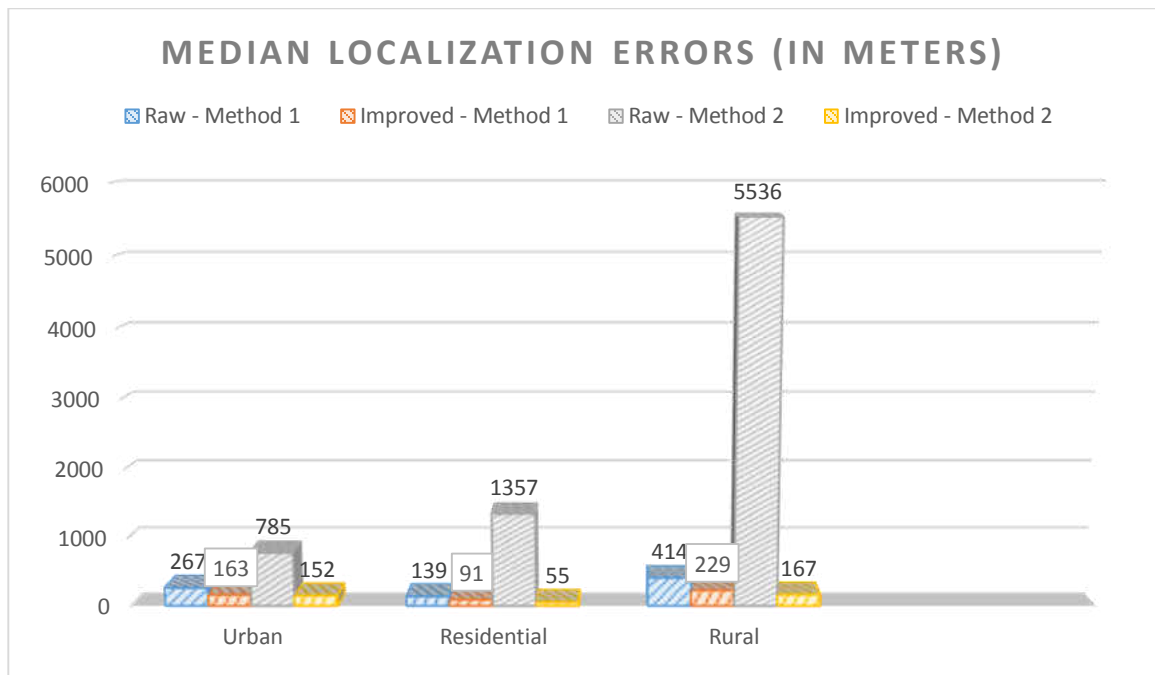


Figure 1 - Cell Tower Localization Errors by Environment

From these results, it would appear that cellular location information may be fairly reliable, generally much more so in urban and residential areas than in rural areas. Other researchers seem to corroborate this generalization, describing accuracy estimates as being within 50 to 100 meters in urban areas and some hundreds of meters elsewhere (Michael, 2013).

Another study focused on identifying when a user is “dwelling,” or stationary, by tracking the user’s mobile device. Such concerns may figure in multiple contexts, from military operations to marketing strategies. The researchers created a custom Android application and collected information from five different devices with seven users. Users were directed to enable various settings at certain times and to track their own movements throughout their use of the devices. These researchers looked beyond cell towers and focused on the addition of GPS and

WLAN resources, as well as geolocation services, like Google location services or rival provider Skyhook (Brouwers & Woehrle, 2012).

As might be expected, these researchers found that tracking information was most accurate when all of the aforementioned resources were involved, meaning GPS and WLAN sensors were enabled and access to geolocation services was facilitated. Again, geolocation services use the detected WLAN access points and signal strengths to query a remote database, which returns a location and estimate of accuracy based on the query information. The researchers noted that the best results typically involved the use of this service, although the quality of such information depends on the accuracy of the database and how many WLAN access points are actually in range at particular place. They found that static users would sometimes appear to jump between two points over 100 meters apart within seconds, as signals were detected and lost, etc. They also found that users moving at constant speed would appear to have clumped locations along their tracked route, rather than continuous travel, likely a sign that devices revert to most recent previously detected location when no new signal is detected. Furthermore, the use of geolocation services comes with a cost in terms of power consumption versus cellular only, especially if the rate of the queries is increased (Brouwers & Woehrle, 2012).

But how are these geolocation service databases developed? Who maintains them? How is their accuracy evaluated and improved?

The “dwelling” researchers highlighted two key points that offer some response to these questions and also serve to guide some of the methodology of this study. First, they noted that Google’s geolocation service boasts reliable accuracy and extensive coverage. Secondly, they

offer some insight to its operation. According to Brouwers and Woehrle (2012), “Google trains its database using a background service built into Android devices that reports GPS coordinates and WLAN scan results to their servers at regular intervals” (p. 667).

This indicates that Google’s database is likely to be extremely well-maintained, given the abundance of Android users and their ongoing participation in contributing to its improvement. It further suggests that Android users may not be particularly cognizant of their role in this maintenance effort, since it is a background service that is built into the devices that facilitates the activity. So does Google actually have a vast repository of location history information associated with its Android customers, and is that information at all accessible? Those questions also guided this study’s methodology and will be addressed in the study findings.

However, other geolocation services, such as Skyhook (Brouwers & Woehrle, 2012), for example, certainly cannot enjoy Google’s access to Android user location updates. There are other ways of building geolocation databases, though. The aforementioned technique of wardriving may be employed. Service users may also contribute known location/access point information directly to the provider, as with Skyhook (Skyhook, 2015). Via wardriving, voluntary user contributions, and in Google’s case, Android background location services, various providers have been able to build geolocation service databases (Michael, 2013).

Having elaborated on how the geolocation services are developed, operated, and maintained, the question turns to their reliability. One study evaluated Skyhook’s service, contrasting the company’s claims of 10-meter accuracy with results closer to 63-meter averages in Sydney, Australia, and 43 to 92 meters in Las Vegas, Miami, and San Diego (Michael, 2013). Yang et al. (2010) reference a study regarding wardriving and WLAN access points that found

that the estimated access point locations have a median error of 40 meters. Another group noted that WLAN access point locations can change (Davydov, 2011). However, Brouwers and Woehrle (2012) generally found geolocation services to be fairly accurate, especially when used in conjunction with other resources like GPS.

In summary, a review of the existing work involving smartphone location resources and their accuracy indicates a broad range of reliability estimates, from a few meters for GNSS information to a few kilometers for positions obtained via a single cell tower connection. The chart below offers a visual representation of the relative error ranges by resource type, with cellular on the order of thousands of meters, GNSS way down in the single to double digits, and WLAN resources overlapping between the two. Part of this study’s objectives will be to determine if any such accuracy trends are noted in the testing.

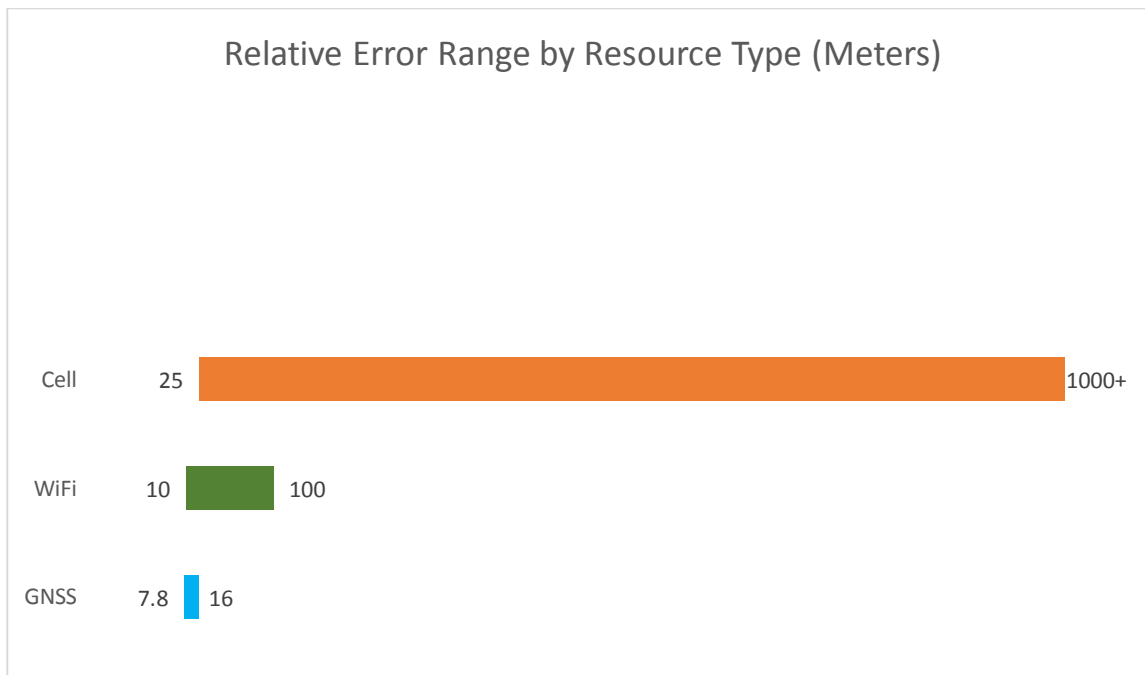


Figure 2 - Relative Error Range by Resource Type

Recovery of Location Data from Devices

Studies related to location data recovered from the devices themselves appeared somewhat limited in scope and become dated quickly due to the fast-paced evolution of devices and applications, but they do offer some instructive insight. Various strategies are described by different researchers. Some are more direct than others, as the subsequent discussion will show.

Several works mentioned the need to “root” a device to extract all of the data (Davydov, 2011; Kroger & Creutzberg, 2012; Maus, Hofken, & Schuba, 2011; Racioppo & Murthy, 2012; Sack, Kroger, & Creutzberg, 2012). The term “root” refers to the process of obtaining administrative, or root, access to the device’s operating system. This enables greater user control of the phone’s functionality and files, and it also ensures a forensic examiner the ability to obtain a complete raw image of the device’s internal memory (Racioppo & Murthy, 2012). Statistics on just what percentage of Android users root their phones proved elusive, but the process is specific to each particular make and model of the device, sometimes even varying depending on the device firmware version as well (Riley, 2015).

Also highlighted by multiple researchers are two files associated with older versions of Android, and therefore, older devices: `cache.cell` and `cache.wifi` (4RENSIKER, 2012; Kroger & Creutzberg, 2012; Yi, 2012). These files stored the 50 most recently detected cell tower locations and 200 most recent WLAN network locations, respectively. The information was timestamped, as well (4RENSIKER, 2012). However, researchers also noted that these files would only be populated with information if the user had enabled two specific settings which are not active by default (Kroger & Creutzberg, 2012). Furthermore, the files would only maintain the information up to the previously stated maximum record number, or for fourteen days (4RENSIKER, 2012).

Instead of focusing on specific files of interest, some chose to hone in on certain file types, specifically SQLite databases and certain picture files. Racioppo and Murphy (2012) stress that SQLite databases are among the most important features of a smartphone, for forensic purposes, storing the bulk of the application data for Android apps. Among others, Yi (2012) discusses geo-tagged photos, images captured by the device's camera that contain embedded latitude and longitude information. However, Yi also offers the more novel approach for photos of possible interest that are not geo-tagged: upload the image in question to Google Image search to find possibly similar images that *are* geo-tagged. More conventionally, Sack, Kroger, and Creutzberg (2012) also point out the potential value of Google Maps' map tiles, which are snapshots of map images viewed via the application.

Other examiners targeted specific apps, or categories of apps. For example, Saliba (2013) identifies the Facebook and Facebook Messenger apps as specific sources of location data, noting that the "threads_db2" database(s) associated with these apps store latitude and longitude values, along with altitude and speed, along with users' message content. Maus et al. (2011) highlight other apps, such as Google Maps, Foursquare, and Twitter, as well as database content recovered from a weather app and a navigation app. Sack et al. (2012) also mention cookies and databases associated with the device's web browser app. Davydov (2011) extends the consideration to all location-aware apps, naming example categories like navigation, social networking, weather, travel services, and banking.

Finally, when the traditional avenues of exploring known file types and applications of likely interest have been exhausted, the task shifts to identifying more elusive types of location artifacts. Artifacts stored as text addresses, points of interest, or navigation routes may require

additional search techniques (Barmpatsolou, Damopoulos, Kambourakis, & Katos, 2013). Strategies involve searching for geodata formats (GPS coordinates) or keywords like *location*, *latitude*, *longitude*, or *address* (Maus et al., 2011). Davydov (2011) suggests looking for logged MAC addresses of WLAN access points the device has detected, then looking up the associated location. The table below summarizes the various strategies utilized by previous researchers for recovering location data from Android devices.

Table 3 - Strategies for Recovery of Location Data from Android Devices

File Types	App Categories	Other
SQLite Databases	Navigation	Text content
Pictures	Social networking	Addresses
Geo-tagged	Weather	Points of interest
Map tiles	Travel	Routes
Google Image Search	Banking	Keywords
		MAC addresses of WLAN access points

While all of these insights are useful and certainly inform the methodology of this study, little comment on the accuracy of any such recovered data was observed. Davydov (2011) offers a discouraging assessment of prospects, noting that phones use cellular, GPS, and WLAN resources collectively to get location information, and there is no way to determine how particular data recovered from a device was obtained by that device. As an evaluation of his own proffered method regarding cached MAC address lookup, he notes that any location estimate so obtained may be of questionable accuracy. The relative lack of resolution regarding the reliability of the location information recovered from Android devices factored heavily into the motivation for this study.

Use of Mobile Device Location Data in Criminal Cases

Of course, the reliability of mobile device location data would be of interest in criminal cases where one or more of the parties involved used a smartphone during or around the incident. Sisak (2012, p. 2) notes that “the technologies that make smartphones so ‘smart’ also make them the closest thing law enforcement officers have to...homing devices.” Sisak also quotes one detective who points out the potential boon that the increasingly ubiquitous smartphone has brought investigators:

It only makes sense for us to look for digital evidence. A crime is committed. People panic. They’re making calls. They’re sending text messages. All of that stuff is being digitally recorded and it’s going to be great evidence for prosecuting a case. (p. 1)

The nature of who or what is recording the information becomes of interest. It could be service providers, law enforcement or other parties actively tracking a live device, or it could be cached by the device itself. Live tracking typically pertains to investigative operations, not prosecution strategies, though courts have ruled that using a phone’s GPS capability to track location does not require a warrant as law enforcement need no physical contact with the device to facilitate such operations (*Harvard Law Review*, 2013). However, the remaining two of these possibilities raises multifaceted concerns regarding the reliability and admissibility of the information, leading to qualified conclusions about how it should be used in court.

The first category, data retrieved from service providers, has been the most studied and arguably, the most controversial. The general process involves a few preliminary steps: associating a device with a person of interest; identifying the phone carrier in question; and serving some sort of legal process (i.e., search warrant, subpoena, etc.) on the carrier for the

relevant call detail records. Then, an analyst examines the records and plots locations of cell towers on a map, assigns a coverage area to each tower, then plots pie-shaped sectors that represent possible locations from which particular cellular activity originated (Daniel, 2014). The resulting maps may be presented in court as evidence of where an individual could have been at or around the time of the crime, or to track past activity (Blank, 2011).

This practice has been targeted from various angles. Critics question everything from how the records are obtained to their relevance and admissibility to their scientific validity. Even service providers have weighed in on the discussions, with AT&T filing a friend-of-the-court brief on a case involving records obtained via a court order, arguing that a search warrant should be required. Whether due to AT&T's intervention or not, the 11th Circuit Court of Appeals ruled that police do indeed need a search warrant for cellular location history (American Civil Liberties Union, 2014). A number of efforts to require search warrants for such records have been undertaken in courts and legislatures in California, Maryland, and Georgia. Though these did not all succeed, Washington state attorneys, for their part, are now advising law enforcement to obtain a search warrant when seeking cellular location data from service providers (State of Washington Senate Judiciary Committee, 2012).

In terms of relevance and admissibility, the path has been a bit smoother but still has some evolving nuances. Establishing the relevance of the records, or for data obtained via live tracking or from the device itself, has proven fairly straightforward. As long as a connection between the device and the person can be established, as when a phone is found in an individual's possession or registered in their name, the location of the phone can certainly be germane. Furthermore, records from providers are admissible under the business records

exception to the hearsay rule, so long as they were obtained appropriately (Blank, 2011). Some argue that an expert witness should be required to testify to cellular records, since the average juror is unlikely to understand the technology well enough to use the information to judge the defendant fairly (Wells, 2014).

Others would argue that introducing historical cellular location data via an expert witness would impart a weight to the information that may be unwarranted. The previous review of existing research attributed a broad and variable range of accuracy to cellular location information. Furthermore, Daniel (2014) asserts that such records do not meet the Daubert standard for scientific evidence. The Daubert standard sets out specific criteria governing the admissibility of expert findings. Daniel particularly highlights the requirement that processes be subject to peer review, have published error rates, and conform to standard, repeatable methodology. Daniel argues that location evidence from call detail records fails to meet any of these conditions. Specific findings may not be repeated or corroborated via peer review, since it is impossible to recreate all of the conditions at the time of the incident and it is likewise infeasible to know the exact distance between phones and towers at any given time. To be forensically sound, Daniel notes that a process must be predictable, repeatable, and verifiable. Once again, he argues that the use of call detail records to track a phone's location fails on all of these counts.

This leaves the data recovered from the device themselves. Presumably, such data would be easily admissible if law enforcement obtained a valid search warrant for the device, and if the device was collected from the subject, the association between user and device would easily cement the relevance of any recovered data. However, an interesting argument emerges, hinging

on application permissions. As stated previously, Android applications require permissions to access (and possibly cache or transmit) the device location, and users accept these permissions upon installing the app. Thus, an inference can be made regarding the user's consent to the collection of their location data, which has been used to rebut their expectation of privacy with regard to the information. However, vague privacy policies presented by apps assuring users that the information is used "to improve services" or the like may obscure users' awareness that their location history may be collected or used by other parties. This may undermine the easy assumption that cached location data should be accorded no special privacy protections (State of Washington Senate Judiciary Committee, 2012).

In any case, privacy concerns continue to inform legal developments regarding mobile device location data, but Michael and Clarke (2013) argue that even if proper legal measures are indeed taken to obtain the data, it can still lead to a "miscarriage of justice" (p. 221) if the tracking data is not accurate. Indeed, both Blank (2011) and Daniel (2014) argue that the data can be helpful in refuting an alibi or demonstrate travel, for example. The issue arises from overstating the accuracy of the presented information. Ultimately, though, as Wells (2014) points out, the question of accuracy is for the jury to decide. The questions of reliability and authority still restrict the utility of mobile device location data in criminal cases.

Emerging Issues

Aside from the trend of heightened legal protection of device location information, additional challenges are emerging in the world of Android forensics, in general, and for location data recovery, in particular. These complications start with the devices themselves. Newer

versions of the Android operating system (KitKat and higher) have inherent security measures that prevent access to key device functions, some of which were previously utilized by forensic software developers and others to obtain full physical images of the phone's internal memory. Now that those exploits may no longer be used, examiners must rely on the built-in Android backup functionality to extract data from the device. The main issue with this is that application developers can set a flag in their app's code to exclude their app's content from the backup process. Unsurprisingly, many stock Google applications appear to have this exclusion flag set. The ultimate consequence of this development is that even with a fully accessible, unlocked device, an examiner may still not be able to extract all of the phone's data, including in some cases, the data of particular interest. This leaves examiners with complex rooting or custom recovery options that require significant research and testing, or they can opt for hardware-based options like the Joint Test Action Group (JTAG), In-System Programming (ISP), or chip-off methods that are time-consuming or potentially destructive, if they need to retrieve excluded or deleted data from a particular device (International Association of Computer Investigative Specialists, 2015).

Given that Google excludes many of its apps from the backup process, this begs the question of what other backup resources are available to users. Here we turn to the cloud. Google, of course, backs up contacts and emails associated with users' Gmail accounts. Other providers like Facebook may keep messages or contacts on their servers, as well. Recovery of this data is a significant challenge, introducing a host of technical and legal issues. For example, the data may be stored across multiple servers that are geographically scattered. This complicates

jurisdictional issues and prevents the traditional approach of capturing a raw image of data, since the data must be identified and isolated before the extraction process (CelleBrite, 2015).

One prominent cell phone forensics company, CelleBrite, has developed a new tool called Cloud Analyzer, specifically designed to retrieve data from these remote resources. The tool utilizes the various providers' own APIs to authenticate credentials and download content. Credentials may be manually entered, as in cases where owner consent has been obtained, or they may be recovered from account information extracted from a target device. In any case, this technique offers two inherent benefits. First, the use of the providers' own interfaces and protocols means that whatever the provider would allow the user to access, the Cloud Analyzer software can also retrieve. Second, because this method also employs the user credentials, concerns over encryption and specificity are rendered moot. The data will be received in its decrypted state and only the authorized user data will be obtained (CelleBrite, 2015).

The latest development involves out-of-the-box encryption for new Android devices, running version 6.0, also known as Marshmallow. Google has mandated full-disk encryption be implemented by the time the user completes their device setup steps, though only for new Marshmallow devices that meet certain performance standards. The requirement will not extend to older devices upgrading to Marshmallow, but the concern from a data recovery standpoint is that this will introduce significant complications going forward and may result in an inability to extract any data from the devices, even via hardware-based methods that would normally extract the full device content (Cunningham, 2015). This development could place even more emphasis on cloud data.

In summary, a heightened emphasis on securing device application functionality and user data has developed, raising new obstacles to software-based methods of capturing a complete extraction from newer Android devices. The simultaneously emerging shift from device-based storage of user data to cloud storage further complicates mobile device data recovery. This could place more emphasis on thorough analysis and understanding of what data *is* recovered from the device, as well as the cloud provider data. In any case, the recovery of any location data obtained from either source could prove crucial in a criminal case, and the ability to assess the accuracy of such data could figure heavily into its admissibility and impact.

CHAPTER THREE: METHODOLOGY

This study focuses on Android devices, since Android currently comprises the largest market share among mobile devices. Four different test phones were obtained, each a different make and model. The table below summarizes the features of each device, highlighting the differences in system version, network type, and hardware capabilities. These distinctions mirror the wide variety of devices likely to be encountered in forensic casework.

Table 4 - Test Device Information

Make/Model	LG VS870 Lucid II	Samsung SGH-i257 Galaxy S4 Mini	OnePlus One A0001	Samsung SM- G900P Galaxy S5
Android OS Version (Status)	4.1.2 JellyBean (Unrooted)	4.4.2 KitKat (Unrooted)	5.0 Lollipop (Unrooted)	5.0 Lollipop (Unrooted)
Carrier	Verizon	AT&T	AT&T	Boost Mobile
Network Type	CDMA	GSM	GSM	CDMA
GNSS Sensor(s)	GPS	GPS GLONASS	GPS GLONASS	GPS GLONASS

The decision not to root the devices was made for a number of reasons. First, a rooted device is the best-case scenario in terms of data recovery capabilities, but the goal of this study is to address the most typical scenario forensic examiners are likely to encounter. Since rooting is a complex and potentially damaging process, it seems likely that most users would not attempt to root their device. Thus, a rooted device would probably be an exception rather than the rule in forensic casework situations.

The devices were then prepped for the testing phase, with various location-aware applications installed and user accounts configured. These apps were specifically selected for their location functionalities, including permissions. Several application categories were chosen

for testing, including navigation/mapping, messaging/chat, fitness, weather, location sharing, and leisure activity apps. Ad-supported games were also installed, to see if any location data would be cached in relation to advertisements transmitted to the device based on its location. Devices and applications were configured with a view to optimizing the recovery of cached location data from the device.

Devices were used in 24 test sessions under controlled conditions. Recorded test parameters ranged from actual time and location to environment (rural, urban, or suburban) to enabled sensors (GNSS, WLAN, cellular, or combination), as well as careful documentation of user activity. Weather conditions were also noted, as well as which devices were used, of course. User activity involved navigation sessions, photo/video captures, chat and location sharing sessions, weather lookup, location searches, web searches, and/or workout recordings. The documentation was maintained to allow for a later evaluation of reliability regarding any extracted data.

After completing the device testing phase, work shifted to recovery of device data. At this point, issues with accessibility due to security implementations of later Android devices were encountered and documented. Successful extractions of each the devices were performed. One test device was subjected to an initial extraction, then a reset operation was performed. Following the hard reset, the device data was acquired again to determine if any location artifacts would be recoverable after the reset operation. All extracted data was then analyzed for location artifacts using various tools.

The analysis strategy began by identifying which apps had permissions to the device location. Permissions information is stored in the file title “packages.xml.” Some software utilities,

such as CelleBrite Physical Analyzer, will parse the permissions information from this file and report it in its list of installed apps. Once the apps with location permissions were identified, the associated application data for each was examined for location information. The examination followed much of the previously outlined strategy, focusing first on SQLite databases and geo-tagged media files. Further analysis involved a search for possible textual location information, including addresses, destinations, points of interest, MAC addresses, etc. This latter step was effected via keyword searches and manual review of application data.

In addition, the Google Location History was retrieved for each test device using the specialized Cloud Analyzer software. These operations were performed using both the credentials obtained from the devices themselves, as well as a manual entry of the credentials, to see if the method used had any effect on the results. A third step involved the collection of one day's worth of location history for one test device via the Google user account interface itself, accessible by logging into the account on the web. This data was retrieved to compare the collected cloud location history to the location data made available by Google to its users.

All recovered location information was compared to the location data recovered from the devices themselves, as well as the known locations recorded in the test session documentation. This was done with a view to confirming the accuracy trends noted in the previous research, ranging from highly accurate GNSS data to the variable reliability of cellular network location information. The data was also examined for trends involving environmental impact, in terms of indoor versus outdoor activity and area type (rural, suburban, or urban).

From there, reviews of system logs, application code, databases, and text-based application data (XML files) were undertaken to see if any determinations about sensor activity could be made.

For example, if a location artifact was found in the web browser cache and timestamped with a particular value, it would be useful to be able to establish if the GNSS sensor was active at this time. If sensor activity could be ascertained for that timeframe, it could help bolster the reliability of the location artifact.

Another consideration was whether the mere presence or absence of certain information could be used to infer which sensors were active. For example, do certain applications only function or log data when the GNSS sensors are in use? Or do they still cache information but with null values for certain metadata like accuracy and altitude values? Will devices still geo-tag photos and videos if the GNSS services are not enabled? If an obvious trend could be observed in regards to these questions, it may have implications for the future accuracy evaluation of certain types of location data recovered from devices.

CHAPTER FOUR: FINDINGS

Extraction Issues

Three of the devices were susceptible to physical extraction methods. One device, the OnePlus One A0001, running Android 5.0 (Lollipop), blocked the physical extraction since the device is running Lollipop and is not rooted. Furthermore, because of additional technical aspects of this device, all solutions for obtaining root access would require wiping the user data first. Thus, this device could only be analyzed logically via the ADB (Android Debugging Bridge) backup method.

The other Lollipop device, the Galaxy S5, did have a physical extraction solution that does not involve wiping the user data, but to test the impact of resetting a device to allow root access, the ADB backup method was used on this device first. Then, the device was used once to take three photos, reset, and a physical extraction was successfully performed. The resulting physical extraction was examined to determine if information previously recovered via the ADB backup could still be located after the reset operation. For instance, keyword searches were performed for known latitude and longitude values recovered via the analysis of the ADB backup. These searches were unsuccessful, though the data was known to be cached in SQLite databases.

Additional research into this apparent complication revealed that the SQLite databases in question store the latitude and longitude coordinates as “REAL” or “DOUBLE” (floating point) or “INTEGER” values. Therefore, a search for numeric strings consistent with coordinate values will not recover these artifacts, even if regular expressions or GREP techniques are used. A

potential solution involved attempts to carve out SQLite databases containing text terms like “lat REAL” and “latitude DOUBLE,” or similar variations. These search terms successfully returned hits on the actual databases containing the location artifacts, but no results of value were obtained when searching the unallocated space of the post-reset physical extraction. Searches of SQLite databases carved from the unallocated space were also negative.

Clearly, recovery of location data stored in SQLite databases after resetting a device presents significant challenges. This underscores the need to mitigate risk when confronted with devices that are not inherently supported for software-based physical extraction. In some cases, where encryption is not involved, hardware-based techniques may be a better solution. Though they may require more time and expense, they can assure access to all of the device content without running the risk of resetting the device.

Furthermore, as anticipated, certain applications were excluded from the ADB backup in the Lollipop device extractions. Notably, WhatsApp, Facebook Messenger, Chrome browser, and Google Maps application data was excluded from the backups of these devices. On a positive note, these items were listed as installed applications by the forensic tools. A review of the “localappstate.db” database confirms that the apps were installed on the devices, as well as their installation time. In a case where Facebook Messenger or Google location data was needed, but the extraction failed to retrieve it, the investigator may wish to turn to the next resource discussed: the cloud.

Cloud Data Recovery

The first notable feature of the Cloud Analyzer software’s “Google Location History” extraction function is the 30-day range limit on location data. Presumably, this is a result of Google’s own imposed limitations capping the retrieval of history information even by users and devices to one month at a time. By repeating the extraction process and selecting different ranges, though, multiple months’ worth of locations were recovered for each test device. The figure below shows the Cloud Analyzer interface with the 30-day range selection requirement.

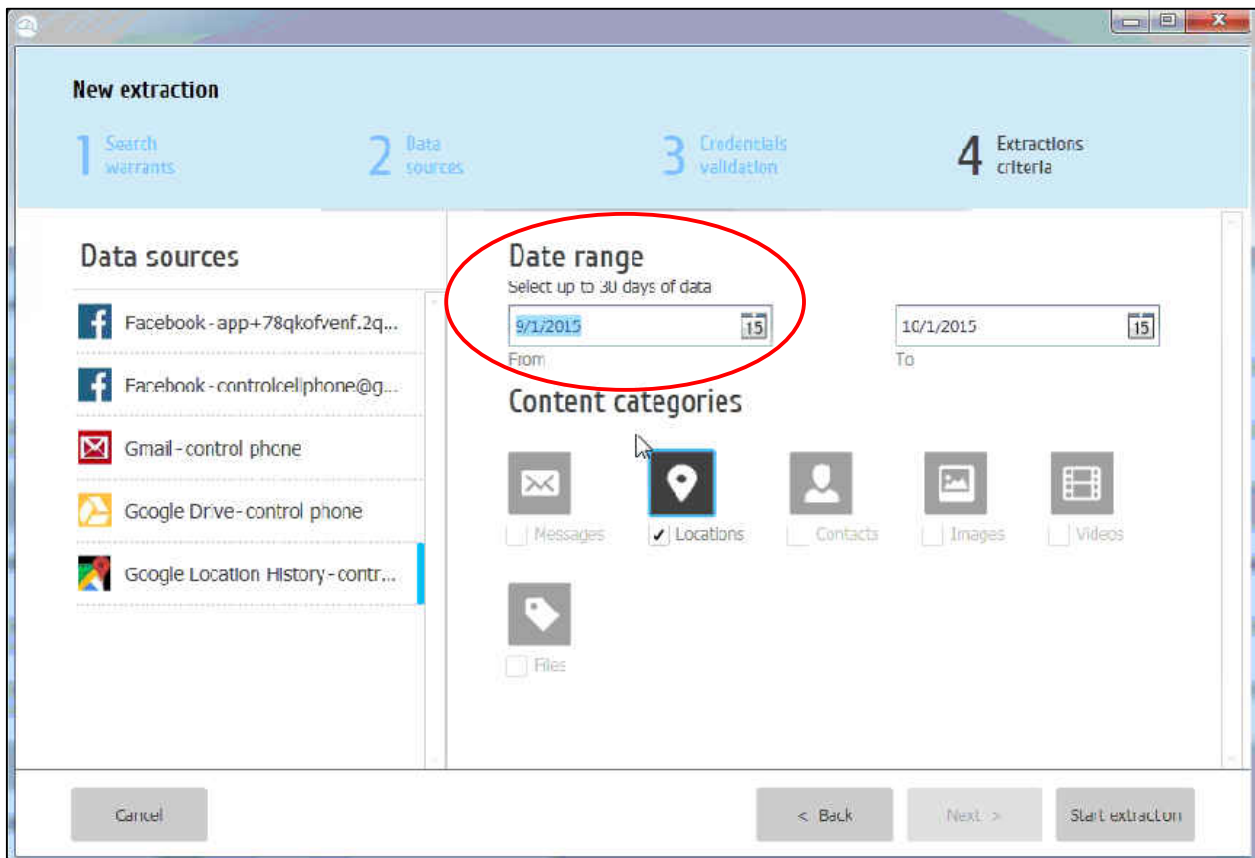


Figure 3 - Cloud Analyzer Location History Options (30-day range)

Two methods exist for retrieving Google Location History. Both were tested and successfully used. The first method involves the export of a specialized account package from the CelleBrite Physical Analyzer (P.A.) software. This account package contains all of the user account credentials parsed by the P. A. software, as well as the unique Android device identifier. By utilizing this information, the software is essentially able to present the request for the location history via Google's own API as though the device itself were retrieving the data. This offers a key advantage over the second method, in which examiners manually enter the account credentials. Manual entry generates a notification email from the provider, Google, to the account holder regarding a new login from an unrecognized device. Use of the account package (device credentials) does not produce this alert to the user. This distinction was confirmed in the tests performed for this study. A notification email was received when the manual credential entry method was used but not with the account package method. Note that account packages could not be created for the Lollipop ADB backup extractions, as the Google account credentials are not recovered via this extraction method. For such situations, account credentials would have to be obtained via the device owner or other source and entered manually.

Both methods yielded the same results, and the results were very impressive. Latitude and longitude coordinates for the device locations were retrieved with a frequency of roughly one location per minute that the devices were up and running. Another initial observation was that the retrieved coordinates were in decimal format, and the precision of all recovered coordinates appeared to be limited to the thousandths place. This could have implications regarding the accuracy of the cloud location data. However, the frequency with which the device reports its location to Google was surprising, and certainly supports the idea that Google does

indeed have a vast cache of user location history information that could potentially be of great value in criminal investigations. Table 5 illustrates the frequency with which Google location updates are recorded for the device, based on a sampling of test results. The results displayed were selected to optimize for visualization of any trends noted among the different environment types and sensor activity, so some sessions with identical environment and sensor parameters were excluded from the table though they displayed similar trends.

Table 5 - Average Cloud Location Frequency

Device	Number of Cloud Locations	Up Time (minutes)	Average Frequency (points/minute)	Environment (Sensors Active)
OnePlus One	22	25	0.88	Suburban, Indoor (Cell, WLAN, GNSS)
LG VS870	118	105	1.12	Suburban, Outdoor
OnePlus One	116	105	1.10	(Cell, WLAN, GNSS)
VS870	84	105	0.80	Suburban, Outdoor
OnePlus One	115	110	1.05	(Cell only)
S4 Mini	85	80	1.06	Urban, Outdoor
Galaxy S5	86	82	1.05	(Cell, WLAN, GNSS)
LG VS870	56	71	0.79	Suburban, Indoor
OnePlus One	63	74	0.85	(Cell only)
LG VS870	51	50	1.02	Rural, Indoor
OnePlus One	110	115	0.96	(Cell, WLAN, GNSS)
LG VS870	43	49	0.88	Rural, Outdoor
OnePlus One	39	34	1.15	(Cell, WLAN, GNSS)
LG VS870	78	80	0.98	Suburban, Indoor
OnePlus One	118	121	0.98	(Cell, WLAN)
S4 Mini	23	19	1.21	Suburban, Outdoor (Cell, GNSS)
LG VS870	27	26	1.04	Rural, Outdoor (Cell only)
Galaxy S5	0	10	0	Suburban, Indoor (WLAN only)
OnePlus One	0	60	0	Suburban, Outdoor (WLAN, GNSS)

Some notable observations from this data include the fact that the trend of roughly one location per minute seems to persist across environment types and regardless of which services are active, with one notable exception: when test devices were used with cellular services disabled, no cloud locations were captured. Interestingly, a fitness app also used in one such test session *did* cache some location data for the same timeframe. However, the phenomenon could be an aberration, given that it involved just two devices in as many test sessions. More testing would be needed to confirm if this trend holds.

Recovery of Device Location Data

Some location information was automatically parsed by the forensic software tools used to analyze the data from the test devices. This included Facebook Messenger, WhatsApp, and Viber chat message locations, some Google Maps and Waze navigation data, and geo-tagged media files. These artifacts are essentially the “low-hanging fruit,” as it were, so not much additional strategy is required to identify them. However, other data of possible value poses greater challenges.

App and File Review

A review of the extracted data confirmed that the location cache files identified in previous studies, “cache.cell” and “cache.wifi,” were not recovered from any of the test devices, as expected based on the ages of the test devices. The file review then shifted to the next most obvious targets: geo-tagged photos and SQLite databases. In terms of geo-tagged photos, although they were identified by the analysis software immediately, the results were somewhat

muddled. For example, one might expect a trend in which geo-tagged photos would only be recovered from instances when the GNSS services were active. However, in some instances where GNSS services were noted as enabled and photos were captured, no geo-tagged photos were recovered. Two of the test devices, the Galaxy S5 and the Galaxy S4 Mini, recorded no geo-tagged photos, even though they were configured to do so and used to capture images. Furthermore, in two sessions on two separate devices, geo-tagged photos were recovered despite the fact that no GNSS services were enabled at the time of the session. Thus, an empirical basis for inferring sensor status from the mere presence or absence of geo-tag metadata in recovered images could not be determined.

Contributing further to the ambiguity, there was also no discernable pattern regarding the geo-tagged images captured at times when test session notes indicate the GNSS services were disabled. For example, in one instance, the test phone was indoors with cellular service only enabled, and the error was within roughly 130 meters from actual location. In another instance, however, the same phone was indoors with cellular and WLAN enabled, and the error was around 30 meters. A second test phone used in the same area as the first in the same outdoor, cellular-only test session appears to have geo-tagged the photo with a location over a kilometer away from the actual site of the photo.

The camera apps for all devices were configured during device setup to geo-tag photos, so a settings issue in the camera should not be responsible for the absence of geo-tagged photos from sessions in which the GNSS services were enabled. Weather conditions during such sessions were noted as being partly cloudy or clear. One possibly notable factor could be that other location-sensitive applications were also used in the sessions in which geo-tagged images

were captured in spite of the disabled GNSS services. Such applications, like Life360, a location-sharing app, could have permissions to change the WLAN connection state, for example. This could potentially activate another resource for devices to obtain their location, though it is unknown if such a phenomenon resulted in the results observed here.

Identifying Location-Permitted Apps

Having revisited the topic of permissions, the focus now shifts to apps with access to the device location. By examining the file titled “packages.xml,” various apps were identified which have access to either coarse or fine location, or more often both. The “packages.xml” file is a simple XML text file in which details about the installed applications are stored, including the app permissions. The permissions appear as a list and are organized by app. The figure below provides a snapshot of the permissions list for the RunKeeper fitness app. Note that this particular app only has access to the device “FINE” location. Based on the previous discussion regarding the maximum accuracies of both “COARSE” and “FINE” location permissions, this could indicate that the RunKeeper app data, if recovered, is likely to be quite accurate.

This idea is corroborated by observations made during test sessions regarding the RunKeeper app, as well as several other apps. Both the RunKeeper and MapMyWalk apps, for example, were noted to display a distance of “0.0” upon conclusion of workouts in which GNSS sensors were not enabled. A review of the associated databases for these apps confirms that no location data was cached for those particular sessions, just start and end times. In addition, the Waze navigation app insisted on “High Accuracy” mode being enabled by the user (including GNSS sensors) before performing navigation functions. Intuitively, this makes sense for the apps’ various functions, as navigation instructions would not be helpful if they were not finely

attuned to the device's actual location, and fitness apps aim to provide users a track of their workout waypoints to facilitate measurements of improvement over successive workouts. For this reason, an investigator may wish to test a particular application of possible interest on a control device for evidence of this type of app-specific settings requirement. Doing so may allow an examiner to gain some insight into the reliability of similar data recovered from the evidence item.

```
- <package name="com.fitnesskeeper.runkeeper.pro" codePath="/data/app/com.fitnesskeeper.pro" flags="0" ft="14dc9831a90" it="14d9b376ae0" ut="14dc98331a3" version="314" userId="1" />
- <sigs count="1">
  <cert index="26"
    key="308201ff30820168a00302010202044bc78966300d06092a864886f70d010" />
</sigs>
- <perms>
  <item name="android.permission.READ_EXTERNAL_STORAGE" />
  <item name="android.permission.USE_CREDENTIALS" />
  <item name="android.permission.WRITE_EXTERNAL_STORAGE" />
  <item name="android.permission.READ_CONTACTS" />
  <item name="android.permission.GET_ACCOUNTS" />
  <item name="com.fitnesskeeper.runkeeper.pro.permission.C2D_MESSAGE" />
  <item name="com.google.android.c2dm.permission.RECEIVE" />
  <item name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
  <item name="android.permission.RECEIVE_BOOT_COMPLETED" />
  <item name="com.android.vending.BILLING" />
  <item name="android.permission.INTERNET" />
  <item name="android.permission.BLUETOOTH" />
  <item name="android.permission.ACCESS_FINE_LOCATION" />
  <item name="android.permission.BLUETOOTH_ADMIN" />
  <item name="android.permission.WAKE_LOCK" />
  <item name="android.permission.ACCESS_NETWORK_STATE" />
  <item name="android.permission.RECORD_AUDIO" />
</perms>
</package>
```

Figure 4 - RunKeeper Permissions from “Packages.xml” with “ACCESS_FINE_LOCATION”

Returning to the topic of permissions, a couple of other comments on the “packages.xml” file are warranted. First, the Physical Analyzer software does parse out a list of installed applications, including a summary of their permissions. However, it does not go into detail about coarse versus fine locations and so on. A quick “Find” search of the “packages.xml” file in

Internet Explorer for the terms “ACCESS_FINE_LOCATION” and “ACCESS_COARSE_LOCATION” will enable an investigator to identify each app that could potentially store location information based on its permissions. The app name and path appear above the list of its associated permissions, as shown in Figure 4. It is also important to note that the “packages.xml” file appears to have been excluded from the Android backup for the Lollipop test devices.

Given that the apps used in the test sessions were specifically chosen because of their location permissions, the focus of this study shifted quickly to reviewing the SQLite databases associated with each app. In general, the examination of app databases quickly made clear that location-sharing and fitness apps seem to cache the most data, frequently with high update rates and accuracy. They also seem most likely to cache other location metadata, like accuracy and altitude. Navigation and mapping apps seemed to do less logging of actual track points, focusing more on search results and recent destinations, but Waze did have a database named “tts.db” that contains timestamped turn-by-turn directions. These could certainly play a key role in reconstructing an individual’s activity, though not necessarily with the minutiae of a true tracking device.

Many databases were found to store location-related content. The trick was in determining the nature of the cached information. Was it consistent with the device’s actual location as noted in the test session documentation? Or was it based more on searched locations or destinations? What metadata was cached within the databases? Table 6 below details twenty-six of the recovered databases of possible interest by filename, along with a brief description of their content. The databases are presented with their associated app, with the apps divided into

seven distinct categories. A detailed description of SQLite database analysis follows, highlighting specific databases of interest that were recovered from the test devices.

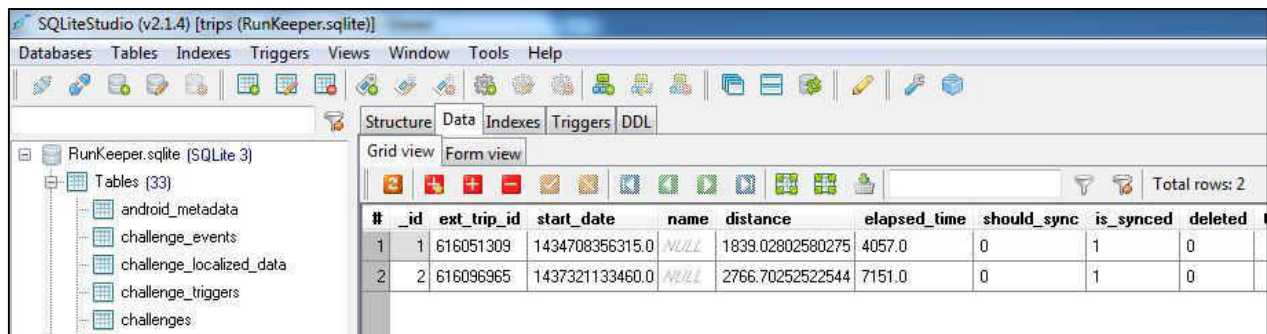
Table 6 - Databases of Interest

App Category	Database Name	Source App	Content
Navigation/ Mapping	gmm_storage.db	Google Maps	searched locations, directions. Stored in BLOB data. Manual review or strings/keyword searches required.
	suggestions.db	Google Earth	searched locations (addresses or points of interest), with timestamps
	tts.db	Waze	turn-by-turn directions (transcript), with timestamps
	user.db	Waze	recent locations with lat/long and timestamps
Fitness	mytracks.db	MyTracks	Workout session history, timestamped trackpoints, with metadata (accuracy, etc.)
	RunKeeper.sqlite	RunKeeper	Workout session history, timestamped trackpoints, with metadata (accuracy, etc.)
	workout.db	Map My Walk	Workout session history, timestamped trackpoints, with metadata (accuracy, etc.)
Location Sharing	360LocationDB	Life360, FriendLocator	Location history with timestamps and metadata (accuracy, altitude, etc.)
	messaging.db	Life360, FriendLocator	Chat messages with lat/long and timestamps
	nc.db	Life360	Notifications with extra text metadata (timestamps and lat/long)
	dumpLogsDatabase	FriendLocator	Detailed activity log, with connections, request details, and locations info, timestamps (lengthy text format, manual recovery or keyword searches needed)

App Category	Database Name	Source App	Content
Location Sharing	fsq.db	Swarm	Recent locations with lat/long and timestamps
Chat	threads_db2	Facebook Messenger	chat messages with user location (lat/long) and timestamps
	viber_messages	Viber	chat messages with user location (lat/long) and timestamps
	msgstore.db	WhatsApp	chat messages with user location (lat/long) and timestamps
	naver_line	LINE	chat messages with user location (lat/long) and timestamps
Leisure	scout.db	FieldTrip	Locations of viewed points of interest with view timestamp
Weather	oneweather.db	OneWeather	saved locations, including lat/long, timestamps of last hit
	weather.db	GO Weather	Recent locations, including lat/long and timestamp
	forecast_accu.db	Accuweather	saved location to display in widget (not necessarily current location)
Other	ContextLog_0.db	Pre-installed Samsung app/feature	Tracks app launches/sessions with timestamps and duration of activity
	event	Pre-installed Amazon shopping app	WLAN/cellular data usage stats with timestamp
	herrevad	Google Mobile Services	WLAN network history with BSSID (MAC address) and timestamps
	https_www.google.com_0.localstorage	web browser	searched terms, lat/long, with timestamps
	NetworkUsage.db	Google Mobile Services	some WLAN/cellular data usage stats with timestamp
	locdatabase	Android Location Tracker	logs with lat/long and timestamps

SQLite Database Analysis – Fitness App Example

SQLite databases store content in tables, which in turn contain records associated with particular rows and columns. Often, content of interest is stored in multiple tables or in a format that is less intuitive to the human reader, such as timestamps stored in UNIX milliseconds format rather than the typical month/day/year, hour/minute/seconds format we are used to seeing. Many utilities exist for viewing and extracting data from these files. In this case, SQLite Studio was used to extract information of interest from the recovered databases. Custom queries were created to retrieve the relevant location content. To illustrate the method used, the following figures depict an example database, the RunKeeper app’s “RunKeeper.sqlite” database, as well as the query and its results. Query results were output into Excel spreadsheets to facilitate further data review, comparison, filtering, and the like.



The screenshot shows the SQLiteStudio interface for the RunKeeper.sqlite database. The left pane displays the database structure with 33 tables, including android_metadata, challenge_events, challenge_localized_data, challenge_triggers, and challenges. The main pane shows the 'trips' table in grid view, displaying two rows of data. The table has columns: #, id, ext_trip_id, start_date, name, distance, elapsed_time, should_sync, is_synced, and deleted.

#	id	ext_trip_id	start_date	name	distance	elapsed_time	should_sync	is_synced	deleted
1	1	616051309	1434708356315.0	NULL	1839.02802580275	4057.0	0	1	0
2	2	616096965	1437321133460.0	NULL	2766.70252522544	7151.0	0	1	0

Figure 5 - RunKeeper.sqlite database, *trips* table

SQLiteStudio (v2.1.4) [points (RunKeeper.sqlite)]

Databases Tables Indexes Triggers Views Window Tools Help

Structure Data Indexes Triggers DDL

Grid view Form view

Total rows: 439

#	_id	ext_point_id	trip_id	has_been_sent	latitude	longitude	altitude	time_at_point	time_interval_at_point
193	632	NULL	2	0	35.106078	-77.04444906	8.0	0	0.0
194	633	NULL	2	0	35.10607838	-77.04444589	7.85714285714286	0	0.0
195	634	NULL	2	0	35.10602588	-77.04452331	7.75	0	74.0
196	635	NULL	2	0	35.10602531	-77.04462476	7.66666666666667	0	83.0
197	636	NULL	2	0	35.10607297	-77.04471048	7.6	0	91.0
198	637	NULL	2	0	35.10615587	-77.04470137	7.54545454545455	0	103.0
199	638	NULL	2	0	35.10623894	-77.04467982	7.27272727272727	0	113.0
200	639	NULL	2	0	35.10624824	-77.04456318	7.0	0	121.0
201	640	NULL	2	0	35.10627922	-77.04446813	7.0	0	131.0
202	641	NULL	2	0	35.10630623	-77.04435941	7.0	0	141.0
203	642	NULL	2	0	35.10638082	-77.04429506	7.0	0	149.0
204	643	NULL	2	0	35.10646234	-77.04427552	7.0	0	158.0
205	644	NULL	2	0	35.10654981	-77.04427734	7.0	0	166.0
206	645	NULL	2	0	35.10663916	-77.04428466	7.0	0	175.0
207	646	NULL	2	0	35.10672203	-77.04427229	7.0	0	183.0

Figure 6 - RunKeeper.sqlite database, *points* table

```

SELECT points.trip_id,
trips.start_date,
datetime( ( trips.start_date + points.time_interval_at_point * 1000 ) / 1000, 'unixepoch' ) AS [Converted Time],
points.latitude,
points.longitude,
points.altitude,
points.time_interval_at_point,
points.speed_from_last_point,
points.distance_from_last_point,
points.point_type,
points.accuracy,
points.distance_at_point
FROM
points,
trips
WHERE points.trip_id = trips._id
ORDER BY points._id ASC;

```

Figure 7 - RunKeeper.sqlite database, query converting timestamps and combining content from *trips* and *points* tables

#	trip_id	start_date	Converted Time (UTC)	latitude	longitude	altitude	time_interval_at_point
193	2	1437321133460.0	2015-07-19 15:52:13	35.106078	-77.04444986	8.0	0.0
194	2	1437321133460.0	2015-07-19 15:52:13	35.10607838	-77.04444589	7.857142857142857	0.0
195	2	1437321133460.0	2015-07-19 15:53:27	35.10602588	-77.04452331	7.75	74.0
196	2	1437321133460.0	2015-07-19 15:53:36	35.10602531	-77.04462476	7.666666666666667	83.0
197	2	1437321133460.0	2015-07-19 15:53:44	35.10607297	-77.04471048	7.6	91.0
198	2	1437321133460.0	2015-07-19 15:53:56	35.10615587	-77.04470137	7.545454545454546	103.0
199	2	1437321133460.0	2015-07-19 15:54:06	35.10623894	-77.04467982	7.272727272727273	113.0
200	2	1437321133460.0	2015-07-19 15:54:14	35.10624824	-77.04456318	7.0	121.0
201	2	1437321133460.0	2015-07-19 15:54:24	35.10627922	-77.04446813	7.0	131.0

Figure 8 - RunKeeper.sqlite database, query results

This database offers a few noteworthy observations. First, the precision of the latitude and longitude values extends well beyond the cloud data’s thousandths-place level, all the way to eight places after the decimal, or to one hundred-millionths place. In addition, the timestamps are also incredibly precise, stored in UNIX epoch milliseconds format. This is typical of many Android applications and was noted in the majority of the examined databases. Furthermore, the update intervals, as noted in the “time_interval_at_point” column, are quite frequent, occurring multiple times per minute. This was also noted to be a common trait among databases associated with the other tested fitness apps, MyTracks and MapMyWalk. The other two fitness apps also exhibited the same behavior in test sessions, failing to report a distance or cache workout waypoints when the GNSS service was not active. If these observations are any indication, recovered fitness app data is likely to be quite precise to the actual device location, quite accurate due to GNSS sensor use requirement, and frequently updated, making it a potentially valuable resource in an investigation in which it is available.

Chat and Location-Sharing Apps

The chat and location-sharing apps also seemed to cache the actual device location with significant precision. However, the chat apps seem more likely to be parsed by the commercial forensic software tools, such as Internet Evidence Finder and CelleBrite Physical Analyzer. The following tables display examples of the content obtained from the various chat applications which cached location data, as recovered by the commercial software tools used in this study.

Table 7 - Selection of content recovered from Viber app's "viber_messages" database as reported by Internet Evidence Finder

Sender	Recipient(s)	Message Sent Date/Time - (UTC) (MM/dd/yyyy)	Message	Message Status	Latitude	Longitude
TestPhone Gsmone		09/16/2015 01:55:16 PM	Orlando today!	Received	27.8014691	-82.3025061
-Not Found-	, TestPhone Gsmone	09/16/2015 01:55:35 PM	Sho nuff!	Sent / Delivered	27.8014233	-82.3024485
-Not Found-	, TestPhone Gsmone	09/16/2015 01:56:18 PM	No location now?	Sent / Delivered	n/a	n/a
TestPhone Gsmone		09/16/2015 01:56:28 PM	We'll see	Received	27.8014691	-82.3025061
-Not Found-	, TestPhone Gsmone	09/16/2015 01:57:04 PM	How bout now? I re-enabled it	Sent / Delivered	27.8014181	-82.3024472
TestPhone Gsmone		09/16/2015 01:57:11 PM	Cool	Received	27.8014691	-82.3025061
TestPhone Gsmone		09/16/2015 01:57:20 PM	Mine is always enabled	Received	27.8014691	-82.3025061
-Not Found-	, TestPhone Gsmone	09/16/2015 01:57:25 PM	Nice	Sent / Delivered	27.8014181	-82.3024472

Table 8 - Selection of content recovered from Facebook Messenger app's "threads_db2" database as reported by CelleBrite Physical Analyzer

From	Body	Location	Timestamp: Date	Timestamp: Time
100009025428140 Testphone Gsm	1 - This message sent from A0001 to VS870 at 4211 N. Lois Ave. Tampa. Active services- cellular, WiFi, and gnss	(27.977825, -82.513403)	5/27/2015	5/27/2015 10:04:33 PM(UTC+0)
100009025428140 Testphone Gsm	2 - This message sent from A0001 to VS870 at 4211 N. Lois Ave. Tampa. Active services- cellular, WiFi, and gnss	(27.977825, -82.513403)	5/27/2015	5/27/2015 10:05:22 PM(UTC+0)
100009025428140 Testphone Gsm	3 - This message sent from A0001 to VS870 at 4211 N. Lois Ave. Tampa. Active services- cellular, WiFi, and gnss	(27.977825, -82.513403)	5/27/2015	5/27/2015 10:06:08 PM(UTC+0)

As illustrated in the tables, the cached latitude and longitude values are quite precise, with Viber recording up to the ten-millionths place and Facebook Messenger up to the millionths. However, although these location coordinates are quite precise, they are only cached when a message is sent. Locations can also be recovered for the remote conversation partner, not just the local device from which the data was retrieved, as seen as in the Viber messages above. No location data was recovered for messages sent during test sessions when the GNSS services were disabled, interestingly. Also, no locations were stored by the other two chat apps tested, WhatsApp and LINE, even though the apps were configured to share locations with chat conversation partners.

In general, the location-sharing apps also report the device's current location, and they do so with great precision. Although no SQLite databases containing location data were recovered for the Glympse app, several source databases were identified for the other applications. Life360 and Locate My Friends are apps from the same developer, generating databases with the same names, just under different directory paths. Swarm is associated with Foursquare, so its primary database is named and formatted similarly, as well. One interesting finding from the review of

these databases is that the precision of the cached coordinates seemed to vary, with coordinates associated with messages sent via the apps' interfaces reported with an even greater number of values after the decimal point. It's also important to note that the coordinates appeared rounded to the nearest degree when viewed within the CelleBrite Physical Analyzer's internal SQLite database view. The following figures show the precision with which the Locate My Friends app (and its sister application, Life360) records latitude and longitude values, as viewed from within the Physical Analyzer and SQLite Studio interfaces, for messages and cached locations.

	type	content	created_at	failed_to_send	sent	dismissed	read	deleted	first	has_location	location_latitude	location_longitude
message	a373c8f	TEXT Hello	1442414116	0	1	0	1	0	1	1	27	-82
message_participant	c3d8e43	TEXT Hmm	1442414160	0	1	0	0	0	0	1	27	-82
thread	a373c8f	TEXT Good grief	1442414174	0	1	0	0	0	0	1	27	-82
thread_participant	c3d8e43	TEXT Too many ways to message	1442414186	0	1	0	0	0	0	1	27	-82
thread_participant	a373c8f	TEXT Yep	1442414196	0	1	0	0	0	0	1	27	-82

Figure 9 - Contents of the Locate My Friends app's "messaging.db" database as viewed within Physical Analyzer (coordinates rounded to nearest degree)

	type	content	created_at	failed_to_send	sent	dismissed	read	deleted	first	has_location	location_latitude	location_longitude
message	373c8f	TEXT Hello	1442414116	0	1	0	1	0	1	1	27.801657177413	-82.302531568476
message_participant	3d8e43	TEXT Hmm	1442414160	0	1	0	0	0	0	1	27.801467326635	-82.302480309631
thread	373c8f	TEXT Good grief	1442414174	0	1	0	0	0	0	1	27.8014598	-82.3024735
thread_participant	3d8e43	TEXT Too many ways to message	1442414186	0	1	0	0	0	0	1	27.801467326635	-82.302480309631
thread_participant	373c8f	TEXT Yep	1442414196	0	1	0	0	0	0	1	27.8014598	-82.3024735

Figure 10 - Contents of the Locate My Friends app's "messaging.db" database as viewed within SQLite Studio (more precise coordinates)

#	time	Converted Time (UTC)	lat	lon	accuracy	speed	altitude	bearing	provide
1	1441759010358	2015-09-09 00:36:50	27.801481	-82.302515	11.09899977111816	0.0	0.0	0.0	fused
2	1441760033898	2015-09-09 00:53:53	27.8014558	-82.3025134	34.93199920654297	0.0	0.0	0.0	fused
3	1441760624127	2015-09-09 01:03:44	27.8017011	-82.3024733	3.9000000953674316	0.0	-4.0	0.0	fused
4	1442410570682	2015-09-16 13:36:10	27.8014573	-82.3025029	20.20800018310547	0.0	0.0	0.0	fused
5	1442411432751	2015-09-16 13:50:32	27.8014771	-82.3024793	39.0	0.0	0.0	0.0	fused
6	1442413236440	2015-09-16 14:20:36	27.8014522	-82.3024968	10.0	0.0	0.0	240.0	fused
7	1442414865632	2015-09-16 14:47:45	27.8014598	-82.3024735	10.0	0.0	0.0	0.0	fused
8	1442415755579	2015-09-16 15:02:35	27.8015202	-82.3025031	28.0	0.0	0.0	0.0	fused
9	1442418459046	2015-09-16 15:47:39	27.801477	-82.3024803	25.0	0.0	0.0	0.0	fused
10	1442606098592	2015-09-18 19:54:58	27.9777954	-82.5133069	15.88599967956543	0.0	0.0	0.0	fused

Figure 11 - Contents of the Locate My Friends app's "360LocationDB" database, SQLite query results. High precision coordinates noted, but less precise than those associated with messages found in "messaging.db."

These figures demonstrate the higher precision ascribed to coordinates associated with the app's chat messages, as recorded in the "messages.db" database, versus the cached locations stored in the "360Location" database. The message coordinates may even be too precise to be genuine, based on the previous discussion of coordinate precision levels in consumer-grade devices. In addition, the timestamps reflect fairly frequent update intervals of roughly fifteen minutes for the stored location points, while the chat message coordinates are dependent on the sending of messages. It is also notable that the "provider" is noted as "fused" in the "360LocationDB" database, indicating that the Google location services' Fused Location Provider API is used by this application.

Other findings from the analysis of the location sharing apps' databases included the obliteration of older records. Test session documentation shows these apps were used in multiple sessions prior to the earliest records recovered from the extracted databases. This was consistent

among the Locate My Friends, Life360, and Swarm apps, across all devices. Also, each of the aforementioned apps cached locations for sessions in which the GNSS services were noted as disabled in the test session documentation. In short, these apps cached very precise data, with moderate or conditional update intervals, appear not to store locations indefinitely, and will store coordinates even when GNSS sensors are not enabled by the user.

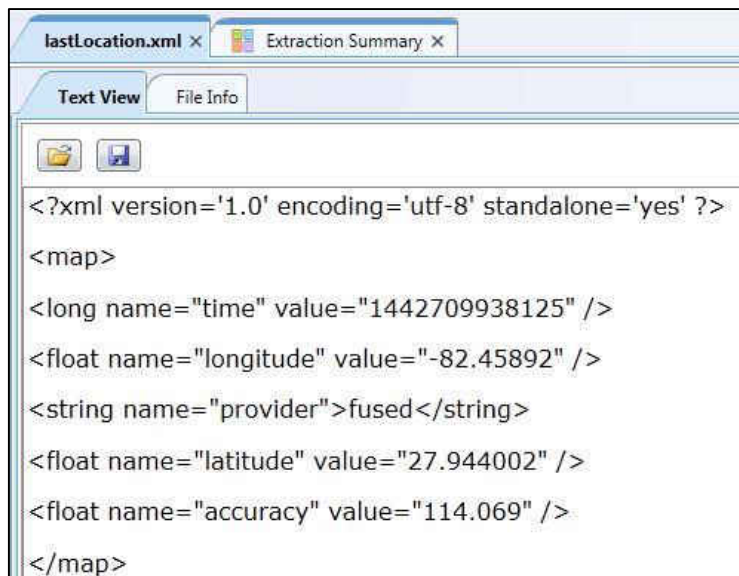
Leisure and Navigation Apps

Alternatively, the leisure apps tested seemed more likely to record searched destinations, rather than the actual device location. As an example, the Foursquare app caches recently viewed venues but does not specify the location of the device at the time the search was executed. Empirical use of these applications suggests that the viewed venues will generally be nearby points of interest, as related to the device location at the time of the search. However, in a retroactive analysis situation, this would be an inference and is not documented directly in the app's databases. It could be corroborated with other sources, however, such as Google Location History or carrier cell tower location records. The table below displays the content retrieved from the Foursquare app's "fsq.db" database, as recovered from the OnePlus One A0001 test device.

Table 9 - Content of Foursquare's "fsq.db" database, extracted via SQLite query

last_viewed	Converted Time (UTC)	name	loc_lat	loc_long	loc_address	loc_city
1442418216	9/16/2015 15:43	East Coast Pizza	27.79071808	-82.34282684	13340 Lincoln Rd	Riverview
1442606556	9/18/2015 20:02	Starbucks	27.98117065	-82.48847961	2720 W Dr Martin Luther King Jr Blvd	Tampa
1442606575	9/18/2015 20:02	Brio Tuscan Grille	27.9652195	-82.52071381	2223 N West Shore Blvd	Tampa
1442606589	9/18/2015 20:03	Cigar City Brewing	27.95913696	-82.50926971	3924 W Spruce St	Tampa

Other leisure apps stored location content in text-based XML files, as did the Field Trip app. Two files titled “lastLocation.xml” and “lastNotification.xml” were recovered from the Field Trip app’s directory. These files contained precise latitude and longitude values with timestamps but they were stored only for the most recent activity. The cached coordinates appear to relate to the actual device location and were consistent with the actual test session location, within 100 meters. Another noteworthy finding is that these coordinates were cached at times when the GNSS services were disabled on the test device, the LG VS870, in this case.

A screenshot of a text editor window. The title bar shows two tabs: "lastLocation.xml" and "Extraction Summary". The window has two tabs: "Text View" (selected) and "File Info". Below the tabs are icons for folder, save, and print. The main text area contains the following XML code:

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<long name="time" value="1442709938125" />
<float name="longitude" value="-82.45892" />
<string name="provider">fused</string>
<float name="latitude" value="27.944002" />
<float name="accuracy" value="114.069" />
</map>
```

Figure 12 - Content of the Field Trip app's "lastLocation.xml" file


```

<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<string name="provider">fused</string>
<float name="longitude" value="-82.4604" />
<long name="time" value="1442706187370" />
<float name="latitude" value="27.948954" />
<float name="accuracy" value="18.659" />
<long name="notificationTime" value="1442619859519" />
</map>

```

Figure 13 - Content of the Field Trip app's "lastNotification.xml" file

Falling somewhere in between the fitness and leisure apps, the navigation apps store both searched locations and in some instances, the actual device location at the time of the search. Two databases among the Waze navigation app data demonstrate this dichotomy. The “user.db” database contains destinations to which the user has navigated via the Waze app. The “tts.db” offers a rather less conventional twist on pinpointing the device location, by transcribing the turn-by-turn directions with timestamps. Extracted content from both databases is displayed in the tables below to illustrate the different storage strategies, highlighting data for the same navigation session from each database.

Table 10- Destination data extracted from the Waze app's "user.db" database, *RECENTS* and *PLACES* tables, using SQLite query

name	city	state	longitude	latitude	created_time	Converted Created Time (UTC)
The Proper Pie Company	Davenport	FL	-81.638595	28.214551	1442438167	9/16/2015 21:16:07

Table 11 - Turn-by-turn directions recovered from Waze app's "tts.db" database, using SQLite query

text	path	Converted Time (UTC)
145 Ridge Center Drive	database//Jane//1442438170-504860-238.tts	9/16/2015 21:16:11
The Proper Pie Company	database//Jane//1442438170-504890-239.tts	9/16/2015 21:16:11
Let's take SR-417 S, and I-4 W	database//Jane//1442438170-504890-240.tts	9/16/2015 21:16:11
exit to Exit 3: Osceola Pkwy (toll)	database//Jane//1442438170-517373-241.tts	9/16/2015 21:16:11
exit right to Exit 3: Osceola Pkwy (toll)	database//Jane//1442438170-517373-242.tts	9/16/2015 21:16:11
stay to the right to Osceola Pkwy	database//Jane//1442438170-517404-243.tts	9/16/2015 21:16:11
stay to the left to I-4 W / Tampa	database//Jane//1442438170-517404-244.tts	9/16/2015 21:16:11
turn left on Citrus Ridge Dr	database//Jane//1442438170-529093-245.tts	9/16/2015 21:16:11
turn left on Majesty Dr	database//Jane//1442438170-529093-246.tts	9/16/2015 21:16:11
then turn left	database//Jane//1442438170-529124-247.tts	9/16/2015 21:16:11
then turn left on Majesty Dr	database//Jane//1442438170-529124-248.tts	9/16/2015 21:16:11
you'll arrive at The Proper Pie Company	database//Jane//1442438170-531260-249.tts	9/16/2015 21:16:11

Clearly, there is some ambiguity regarding the timestamps of the turn-by-turn directions, with multiple instructions timestamped identically. However, they do show evidence of a navigation route request, which may be indicative of travel or intent to travel by the user. Digging a little deeper, we see that Waze is actually caching the original location and timestamp of the search in a separate text file, titled "waze_log.txt." This file actually does contain the device's precise latitude and longitude value at the time the navigation request was initiated, as well as the coordinates of the destination. Figure 14 shows the relevant excerpt from this log file, with the timestamp (blue), origin coordinates (yellow), and destination coordinates (green)

highlighted. The recorded data is consistent with test session activity, and since this app's navigation function requires the GNSS service, the cached location is quite accurate.

```
[17:16:07.696 Warning] First routing id: 1442438168 [navigate_route_trans.c:381
(navigate_route_init_context)]
[17:16:07.696 Warning]
UID,842961612,ChBWNTBHNnNYa2xZN2hvd3RBEJK3568FGAwiA3VzYSjMpfqRAw,203
At,-81.301666,28.370980,-0.000006,257,3,53135374,53135286
RoutingRequest,1442438168,3,-10,1,-1,1000,-81301679,28371032,-1,53135374,53135286,SR-417
S,F,-81638595,2821455,-1,-1,-1,Ridge Center
Drive,T,T,T,26,1,F,2,T,3,F,4,T,5,F,6,T,7,T,8,T,10,F,12,F,13,F,16,T,32,T,0,145,Davenport,FL,62167,0,
F,-1,4,257,-1,-1,-1,-1,-
81301679,28371032,257,F,2,twitter,0,facebook,0,0,googlePlaces.ChIJNUVHYI5w3YgRUB2XohCuq
bE,, [RealtimNet.c:3902 (RTNet_RequestRoute)]
```

Figure 14 - Navigation request from Waze app's "waze_log.txt" text file

Browser, Weather Apps, and Games

In addition, some less intuitive sources of location data include files associated with the web browser, weather apps, and ad-supported games. The web browser stores “localstorage” databases containing website-specific content cached for later visits, named with the website URL with the “.localstorage” extension. Other apps can store these databases within their own directories, as well, but the web browser’s collection are discussed here. The browser’s “localstorage” databases are specific to the particular website visited and can contain probative user-generated information. In the case of the www.google.com website’s “localstorage” database, this can include search terms and location data, as shown in the figure below.

The screenshot shows a web browser's local storage database view for the URL "https://www.google.com". The database contains a table named "ItemTable (37)" with two columns: "key" and "value". The table contains 17 rows of data, including search history and location coordinates.

key	value
msuggest.history.n.ans	
msuggest.history.n.history	uswnt
msuggest.history..ans	
msuggest.history..history	mullet publix sushi target walmart starbucks ucf library
devloc::swml.location	"Ruskin, FL"
devloc::swml.location.isdev	"1"
devloc::swml.location.ts	"1442677187528"
devloc::web.rgc.en.0.acc	21
devloc::web.rgc.en.0.last	1442677187515
devloc::web.rgc.en.0.lat	27.72073115
devloc::web.rgc.en.0.lon	-82.43370627
devloc::web.rgc.en.0.rgc	"Ruskin, FL"
devloc::web.rgc.en.1.acc	4704
devloc::web.rgc.en.1.last	1442674949052
devloc::web.rgc.en.1.lat	27.6956312
devloc::web.rgc.en.1.lon	-82.2123443
devloc::web.rgc.en.1.rgc	"Wimauma, FL"

Figure 15 - Content of web browser's "https://www.google.com_0.localstorage" database, showing search terms, location coordinates, and timestamps

The content is stored as BLOB data, and its organization makes the recovery of the data a bit cumbersome, but the data recovered from this database was consistent with the test session activity. This database was only recovered for the two non-Lollipop devices for which physical extraction without root privileges was supported, but in both instances, the cached coordinates had timestamps associated with test sessions in which the GNSS services were enabled, and their precision was at least millionths-place level. The data is only recorded when a user actually conducts a search via Google.

Weather apps also appeared to cache some locations, although there may be some ambiguity regarding whether they were the actual device location or just a user-requested

location. The OneWeather app did have one database that cached apparent device locations, but the precision was fairly low, only one hundredths-place level. Alternatively, the AccuWeather's "forecast_accu.db" database stores more precise coordinates, but they may not be associated with the device's actual location. Fortunately, in this case, the database actually flags whether or not the stored information is the device's actual current location. The following tables display the varying types of location information recovered from the tested weather apps.

Table 12 - OneWeather app's "oneweather.db" cached locations, from *geocodes* table, showing device location history

city	state	country	lat	lng	lastHit Converted Timestamp (UTC)	hits
Riverview	FL	US	27.8	-82.3	9/16/2015 15:11:24	4
Tampa	FL	US	28.13	-82.38	6/6/2015 23:17:10	1
Tampa	FL	US	27.98	-82.51	6/9/2015 17:26:24	1
Tampa	FL	US	28.11	-82.37	6/9/2015 23:24:27	1
Tampa	FL	US	27.98	-82.52	6/10/2015 13:57:22	1
Durham	NC	US	35.94	-78.92	7/14/2015 15:41:23	3
Hamptonville	NC	US	36.07	-80.81	7/15/2015 11:58:59	3
Hamptonville	NC	US	36.05	-80.79	7/15/2015 15:01:54	1
Goldsboro	NC	US	35.34	-77.9	7/17/2015 14:46:33	1
New Bern	NC	US	35.14	-76.97	7/17/2015 15:52:07	1
Tampa	FL	US	28.14	-82.33	9/5/2015 14:04:15	1
Wesley Chapel	FL	US	28.19	-82.35	9/5/2015 15:09:51	1
Wimauma	FL	US	27.76	-82.26	9/19/2015 14:18:41	1
Tampa	FL	US	27.95	-82.46	9/19/2015 23:40:47	1

Table 13 - AccuWeather app's "accu_forecast.db" database content showing stored user-configured location, not actual device location

current_city_flag	Converted Timestamp (UTC)	city	country	lat	lon
FALSE	10/2/2015 11:16:04	Bremen	Germany	53.07561	8.80934

Venturing beyond database review, the weather apps also seemed to cache location data within text-based files, such as the AccuWeather app's "accuwx_locations" and "accuwx_geocoder_cache" files. These files appear to contain historical location information for the device, including coordinates and addresses, but they do not appear to store associated timestamps for each entry. The following figures show snippets of some the content recovered from these files.

```
{"adminAreaId":"FL","alias":"Balm","canonicalLocationKey":"2245533","canonicalPostalCode":"33598","countryId":"US","dmaId":"539","geocodedAddress":{"addressLine1":"13012 CR-672","addressLines":[],"adminArea":"FL","country":"United States","formattedAddress":"13012 CR-672, Riverview, FL 33579","latitude":27.7632628,"locality":"Riverview","longitude":-82.2658239,"maxAddressLineIndex":0}
```

Figure 16 - Content of AccuWeather app's "accuwx_locations" file

```
[{"Latitude":27.763,"Longitude":-82.266},[{"addressLine1":"13012 CR-672","addressLines":[],"adminArea":"FL","country":"United States","formattedAddress":"13012 CR-672, Riverview, FL 33579","latitude":27.7632628,"locality":"Riverview","longitude":-82.2658239,"maxAddressLineIndex":0}]
```

Figure 17 - Content of AccuWeather app's "accuwx_geocoder_cache" file

Finally, some ad-supported games obtain device location information, enabling advertisers to tailor their ads to a user's surroundings. In the case of the apps tested, the location information was not found in SQLite databases but in various text-based files. One example involves the Words With Friends app's "iad.dat" file, which was located in the "data\com.zynga.wwf2.free\files\mmsyscache" directory. This file contained some apparent location coordinates, however, they were not consistent with documented test session activity, though still within the same general geographic region (same county). This was true even though the device had access to the device FINE location, based on its permissions. The figure below

shows the content as recovered from the file, with very precise coordinates (yellow), as well as a notable reference to the device connection type (green).

```
"pkid":"com.zynga.wwf2.free","campaignid":"172693","idfa":"","","response":{"adType":"INTERSTITIAL","creativeHeight":-1,"creativeWidth":-1,"cdfId":"e1e4a6d7-1780-46b3-9876-616b5b1393c1"},"ruleVariableValues":{"portraitwpx":360,"osVersion":{"versions":[4,1,2],"versionCount":3},"mobileOS":"ANDROID","compilersdk":{"major":"FIVE","minor":"FOUR","subminor":"ZERO","type":"a"},"instanceCompilerOptions":{},"creativeId":"e1e4a6d7-1780-46b3-9876-616b5b1393c1","mobileOS":"ANDROID","osVersion":{"versions":[4,1,2],"versionCount":3},"deviceHeight":640,"deviceWidth":360,"location":{"latitude":28.082199096679688,"longitude":-82.5239028930664,"accuracy":0.0},"placementWidth":-1,"placementHeight":-1,"userAgent":"Dalvik/1.6.0 (Linux; U; Android 4.1.2; VS870 4G Build/JZO54K)","language":"en","connectionType":"WIFI",""
```

Figure 18 - Content of Words With Friends app's "iad.dat" file, showing latitude and longitude values and connection type

App Trends

As the preceding analyses indicate, location data is cached on these devices in a variety of file types, and many different types of applications can be potential sources of valuable data. Searches for location data may be facilitated by manual review of database content, keyword searches for possible text content and database column names of interest, and of course, review of geo-tagged photos. The recovered location data may reflect the device's location or destinations and points of interest searched for by the device user.

Chat, location sharing, and fitness apps typically record the actual device location, with varying, usually frequent update intervals and typically requiring a user interaction to initiate the location caching. The web browser's "localStorage" databases also seem to follow this trend. The chat and web browser apps only update location information upon a user action, such as a sent

message or a web search, respectively. These apps seem to save this data indefinitely or until user deletion, and they appear to require enabled GNSS services to cache location data.

Leisure apps, on the other hand, seem more likely to store searched or viewed locations, with only most recent device location information cached. They do appear to retain points of interest permanently but do require user interaction to initiate any data caching, as well. They do not appear to require GNSS sensor activity. Navigation and weather apps present a hybrid approach, recording some actual device coordinates, not always with timestamps, but also including searched locations or destinations. These apps appear to preserve the cached information but also require user activity to initiate record keeping. The Waze navigation app specifically requires GNSS functionality to perform the navigation function. Location artifacts recovery from ad-supported game data proved somewhat nebulous, with the highlighted example demonstrating possible unreliability of the stored coordinates.

Depending on the nature of the application, the recovered location data could be used to pinpoint a device's location at a particular time, demonstrate dwelling or travel, or illustrate a user's interest or intent to travel to a particular point of interest. The resulting findings could be used to implicate a particular individual, refute or confirm an alibi, or corroborate witness statements, as a few examples. While certain artifact or application types may be automatically parsed by commercial forensic tools, examiners should also target both SQLite databases and text-based files to recover possible location data of interest, focusing on apps with permissions to the device location.

If a particular application is found to store data of interest, an examiner could further consider installing the app of interest on a test device to better understand its behavior. For

example, the examiner could attempt to determine if the app requires GNSS services to perform its function or cache location data. This could have implications regarding the accuracy of the recovered data. He or she could also evaluate if the application continues to cache data in the background, or if any recovered data must be the result of direct user interaction. Of course, app functionality can vary from version to version, so an examiner would need to consider this limitation when attempting such evaluations. However, by making this effort, an examiner may be able to obtain greater insight into his/her findings and thus be equipped to present the results with greater confidence and clarity.

Metadata and Logs

As seen in the previous review of the Waze app data, apps can and do store location data in text and log files, not just in SQLite databases. However, as shown in Table 6, some app files also appear to store metadata associated with device network connectivity that may be helpful in evaluating the accuracy of cached coordinates. For example, the Locate My Friends app's "dumpLogsDatabase" from the OnePlus One A0001 device contained very detailed information on the location requests, accuracy figures, and location coordinates. An example entry is shown below, having been copied from the SQLite database record into Excel for easier viewing:

{"locationInfo":{"epochTime":1444228846496
"dateTime":"Oct 7 2015 10:40:46 AM"
"data":{"locDB":{"timestamp":1444228823366
"provider":"fused"
"lat":27.9778361
"lon":-82.5132756
"accuracy":43.5
"alt":0
"age":"23129"}
"locNew":{"timestamp":1444228846236
"provider":"fused"
"lat":27.9778371
"lon":-82.5132754
"accuracy":36
"alt":0
"age":"259"}
"geolocation_meta":{"emode":"storeLoc"
"info":"DB: New loc with higher accuracy. Replacing loc in database"}
"device":{"battery":"82"
"charge":"1"
"wifi_state":"1"
"build":"10827"} } } }

Figure 19 - Sample entry from "dumpLogsDatabase" file

Aside from the actual latitude and longitude values, this database was also caching the accuracy information, WLAN (“wifi”) state, altitude (“alt”), and battery status. Other entries detail the location request type, such as “PRIORITY_BALANCED_POWER_ACCURACY,” as well as cell tower connectivity, with one example reading "info":"New cell tower location detected. Notifying policies." This level of detail is certainly illuminating, especially since this log had such high granularity, sometimes recording multiple updates for a single second! Of course, the extreme detail of this particular log did have a significant tradeoff: it only covered the most recent day’s activity. Still, for one session running approximately 128 minutes, this log had over 3000 entries! A small window, surely, but it is a very detailed one.

Another database associated with one of the ad-supported free games, QuizUp’s “mixpanel” database, as recovered from the Galaxy S5 test device, stored some information on the WLAN connectivity history, but it was embedded in lengthy text records and was limited in both quantity and scope. The information was timestamped, but the timestamps only pertained to sessions in which this game was actually in use. Furthermore, the entries reflect the wireless network connectivity state, not necessarily whether the WLAN sensor was active. As seen in the figure below, these entries show “\$wifi:false” even though the WLAN services were enabled at the specified timestamps appearing toward the right side of each entry, based on test session documentation. However, the device was not connected to a network at the time, so the reported “false” status is consistent with the connectivity state.



Figure 20 - Contents of the QuizUp game's "mixpanel" database showing WLAN network state with timestamp

The Amazon shopping app’s “event” database recovered from the LG VS870 device, covered a much broader range of dates and times, and tags events with a plain-text “connectionType:” parameter, listing either “WIFI” or “mobile” for each entry. This database was reviewed to determine if the connection types reported were consistent with those noted in the test session documentation. In most cases, the recorded connection types matched the active services noted in the test session documentation. The exceptions again included sessions in which the WLAN sensor was enabled but the device was not actually connected to a network.

Exceptions are noted in the following table, which contains formatted samples of the data retrieved from the database to facilitate review of the information. Actual data pulled from the database is shown in the green columns, while author-added columns are presented in gray.

Table 14 - Examples of connection types logged in the Amazon app's "event" database that are not consistent with test session documentation

Connection Type	Timestamp	Converted Timestamp (UTC)	Indoor/ Outdoor	Reason for discrepancy
connectionType:"mobile"	1433634088961	6/6/2015 23:41:28	Outdoor	On but not connected to network
connectionType:"mobile"	1436961354568	7/15/2015 11:55:54	Indoor	On but not connected yet, immediately subsequent entries show connection
connectionType:"mobile"	1436972419082	7/15/2015 15:00:19	Outdoor	Driving
connectionType:"mobile"	1436974483466	7/15/2015 15:34:43	Outdoor	On but out of range of network access point
connectionType:"mobile"	1437147300840	7/17/2015 15:35:00	Outdoor	Driving
connectionType:"mobile"	1442675854508	9/19/2015 15:17:34	Outdoor	Driving
connectionType:"mobile"	1442706821217	9/19/2015 23:53:41	Outdoor	On but not connected to network

One more log file of possible interest was recovered from both of the non-Lollipop devices (the LG VS870 and Galaxy S4 Mini) and was stored in the Google Mobile Services directory. This database, titled "herrevad," appears to track connections to saved WLAN networks, including SSID and BSSID (MAC address) info, as well as timestamps. This content could be valuable in placing a particular device within range of a known access point at a specific time. However, during a review of the databases' contents, it became clear that the database did not capture all of the instances in which the devices were connected to WLAN networks, based on test session activity. However, for those instances it did report connectivity,

the information was consistent with test session documentation. An excerpt of the database content appears in the table below, with author-added content in gray.

Table 15 - Excerpt of the Google Mobile Services' "herrevad" database showing WLAN network connection details

ssid	security_type	bssid	timestamp_millis	Converted timestamp (UTC)
COYG	4	20:aa:4b:32:21:3c	1436961710670	7/15/2015 12:01:50
COYG	4	20:aa:4b:32:21:3c	1436961823419	7/15/2015 12:03:43
COYG	4	20:aa:4b:32:21:3c	1436961964698	7/15/2015 12:06:04
COYG	4	20:aa:4b:32:21:3c	1436962208837	7/15/2015 12:10:08
COYG	4	20:aa:4b:32:21:3c	1436962385574	7/15/2015 12:13:05
COYG	4	20:aa:4b:32:21:3c	1436964198663	7/15/2015 12:43:18
COYG	4	20:aa:4b:32:21:3c	1436964362487	7/15/2015 12:46:02

From these examinations, it seems clear that connectivity-related log artifacts may be quite useful in ruling out the possibility that the WLAN sensor was disabled at a particular time. However, it may be more difficult to affirm that the sensor was indeed enabled at a particular time, since these logs seem to only document when the device is actually connected to a network. A device may have the WLAN functionality enabled but be out of range or not connected due to wireless network security, for example. In situations like these, it seems the log files would not indicate that the device WLAN feature was active, since the device would then default to cellular data services.

Knowing this, and knowing that the Waze app requires “High Accuracy” mode to be enabled for navigation functionality, it could be helpful to know when the Waze app is in use. Fortunately, one database recovered from the Galaxy S4 Mini test device seems to do just that. This database, titled “ContextLog_0.db,” tracks app usage in one of its tables, including start and stop times, as well as duration and activity type. It seems to be specific to Samsung devices,

located within the “/Root/data/com.samsung.android.providers.context/databases/” directory.

Figure 21 below shows an excerpt of this database, filtered to show the Waze app’s activity on a particular test date.

	app_sub_id	launcher_type	start_time	stop_time	duration
android_metadata (1)					
browse_web (0)	om.waze.MainActivity	0	9/16/2015 9:14:29 PM	9/16/2015 9:14:58 PM	29169
capture_content (0)	om.waze.navigate.NavigateActivity	0	9/16/2015 9:14:58 PM	9/16/2015 9:15:02 PM	3360
change_activity (0)	om.waze.MainActivity	0	9/16/2015 9:15:02 PM	9/16/2015 9:15:05 PM	2912
change_device_status (0)	om.waze.FreeMapAppActivity	0	9/16/2015 9:15:18 PM	9/16/2015 9:15:18 PM	1
exchange_call (0)	om.waze.MainActivity	0	9/16/2015 9:15:18 PM	9/16/2015 9:15:32 PM	14170
exchange_email (0)	om.waze.navigate.social.AddFriendsActivity	0	9/16/2015 9:15:32 PM	9/16/2015 9:15:36 PM	4024
exchange_message (0)	om.waze.MainActivity	0	9/16/2015 9:15:36 PM	9/16/2015 9:15:43 PM	6249
manage_app (262)	om.waze.navigate.NavigateActivity	0	9/16/2015 9:15:43 PM	9/16/2015 9:15:51 PM	7750
move_area (0)	om.waze.navigate.NavigateActivity	0	9/16/2015 9:15:56 PM	9/16/2015 9:15:59 PM	3094
move_location (0)	om.waze.navigate.SearchActivity	0	9/16/2015 9:15:59 PM	9/16/2015 9:16:04 PM	5239
move_place (0)	om.waze.navigate.AddressPreviewActivity	0	9/16/2015 9:16:04 PM	9/16/2015 9:16:08 PM	3575
play_music (0)	om.waze.MainActivity	0	9/16/2015 9:16:08 PM	9/16/2015 9:21:09 PM	301161
play_video (0)	om.waze.main.navigate.NavigationListActivity	0	9/16/2015 9:21:09 PM	9/16/2015 9:22:57 PM	107572
record_audio (0)	om.waze.MainActivity	0	9/16/2015 9:22:57 PM	9/16/2015 9:27:57 PM	300294
record_video (0)	om.waze.navigate.social.AddFriendsActivity	0	9/16/2015 9:27:57 PM	9/16/2015 9:28:08 PM	10945
search_keyword (0)	om.waze.MainActivity	0	9/16/2015 9:28:08 PM	9/16/2015 9:30:11 PM	122585
sqlite_sequence (2)	om.waze.navigate.social.AddFriendsActivity	0	9/16/2015 9:30:11 PM	9/16/2015 9:31:05 PM	53652
take_photo (0)	om.waze.MainActivity	0	9/16/2015 9:30:11 PM	9/16/2015 9:31:05 PM	832049
track_legacy (0)	om.waze.MainActivity	0	9/16/2015 9:31:05 PM	9/16/2015 9:44:57 PM	2906
track_roaming (0)	om.waze.main.navigate.NavigationListActivity	0	9/16/2015 9:44:57 PM	9/16/2015 9:45:00 PM	2906
use_app (1970)	om.waze.MainActivity	0	9/16/2015 9:45:00 PM	9/16/2015 9:52:10 PM	429704
use_wifi (0)					
write_document (0)					

Figure 21 - Content of the "ContextLog_0.db" database, filtered to show Waze app activity

The Waze “NavigateActivity” entries are particularly interesting, knowing that the Waze app requires all sensors to be enabled to perform the navigation function. Because the duration of this activity is also reported, it could be of value in evaluating not just Waze application data reliability, but also the potential accuracy of any location artifacts cached by other applications during the same timeframe.

If another app had cached location data for a particular time, or if an investigator had obtained a timestamped location from either cellular service providers or the cloud resources previously discussed, a review of such metadata cached by seemingly unrelated apps could

provide insight into the reliability of the location artifact, even if the information pertains just to WLAN services. This would especially apply if the accuracy trends outlined in the literature review section hold true. The next section addresses the evaluated accuracy of the location data recovered in this study.

Accuracy Evaluation

Large and disparate quantities of location artifacts were recovered from the various test devices. As previously reviewed, the location data is stored in varying formats and intervals. To evaluate the reliability of the device location services, the recovered geo-tagged photos were examined and compared to the actual locations documented both via the photos themselves and the test session documentation. Additional examinations of the recovered cloud data were performed to investigate its reliability, as well. From these reviews, a number of general trends were noted.

Geo-Tagged Photos

As previously discussed, only two of the four test devices were found to store any geo-tagged photos, despite all devices being configured to geo-tag camera images and being used in test sessions involving captured photos. Analysis of the geo-tagged photos from the LG VS870 and the OnePlus One A0001 test phones was performed to determine if any accuracy determinations could be made. Each photograph's embedded latitude and longitude values were compared to the known location as captured in the image and noted in the test session documentation, using an online distance calculator tool. A secondary tool was then used to verify

the calculated distance. The figure below shows an example coordinate from one A0001 geo-tagged photo and its corresponding actual location, with the error distance displayed, as well.

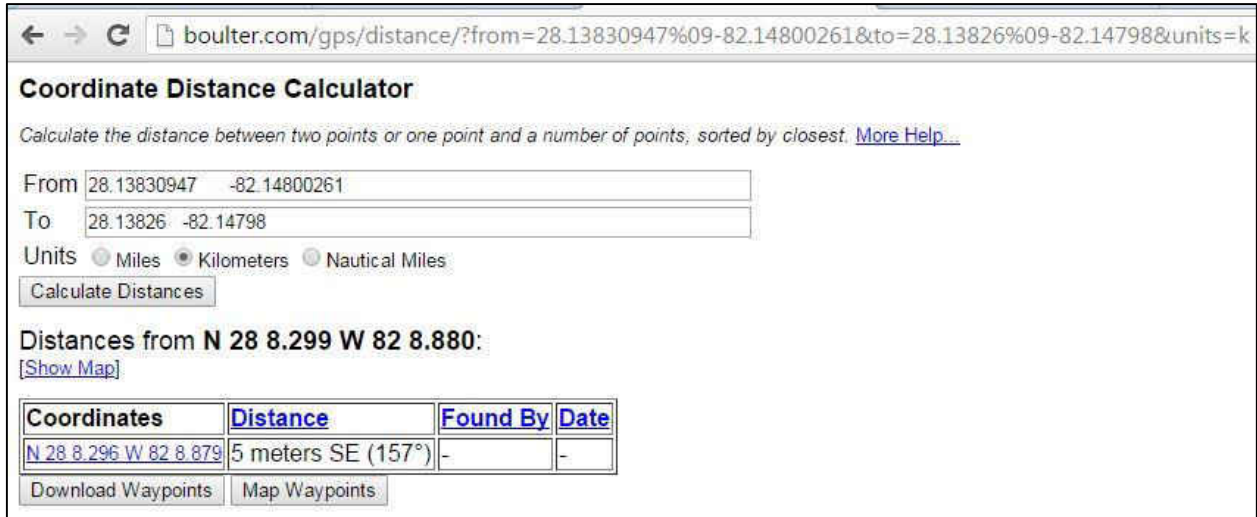


Figure 22 - Geo-tagged photo example latitude and longitude accuracy check

Via this method, each coordinate recovered from the test devices' geo-tagged photos was evaluated for accuracy and an average value was calculated for the various test sessions. Test session environmental and sensor parameters were also examined, with the results for each device documented in the tables below. Fifty-eight photos from the A0001 and sixteen photos from the VS870 were evaluated.

Table 16 - LG VS870 geo-tagged photos, average accuracy

Device	Date	Number of Photos	Environment	Indoor/ Outdoor	Cell/WLAN/GNSS	Average Accuracy (m)
VS870	9-Jun	3	Suburban	Outdoor	Cell	130
VS870	16-Sep	2	Suburban	Indoor	Cell+WLAN	29.5
VS870	19-Sep	3	Urban	Outdoor	Cell+WLAN	67.3
VS870	15-Jul	3	Rural	Outdoor	Cell+WLAN+GNSS	19.7
VS870	6-Jun	2	Suburban	Outdoor	Cell+WLAN+GNSS	28.5
VS870	15-Jul	3	Rural	Indoor	Cell+WLAN+GNSS	30.3

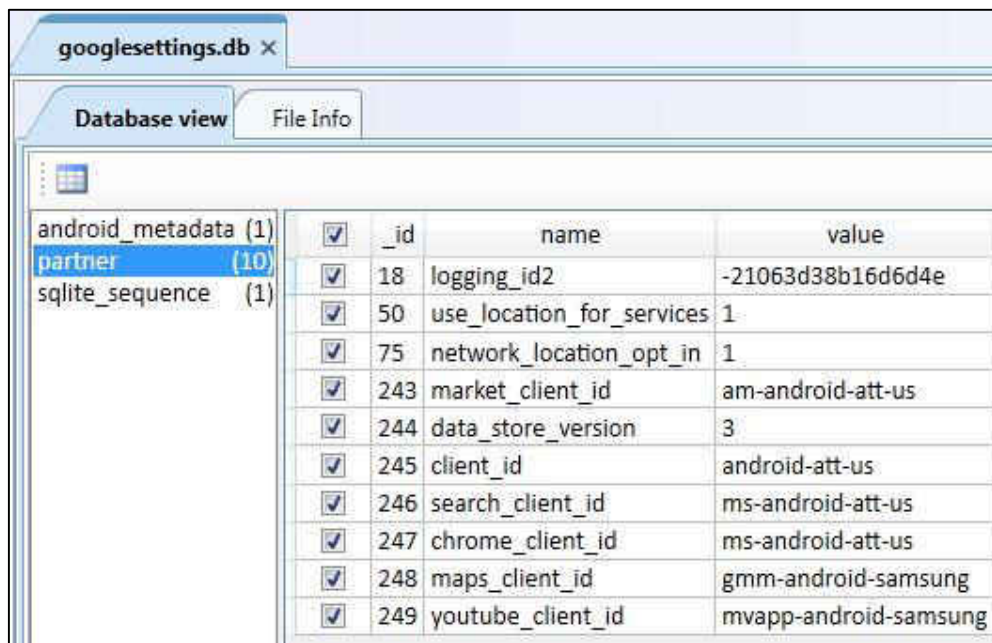
Table 17 - OnePlus One A0001 geo-tagged photos, average accuracy

Device	Date	Number of Photos	Environment	Indoor /Outdoor	Cell/WLAN/GNSS	Average Accuracy (m)
A0001	14-Jul	3	Suburban	Indoor	Cell	24
A0001	9-Jun	3	Suburban	Outdoor	Cell	1373.3
A0001	6-Jun	3	Suburban	Outdoor	Cell+WLAN+GNSS	3.7
A0001	8-Sep	6	Suburban	Indoor	Cell+WLAN+GNSS	3.7
A0001	15-Jul	3	Rural	Outdoor	Cell+WLAN+GNSS	4.3
A0001	19-Jul	3	Rural	Outdoor	Cell+WLAN+GNSS	5.7
A0001	6-Sep	3	Suburban	Indoor	Cell+WLAN+GNSS	10.7
A0001	6-Aug	25	Suburban	Outdoor	Cell+WLAN+GNSS	10.9
A0001	19-Jul	1	Rural	Indoor	Cell+WLAN+GNSS	15
A0001	15-Jul	3	Rural	Indoor	Cell+WLAN+GNSS	16
A0001	27-May	5	Suburban	Indoor	Cell+WLAN+GNSS	118

The results seem to support the notion that cellular-only derived locations are less reliable than those in which the GNSS resources are active. Results from the two test sessions for which geo-tagged photos were captured when only cellular and WLAN sensors were active also seem to corroborate the notion that location data reliability does not suffer from a device's use indoors. Indeed, these results seem to indicate that the location data may be more accurate when WLAN services are used indoors, but due to the small sample size, more testing would likely be needed to determine if this observation can be applied generally.

Cloud Data Limitations

Two preliminary issues present potential obstacles to the utility of the Google Location History cloud data. The first involves the user's choice to utilize Google's location services. A user may opt not to do so, though this option could severely limit the device's functionality in the conventional sense of how consumers utilize their smartphones. For example, a user would not be able to utilize the Google Maps app. In short, most users likely do choose to use Google's location services, making this particular issue probably less significant. The "googlesettings.db" database appears to store a record indicating whether a user has opted to allow Google's services access to the device location, as shown in the figure below.



	<input checked="" type="checkbox"/>	_id	name	value
android_metadata (1)				
partner (10)	<input checked="" type="checkbox"/>	18	logging_id2	-21063d38b16d6d4e
sqlite_sequence (1)				
	<input checked="" type="checkbox"/>	50	use_location_for_services	1
	<input checked="" type="checkbox"/>	75	network_location_opt_in	1
	<input checked="" type="checkbox"/>	243	market_client_id	am-android-att-us
	<input checked="" type="checkbox"/>	244	data_store_version	3
	<input checked="" type="checkbox"/>	245	client_id	android-att-us
	<input checked="" type="checkbox"/>	246	search_client_id	ms-android-att-us
	<input checked="" type="checkbox"/>	247	chrome_client_id	ms-android-att-us
	<input checked="" type="checkbox"/>	248	maps_client_id	gmm-android-samsung
	<input checked="" type="checkbox"/>	249	youtube_client_id	mvapp-android-samsung

Figure 23- Content of the "googlesettings.db" database

The "use_location_for_services" option is set to "1," indicating that the feature is enabled. This should indicate that Google Location History should be stored on Google's servers and therefore be retrievable using the Cloud Analyzer software or legal process to Google.

However, this database was located in the Google Services Framework directory (“/Root/data/com.google.android.gsf/databases/”) as recovered from two of the test devices. This is not a coincidence, as these were the two non-Lollipop devices. This brings up the second fundamental issue.

In the newer versions of Android, as previously discussed, many of the core Google application data is excluded from the ADB backup process. This not only includes the database that indicates whether a user’s location history may be recoverable using the Cloud Analyzer software, but it also extends to a key resource used to generate the account package file containing the necessary credentials to obtain the information: the “accounts.db” database. This database stores the Google account information, with the login email address and an encrypted form of the password, as shown in the following figure.

	<input checked="" type="checkbox"/>	_id	name	type	password
accounts (4)	<input checked="" type="checkbox"/>	1	test.phone.gsm1@gmail.com	com.google	oauth2rt_1/vbA-Yjq2lwXQ4gdTpXUoC5Kpne3rx3_LNDEKxbsJe2Y
android_metadata (1)	<input checked="" type="checkbox"/>	8	com.zynga.wfnaccount	com.zynga.wfn	com.zynga.wfnaccountpwd
authtokens (51)	<input checked="" type="checkbox"/>	10	WhatsApp	com.whatsapp	
extras (139)	<input checked="" type="checkbox"/>	11	test.phone.gsm1@gmail.com	com.etermax.games.account	
grants (0)					
meta (0)					
shared_accounts (0)					
sqlite_sequence (3)					

Figure 24 - Content of the "accounts.db" database

This database is stored under the “/Root/system/users/” directory, which evidently is also excluded from the ADB backup process, as it was not recovered for either of the Lollipop devices for which physical extraction was not supported. Without this file, the account package file cannot be created, meaning credentials will have to be manually entered in the Cloud Analyzer extraction process. Of course, this would require the examiner to know the user’s

credentials or obtain them somehow. Furthermore, the manual entry of credentials does prompt the notification email to the user that a new login from an unrecognized device has occurred. Without a non-disclosure order of some kind, investigators risk alerting a suspect of their activity, even if they are fortunate enough to have obtained the credentials somehow. Still worse, if investigators cannot gain root access, obtain a physical extraction of the device data, or ascertain the Google account credentials, they may not be able to retrieve the cloud data at all.

However, even presuming that the extraction of the cloud data is successful, another limitation of its utility presents itself. This stems from the relatively less precise nature of the retrieved coordinates. As described before, the Google Location History data extracted via the Cloud Analyzer software appears restricted to a precision level of one thousandth of a degree. This means the coordinates should be roughly within 100 meters of the actual device location. Contrast this level of reliability with that obtained via the various apps reviewed, many of which were caching coordinates on the order of millionths-level precision or better, and that makes the Google Location History seem more of a general outline than a specific track. Note the higher frequency and precision of the RunKeeper app's cached data for the same timeframe as shown in the figure below.







559		7/19/2015 12:58:46 PM -07:00 location	35.106, -77.043	Converted Time (UT)	latitude	longitude
				7/19/2015 15:58:45	35.10599137	-77.04331612
				7/19/2015 15:58:39	35.10607412	-77.04331825
				7/19/2015 15:58:31	35.10616206	-77.04329149
				7/19/2015 15:58:21	35.1062525	-77.04327185
560		7/19/2015 12:58:00 PM -07:00 location	35.107, -77.043	7/19/2015 15:58:16	35.10633332	-77.04324095
				7/19/2015 15:58:09	35.10642678	-77.0432187
				7/19/2015 15:58:01	35.10650928	-77.04321579
				7/19/2015 15:57:49	35.10659098	-77.04319956
561		7/19/2015 12:57:14 PM -07:00 location	35.107, -77.043	7/19/2015 15:57:43	35.10668206	-77.04321179
				7/19/2015 15:57:36	35.10676328	-77.04322879
				7/19/2015 15:57:18	35.10684428	-77.04325402
				7/19/2015 15:57:06	35.10689561	-77.04335171
562		7/19/2015 12:56:28 PM -07:00 location	35.107, -77.044	7/19/2015 15:56:52	35.10692131	-77.04344724
				7/19/2015 15:56:40	35.10688501	-77.04354843
				7/19/2015 15:56:32	35.10689425	-77.04364952
				7/19/2015 15:56:23	35.10690638	-77.0437562
563		7/19/2015 12:55:43 PM -07:00 location	35.107, -77.044	7/19/2015 15:56:15	35.10689029	-77.04386281
				7/19/2015 15:56:08	35.10686313	-77.04396034
				7/19/2015 15:55:59	35.10681822	-77.04404375
				7/19/2015 15:55:49	35.10687221	-77.04412523
564		7/19/2015 12:54:58 PM -07:00 location	35.107, -77.044	7/19/2015 15:55:36	35.10688684	-77.04423598
				7/19/2015 15:55:23	35.10680213	-77.0442528
				7/19/2015 15:55:16	35.10672203	-77.04427229
				7/19/2015 15:55:08	35.10663916	-77.04428466
				7/19/2015 15:54:59	35.10654981	-77.04427734

Figure 25 - Cloud Analyzer (left) and RunKeeper (right) location data for same timeframe

Curiously, this same discrepancy in precision seems also to extend to the Google location data obtained by downloading a .kml file via the Google account web interface. This involves logging into the Google user account and selecting the “Control My Content” option from the menu, then navigating to the “Manage Activity” option for the “Places you go” category, as shown in the figure below. Notably, users can also opt to delete content via this utility, as well.

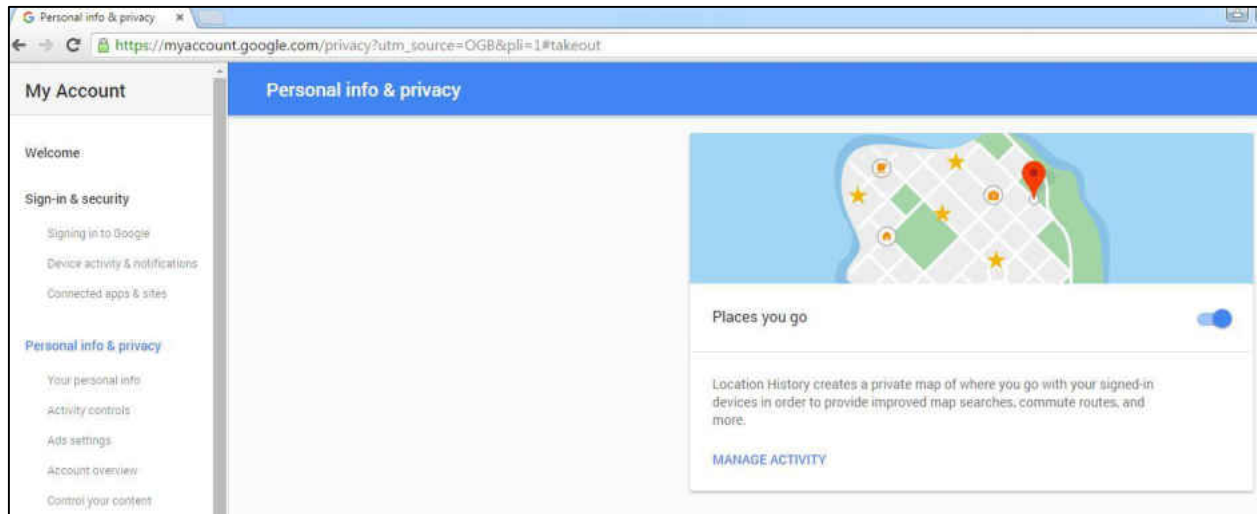


Figure 26 - Google user account interface content management

From there, a user may view their Google location history for a specified day, or they may opt to download or delete the content. Presumably, any such downloaded content would resemble the corresponding timeframe's data obtained via the Cloud Analyzer software for the same device/user. To test this notion, the content for the date referenced in Figure 25 above was saved to a .kml file, as shown below.

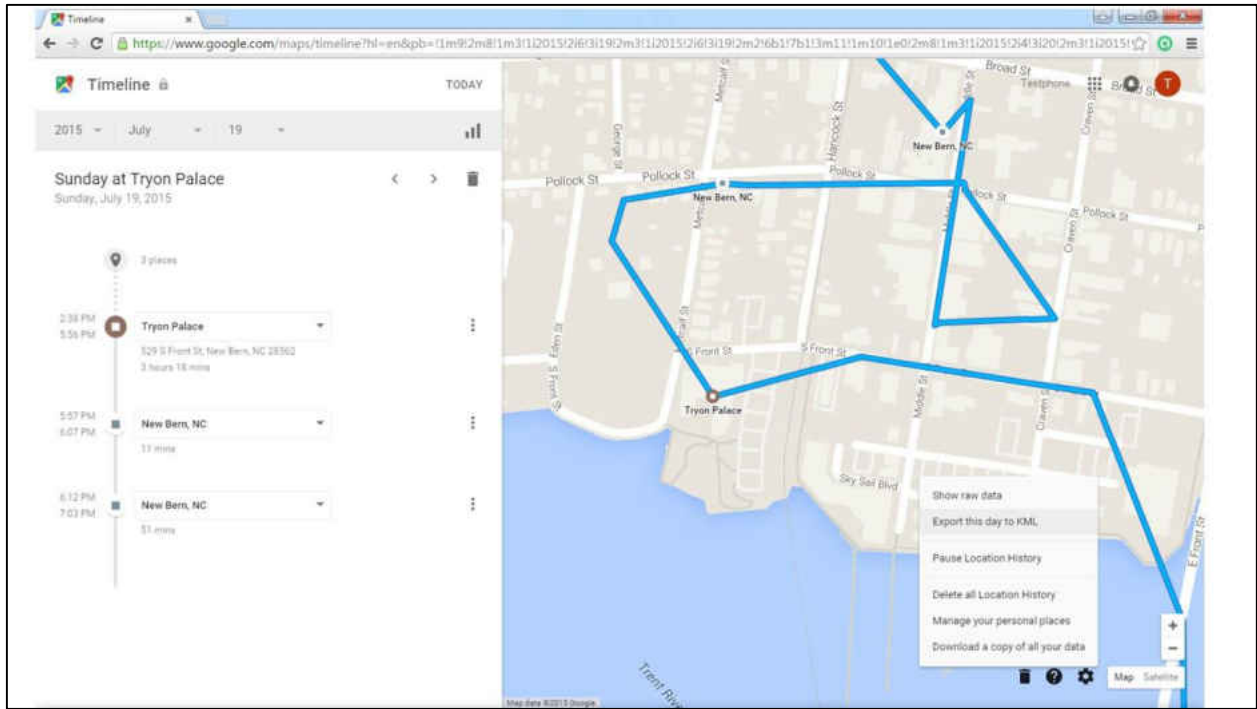


Figure 27 - Option to save timeframe as KML file via Google user account web interface

Once this data was downloaded, it was mapped along with the corresponding data from the RunKeeper app and the Cloud Analyzer locations. Curiously, the data downloaded directly from Google’s user account interface appears far more precise than the associated Cloud Analyzer data. So the initial assumption that the Cloud Analyzer data’s precision level was due to a limitation imposed by Google appears to be incorrect. Google’s servers appear to be storing much more precise information. Additional testing or inquiry would be needed to determine if this is a limitation inherent to the Cloud Analyzer software, or some other reason. Figure 28 displays the map comparing the RunKeeper, Google account KML file, and Cloud Analyzer data for the same timeframe.



Figure 28 - Comparison of cached coordinates for same timeframe from RunKeeper (purple), Google user location history (yellow), and Cloud Analyzer (cyan)

A number of clear observations can be made from this map. First, the Google user account location history is clearly more precise than the Cloud Analyzer data obtained, even though the Cloud Analyzer theoretically should be accessing and retrieving the very same Google account data. Furthermore, the Google user data appears to correspond more closely to the RunKeeper app's data, with the Cloud Analyzer data forming broad geometric patterns surrounding the more detailed tracks presented by the other sources. Given that the more accurate location is likely to be the more precise one, it seems that the Cloud Analyzer data may be limited to the 100 meter range imposed by its thousandths-level precision. Still, this level of

accuracy will certainly be enough in many cases, and the high update interval would still be useful in establishing patterns of movement or dwelling.

At a minimum, the Cloud Analyzer software offers another tool to investigators for corroborating data or testimony from other sources. It is very dense, timewise, voluminous, and it is clearly accurate enough to place an individual device within a block or so of a particular location, at worst. Witness statements, suspect alibis, call detail records, or other resources could all be better evaluated with this information. Given that the majority of Android users likely utilize the Google location services that render this software's function possible, it seems that the cloud data resource could become a potential boon for investigators in criminal cases, provided they are able to extract the necessary data to access the content or otherwise obtain the user's credentials.

General Trends

To evaluate the accuracy of the location data, content from both the information recovered from the device and the Google Location History cloud data were reviewed for coordinates with corresponding timestamps, then compared to actual locations documented in test session records. Overall, the results seemed to support the previously stated trends involving accuracy based on resource type. For example, locations obtained when only cellular service was active were significantly less accurate than the same location information recorded in the same place at approximately the same time when all sensors were active, as illustrated in Figures 29 and 30 below.

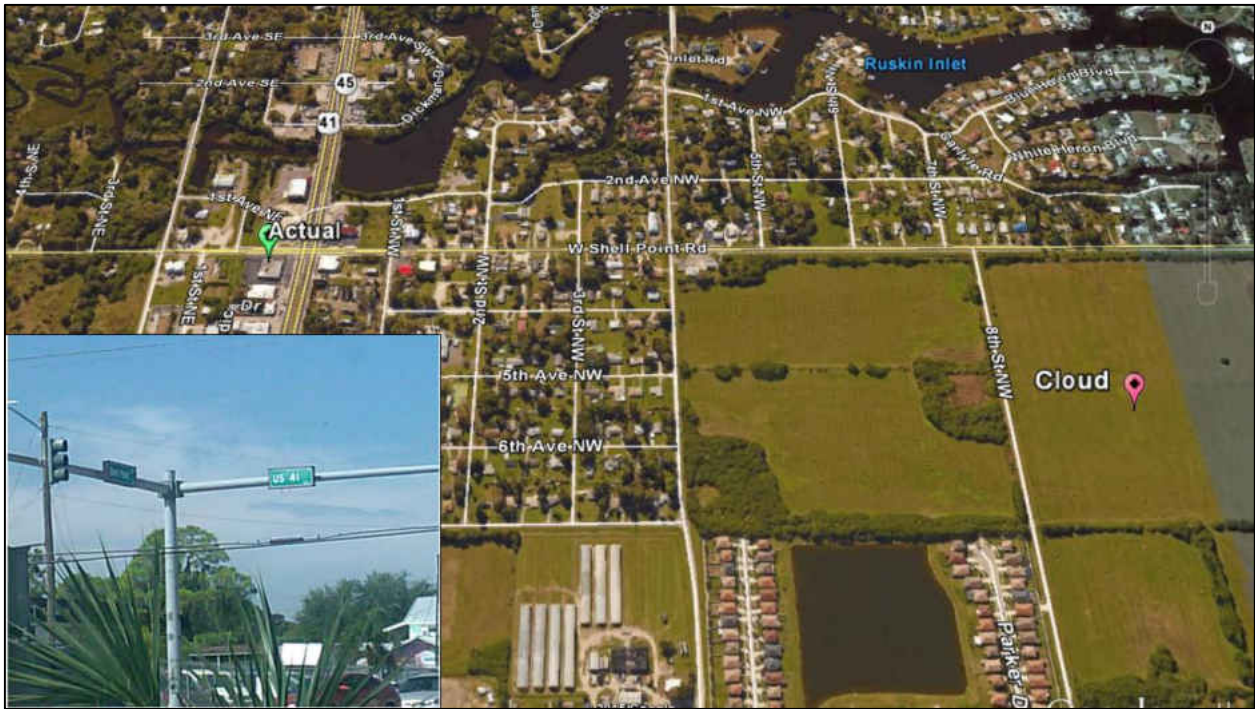


Figure 29 - Actual versus Cloud location with cellular service only (error of over 1.5 kilometers)



Figure 30 - Actual, Device, & Cloud locations with all sensors active (errors from 5 to 45 meters)

These figures depict how significant the impact of sensor activity can be. With all of the sensors active, the error margin was reduced from over a kilometer down to a few meters for the device cached information, a little more for the corresponding cloud location. Interestingly, the cloud data point was not quite as accurate as what the device cached, even though all services were active. This could very well be the result of the cloud data's limitation to thousandths-level precision. Figure 31 again illustrates the way in which cloud points, even with high frequency and volume of locations, are hampered by the lower precision. The device is caching coordinates on the order of ten-thousandths in the Map My Walk app's "workout.db" database, and just that extra power of ten clearly shows a much more refined track of the actual device location.



Figure 31 - Map My Walk device (green) versus cloud (red) locations

The cloud track clearly is more angular and less precise, but it does capture the basic outline of the movement, with errors generally in the 20 to 50 meter range. In another instance, when only cellular and WLAN sensors were active, the error was in the over a kilometer range. Figures 32 and 33 demonstrate how much of an impact activating the GNSS services had for the particular area in question (traversed during a drive along the junction of Interstates 10 and 75, and back again). In Figure 32, the error reaches up to about 1.5 kilometers without the GNSS services. However, with the GNSS sensor enabled, the max error appears to go down to roughly 300 meters, though it is generally much less than that, as shown in Figure 33.

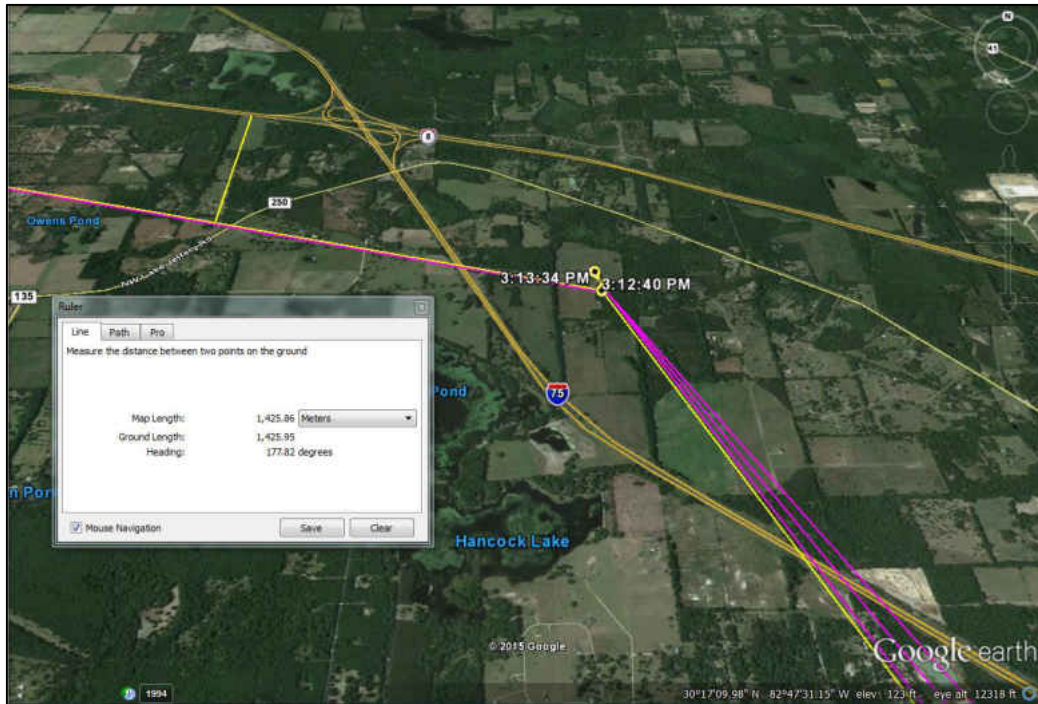


Figure 32 - Location Tracker device data (yellow) versus cloud data (magenta) - no GNSS

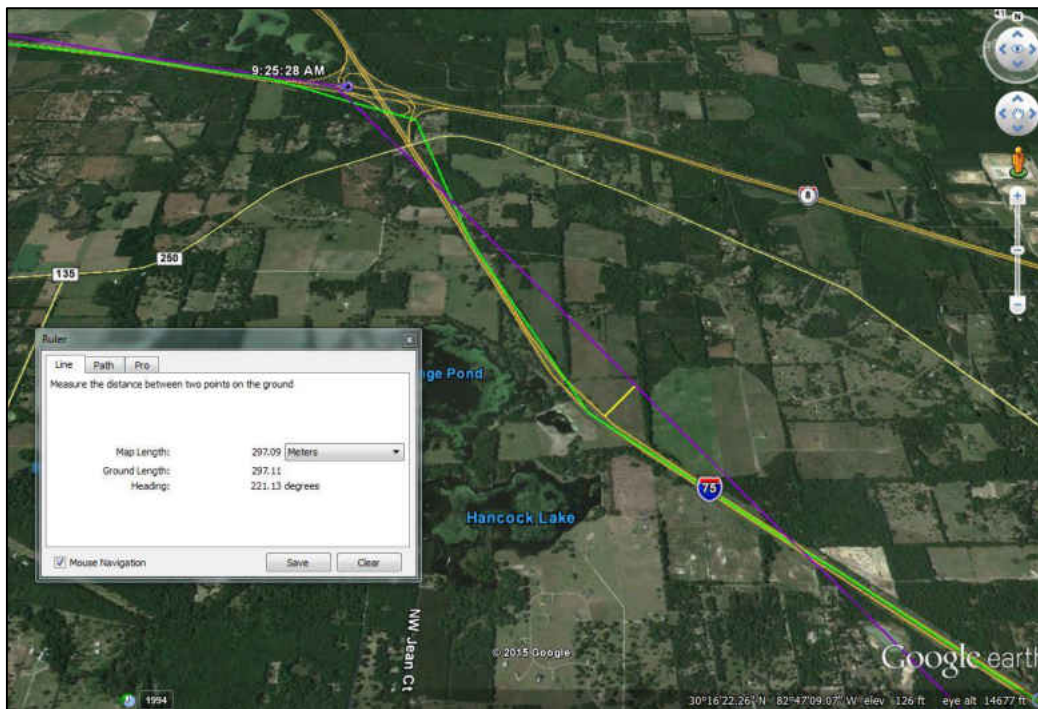


Figure 33 - Location Tracker device data (purple) versus cloud data (green) - with GNSS

The above figures demonstrate the relative accuracies in suburban and rural environments, with and without GNSS services enabled. In general, there was no notable cost to accuracy of cached device data when GNSS services were enabled, regardless of whether the surrounding were rural or suburban, and very little cost noted for indoor use. In urban areas, there did appear to be relatively little tradeoff for disabling GNSS services, perhaps due to the higher density of cell towers and WLAN access points, as well as the possible interference with GNSS signals due to concentrations of tall buildings. Figures 34 and 35 show some examples of the location data obtained from testing in an urban environment with GNSS disabled and enabled, respectively.

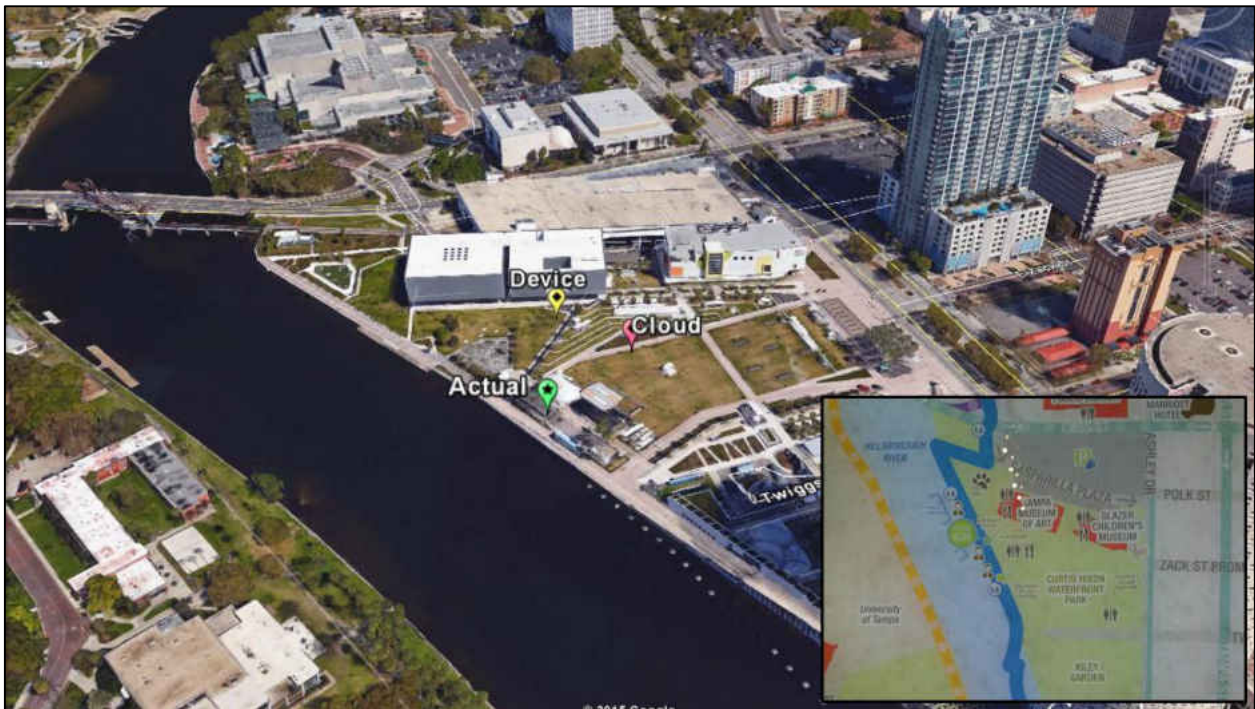


Figure 34 - Device versus actual versus cloud locations in urban environment without GNSS

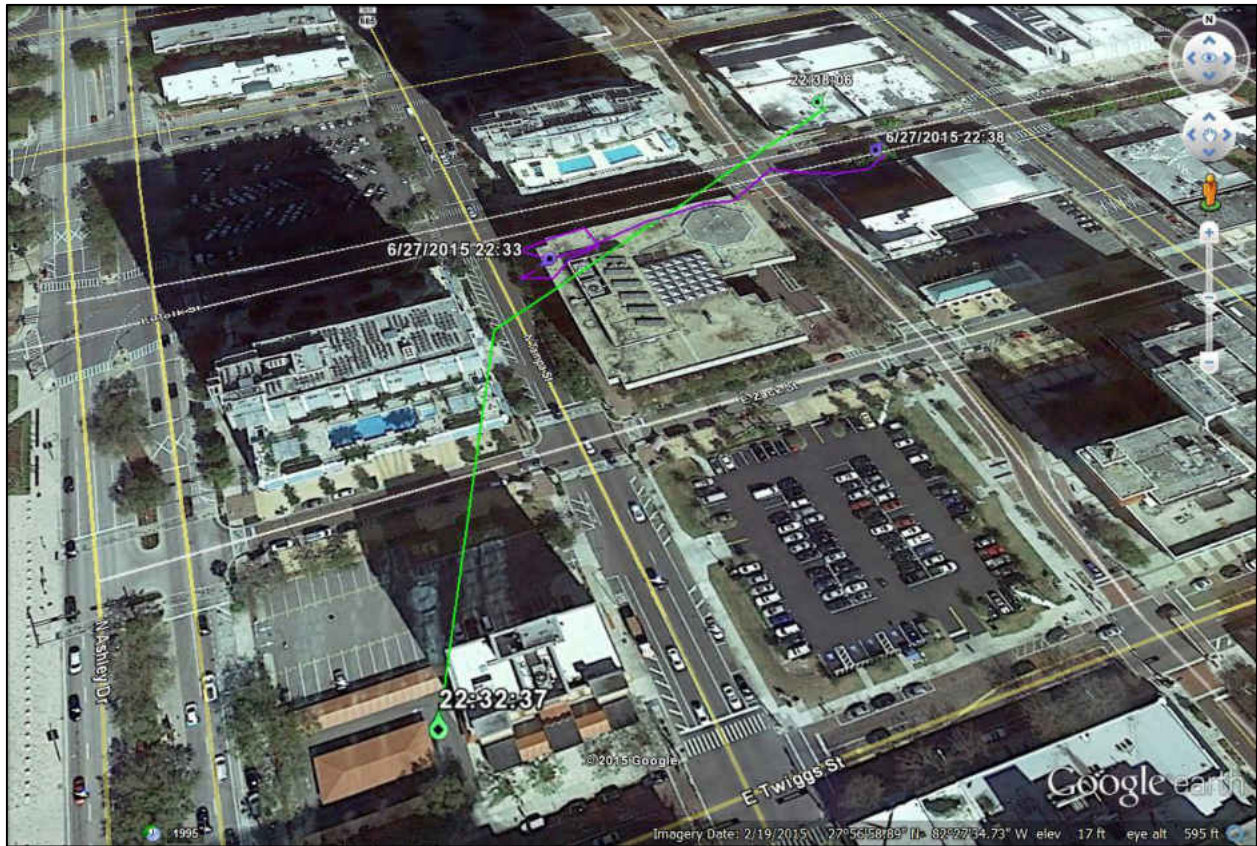


Figure 35 - Device (purple) versus cloud (green) locations in urban environment with GNSS

The purple route in Figure 35 reflects the more accurate route based on the test session activity and was cached on the device by the Run Keeper app. There certainly is some improvement over the relative locations displayed in Figure 34, in which only cellular and WLAN services were enabled, at least for the device data. The cloud data is clearly less precise and curiously less frequently updated (roughly every three minutes) for this particular timeframe. The relatively less accurate cloud data for this instance could again be attributed to the thousandths-level precision that characterizes it. Furthermore, the device was in transit in this test session, and in similar sessions involving transit, the cloud locations also displayed a small time lag.

Efforts to evaluate the reliability of recovered location data using the cached accuracy metadata focused on applications that reported their accuracy in real time via the user interface during test sessions. These accuracy estimates were noted in the test session documentation and later compared to the cached accuracy values. In the user interface, the accuracy values were reported in feet. No units are specified in the accuracy metadata recovered from device SQLite databases for the apps in question, namely Life360 and Locate My Friends, but numerically, they were roughly in line with those reported via the user interface. These locations were plotted on maps along with cloud location points with similar timestamps, then compared to documented test session locations. In general, the calculated error values noted in the maps seemed as though they'd be more consistent with device and user interface values if the reported error were measured in meters rather than feet.

Overall, however, the general trends in terms of a hierarchy of source accuracy seemed to hold true, with GNSS-enabled sessions recording the most accurate data. In some instances, individual location points appeared to be within just a few feet of the actual test location. This was especially true when GNSS services were enabled, but was also noted a few times when cellular and WLAN services were running without GNSS. The maximum error range noted in this study was still less than two kilometers off, and it was noted in test sessions involving only cellular service and in less populated areas. Thus, even in the worst case scenarios of this study's conditions, both cloud and device-cached location data proved reliable enough to place a device within a city, at the very least, and possibly into specific buildings, under ideal circumstances. As both cell tower and WLAN access point infrastructure expands and increases in concentration accordingly, it seems only likely that mobile device location data will become even more precise.

Location Data Recovery and Evaluation Strategy

First, if an examiner has access to the various paid, specialized mobile device forensics tools, a good deal of location of artifacts will be parsed automatically, including geo-tagged photos, Google Maps searches, Waze recent destinations, and coordinates from various messaging apps like Facebook or WhatsApp. If not, these are relatively easy to identify via the use of free metadata parsers, in the case of geo-tagged media files, or SQLite database viewers, for the various navigation and chat apps mentioned.

From there, Internet Explorer can be used to examine the “packages.xml” file, using a simple “find” search for “ACCESS_FINE_LOCATION” and “ACCESS_COARSE_LOCATION” to identify applications with location permissions and the directory paths in which their data is cached. The examiner may then target these directories for further review, especially SQLite databases and XML configuration files. SQLite databases may contain detailed caches of latitude and longitude coordinates, with metadata, and they may also contain embedded images like map tiles.

It is also important to remember that some SQLite databases may store content of value in BLOB data, which may be less intuitive to examine. Google Maps’ “gmm_storage.db” database is a prime example, storing search terms and possible navigation history amongst other proprietary content in BLOB data records. Keyword searches are appropriate for analyzing this file, if an examiner is attempting to recover evidence regarding a particular location. Alternatively, a very primitive approach could begin with using the Strings command-line tool to output string content from the database to a text file, then filter the data in Excel to display content beginning with “/dir/” to identify searches for directions with latitude and longitude of

the position from which the search initiated, as well as the latitude and longitude of the searched destination. Figure 36 shows some of the content recovered in this way from one test device's "gmm_storage.db" database.

2	Strings v2.51
1760	/dir/27.7207419,-82.4336623/Dunkin'+Donuts,+6190+N.+US+Highway+41,+Apollo+Beach,+FL+33572,+United+States/da
2701	/dir/27.7291619,-82.4337426/Sam's+Club,+10385+Big+Bend+Road,+Riverview,+FL+33578/data=!4m14!4m13!1m1!4e1!1
3116	/dir/27.7291619,-82.4337426/Sam's+Club,+10385+Big+Bend+Road,+Riverview,+FL+33578/data=!4m14!4m13!1m1!4e1!1
4177	/dir/27.9507488,-82.4588339/US+Post+Office,+11126+US+Hwy+41+S,+Gibson,+FL+33534/data=!4m14!4m13!1m1!4e:
5236	/dir/27.9507488,-82.4588339/US+Post+Office,+11126+US+Hwy+41+S,+Gibson,+FL+33534/data=!4m14!4m13!1m1!4e:
6418	/dir/27.9507488,-82.4588339/US+Post+Office,+11126+US+Hwy+41+S,+Gibson,+FL+33534/data=!4m14!4m13!1m1!4e:
6738	/dir/27.7207419,-82.4336623/Dunkin'+Donuts,+6190+N.+US+Highway+41,+Apollo+Beach,+FL+33572,+United+States/da
7278	/dir/27.7291619,-82.4337426/Sam's+Club,+10385+Big+Bend+Road,+Riverview,+FL+33578/data=!4m14!4m13!1m1!4e1!1
7429	/dir/27.7291619,-82.4337426/Sam's+Club,+10385+Big+Bend+Road,+Riverview,+FL+33578/data=!4m14!4m13!1m1!4e1!1
7570	/dir/27.7207334,-82.4337281/Riverview,+FL+33579/data=!4m14!4m13!1m1!4e1!1m5!1m1!1s0x88c2d475c820ce31:0xf5b
8119	/dir/28.506043,-81.4216954/University+of+Central+Florida,+4000+Central+Florida+Blvd,+Orlando,+FL+32816/data=!4m
9331	/dir/27.7291619,-82.4337426/Sam's+Club,+10385+Big+Bend+Road,+Riverview,+FL+33578/data=!4m14!4m13!1m1!4e1!1
10038	/dir/27.7207419,-82.4336623/Dunkin'+Donuts,+6190+N.+US+Highway+41,+Apollo+Beach,+FL+33572,+United+States/da
10266	/dir/27.7207334,-82.4337281/Riverview,+FL+33579/data=!4m14!4m13!1m1!4e1!1m5!1m1!1s0x88c2d475c820ce31:0xf5b
10415	/dir/27.7291619,-82.4337426/Sam's+Club,+10385+Big+Bend+Road,+Riverview,+FL+33578/data=!4m14!4m13!1m1!4e1!1

Figure 36 - Google Maps directions artifacts recovered via Strings command utility from "gmm_storage.db" database

It's also important to note that embedded toward the end of these entries is a UNIX epoch timestamp. Figure 37 shows this timestamp highlighted for record number 5236, which when converted to readable time using DCode is: Saturday, 27 June 2015 at 22:40:00. UTC. A review of test session documentation confirms that a Google Maps search was indeed performed using the test device from which this database was recovered. Curiously, the timestamp converts to a UTC value which should be four hours ahead of local time for the test location timezone, but the converted timestamp is actually consistent with the actual local time at the time of the Google Maps search. Incidentally, the source latitude and longitude from which the search was executed appear at the beginning of the record and are extremely accurate based on the test session

documentation. The corresponding cloud location for the same time was also quite accurate, with all sensors active and the device located outdoors in an urban environment.

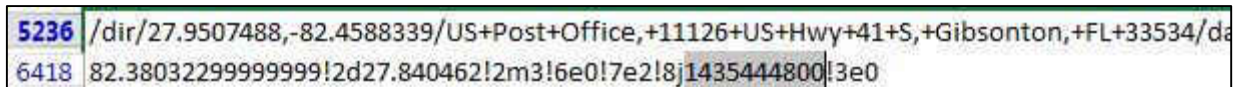


Figure 37 - Timestamp from Google Maps "gmm_storage.db" database directions search record

This is just one example of how examiners need to remain open to digging deeper than what the paid forensic tools automatically parse, in their efforts to recover Android location data. Beginning with SQLite databases and XML configuration files associated with known location-accessing apps, examiners should perhaps worry less about reconstructing the particular activity or resources associated with recovered location information and more about the details of the app they have recovered the data from.

For example, based on this study's results, it would seem that location sharing and fitness apps cache extremely precise and reliable data, typically. Furthermore, navigation apps, which are designed for safety reasons to provide a very accurate idea of the user's location, store very reliable information, though perhaps with less extensive local caching. This makes sense, since users of fitness apps may want to review old workout routes and times to track their progress, but users of navigation apps are more likely to search their current destination of choice each session. Furthermore, if an app has only coarse location permissions, it may not be the most reliable of sources of location data, though certainly still good enough to refute an individual's argument that they were in another town at the time, for example. Weather apps and ad-supported games can fall into this latter category.

In cases where investigators are particularly determined to use device location data to pinpoint a user’s whereabouts, it could be helpful to check for files that may contain timestamped logs of network connectivity or sensor states, even though these logs could be associated with other apps that stored no location information. Keyword searches will do little to directly recover latitude and longitude coordinates, but searching for terms like “latitude REAL” or “lat DOUBLE” may help identify active or deleted SQLite database content that may be of value. Databases and unallocated space can be carved for geo-tagged pictures or map tile images, as well, though attributing context to these can be difficult, as it is hard to say if they reflect a device’s actual location or just a browsing session or location search.

Also, if an examiner has a particular artifact of interest recovered from a known application, he or she could utilize a test device, install the application of interest, and run some tests to try to evaluate that app’s reliability in terms of location accuracy. From this study, it was noted that some apps will not perform their designated function unless all location resources are enabled, like Waze navigation, for example. Furthermore, some apps actually inform the user during the use of accuracy estimates, as noted for Life360 and Friend Locator. Table 18 summarizes some location data recovery and evaluation strategies utilized in this study.

Table 18 - Recovery and evaluation strategies

Tier 1 – The obvious	Tier 2 – Dig deeper	Tier 3 – Beyond the deep end
Forensic tools’ parsed artifacts	Identify apps with location permissions	Carve for images, SQLite databases
SQLite databases	Examine SQLite databases and XML configuration files associated with these apps	Keyword searches for possible metadata/code terms
Geo-tagged media files		Test device analysis on app of interest
Map Tiles		

Finally, examiners may wish to think beyond the device itself. Using call detail records and cloud resources may allow for the corroboration of points recovered from the device, as well as an expansion of the location data set. Although the data obtained via Cloud Analyzer was less precise, it has a number of advantages. First, the user does not need to initiate any process for the Google location service to begin collecting device location information, beyond agreeing to use Google's location services. So the process occurs in the background, virtually all of the time, unlike most apps that cache location data which require a user to initiate a session or event. Furthermore, the coordinates are updated almost every minute or so, so the information is very dense and can indicate dwelling or travel, etc. Finally, because it is a background service, it is unlikely a user could or would disable it, leading to potentially large volumes of location data during timeframes when the device might otherwise not cache any.

Limitations and Future Research

This research was performed with some fundamental limitations. First, it obviously utilized a small sample size of just four unrooted test devices, and the device and application settings were optimized for location caching, which may not be typical configurations. Second, the number and variety of test sessions clearly cannot account for all possible conditions in which devices are used. Testing was mostly performed in suburban (residential) areas, with access to rural and urban areas less frequent. Finally, devices were obviously accessible and in an unlocked state, mitigating the potential hurdles to data extraction one might encounter in forensic casework. This study does not address means of gaining access to locked devices or encrypted data.

Based on the emerging trends noted, these last issues could become serious impediments to data recovery from Android devices. It remains to be seen how encryption will impact Android forensics in the future. Furthermore, the shift to cloud storage definitely warrants future study. Investigators and legal professionals need to be aware of these potential resources. Google Location History, for example, could be used to refute or confirm an individual's alibi, especially if there is corroborating evidence showing the individual was in possession of the device at the time of the incident. However, confusion about legal issues of jurisdiction and authority surrounding the retrieval of this data certainly suggest that the criminal justice community will need to continue learning about the technical and logistical details involved in the process.

Additional research into the Cloud Analyzer's thousandths place precision level and its relationship to the Google user account location history could also be interesting and beneficial. Furthermore, one could investigate how long Google location history is maintained and if any data can be retrieved using the Cloud Analyzer software after a user opts to delete it via the Google account interface. It may also be of interest to further study the cloud data to determine if the trend of no cloud data points being collected when cellular service is inactive holds true, including whether or not airplane mode impacts this.

Perhaps some evolution in both investigative strategies and legal statutes is needed. Certainly, this issue will only become more prominent as providers continue to emphasize security and move toward cloud-based data storage. It would also be a very useful study to develop a method for automating some of the techniques outlined in this paper. The volume and complexity of data cached on these devices means manual review and correlation is tedious and time-consuming. Finally, because there are now varied resources, from the carrier records to the

device data to the cloud information, investigators should consider exploring all of these avenues in their cases. An automated solution capable of incorporating all of these different resources and streamlining their review would be a very valuable tool indeed.

CHAPTER FIVE: CONCLUSION

Smartphones and tablets are becoming more and more entrenched in everyday life, and users knowingly and unknowingly store intimate information within these devices. Because of the location aware features offered by mobile devices, as well as users' tendency to keep them on their person much of the time, device location history is both a real phenomenon and a potentially valuable resource in criminal investigations. Android devices make up a majority of the mobile market share, and this study's exploration of their capabilities, strategies, and stored data indicates that they do indeed possess large amounts of historical location data associated with a particular device, and by extension, its owner.

One critical point to consider is that mobile device investigations now go beyond the phone itself, extending not just to carrier records but also to potential cloud artifacts. As this study demonstrated, vast quantities of Android device location points are archived on Google's servers. This information is updated often enough to demonstrate if a user is dwelling or traveling, and to provide an estimate of their location likely accurate to within 100 meters, or less. Furthermore, what really sets this resource apart is its omnipresent activity, running in the background without any need for user-initiated sessions or events. Potential impediments to the acquisition and use of the cloud resource include issues involving access or credentials, as well as perceived ambiguity regarding the legality of obtaining the cloud data via a commercial tool, such as CelleBrite's Cloud Analyzer software.

In terms of the devices themselves, examiners should begin by examining data associated with applications known to have permissions to the device location. Text-based files may be triaged via keyword searching, and this technique may also extend to SQLite database column

headers, though not floating point values, which is typically how latitude and longitude coordinates are stored within the databases. Databases should be reviewed for location artifacts. This study describes a number of example databases and their content. Depending on the type of application, the cached location points could reflect the device's actual location, or some searched point of interest or destination. In other words, a device's location at a particular time may be either directly documented, or it may be inferred based on a viewed point of interest nearby, or a navigation route with transcribed directions.

Different types of apps seemed to handle location data differently, with almost all requiring some sort of user-initiated impetus to record the information. Fitness, location sharing, and chat apps appear to store device locations with high precision. In general, these apps also seem to require GNSS sensors to be enabled, although perhaps less so for the location sharing apps. The update intervals are tied to the user activity. When a user employs a fitness app, the device location may be updated up to every few seconds, as opposed to a chat app in which the location is recorded only when a message is sent, with the proper settings configured to enable location sharing.

Navigation and leisure apps also require user interaction, but these apps may store locations more likely to be searched or viewed by the user, not necessarily the location of the device itself. The cached locations do appear to be quite precise, but update intervals are again tied to the user's activity. In the case of leisure apps, and the Waze navigation app's transcribed turn-by-turn directions, the device's location could possibly be inferred from these apps' data. Weather apps and ad-supported games did cache some location artifacts but were not as reliable in terms of accuracy, and the update intervals were intermittent.

It may also benefit an examiner to look further than the files or apps that may store location data itself. Various log files may offer additional information regarding sensor activity that could prove valuable in evaluating the accuracy of any other recovered device locations. While GNSS sensor activity logs were lacking in this research, WLAN connectivity was documented by several apps in several different contexts. Other log files monitored app usage, which could be of interest when dealing with an app that requires GNSS operations to perform particular functions or cache data. For example, if a coordinate pair of interest is found in an examination, and then details from such a seemingly unrelated log files regarding app usage reveal that a separate app, which requires GNSS services, was active at the time, it may bolster confidence in the reliability or accuracy of the particular point.

Previous studies regarding accuracy trends noted during live tracking of test devices noted that reliability improved whenever GNSS resources are used, with location points derived merely from cell tower signals proving least accurate. Somewhere in between either extreme, WLAN-supplemented services were reported as having decent accuracy of within 100 meters or so. Perhaps unsurprisingly, these trends were corroborated in this study by the location data recovered from the mobile devices themselves. The maximum error in this testing was roughly around 1.5 kilometers, with a minimum error of within 3 meters noted. The maximum error was in fact obtained when only cellular services were active, and in a rural area. The minimum, however, was associated with a test session in which cellular, WLAN, and GNSS sensors were all active, and in a suburban area.

Such an accuracy range is likely adequate for refuting an alibi, but investigators may wish to have a better idea of how accurate a particular cached coordinate pair may be.

Fortunately, multiple resources exist to facilitate such an effort, in the form of metadata and logs stored on the devices themselves, to the option to review carrier records and cloud data as corroborative sources. Examiners should not rely solely on commercial tools to parse location data for them. They should actively search application data for possible contemporaneous artifacts that may provide insight or verify a particular point, focusing especially on databases and text files. The search should initially focus on apps with location permissions, but one should not overlook the potentially valuable logging contributions of the other app types.

These strategies, along with the incorporation of external resources such as carrier records and cloud data, may enable an examiner to be able to do more than just recover the location artifacts, but also comment on their potential reliability as well. As device storage and extraction techniques evolve, and cloud storage becomes more prominent, mobile device forensic examinations will need to evolve, as well. Understanding the device functionality and all of the possible resources available will help ensure that practitioners perform the most effective and complete analyses they can, improving investigations and promoting informed adjudication of civil investigations or criminal cases.

APPENDIX A: TEST SESSION WORKSHEET

Masters Thesis – Test Phone Worksheet

Connie Bell

Date: _____

Start Time: _____

End Time: _____

Location: _____

Test Device Used:

LG VS870 One + One (A0001) Samsung SGH-i257 Samsung SM-G900P

Environment: Indoors Outdoors Urban Suburban Rural

Active Services: Wi-Fi Cellular GNSS

Weather Conditions:

-
- 5 Calls Out
 - 5 Calls In
 - 5 SMS messages sent
 - 5 SMS messages received
 - 5 geo-tagged photos taken
 - 5 MMS messages sent (with geo-tagged photo attached)
 - 5 MMS messages received
 - 5 chat messages sent (per app)
 - 5 chat messages received (per app)
 - 5 Google Maps searches (stationary) 5 Google Maps searches* (transient)
 - 3 Ad-supported game sessions
 - 1 Weather check (per app)
 - 1 Navigation route* (per app)
 - 1 Running/Walking route* (per app)
 - 5 Google Earth searches
 - 5 Yelp/Foursquare/Field Trip/~~Shopular~~ searches (stationary) 5 Yelp/Foursquare/Field Trip searches* (transient)
 - 1 Location share session (per app)
 - 5 Web browser searches via various websites (check Browser CachedGeoposition and localstorage databases)

APPENDIX B: APPS AND SETTINGS INFORMATION

Table 19 - Apps and Settings Information

Category	Apps Tested	Custom Location Settings	Location Permissions
Navigation	Google Maps Waze	Google Location Settings enabled <i>No track log settings noted</i>	<input checked="" type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine <input checked="" type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine
Fitness	RunKeeper Map My Walk My Tracks	Linked to Facebook account Linked to Facebook account Set recording time interval to smallest option Recording distance interval set to 32 feet Default track name set to Date and Location	<input type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine <input checked="" type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine <input checked="" type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine
Location Sharing	Glympse Life360 Locate My Friends Swarm	<i>No logging options noted</i> Linked to Facebook Location sharing enabled Location sharing enabled Linked to Facebook	<input checked="" type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine <input checked="" type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine <input checked="" type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine <input checked="" type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine
Chat	Facebook Messenger WhatsApp Viber LINE	Messages include location <i>No location options noted</i> <i>No location options noted</i> <i>No location options noted</i>	<input checked="" type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine <input checked="" type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine <input checked="" type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine <input checked="" type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine
Weather	AccuWeather One Weather GO Weather Weather Bug	Enabled alerts, Use current location “Follow my location” enabled Use current location “Enable my location” checked	<input checked="" type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine <input checked="" type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine <input checked="" type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine <input checked="" type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine
Ad-Supported Games	Words With Friends Trivia Crack Quiz Up	Linked to Facebook Linked to Facebook Linked to Facebook	<input type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine <input checked="" type="checkbox"/> Coarse <input type="checkbox"/> Fine <input type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine
Leisure	Yelp Foursquare Field Trip	<i>No location options noted</i> Location services enabled <i>No location options noted</i>	<input checked="" type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine <input checked="" type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine <input checked="" type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine
Other	Location Tracker *(OnePlus One A0001 only) GPS Status Tracker	Time interval set to smallest (every 5 minutes) Location resource options: Mobile/WLAN, GPS, or Both NMEA logging enabled	<input checked="" type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine <input checked="" type="checkbox"/> Coarse <input checked="" type="checkbox"/> Fine

APPENDIX C: TEST SESSIONS

Table 20 - Test Session Information

Date	Device(s) Used	Environment	Setting Category	Active Sensors
May 27	OnePlus One LG VS870	<input checked="" type="checkbox"/> Indoor <input type="checkbox"/> Outdoor	<input type="checkbox"/> Rural <input checked="" type="checkbox"/> Suburban <input type="checkbox"/> Urban	<input checked="" type="checkbox"/> GNSS <input checked="" type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular
June 6	OnePlus One LG VS870	<input type="checkbox"/> Indoor <input type="checkbox"/> Outdoor	<input type="checkbox"/> Rural <input checked="" type="checkbox"/> Suburban <input type="checkbox"/> Urban	<input checked="" type="checkbox"/> GNSS <input checked="" type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular
June 9	OnePlus One LG VS870	<input type="checkbox"/> Indoor <input checked="" type="checkbox"/> Outdoor	<input type="checkbox"/> Rural <input checked="" type="checkbox"/> Suburban <input type="checkbox"/> Urban	<input type="checkbox"/> GNSS <input type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular
June 13	SM-G900P S5	<input checked="" type="checkbox"/> Indoor <input type="checkbox"/> Outdoor	<input type="checkbox"/> Rural <input checked="" type="checkbox"/> Suburban <input type="checkbox"/> Urban	<input type="checkbox"/> GNSS <input checked="" type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular
June 14	SM-G900P S5 SGH-i257 S4 Mini	<input checked="" type="checkbox"/> Indoor <input type="checkbox"/> Outdoor	<input type="checkbox"/> Rural <input checked="" type="checkbox"/> Suburban <input type="checkbox"/> Urban	<input checked="" type="checkbox"/> GNSS <input checked="" type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular
June 19	SGH-i257 S4 Mini	<input type="checkbox"/> Indoor <input checked="" type="checkbox"/> Outdoor	<input type="checkbox"/> Rural <input checked="" type="checkbox"/> Suburban <input type="checkbox"/> Urban	<input checked="" type="checkbox"/> GNSS <input checked="" type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular
June 27	SGH-i257 S4 Mini SM-G900P S5	<input type="checkbox"/> Indoor <input checked="" type="checkbox"/> Outdoor	<input type="checkbox"/> Rural <input type="checkbox"/> Suburban <input checked="" type="checkbox"/> Urban	<input checked="" type="checkbox"/> GNSS <input checked="" type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular
July 14	OnePlus One LG VS870	<input checked="" type="checkbox"/> Indoor <input type="checkbox"/> Outdoor	<input type="checkbox"/> Rural <input checked="" type="checkbox"/> Suburban <input type="checkbox"/> Urban	<input type="checkbox"/> GNSS <input type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular
July 15	OnePlus One LG VS870	<input checked="" type="checkbox"/> Indoor <input type="checkbox"/> Outdoor	<input checked="" type="checkbox"/> Rural <input type="checkbox"/> Suburban <input type="checkbox"/> Urban	<input checked="" type="checkbox"/> GNSS <input checked="" type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular
July 15	OnePlus One LG VS870	<input type="checkbox"/> Indoor <input checked="" type="checkbox"/> Outdoor	<input checked="" type="checkbox"/> Rural <input type="checkbox"/> Suburban <input type="checkbox"/> Urban	<input checked="" type="checkbox"/> GNSS <input checked="" type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular
July 17	OnePlus One LG VS870	<input type="checkbox"/> Indoor <input checked="" type="checkbox"/> Outdoor	<input checked="" type="checkbox"/> Rural <input checked="" type="checkbox"/> Suburban <input type="checkbox"/> Urban	<input checked="" type="checkbox"/> GNSS <input checked="" type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular
July 19	OnePlus One	<input type="checkbox"/> Indoor <input checked="" type="checkbox"/> Outdoor	<input type="checkbox"/> Rural <input checked="" type="checkbox"/> Suburban <input type="checkbox"/> Urban	<input checked="" type="checkbox"/> GNSS <input checked="" type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular

Date	Device(s) Used	Environment	Setting Category	Active Sensors
August 6	OnePlus One SM-G900P S5	<input type="checkbox"/> Indoor <input checked="" type="checkbox"/> Outdoor	<input checked="" type="checkbox"/> Rural <input type="checkbox"/> Suburban <input type="checkbox"/> Urban	<input checked="" type="checkbox"/> GNSS <input checked="" type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular
September 6	OnePlus One	<input checked="" type="checkbox"/> Indoor <input type="checkbox"/> Outdoor	<input type="checkbox"/> Rural <input checked="" type="checkbox"/> Suburban <input type="checkbox"/> Urban	<input checked="" type="checkbox"/> GNSS <input checked="" type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular
September 8	OnePlus One	<input checked="" type="checkbox"/> Indoor <input type="checkbox"/> Outdoor	<input type="checkbox"/> Rural <input checked="" type="checkbox"/> Suburban <input type="checkbox"/> Urban	<input checked="" type="checkbox"/> GNSS <input checked="" type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular
September 16	OnePlus One LG VS870	<input checked="" type="checkbox"/> Indoor <input type="checkbox"/> Outdoor	<input type="checkbox"/> Rural <input checked="" type="checkbox"/> Suburban <input type="checkbox"/> Urban	<input type="checkbox"/> GNSS <input checked="" type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular
September 16	SGH-i257 S4 Mini	<input type="checkbox"/> Indoor <input checked="" type="checkbox"/> Outdoor	<input type="checkbox"/> Rural <input checked="" type="checkbox"/> Suburban <input type="checkbox"/> Urban	<input checked="" type="checkbox"/> GNSS* <input checked="" type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular
September 18	OnePlus One SM-G900P S5	<input checked="" type="checkbox"/> Indoor <input type="checkbox"/> Outdoor	<input type="checkbox"/> Rural <input checked="" type="checkbox"/> Suburban <input type="checkbox"/> Urban	<input type="checkbox"/> GNSS <input checked="" type="checkbox"/> WLAN <input type="checkbox"/> Cellular
September 19	LG VS870 SGH-i257 S4 Mini	<input type="checkbox"/> Indoor <input checked="" type="checkbox"/> Outdoor	<input checked="" type="checkbox"/> Rural <input type="checkbox"/> Suburban <input type="checkbox"/> Urban	<input checked="" type="checkbox"/> GNSS <input checked="" type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular
September 19	LG VS870	<input type="checkbox"/> Indoor <input checked="" type="checkbox"/> Outdoor	<input checked="" type="checkbox"/> Rural <input type="checkbox"/> Suburban <input type="checkbox"/> Urban	<input type="checkbox"/> GNSS <input type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular
September 19	SGH-i257 S4 Mini	<input type="checkbox"/> Indoor <input checked="" type="checkbox"/> Outdoor	<input type="checkbox"/> Rural <input checked="" type="checkbox"/> Suburban <input type="checkbox"/> Urban	<input checked="" type="checkbox"/> GNSS <input type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular
September 19	LG VS870 SGH-i257 S4 Mini	<input type="checkbox"/> Indoor <input checked="" type="checkbox"/> Outdoor	<input type="checkbox"/> Rural <input type="checkbox"/> Suburban <input checked="" type="checkbox"/> Urban	<input checked="" type="checkbox"/> GNSS* <input checked="" type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular
October 8	OnePlus One	<input type="checkbox"/> Indoor <input checked="" type="checkbox"/> Outdoor	<input checked="" type="checkbox"/> Rural <input checked="" type="checkbox"/> Suburban <input type="checkbox"/> Urban	<input checked="" type="checkbox"/> GNSS <input checked="" type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular
October 8	OnePlus One	<input type="checkbox"/> Indoor <input checked="" type="checkbox"/> Outdoor	<input checked="" type="checkbox"/> Rural <input checked="" type="checkbox"/> Suburban <input type="checkbox"/> Urban	<input type="checkbox"/> GNSS <input checked="" type="checkbox"/> WLAN <input checked="" type="checkbox"/> Cellular

*GNSS enabled for navigation sessions only. All other test activity performed with no GNSS.

APPENDIX D: ANALYSIS TOOLS

Table 21 - Analysis tools used in this study

Tool (Vendor/Tool Name)	Version	Purpose/Usage
CelleBrite UFED 4PC	4.2.2.86	Extraction of data from test devices
CelleBrite Physical Analyzer	4.2.6.4	Analysis of data extracted from test devices
CelleBrite UFED Cloud Analyzer	4.3.0.412	Collection of Google Location History data for test devices
Magnet Forensics Internet Evidence Finder	6.6.3.0736	Analysis of data extracted from test devices
Magnet Forensics ACQUIRE	0.6.0.0351	Extraction of data from test devices
Kali Linux	1.0 (64-bit)	Extraction of data from test device
Guidance Software EnCase	7.10.05.11	Analysis of data extracted from test devices
SQLite Studio	2.1.4	Analysis of databases extracted from test devices
DCode	4.02a	Timestamp conversion/analysis
Microsoft Excel	Pro 2013	Analysis of databases and cloud data extracted from test devices
Earth Point Excel to KML	2015	Conversion of Excel content to KML format for use in Google Earth
Google Earth	7.1.5.1557	Import of recovered location data for creation of figures
X-Ways Forensics	18.5	SQLite database carving, keyword searches
Strings	2.51	Analysis of Google Maps “gmm_storage.db” database
Coordinate Distance Calculator	N/A (web tool)	Web-based calculator to determine distance between two sets of latitude/longitude coordinates
Movable Type Scripts – Calculate Distance between Latitude/Longitude Coordinate Points	N/A (web tool)	Secondary (verification) tool for distance between two sets of latitude/longitude coordinates

APPENDIX E: POSSIBLE KEYWORD SEARCH TERMS

Table 22 - Suggested keywords for recovery of metadata, application code, or sensor activity

Request Types	Metadata	Active Services	Databases of Interest
PRIORITY_BALANCED_POWER_ACCURACY PRIORITY_HIGH_ACCURACY PRIORITY_LOW_POWER PRIORITY_NO_POWER	accuracy altitude elevation recent	connectionType mobile WIFI BSSID SSID	latitude REAL latitude DOUBLE latitude INTEGER lat REAL lat DOUBLE lat INTEGER

APPENDIX F: SQLITE QUERIES USED

Table 23 - SQLite query details

Database Filename	Associated App	Query Used
360LocationDB	Life360, Locate My Friends	select locations.time, datetime((locations.time)/1000,'unixepoch') as "Converted Time (UTC)", locations.lat, locations.lon, locations.accuracy, locations.speed, locations.altitude, locations.bearing, locations.provider from locations order by locations.time asc
dumpLogsDatabase	Locate My Friends	select logsTable.log, logsTable.utc, datetime((logsTable.utc)/1000,'unixepoch') as "Converted Time (UTC)" from logsTable order by logsTable.utc asc
event	Amazon preinstalled app	select events.body, events.timestamp, datetime((events.timestamp)/1000,'unixepoch') as "Converted Timestamp (UTC)" from events order by events.timestamp asc
Forecast_accu.db	AccuWeather	select forecasts.current_city_flag, forecasts.device_updated_millis, datetime((forecasts.device_updated_millis)/1000,'unixepoch') as "Converted Timestamp (UTC)", forecasts.city, forecasts.country, forecasts.lat, forecasts.lon from forecasts
Fsq.db	Foursquare, Swarm	select comments.createdAT, datetime((comments.createdAt),'unixepoch') as "Converted Time (UTC)", comments.lat, comments.lng, comments.geoId, comments.contextLine from comments order by comments.createdAT asc
herrevad	Google Mobile Services	select local_reports.network_type, local_reports.ssid, local_reports.security_type, local_reports.bssid, local_reports.timestamp_millis, datetime((local_reports.timestamp_millis)/1000,'unixepoch') as "Converted timestamp (UTC)" from local_reports order by local_reports.timestamp_millis asc

Database Filename	Associated App	Query Used
Messaging.db	Life360, Locate My Friends	select message.content, thread_participant.participant_name, message.created_at, datetime((message.created_at),'unixepoch') as "Converted Time (UTC)", message.has_location, message.location_latitude, message.location_longitude, message.location_timestamp, datetime((message.location_timestamp),'unixepoch') as "Converted Time (UTC)", message.location_accuracy, message.location_address1, message.location_address2 from message, thread_participant where message.sender_id=thread_participant.participant_id order by message.created_at asc
Mytracks.db	My Tracks	select tracks.name, trackpoints.longitude, trackpoints.latitude, trackpoints.time, datetime((trackpoints.time)/1000,'unixepoch') as "Converted Time (UTC)", trackpoints.elevation, trackpoints.accuracy, trackpoints.speed, trackpoints.bearing, trackpoints.sensor from tracks, trackpoints where tracks._id=trackpoints.trackid order by trackpoints.time asc
Oneweather.db	1 Weather	select geocodes.city, geocodes.state, geocodes.country, geocodes.lat, geocodes.lng, geocodes.lastHit, datetime((geocodes.lastHit)/1000,'unixepoch') as "Converted Timestamp (UTC)", geocodes.hits from geocodes order by geocodes._id asc
RunKeeper.sqlite	RunKeeper	select points.trip_id, trips.start_date, datetime((trips.start_date+points.time_interval_at_point*1000)/1000,'unixepoch') as "Converted Time", points.latitude, points.longitude, points.altitude, points.time_interval_at_point, points.speed_from_last_point, points.distance_from_last_point, points.point_type, points.accuracy, points.distance_at_point from points, trips where points.trip_id=trips._id order by points._id asc
Tts.db	Waze	select Jane.text, Jane.path, Jane.update_time, datetime((Jane.update_time),'unixepoch') as "Converted Time (UTC)" from Jane order by Jane.update_time asc

Database Filename	Associated App	Query Used
User.db	Waze	<pre>select places.name, places.street, places.city, places.state, places.house, places.longitude, places.latitude, places.venue_id, places.created_time, datetime((places.created_time),'unixepoch') as "Converted Created Time (UTC)", recents.access_time, datetime((recents.access_time),'unixepoch') as "Converted Access Time (UTC)" from places, recents where places.id=recents.place_id order by places.id asc</pre>
Viber_messages	Viber	<pre>select messages.date, datetime((messages.date)/1000,'unixepoch') as "Converted Time (UTC)", messages.type, messages.body, messages.location_lat, messages.location_lng, participants_info.display_name, messages.deleted from messages, participants_info where messages.participant_id=participants_info._id order by messages.date asc</pre>
Weather.db	GO Weather	<pre>select citynow.myLocation, citynow.cityName, citynow.updateTime, datetime((citynow.updateTime)/1000,'unixepoch') as "Converted Time (UTC)", citynow.city_my_location, citynow.state, citynow.country, citynow.timestamp, datetime((citynow.timestamp)/1000,'unixepoch') as "Converted Timestamp (UTC)", citynow.latitude, citynow.longitude from citynow order by citynow.updateTime asc</pre>
Workout.db	Map My Walk	<pre>select workouts.name, timeSeries.timestamp, datetime((timeSeries.timestamp)/1000,'unixepoch') as "Converted Time (UTC)", timeSeries.distance, timeSeries.speed, timeSeries.longitude, timeSeries.latitude, timeSeries.altitude from timeSeries, workouts where timeSeries.localID=workouts.localId order by timeSeries.timestamp asc</pre>

REFERENCES

- 4RENSIKER. (2012, February). Android tracking – from a forensic point of view. [Weblog article]. Retrieved from: <http://articles.forensicfocus.com/2012/02/27/android-tracking-from-a-forensic-point-of-view/>
- American Civil Liberties Union. (2014, November 17). *AT&T comes out in support of stricter standards for police cell location phone tracking*. Retrieved from: <https://www.aclu.org/technology-and-liberty/att-comes-out-support-stricter-standards-police-cell-location-phone-tracking>
- Bairstow, D. (2015, July). The difference between location accuracy and precision and why you need to know. [Weblog article]. Retrieved from: <http://blog.skyhookwireless.com/advertising/the-difference-between-location-accuracy-and-precision-and-why-you-need-to-know>
- Barmpatoulou, K., Damopoulos, D., Kambourakis, G., & Katos, V. (2013). A Critical Review of 7 Years of Mobile Device Forensics. *Digital Investigation*, 10. Retrieved from: http://www.cs.stevens.edu/~ddamopou/files/A_critical_review_of_7_years_of_Mobile_Device_Forensics.pdf#%FE%FF%00b%00i%00b%006%007
- Blank, A. (2011). The limitations and admissibility of using historical cellular site data to track the location of a cellular phone. *Richmond Journal of Law & Technology*, XVIII, issue 1. Retrieved from: <http://jolt.richmond.edu/v18il/article3.pdf>
- Brouwers, N. & Woehrle, N. (2012, July). Dwelling in the canyons: Dwelling detection in urban environments using GPS, Wi-Fi, and geolocation. *Pervasive and Mobile Computing*, 9. Retrieved from: <http://www.sciencedirect.com/science/article/pii/S1574119212000752>

- Cai, C. & Gao, Y. (2013). GLONASS-based precise point positioning and performance analysis. *Advances in Space Research*, 51. Retrieved from:
<http://www.sciencedirect.com.ezproxy.net.ucf.edu/science/article/pii/S0273117712005285>
- CelleBrite Mobile Synchronization, LTD. (2015). *Extracting Legally Defensible Evidence from the Cloud*. Retrieved from:
http://www.cellebrite.com/Media/Default/Files/Forensics/White-Papers/Extracting-Legally-Defensible-Evidence-From-Cloud_WhitePaper.pdf
- Cunningham, A. (2015, October). Android 6.0 re-implements mandatory storage encryption for new devices. Retrieved from: <http://arstechnica.com/gadgets/2015/10/android-6-0-re-implements-mandatory-device-encryption-for-new-devices/>
- Daniel, L. (2014). Cell phone tracking evidence. Excerpt from the upcoming book: *Cellular location evidence for legal professionals*. Retrieved from:
<http://www.ncids.org/Defender%20Training/2014SpringConf/CellPhoneTracking.pdf>
- Davydov, O. (2011). Proceedings from Techno Forensics 2011: *Geo-location Data in iOS and Android Services and Applications: Finding, Processing, and Validation*. Myrtle Beach, SC. Retrieved from: http://android-forensics.com/download/presentation/Oxygen_Software_Geo_location_Android.pdf
- Fox, A. (2015, January 28). Better cell phone location accuracy for emergency calls needed. *amNewYork*. Retrieved from: <http://www.amny.com/news/better-cell-phone-location-accuracy-for-emergency-calls-needed-1.9876745>
- International Association of Computer Investigative Specialists (IACIS). (2015). *Procedure for gaining physical access to certain Android phone running Kit Kat*. Michael, Ed.

Kroger, K. & Creutzberg, R. (2012). Proceedings from SPIE 2012: *Forensics of Location Data Collected by Google Android Mobile Devices*. San Diego, CA.

Last, D. (2015, May). Understanding location information recovered from mobile devices. [Webinar]. Retrieved from: <http://www.cellebrite.com/Mobile-Forensics/Webinars/understanding-location-information-extracted-from-mobile-devices>

Lifchitz, R. (2010). Proceedings from the 27th Chaos Communication Congress: *Android Geolocation Using GSM Network: Where was Waldroid?* Berlin, Germany. Retrieved from: http://events.ccc.de/congress/2010/Fahrplan/attachments/1781_27c3-android-geolocation.pdf

Making your app location aware [online training series].

Retrieved from Android Developer website:

<https://developer.android.com/training/location/index.html>

Mander, J. (2014, November 18). *GWI Device Summary: Q3 2014*. Retrieved from: http://cdn2.hubspot.net/hub/304927/file-2301369304-pdf/Reports/GWI_Device_Summary_Q3_2014.pdf?submissionGuid=02e0203b-d994-4909-be35-d5fde304b774

Maus, S., Hofken, H., & Schuba, M. (2011). Proceedings from Cyberforensics 2011: *Forensic Analysis of Geodata in Android Smartphones*. Glasgow, Scotland, UK. Retrieved from: <http://www.schuba.fh-aachen.de/papers/11-cyberforensics.pdf>

Michael, K. & Clarke, R. (2013). Location and tracking of mobile devices: Uberveillance stalks

- the streets. *Computer Law & Security Review*, 29. Retrieved from:
<http://www.sciencedirect.com.ezproxy.net.ucf.edu/science/article/pii/S0267364913000587>
- Paiva, S. & Abreu, C. (2012). Proceedings from HCIST 2012 – International Conference on Health and Social Care Information Systems and Technologies: *Low Cost GPS Tracking for the Elderly and Alzheimer Patients*. Algarve, Portugal. Retrieved from:
<http://www.sciencedirect.com/science/article/pii/S2212017312005191>
- Pew Research Center. (2014). *Mobile Technology Fact Sheet*. Retrieved from:
<http://www.pewinternet.org/fact-sheets/mobile-technology-fact-sheet/>
- Racioppo, C. & Murthy, N. (2012). Proceedings from Student-Faculty Research Day, CSIS 2012: *Android Forensics: A Case Study of the “HTC Incredible” Phone*. Pace University, NY. Retrieved from: <http://csis.pace.edu/~ctappert/srd2012/b6.pdf>
- RECENT Case: Sixth Circuit holds that “pinging” a target’s cell phone to obtain GPS data is not a search subject to the warrant requirement. (2013, January). *Harvard Law Review*, 126:802. Retrieved from: http://harvardlawreview.org/wp-content/uploads/pdfs/vol126_united_states_v_skinner.pdf
- Reisinger, D. (2013, November). 10 Android location-based apps to help you find your way around. Retrieved from: <http://www.eweek.com/mobile/slideshows/10-android-location-based-apps-to-help-you-find-your-way-around.html>
- Riley, S. (2015, August). Should you root your Android phone? Retrieved from:
<http://www.tomsguide.com/us/should-you-root-your-phone,review-2999.html>
- Sack, S., Kroger, K., & Creutzburg, R. (2012). Overview of potential forensic analysis of an

Android smartphone. Retrieved from:

http://www.researchgate.net/publication/258332974_Overview_of_potential_forensic_analysis_of_an_Android_smartphone

Saliba, J. (2013, December). Using geolocation artifacts and timeline analysis to solve the case: a digital forensics case study. [Webinar]. Retrieved from:

<http://www.forensicfocus.com/c/aid=69/webinars/2013/using-geolocation-artifacts-and-timeline-analysis-to-solve-the-case-a-digital-forensics-case-study/>

Schumer, C. (2015, January 28). Schumer: First responders too often can't locate victims when 911 calls are made from cell phones; could lead to potentially deadly delays in emergency response time. [Press release]. Retrieved from:

http://www.schumer.senate.gov/newsroom/press-releases/schumer-first-responders-too-often-cant-locate-victims-when-9-1-1-calls-are-made-from-cell-phones-could-lead-to-potentially-deadly-delays-in-emergency-response-time_schumer-urges-fcc-to-approve-and-implement-strong-new-rules-during-meeting-tomorrow-to-strengthen-9-1-1-call-accuracy-from-cell-phones

Selyukh, A. (2014, November 16). Deal to spur better 911 call locating for U.S. cellphone users.

Retrieved from: <http://www.reuters.com/article/2014/11/15/us-usa-wireless-idUSKCN0IZ00L20141115>

Sisak, M. R. (2012, August 27). Cell phone data used to solve crimes. *The Citizens' Voice*.

Retrieved from: <http://citizensvoice.com/news/cell-phone-data-used-to-solve-crimes-1.1364074>

Skyhook Wireless, Inc. (2015). *Submit Wi-Fi Access Point to Provide Location*. Retrieved from:

<http://www.skyhookwireless.com/submit-access-point>

State of Washington Senate Judiciary Committee. (2012). *Cell phone location data*. Retrieved from:

<http://www.leg.wa.gov/Senate/Committees/LAW/Documents/CellPhoneLocationDataMemo.pdf>

Wells, A. (2014). Ping! The admissibility of cellular records to track criminal defendants. *Saint Louis University Public Law Review*, 33:487. Retrieved from:

http://www.slu.edu/Documents/law/PLR/Archives/XXXIII-2-14/Wells_Article_0.pdf

Yang, J., Varshavsky, A., Liu, H., Chen, Y., & Gruteser, M. (2010). Accuracy characterization of cell tower localization. Retrieved from:

<http://www.winlab.rutgers.edu/~gruteser/papers/tower10.pdf>

Yi, S. (2012). *Geo-location Forensics on Mobile Devices*. Retrieved from:

http://secmeeting.ihep.ac.cn/paper/Paper_Yi_Sun_ICDFI2012.pdf