Mobile phone forensics is the science of retrieving data from a mobile phone under forensically sound conditions. This book is an update to Practical Mobile Forensics, and it delves into the concepts of mobile forensics and its importance in today’s world.

We will explore mobile forensics techniques in iOS 8 - 9.2, Android 4.4 - 6, and Windows Phone devices. We will demonstrate the latest open source and commercial mobile forensics tools, enabling you to analyze and retrieve data effectively. You will learn how to introspect and retrieve data from the cloud, and document and prepare reports for your investigations.

By the end of this book, you will have mastered the current operating systems and techniques so you can recover data from mobile devices by leveraging open source solutions.

Who this book is written for

This book is for forensics professionals who are eager to widen their forensics skillset to mobile forensics and acquire data from mobile devices.

What you will learn from this book

- Discover the new features in practical mobile forensics
- Understand the architecture and security mechanisms present on the iOS and Android platforms
- Identify sensitive files on the iOS and Android platforms
- Set up a forensic environment
- Extract data from the iOS and Android platforms
- Recover data on the iOS and Android platforms
- Understand the forensics of Windows devices
- Explore various third-party application techniques and data recovery techniques
In this package, you will find:

- The authors biography
- A preview chapter from the book, Chapter 13 'Parsing Third-Party Application Files'
- A synopsis of the book’s content
- More information on Practical Mobile Forensics Second Edition
About the Authors

**Heather Mahalik** is a principal forensic scientist with Oceans Edge, Inc., where she leads the forensic effort focusing on mobile and digital exploitation. She is a senior instructor and author for the SANS Institute, and she is also the course leader for the FOR585 Advanced Smartphone Forensics course. With over 13 years of experience in digital forensics, she continues to thrive on smartphone investigations, forensic course development and instruction, and research on application analysis and smartphone forensics.

Prior to joining Oceans Edge, Heather was the Mobile Exploitation Team Lead at Basis Technology. When starting her career, she worked at Stroz Friedberg and for the U.S. Department of State Computer Investigations and Forensics Lab as a contractor. Heather earned her bachelor's degree from West Virginia University. She co-authored Practical Mobile Forensics (First edition) and was the technical reviewer for Learning Android Forensics. She has authored white papers and forensic course material and has taught hundreds of courses worldwide to Law Enforcement, Military, Government, IT, eDiscovery, and other forensic professionals focusing on mobile device and digital forensics.

*My first book was dedicated to the people who afforded me the opportunity to grow into the examiner I am today. This book is dedicated to those who push me to keep learning and allow me to share my knowledge – my students. Without you, I would not have had a reason to stay ahead of the curve, find those odd artifacts, and learn ways to outsmart the tools. You give me motivation to keep charging ahead. I would also like to thank metr0 for affording me opportunities to do things in my career that stretch far outside of what the norm is in forensics. I will be forever grateful.*

*To my husband, thank you for being such a great dad and for picking up the slack so that I can work as hard as I do. To Jack, always remember that your mama wants to be home with you and misses you while she’s away. Remember that my work is important and teaching others the right way to conduct digital examinations may make your world a safer and better place. ”The students” are happy you let them borrow your mommy. I would not be where I am today or able to travel and teach as much as I do without my amazing family and students.*
Rohit Tamma is a security analyst currently working with Microsoft. With over 7 years of experience in the field of security, his background spans consulting/analyst roles in the areas of application security, mobile security, penetration testing, and security training. His past experiences include working with Accenture, ADP, and TCS, driving security programs for various client teams. Rohit has also coauthored Learning Android Forensics, which explains various techniques to perform forensics on the Android platform. You can contact him at tamma.rohit5@gmail.com or on Twitter at @RohitTamma.

Writing this book has been a great experience as it has taught me several things, which could not have been possible otherwise. I would like to dedicate this book to my parents for helping me in every possible way throughout my life.

Satish Bommisetty is a security analyst working for a Fortune 500 company. His primary areas of interest include iOS forensics, iOS application security, and web application security. He has presented at international conferences, such as ClubHACK and C0C0n. He is also one of the core members of the Hyderabad OWASP chapter. He has identified and disclosed vulnerabilities within the websites of Google, Facebook, Yandex, PayPal, Yahoo!, AT&T, and more, and is listed in their hall of fame.

I would like to thank everyone who encouraged me while producing this book.
The exponential growth of mobile devices has revolutionized many aspects of our lives. In what is called as the post-PC era, smartphones are engulfing desktop computers with their enhanced functionality and improved storage capacity. This rapid transformation has led to increased usage of mobile handsets across all the sectors.

Despite their small size, smartphones are capable of performing many tasks: sending private messages and confidential e-mails, taking photos and videos, making online purchases, viewing our salary slips, completing banking transactions, accessing social networking sites, managing business tasks, and more. Hence, a mobile device is now a huge repository of sensitive data that can provide a wealth of information about its owner. This has in turn led to the evolution of Mobile Device Forensics, a branch of digital forensics that deals with retrieving data from a mobile device. Today, there is huge demand for specialized forensic experts, especially given the fact that the data retrieved from a mobile device is court admissible.

Mobile forensics is all about utilizing scientific methodologies to recover data stored within a mobile phone for legal purposes. Unlike traditional computer forensics, mobile forensics has limitations in obtaining evidence due to rapid changes in the technology and the fast-paced evolution of mobile software. With different operating systems and with a wide range of models being released into the market, mobile forensics has expanded over the last few years. Specialized forensic techniques and skills are required in order to extract data under different conditions.

This book takes you through the challenges involved in mobile forensics and practically explains detailed methods of collecting evidence from different mobile devices with iOS, Android, and Windows mobile operating systems.

This book is organized in a manner that allows you to focus independently on chapters that are specific to your required platform.
What this book covers

Chapter 1, Introduction to Mobile Forensics, introduces you to the concepts of mobile forensics, its core values, and its limitations. This chapter also provides an overview of practical approaches and best practices involved in performing mobile forensics.

Chapter 2, Understanding the Internals of iOS Devices, provides an overview of the popular Apple iOS devices, including an outline of different models and their hardware. Throughout this book, we explain iOS security features and device security and its impact on the iOS forensics approaches. This chapter also gives an overview of the iOS file system and outlines the sensitive files that are useful for forensic examination.

Chapter 3, iOS Forensic Tools, gives an overview of existing open source and commercial iOS forensics tools. These tools differ in the range of mobile phones they support and the amount of data that they can recover. This chapter describes the advantages and limitations of those tools.

Chapter 4, Data Acquisition from iOS Devices, covers various types of forensic acquisition methods that can be performed on iOS devices and guides you through preparing your desktop machine for forensic work. This chapter also discusses passcode bypass techniques and physical extraction of the devices and explains different ways in which the device can be imaged.

Chapter 5, Data Acquisition from iOS Backups, provides detailed explanations of different types of iOS backup and details what types of file are stored in the backup. This chapter also covers logical acquisition techniques of recovering data from the backups.

Chapter 6, iOS Data Analysis and Recovery, discusses the types of data that is stored on iOS devices and the general location of this data storage. Common file types used in iOS devices such as plist and SQLite are discussed in detail to provide an understanding of how the data is being stored on the device, which will help the forensic examiners to efficiently recover data from these files.

Chapter 7, Understanding Android, introduces you to the Android model, file system, and its security features. It provides an explanation of how data is stored in any Android device, which will be useful while carrying out forensic investigation.

Chapter 8, Android Forensic Setup and Pre Data Extraction Techniques, guides you through the Android forensic setup and other techniques to follow before extracting any information. Screen lock bypass techniques and gaining root access are also discussed in this chapter.
Chapter 9, *Android Data Extraction Techniques*, provides an explanation of physical, file system, and logical acquisition techniques for extracting relevant information from an Android device.

Chapter 10, *Android Data Analysis and Recovery*, talks about extracting and analyzing data from Android image files. This chapter also covers possibilities and limitations for recovering deleted data from Android devices.

Chapter 11, *Android App Analysis, Malware, and Reverse Engineering*, covers the analysis of some of the widely used Android apps to retrieve valuable data. This chapter also covers Android malware and techniques to reverse engineer an Android app.

Chapter 12, *Windows Phone Forensics*, provides a basic overview of forensic approaches when dealing with Windows Phones.

Chapter 13, *Parsing Third-Party Application Files*, covers forensic approaches to include acquisition and analysis techniques when dealing with BlackBerry devices. BlackBerry encryption and data protection is also addressed.
Third-party applications have taken the smartphone community by storm. Most smartphone owners have more than one app on their device that they rely on to chat, game, get directions, or share pictures. According to http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/, there are almost 4 million apps existing worldwide for the various smartphones. Apple App Store offers approximately 1.5 million apps, Google Play offers 1.6 million, Amazon offers 400,000 apps, and Windows offers 340,000. This number is expected to grow exponentially through 2017.

The goal of this chapter is to introduce you the various applications seen on Android, iOS devices, and Windows Phones. Each application will vary due to versions and devices, but their underlying structures are similar. We will look at how the data is stored and why preference files are important to your investigation.

We will cover the following topics in detail in this chapter:

- Different third-party applications
- How applications are stored on iOS devices
- How applications are stored on Android devices
- Windows Phone 8 application storage
- How to use both commercial and open source solutions to parse application data
Third-party application overview

Third-party applications are an integral part of mobile device investigations. Often, the key artifacts seem to exist within an application. This requires the examiner to understand where application data is stored on the device, how application data is saved for this platform, and which tool best helps uncover the evidence. Manual parsing is often a key factor when examining third-party applications on any smartphone. While some commercial tools, such as Magnet IEF, are known for application parsing support, no tool is perfect and it’s virtually impossible for tools to keep up with the frequent updates that are released for each application. Most often, you will realize that the commercial tools parse the most popular applications on the market. For example, when Facebook purchased WhatsApp, Cellebrite, IEF, and Oxygen Forensics started supporting this application. Facebook is extremely popular, but data isn’t always extracted or parsed, due to security features that are built into the app. This is where all apps differ. Our best advice is to test, test, and test! You can download an app, populate data, and examine the results to see how your view of the evidence compares to your actual evidence. This practice will enable you to understand how updates change the artifacts, how evidence locations have changed, and how to manually extract artifacts that your tools are missing. Additionally, reverse-engineering an app and analyzing its code will help us identify where the data is stored and how it is stored.

Most applications do not require a data plan for use. They can fully function off a WiFi network, which means that apps function when a person travels to a region in which their device will not work. For example, when I travel, I rely on Skype, Viber, and WhatsApp to call and text family and friends. All that is required is that my smartphone is connected to WiFi.

We have already addressed some third-party application extraction and analysis tips in this book. In addition to this, we discussed the files that need to be examined to understand and analyze application data in Chapter 6, iOS Data Analysis and Recovery, Chapter 10, Android Data Analysis and Recovery, Chapter 11, Android App Analysis, Malware, and Reverse Engineering, and Chapter 12, Windows Phone Forensics. This chapter will dive deeper into the applications and relevant files and prepare the examiner for the analysis of these artifacts. Each application has a purpose. Most tools provide support for the most popular application in each category. The rest is up to you. A glimpse of applications as presented by Oxygen Detective is shown in the following screenshot. As expected, these are not all of the applications that are present on the device; rather, these are just the ones that the tool knows how to parse:
Chat applications

Chat applications are among the most common applications on the market. These applications provide users with the ability to chat outside the standard SMS services offered by the network service provider and device and sometimes in a secure method. By secure, the apps may offer encryption, private profiles, private group chats, and more. Additionally, these apps enable the user to message others without the need for a data plan, as WiFi provides all the access that they need. Tango, Facebook Messenger, WhatsApp, Skype, and SnapChat are some of the more popular applications.

Parsing artifacts from chat applications is not always simple. Often, multiple tools and methods will be required to extract all of the data. Commercial tools may only parse a portion of the data, forcing the examiner to learn how to examine and recover all data or miss evidence. Oxygen Detective is being used to parse chat messages from Tango on an Android device in the following screenshot. Note that the message does not show the image in the table. However, this image can be “pieced” back into the message, as shown in the following screenshot, to provide the total picture of what was being shared in the conversation. In this example, the graphic was located and is shown with an arrow pointing to the message to which it belongs. This was a manual process and was not performed by the tool:
Parsing Third-Party Application Files

<table>
<thead>
<tr>
<th>Dir ct...</th>
<th>Remote party</th>
<th>Text</th>
<th>Image URL</th>
<th>Time stamp (Device ...</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hank Fresh</td>
<td>Sweet</td>
<td>N/A</td>
<td>1/25/2016 4:27:52 PM</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td>Hank Fresh</td>
<td>/data/media/0/Android/data/com.s...</td>
<td><a href="http://u.tango.net/faw">http://u.tango.net/faw</a>...</td>
<td>1/25/2016 4:27:30 PM</td>
<td>Image</td>
</tr>
<tr>
<td></td>
<td>Hank Fresh</td>
<td>/data/media/0/Android/data/com.s...</td>
<td><a href="http://u.tango.net/cub">http://u.tango.net/cub</a>...</td>
<td>1/25/2016 4:27:00 PM</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td>Hank Fresh</td>
<td>New $47</td>
<td>N/A</td>
<td>1/25/2016 4:26:36 PM</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td>Hank Fresh</td>
<td>Hi its felicia</td>
<td>N/A</td>
<td>1/25/2016 4:25:47 PM</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td>Hank Fresh</td>
<td>Hello I would like to chat with you.</td>
<td>N/A</td>
<td>1/25/2016 4:24:06 PM</td>
<td>Text</td>
</tr>
</tbody>
</table>

An example of piecing application chat logs back together

**GPS applications**

Most users branch outside their standard phone apps for GPS support. This includes getting directions to locations and obtaining maps for areas of interest. Common GPS applications include Waze, Google Maps, and more. Waze goes beyond just providing directions, as it also alerts the user to road hazards, traffic, and police officers that are along the path they are driving:
Other applications that store location information include Twitter, Instagram, Facebook, FourSquare, and so on. These applications enable a user to alert friends and followers to their location when they create a post or share an image/video. All of these transactions are tracked within the app. Understanding this is key to uncovering additional artifacts that are not reported by your forensic tool.

When examining location information from GPS applications, it is best to assume that you need to manually examine the databases and preference files that are associated with that application. We recommend using your forensic tool to triage the data on the device and then dive deeply into the artifacts, which will be discussed later in this chapter. An example of Waze being parsed by UFED Physical Analyzer is in the following screenshot. Here, we can see that the user had five favorite locations, 74 mapped locations, and 70 recent directions. All of this information must be manually verified if it pertains to the investigation. This is due to the fact that the tool cannot determine whether the user typed
the address, whether it was suggested, or whether the user even traveled to that location. Proper skills are required by the examiner to tie a user to a specific location and this takes more than a forensic tool.

![Device Locations in UFED physical analyzer](image)

The Waze application in UFED physical analyzer

### Secure applications

Secure, self-destructing, did it ever even happen? Ignore the claims of data retention and hunt for that data! These apps often make claims that are simply untrue. These applications are designed with security in mind. However, updates are released so quickly, and quality assurance checks seem to not be strong enough to catch everything. On occasion, you will find an app with an encrypted or nonexistent database, but the file has journal, write ahead logs, or shared memory files that contain portions of the chats that were supposed to be encrypted. In addition to this, the user can save media files that are shared, take screenshots of the conversations, and do much more. Often, you may uncover the images, audio, and video files that were shared and supposed to be encrypted.

Some popular secure messaging applications include Telegram, Wickr, and Signal. Some of these are encrypted, and nothing is recoverable. However, this all depends on the device, the OS running on the smartphone, and the version of the app. The security level of these apps is publicly advertised, but again take this with a grain of salt. You should always assume that there could be a vulnerability in the app that may provide you with access forensically. Dig for this evidence!
Information on how secure some of these apps are can be found at https://www.eff.org/secure-messaging-scorecard.

Financial applications

Applications that utilize financial information, such as credit card information and personal banking, are required to be encrypted and secure. iOS devices will not acquire these apps without an Apple ID and password. Even if you have the user’s Apple ID and password, the data extracted should still be encrypted. Some examples of financial applications include Google Wallet, Windows Phone Wallet, PayPal, Apple Pay, and In-App Purchases. When you examine a device, you may see that the app was installed with the associated application metadata, but account information and transactions will not be accessible.

Social networking applications

Commercial support for social networking applications is strong as they are the most popular apps that are downloaded from the app stores. These applications allow users to make posts, share locations, chat publically, and privately and essentially catalog their life. Common social networking applications include Facebook, Twitter, and Instagram. Often, users will enable one app, such as Instagram, to have access to Facebook and Twitter so that posting is seamless. Thus, when examining devices, the user may find multiple copies of the same file or conversation due to the sharing between apps.

When examining these apps with commercial tools, it is common for chats and contacts to be parsed. Other data is often overlooked. Again, this means the examiner must look at the data dump to ensure that nothing is missed. As an example, we are going to take a look at Twitter. This application stores a lot of information that may require more than one tool to parse. Additionally, the user may have to manually examine the database files to ensure that all artifacts have been recovered.

Let's take a look at what the tool was able to extract. As stated several times in this book, start with what the tool is telling you is installed, and then formulate keywords and methods to dig deep into the file system. We can see the user account information for Twitter, as well as the file path where this data is being extracted, in the following screenshot:
The next logical step is to view what the tool can tell you about the application and how it was used. Oxygen Detective provided the following information for Twitter account usage. Note that both public Tweets and private messages (DM) are recovered:
After examining what was parsed by the tool, the database files should be examined to ensure nothing was missed. This is not always simple, as each account and function may have a unique database. By function, we mean that contacts may be stored in one database while chats and account information are stored in another. Once you become more familiar with common applications, you will know where to look first. At the time of writing this book, the following databases were the most relevant:

- **Global.db**: This database contains account information, such as the username
- **<User-id>.db**: This database contains notifications, messages, contacts, and statuses

In the following screenshot, we can see all of the databases that are associated with Twitter. Again, start with what you know and dig deeper:
Each database may contain unique data that can be parsed for additional artifacts. These applications also contain unique user_id values, which can be used as keywords to search for other devices with traces of communication within an investigation. For this example, we can see user_id values, the creation date (UNIX timestamp), and the data, which is the result of private messaging on Twitter:

```
<table>
<thead>
<tr>
<th>user_id</th>
<th>created</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>475222380</td>
<td>1404903823000</td>
<td>I-Ø?u? ...</td>
</tr>
<tr>
<td>29574511</td>
<td>1404905109000</td>
<td>I-Ti’? j- No, I’m with myself at E... my ca...</td>
</tr>
<tr>
<td>475222380</td>
<td>1404913177000</td>
<td>I-s3E jaWe need test data. Not really doing rese...</td>
</tr>
<tr>
<td>29574511</td>
<td>1404913470000</td>
<td>I-tQ??Do you want just the chats? I can toss togeth...</td>
</tr>
<tr>
<td>475222380</td>
<td>1404916910000</td>
<td>I-Øq? j? It’s not even for me. A student asked me if I h...</td>
</tr>
<tr>
<td>29574511</td>
<td>1454683850000</td>
<td>I $W@@ZjI Shoot me your address and I’ll try to get th...</td>
</tr>
</tbody>
</table>
```

Custom queries can be written to parse Twitter databases of interest. A good example of how to do this is shown, as follows. This query is specific to parsing Twitter contacts:

```
SELECT
    _id AS "Index",
    user_id,
    username,
    name,
    datetime (profile_created/1000,'UNIXEPOCH','localtime') AS "Profile Created",
    description AS "Twitter Description",
    web_url,
    location,
    followers,
    friends AS "Following",
    users.statuses AS "Number of Tweets",
    datetime (profile_created/1000,'UNIXEPOCH','localtime'),
    image_url,
    datetime (updated/1000,'UNIXEPOCH','localtime') AS "Profile Updated",
    datetime (friendship_time/1000,'UNIXEPOCH','localtime') AS "Became Friends"
FROM users
```
Encoding versus encryption

The terms encoding and encryption are used so frequently when discussing applications and smartphone data that they are often confused. Encoding is essentially the process of obfuscating a message or piece of information to appear as raw code. In some cases, the goal of encoding is to make the data unrecognizable to the computer or the user. In reality, the primary goal of encoding is to transform the input into a different format using a publicly available scheme. In other words, anyone can easily decode an encoded value. Encryption, however, transforms the data using a key in order to keep it secret from others. So, encrypted text can be reversed only if you have the key. Most applications claim that they encrypt the data or that the data is never saved to disk. While this is true for some, most are simply encoded. Encoding options can vary, but the most common for smartphone data is **Base64**. Messaging apps often rely on Base64 encoding to make the data appear to be hidden or “safe.” A common artifact of Base64 is the padding of the data with an “=” when the encoded bytes are not divisible by three.

Until a little over a year ago, Oxygen Forensics and Autopsy were two of the few tools supporting the decoding of Base64 payloads from applications derived from smartphones. For these tools to parse the data, they must support the application containing the encoding. Currently, MSAB, UFED Physical Analyzer, and Magnet IEF also provide Base64 decoding support.

An example of Base64-encoded messages is shown in the following screenshot. This data is from the Tango chat application:

![Base64-encoded Tango messages](image-url)
Encryption is a bit more difficult as the app may not even provide access to the encrypted data. For example, the database directory may be empty or the cells containing the encrypted data are simply empty. Occasionally, you will have access to the encrypted blobs within the databases, but this data cannot always be decrypted. Again, when you face encrypted data, look elsewhere. Have you examined the journal and write ahead logs? Have you examined the cache and media directories? Have you examined the SD card? These are common questions you will often have to ask yourself to ensure you are not relying on your forensic tools too much and that you are covering your bases to ensure nothing is overlooked. As explained, start with what you know. We know that the cache and database directories store user data, so this is a great place to start your manual examination:
Application data storage

Almost all applications rely on SQLite for data storage. These databases can be stored internally on the device or on the SD card for relevant phones. When SQLite is used, temporary memory files are commonly associated to each database to make SQLite more efficient. These files, which were previously mentioned, are write ahead logs (WAL) and shared memory files (SHM). These files may contain data that is not present in the SQLite database. Few tools will parse this information, but the ones that are offered by Sanderson Forensics, will get you started. Go to http://sandersonforensics.com/forum/content.php?261-Timelining-events-in-a-WAL-based-SQLite-DB. We can see several WAL and SHM files associated with various WhatsApp database files in the following screenshot:

An SHM and WAL example

In addition to SQLite databases, other devices rely on Plist, XML, JSON, and DAT files for application data storage, account data storage, purchase information, and user preferences. These files will be discussed in the Android, iOS, and Windows Phone sections.
iOS applications

Apple relies on SQLite and Plists as common locations for application data storage. On occasion, JSON files will be used for application data. Examining applications recovered from an iOS device can be overwhelming. We suggest you start with what you know and what your tool is telling you. Examine the Installed Applications listed by your tool of choice. From here, go directly to the applications directory and ensure that nothing is being overlooked. When a user deletes an app, the databases often remain, and the link to the installed application is simply broken. Examining all areas of the iOS device will prevent the examiner from missing data:

![File Browser Screenshot]

After examining the installed applications, search the Library and Documents directories for relevant Plist files that may contain application artifacts. Finally, examine the Media directory on the iPhone as well as the one associated with the app to recover additional artifacts, such as shared photos, videos, audio files, and profile pictures. In the following screenshot, we are examining the Media directory associated to the WhatsApp application:
Android applications

Android devices heavily rely on SQLite for application storage. The preference files for each application are often in the DAT or XML files. More so than an iOS device, examining application on an Android may be one of the most tedious tasks due to the various locations that data may be stored in. The best place to start is with a tool that will provide a listing of what is installed on the device. Next, go to the subdirectories off the /Root directory. Remember, these applications may possess unique names and may be difficult to locate. You may have to research the application to gain a better understanding of the filenames that are associated with each of them. The following screenshot is an example of application directories on an Android device:
Each of these application directories will contain a lot of data to examine. We recommend starting with the Databases and Cache directories and then expanding your analysis to other locations on the device. The next locations to examine include the Media and Cache partitions. If the data appears to be missing or is claimed to have been deleted, do not forget to examine the Downloads directory on the device and SD card.

Application data can exist in several locations in the Media directories. Using a tool, such as UFED Physical Analyzer, which provides keyword-searching capabilities spanning beyond parsed items, will really help in locating artifacts pertaining to specific applications. We are looking at the large amount of data stored in the Media directory on an Android device in the following screenshot. This data is unique from what is stored in the application directory that was discussed previously. Each location needs to be thoroughly examined to ensure nothing is missed. It is important that you take what you learned in previous chapters to analyze Android application data:
Windows Phone applications

Applications found on Windows Phones are no different than those found on iOS and Android devices. SQLite is the most common format used for data storage. However, not all devices allow for SQLite files to be stored internally on the phone. For these devices, all application data will be found on the SD card. Some may view this as lucky because it saves us from having to examine several locations on the device, but the SD card and the applications themselves may be encrypted.

Where possible, it is best to remove the SD card and acquire it using a forensic tool. When this is not possible, the next best method would be to try to acquire the SD card through the
phone using a forensic tool. Again, this will often result in missed data. As a final effort, live
analysis can be completed by mounting the device and using Windows Explorer to view the
applications stored on the device and SD card, as discussed in Chapter 12, Windows Phone
Forensics.

Forensic methods used to extract third-party application data

Almost all commercial tools will attempt to support the extraction of third-party
applications. We recommend that you test your tools thoroughly and often if you rely on
tool output for your investigative results. This is because the apps are updated so frequently
that it is nearly impossible for the tools to not miss something. You must learn the
applications, how they work, and how the devices store the data for each. We strongly
recommend that you use your tool to triage the case and then dive into the data to manually
extract anything that the tools miss. Make sure that you only include factual data in your
forensic report and not everything that the tools parses. The tools cannot decipher the
difference between device and human creation. Only a trained examiner can do this with
confidence.

Commercial tools

As you have seen in this book, there are many tools that can handle the job of smartphone
forensics. However, there are a few that really shine when it comes to parsing application
data. Magnet IEF, Oxygen Detective, Forensics Suite, and UFED Physical Analyzer are a
few that do a good job of recovering data from the application categories discussed in this
chapter. We will take a quick glance at how to leverage each of these tools to parse
application data. Keep in mind that these tools will not find every application and will not
parse all data for applications.

Oxygen Detective

Oxygen Detective can be used to examine application data. For this example, we are
assuming the acquisition is complete, and we are simply attempting to analyze the data.
Note that Oxygen is capable of acquiring and analyzing smartphones. In this example, we
acquired the device with Cellebrite UFED and analyzed it with Oxygen. To load a data
dump of a device and examine application artifacts, follow these steps:
1. Launch Oxygen Detective.
2. Select the **Import File** option and choose your image. Multiple image formats are supported for ingestion into Oxygen.
3. After parsing is complete, start examining the parsed applications:

![Oxygen Detective application view](image)

4. Next, start examining applications of interest by clicking on the application and examining all of the associated files.
5. Once you select the application, you will be presented with the data that was parsed and the full file path of where the data was extracted. Use this path to manually verify the findings. We are looking at the Pinterest application in the following screenshot. Note how the container, file, and table of interest are provided and hyperlinked for the user. The tool is even encouraging you to dig deeper and verify the findings:

Oxygen Detective has built-in features for keyword searching, bookmarking, and reporting. In addition, the SQLite Database and Plist Viewer will provide a method to examine relevant application data.

6. Report all account information, chats, messages, locations, and any other data of interest as this provides relevance to your investigation.

**Magnet IEF**

Magnet IEF has been known as one of the leaders in Internet and application parsing for digital media. They are just as strong with mobile devices. Again, one tool cannot do the job, but IEF proves to be the strongest and parses the most applications from Android, iOS,
and Windows Phones. The downside to this tool is that we are forced to rely on the reported artifacts as the file system is not normalized and provided for manual examination. To use IEF to examine application artifacts, follow these steps:

1. Launch IEF and then select **MOBILE** (note that, if **MOBILE** is grayed out, you need to obtain a license that provides mobile support from Magnet Forensics):

   ![Magnet IEF](image)

2. Select **IMAGES** and navigate to your image file. More than one image can be loaded and parsed at the same time.

3. Select **NEXT** and determine what you want to parse. We recommend selecting **CHECK ALL**:
4. Browse to the location where you wish to save the case file and select Find Evidence.
5. Once complete, the IEF Report Viewer will be displayed:
Chapter 13

The first step in examination is to review what is parsed by IEF. In the preceding screenshot, we can see that Telegram was parsed. Start your examination in the most relevant location. For example, if you are looking for Telegram chats, go right to that location and start examining the artifacts. Note that Messages and Chats are pulled into two different categories. This is common when Private Messaging is used. All relevant application containers should be examined. Additionally, IEF provides the full file path from which the data was recovered. Use another tool to navigate to this file for verification and manual examination.

IEF also provides logical keyword search; essentially it will search what it can parse and nothing else, bookmarking and reporting. Make sure that you only report factual application artifacts and incorporate this into your final forensic report.
UFED Physical Analyzer

Physical Analyzer is one of the most well-known mobile forensic tools on the market. This tool is one of the best platforms to manually conduct an examination in addition to leveraging the data parsed by the tool. For application analysis, Physical Analyzer is good at parsing chats and contacts for each supported application. For data that in not parsed, Physical Analyzer provides an analytical platform that enables the user to browse the file system to uncover additional artifacts. Keyword searching is robust in this tool and is capable of searching raw Hex as well as parsed data. In addition, a SQLite viewer is included.

To conduct a forensic examination of application data in Physical Analyzer, follow these steps to get started:

Launch Physical Analyzer by double-clicking on the UFD shortcut image file or by double-clicking on the tool icon.

Load the image file and wait until parsing completes.

Examine the parsed artifacts, as shown in the following screenshot. For this example, we are examining Tango. Physical Analyzer recovered Tango data in **Chats**, **Contacts**, **Installed Applications**, **Passwords**, and **User Accounts**:

![Tango as parsed by Physical Analyzer](image)

We recommend examining what is parsed and referring to the hyperlink of where the data is being extracted. Navigate to this path and then examine the entire application directory.
To find the application directory, leverage built-in keyword searching capabilities to aid in the investigation. Remember, you may have to conduct research to determine the file names associated to the app if this is not apparent. Tango, for example, does not use the term Tango in the file paths or filenames. The directory is `.sgiggle` and the primary database is `tc.db`. This makes our job harder because we can’t simply search for Tango and get accurate results.

**Open source tools**

For those on a budget, it is possible to examine application data from smartphones using open source solutions and cheap tools. These solutions are more difficult, and they are often not the answer for those new to forensics who need the assistance of a tool to aid in data extraction and analysis. Examining application data is tedious, and if you do not know where to look, the chances are that you will need to spend some money to get a head start. Tools, such as Andriller, can be purchased for around $500. This not free, but it’s also not $10,000, which is what some of the other commercial tools cost. We will cover a few of our favorite tools that are useful in parsing application data from smartphones.

**Autopsy**

Autopsy is one of the best tools to examine Android and Windows Phones. Unfortunately, iOS parsing is not provided in Autopsy. Autopsy can be downloaded from [http://sleuthkit.org/autopsy/](http://sleuthkit.org/autopsy/). When using Autopsy, the Android Analyzer module will parse some application data from the device. This module is unique in that it is currently the only tool that parses WordsWithFriends, a gaming application, and was the first tool, other than Oxygen Forensics, to provide Base64 decoding support for Tango chat messages. Some say that Autopsy is the free solution for those who cannot afford Physical Analyzer.

To use Autopsy, download the software and install it on a Windows machine and follow these instructions. Make sure that you are always using the latest version:

1. Launch Autopsy.
2. Create a new case:
3. Select **Next** and then click on **Finish**.
4. Navigate to your image file and select **Next**.
5. Select the modules that you wish to run. Keyword Search and Android Analyzer will be the most fruitful for an Android device. These modules can also be run after the image is ingested. The Keyword search will prove to be just as robust as Physical Analyzer:
6. Autopsy provides access to file system data faster than any commercial or open source tool available. Knowing where to go from there is the hard part. Again, start with anything that is in the extracted content and then dive into the file system and examine the files that we discussed in this book and relevant application data:
Once you have identified applications of interest, start with what is parsed and then examine the relevant database, cache, and preference files. At the time of writing, Autopsy did not have a SQLite viewer available. All databases must be exported and examined in a SQLite viewer. We like SQLite Forensic Browser, which has been discussed in this book.

Autopsy was able to parse Tango chat messages and contacts, similarly to Physical Analyzer, IEF, and Oxygen. The following screenshot shows the results of the decoded messages:
Other methods to extract application data

One of the easiest ways to parse application data is to create custom SQLite queries and Python scripts to parse data of interest. We discussed several suggestions and examples of queries and scripts throughout this book. Python is one of the best solutions because it is free and we have full access to the libraries. One thing to keep in mind is that our scripts have to be updated frequently to keep up with application updates. Also, make sure your encoding schemas are correct to prevent application artifacts from being missed or not interpreted correctly.

In addition to Python scripts, free parsers that support application extraction already exist. WhatsApp Extract is a free tool for both Android and iOS that will extract WhatsApp application data from devices. Often, this free tool will extract more data than the commercial solutions, depending on the permissions the user allocated during installation. Others, such as Mari DeGrazia (http://az4n6.blogspot.com/p/downloads.html) and Adrian Leong (https://github.com/cheeky4n6monkey/4n6-scripts), have developed scripts to parse applications, recover deleted data from SQLite free pages, decode Base64, and more. We recommend using what is already available before developing your own.
Summary

Many apps are not what they claim to be. Never trust what you read about the apps, as Quality Assurance testing across these apps is not consistent, and we have determined several vulnerabilities and security flaws over the years that provide us with methods of piecing application data back together. In addition, application updates will change the way we need to look at the data. Understanding each smartphone and how it stores application data is the first step in successfully examining applications on smartphones. Knowing that updates may change data locations, encoding, and encryption, and how your tool functions, is one of the hardest concepts for examiners to grasp. It is your job to learn the capabilities of the application to uncover the most data from the mobile device.

Understanding how an application works is hard enough, and then we have to consider how to extract the artifacts. As you have read in this book, there are so many ways to parse data from smartphones. One tool is never enough and the reality is that mobile forensics can be expensive. We hope that we have provided you with a practical guide that teaches you to acquire and analyze artifacts that are recovered from smartphones. Take what you learned and apply it immediately to your methods to conduct mobile forensics or use it to make you more prepared for your next job. Remember that practice, testing, and training will make you better at your job and help you perfect the art of mobile forensics.
Where to buy this book
You can buy Practical Mobile Forensics Second Edition from the Packt Publishing website.
Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.
Click here for ordering and shipping details.