da, MD
20852
301.65
4.SANS
(7267)

DFIR RESOURCES
info@sans.org
digital-forensics.sans.org
registration@sans.org
Twitter: @sansforensics

SANS EMAIL
GENERAL INQUIRIES:

REGISTRATION:

TUITION: tuition@sans.org
PRESS/PR: press@sans.org

Author: Heather Mahalik
hmahalik@gmail.com
Twitter:
@heathermahalik

585.3

iOS Device Forensics

If any provision of this CLA is declared unenforceable in any jurisdiction, then such provision shall be deemed to be severable from this CLA and shall not affect the remainder thereof. An amendment or addendum to this CLA may accompany this Courseware.

SANS acknowledges that any and all software and/or tools, graphics, images, tables, charts or graphs presented in this Courseware are the sole property of their respective trademark/registered/copyright owners, including:

AirDrop, AirPort, AirPort Time Capsule, Apple, Apple Remote Desktop, Apple TV, App Nap, Back to My Mac, Boot Camp, Cocoa, FaceTime, FileVault, Finder, FireWire, FireWire logo, iCal, iChat, iLife, iMac, iMessage, iPad, iPad Air, iPad Mini, iPhone, iPhoto, iPod, iPod classic, iPod shuffle, iPod nano, iPod touch, iTunes, iTunes logo, iWork, Keychain, Keynote, Mac, Mac Logo, MacBook, MacBook Air, MacBook Pro, Macintosh, Mac OS, Mac Pro, Numbers, OS X, Pages, Passbook, Retina, Safari, Siri, Spaces, Spotlight, There's an app for that, Time Capsule, Time Machine, Touch ID, Xcode, Xserve, App Store, and iCloud are registered trademarks of Apple Inc.

PMP® and PMBOK® are registered trademarks of PMI.

SOF-ELK® is a registered trademark of Lewes Technology Consulting, LLC. Used with permission.

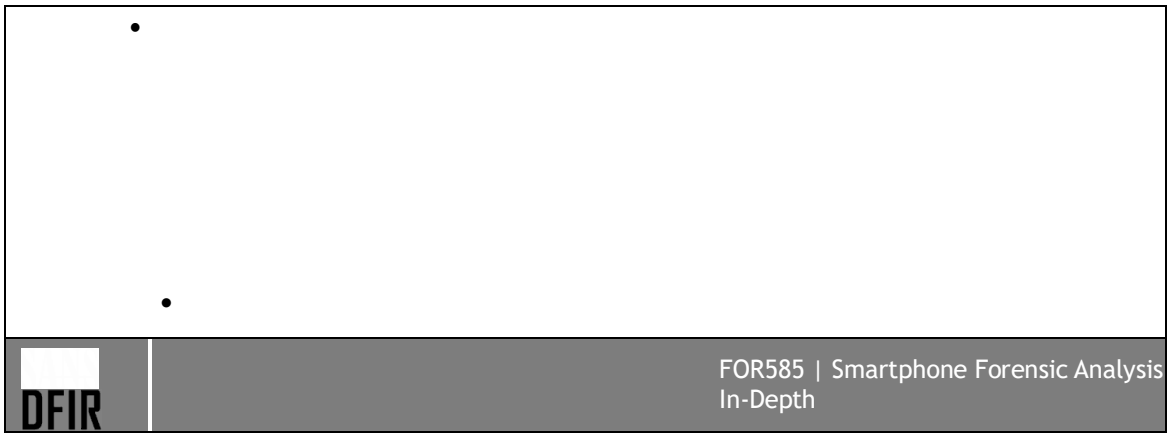
SIFT® is a registered trademark of Harbingers, LLC. Used with permission.

Governing Law: This Agreement shall be governed by the laws of the State of Maryland, USA.

This page intentionally left blank.

This page intentionally left blank.

This page intentionally left blank.



For the matters of this section and in the real world, iOS devices include the iPhone, iPad, and iPod touch. The Apple Watch runs Watch OS, which is similar to iOS. We do not cover Apple TV in this course, but some concepts apply. Although some may agree that iOS and Mac OS X are similar, the main differences are the gesture-based touch interface on the iOS device and some logging. The two operating systems also manage files in different manners. The internal components of iOS devices are broken down in detail in the book written by Heather Mahalik, Rohit Tamma, and Satish Bommisetty, *Practical Mobile Forensics*, Second Edition (Birmingham, UK: Packt, 2016).

File systems on iOS devices are similar to HFS+ and APFS, and the data is primarily stored in SQLite database and plists. APFS (Apple File System) fixed core problems and bugs from HFS. APFS is used on devices running iOS 10.3 and later. Regardless of the file system, iOS device applications sandbox their data, primarily for security purposes, which also makes our investigations easier because the data should be stored within a cached database file for each application. Data storage and application decoding are covered later in this section and in Section 5.

iOS devices differ from other smartphones and mobile devices in that they do not have a removable battery and do not have slots for external storage. Prior to the iPhone 4s, only GSM iPhones contained SIM cards. Verizon added a SIM card to the CDMA version of the iPhone 4s to give it "world phone" capabilities as discussed in Section 1. All iOS devices released after 2011 now contain SIM cards in various physical (micro-SIM and
mini-SIM) to support LTE (Long Term Evolution) and digital forms as ESIM. A great reference for every Apple device is listed at https://for585.com/specs.

The data residing within the NAND Flash Memory consists of SQLite databases and plists. The SQLite database files are comprised of everything imaginable (contacts, locations, communications, preferences, and more). The plists are generally used by applications to store, organize, and access data on the iOS device. The plists are stored in XML or binary format and often contain embedded BLOBs that store items of interest. BLOBs are covered in detail in Sections 4 and 5 of this course.

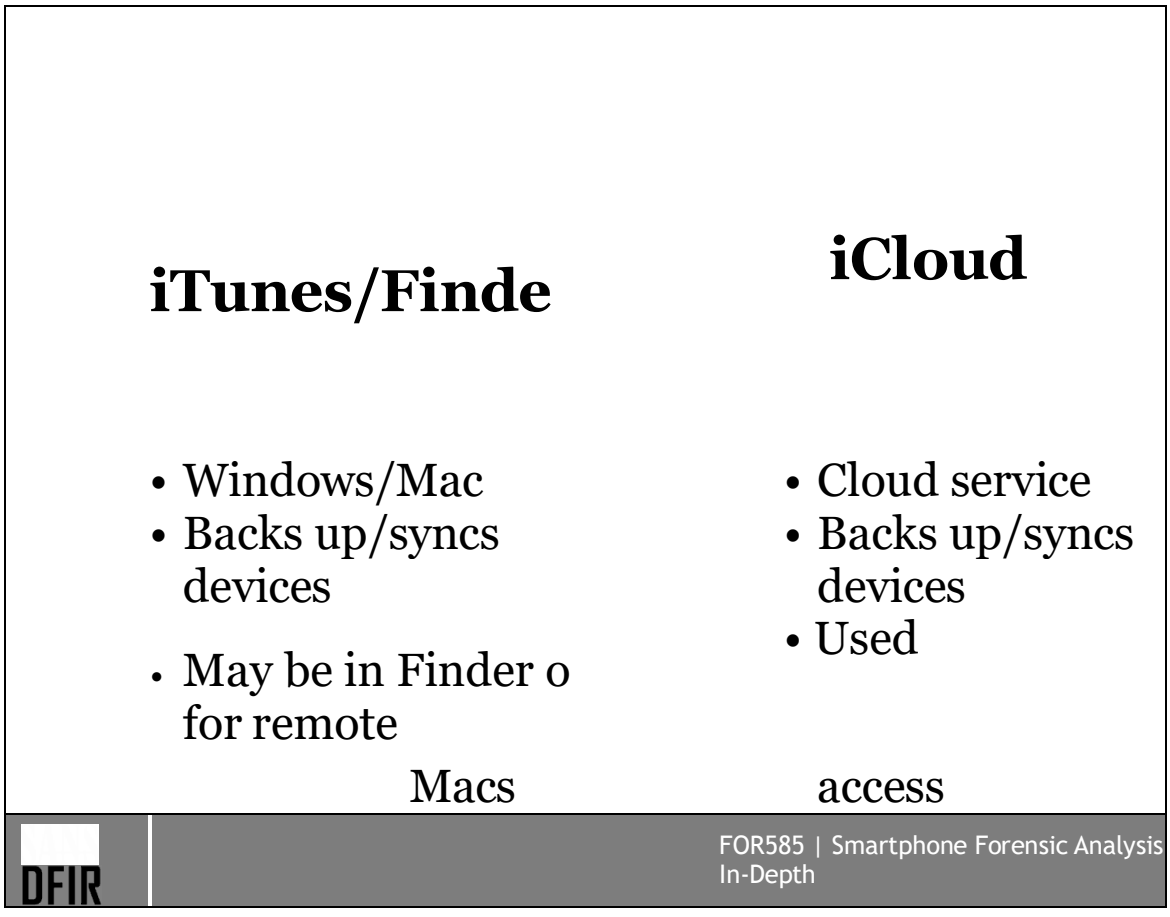
Property List Files (plist) are structured text files containing information needed for the iOS device and applications to function. The format looks similar to an XML file and is not always easy to read unless you utilize a plist editor/viewer. The plist files should be examined for relevance to your investigation.

The iOS architecture of the iPhone is detailed in this slide. The User Space is comprised of the Core Services, which run at the lowest level of this partition. Core Services include the Address Book, SQLite databases, XML files, plists, and other core files. The next level is comprised of multimedia files, which run under the Cocoa Touch framework. The Cocoa Touch framework handles the iOS device user interface. The final layer consists of Apple and third-party applications. The Kernel or System Space is comprised of the hardware layer and the System Area.

This image was composed with information found on https://for585.com/bada.

Reference:

[1] https://for585.com/mahalik



iTunes and iCloud are the most used methods for iOS device support. iTunes/Finder provides free methods for backing up, syncing, and restoring data to your iOS device from your Macintosh or Windows computer. iTunes can also be used to create a forensic backup, which is discussed in Backup File Forensics in Section 4.

Once an iOS device is backed up as an “encrypted device," the password used for the initial encryption is required for future access to the device backup files and for opting to no longer back up using encryption. For example, if I backed up my iPad using the password *HeatherM11*, the next time I attempt to back up the iPad on ANY computer, iTunes automatically checks the Encrypt Local Backup option. To uncheck this option, I am required to enter *HeatherM11*, which releases the encryption restriction on this device. Keep this in mind in the next section discussing iOS forensic acquisition. This backup encryption can cause issues for you as an examiner.

iCloud provides 5 GB of free data storage for each iOS device. iCloud functions just like iTunes but omits the need to connect to a computer. iCloud also offers a feature called Find My iPhone, which allows the user to remotely locate their device, send a message, or erase the iOS device. As trained examiners, we must prevent remote access from occurring on the iOS device.

Knowing how the user backs up their data may lead you to another device (host computer) or permit access to cloud data, which can open up a whole world of artifacts. Remember, just because we have moved on from Android, we still have to consider Google cloud data for iOS devices. This is something that should also be extracted and examined if possible.

iCloud Sync

- Allows data to be updated and synced across all devices
- Photos and videos (Photo Stream)
- Contacts, calendar, email, etc.
- Pushes data from one device to iCloud account, and then back to all your other iOS devices
- Consider the potential unintended consequences of a shared iCloud

account
- Who is the responsible party?
- Could the data you're interested in be on other devices or the iCloud account?
- CloudConfigurationDetails.plist may reveal the truth

Through iCloud, the user can access, update, and sync their contacts, email, and calendar data from all devices that share the iCloud account. For instance, if a new contact is entered on the iPhone, it will automatically get synced to the user's computer, and vice versa. Reminders (iOS 5 and above), Safari bookmarks, iBooks bookmarks, and Notes can also be synced when enabled on the device. Photo Stream pushes photos taken with one device to all other devices associated with the iCloud account.

When enabled, these items sync across all devices associated with the iCloud account. This can potentially be problematic to an investigation, as it is possible that data could end up on a device without the user's knowledge if someone else on the same iCloud account knowingly or unknowingly synced the data via iCloud. It is important to be sure you know exactly where the data in your investigation came from to avoid the wrong person being held responsible. Make sure you examine the CloudConfigurationDetails.plist to ensure you are not missing the truth.

References:

- https://for585.com/9ykjn (iCloud Sync information)
- https://for585.com/r2x4b (Murphy's Laws of Digital Forensics blog post – Apple Continuity and iCloud Data Leakage when Apple Bites Back)

- Apple Continuity allows for instantaneous activity on multiple devices
- "Handoff": Copy text from a document in one device, paste to

another device

- Receive a call on an iPhone, answer with MacBook Pro or Apple Watch
- Send and receive SMS messages from MacBook and have them instantly updated on iPhone.
- This results in multiple potential sources of evidence!
- What device was used to create the data?

Apple Continuity allows a user to send contact information, presentations, and documents to other iOS users instantly and without network setup or to take a picture or video with your iPad and share it effortlessly across all associated devices.

An Apple Continuity feature called ”handoff” allows users to copy text from a document from one device and paste it into an email on a completely different device. It also allows the user to receive a phone call on their iPhone and answer it with their MacBook Pro or Apple Watch.

Continuity can result in multiple potential sources of evidence. If you can’t get into one device, is there a possibility you might be able to get the evidence you’re looking for from another? Also, it's important to realize that just because data exists on a particular device, it doesn’t mean the user was necessarily ON that device at the time the data was created.

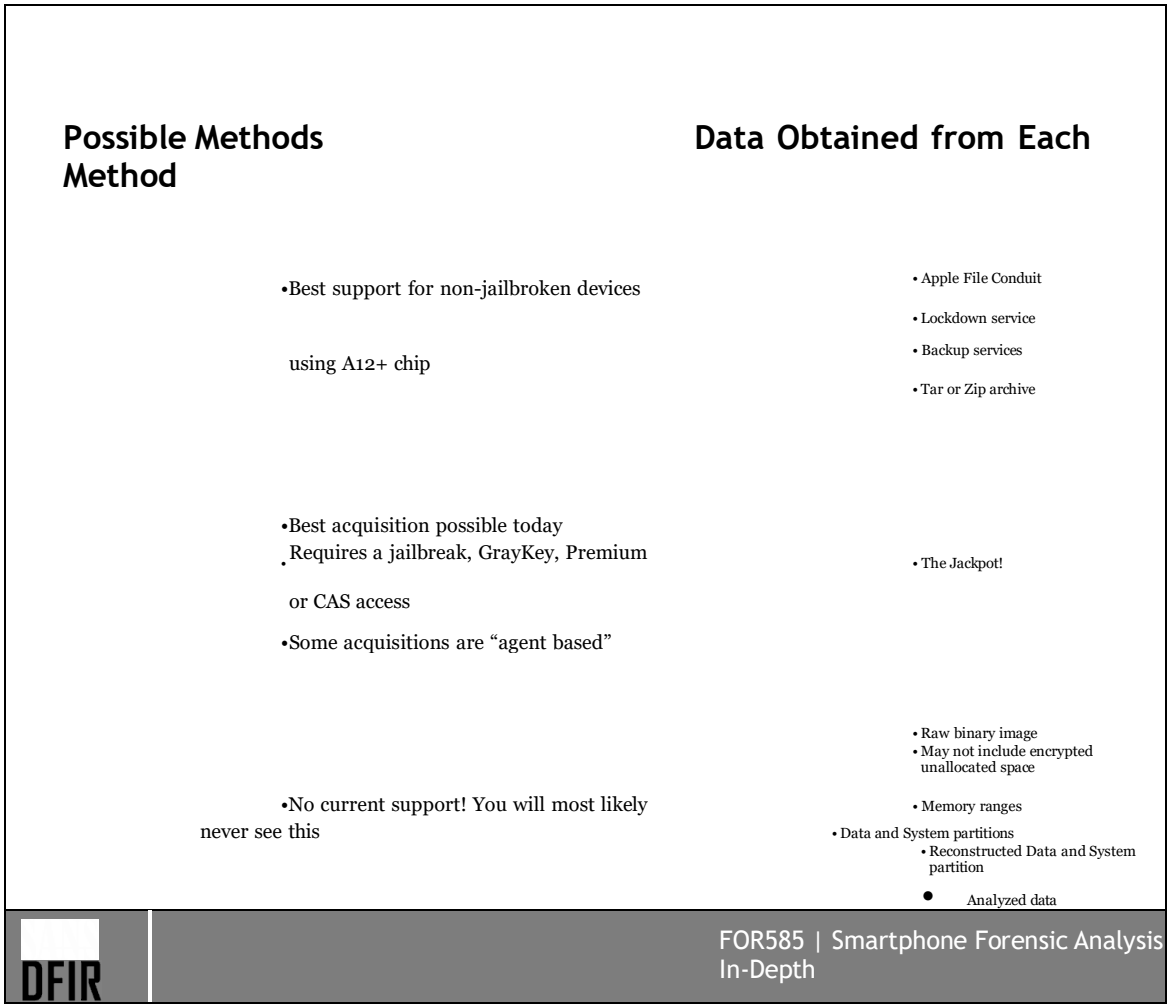
Reference:

https://for585.com/h9tpd (Apple Continuity)

- This means you must be ready for older iOS versions!

This page intentionally left blank.

This page intentionally left blank.



The method for acquiring an iOS device often depends on three variables:

- Is the device locked?
- To which tools do you have access?
- Is the device jailbroken? If not, can you use checkra1n or checkm8?

If the device is unlocked, you can essentially access the device using Logical, Advanced Logical, or full file system acquisition methods with your preferred smartphone forensic tool or forensic workstation. Keep in mind, you must jailbreak in order to get a full file system extraction if you don’t have the specialized tools. Physical acquisition is unlikely present day unless you come across an iPhone preceding the iPhone 5s that is not jailbroken. If you have one of these old devices, we recommend you refer to the

bonus slides and lab within the media files as passcodes could be brute forced and you could simply bypass them. That is a thing of the past unless you have Cellebrite Premium, GrayKey or send your phone to CAS. I haven't been able to physically acquire an iOS device in years. There is a bonus lab on iPhone Physical Analysis within your media files should you decide you want to examine one or even poke around to see the differences.

For locked devices containing the A5+ chip there is not a simple solution that can consistently bypass the locked device, so access to the data may not be possible.[1] If the device is locked and it is running iOS 11+, this is even more difficult. We cover methods for accessing locked devices later in this section. The exception here is jailbroken devices. Elcomsoft iOS Forensic Toolkit can physically acquire jailbroken iOS devices as long as they are not utilizing 64 bit, as the iPhone 6 and current devices implement. The other exception is leveraging BFU or before first unlock. That will be discussed in the lock section coming up.

Some Full file system extractions may be Agent based. Elcomsoft and Belkasoft leverage the agent-based method for obtaining the FFS extraction. At the time of writing this, supported for up to iOS 14.8.

File system acquisitions, by normal standards, provide a logical representation of the files on the device. For iOS devices, users are blocked from accessing certain areas of the file system. Forensic tools, such as Cellebrite, provide access to the file system of the device. The logical data is still obtained during a file system acquisition, but the examiner is also presented access to raw files stored within the file system.

The full file system terminology is one that is exciting everyone and is becoming the new standard! In order to obtain a full file system dump, the device must be jailbroken or acquired using GrayKey, Cellebrite Premium, or a dump created by Cellebrite Advanced Services (CAS). Keep in mind, we have a jailbreak now. We have checkra1n. We also have checkm8 built into UFED.

Physical acquisition is a full memory dump of the device. Tools such as Cellebrite use custom bootloaders to access the areas of flash memory storing data. This is why locked devices can be brute forced into accessing the full flash memory. Most devices are not supported for Physical Acquisition. Physical acquisition of iOS devices is not a true physical dump in that the encrypted areas of unallocated space are not provided to the examiner and are simply not captured. Cellebrite's CAIS may be able to access locked iOS devices and provide physical dumps of the data for a fee.

Free methods for acquiring iOS devices are possible if the device is unlocked and/or jailbroken. Sarah Edwards, the author of FOR518, released a blog on acquiring iOS devices for free, which can be found at mac4n6.com.

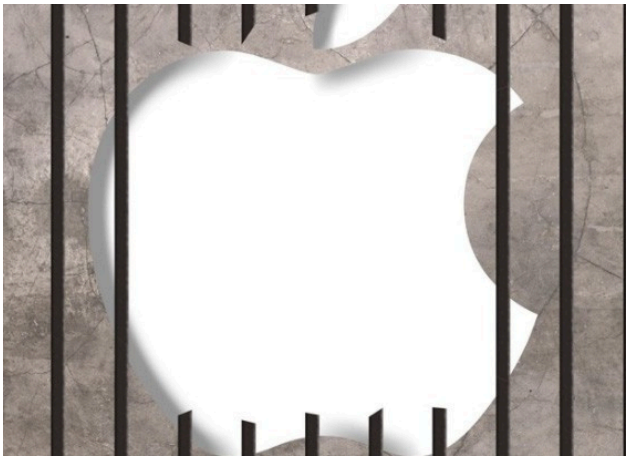
Several tools are available commercially that support Logical, Advanced Logical/File System, Full file system and Physical acquisition of iOS devices. The table shown on the slide in this section titled "iOS Forensic Acquisition Support" remains true for all commercial tools when it comes to device locks and passcodes. Some of the most popular iOS acquisition tools include Cellebrite UFED, Cellebrite Physical Analyzer, Oxygen, Magnet Acquire, Elcomsoft iOS Forensic Toolkit, Belkasoft, MSAB XRY, and more.

To determine the best tool for your investigation, make sure to test all aspects of the tool, not just acquisition, but also analysis.

Reference:
[1] https://for585.com/models

Finally - The Jailbreak and Exploit We Have Been Wanting

- Tethered vs non-tethered – a thing of the past
- The Golden Era is here!
  - ✓ checkm8 – the exploit
  - ✓ checkra1n – the 1st jailbreak
  - ✓ unCover – supports A12 – A14 devices (up to iOS 14.8)
    - ► Others have surfaced
- **You will get full access to the file system**
- Depending on your choice, you may leave traces behind
- Know the difference



Let's all take a moment and reflect – we have come so far. We have been locked out of non-tethered jailbreaks on iOS devices for almost a decade! November of 2019 changed that for us when checkra1n, the first public jailbreak leveraging the checkm8 exploit was released. Now everyone, not just LE, can get a full file system extraction from an iOS device. Based on the current state of iOS forensics, a jailbreak may be your only way to access third-party application files of interest. More on this topic will be covered in detail in Section 5.

You need to be aware in the chance that you stumble upon a device that was previously jailbroken. Determining if the iPhone you have was previously jailbroken can be as simple as viewing the extraction log from your smartphone forensic tool or by manually verifying the settings in the System partition. However, don't expect your tool to always be correct. Jailbreaks have changed over the last decade and most tools are most likely leveraging checkra1n.

The Golden Age of Mobile

Checkm8

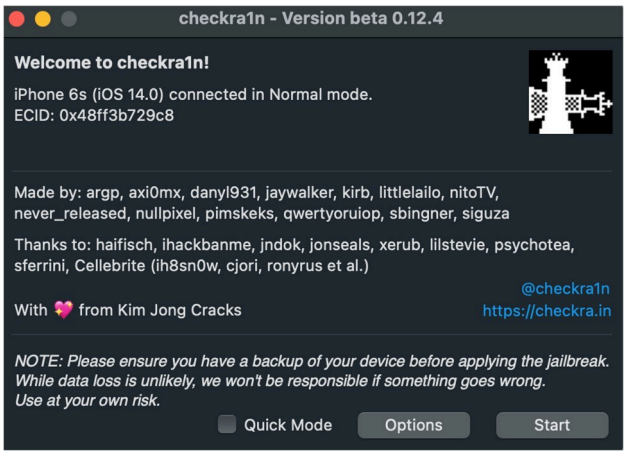
- The exploit
- Built into UFED – runs in RAM – no permanent jailbreak
- Discovered by axi0mX
- Supported on unlocked devices

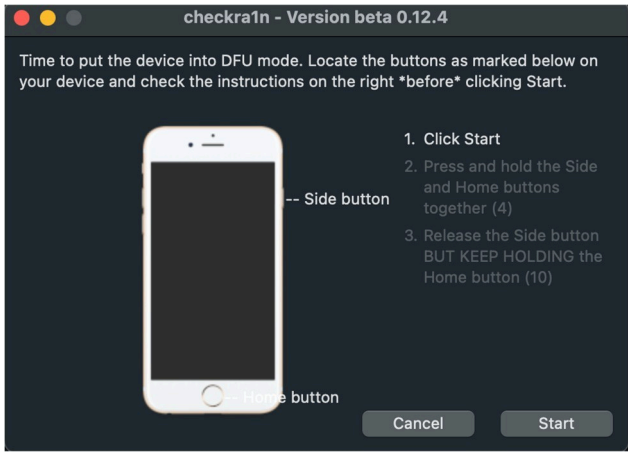
Checkra1n

- The first jailbreak
- An effort of many, including Cellebrite
- Leaves a permanent trace
- Supported up to iPhone X

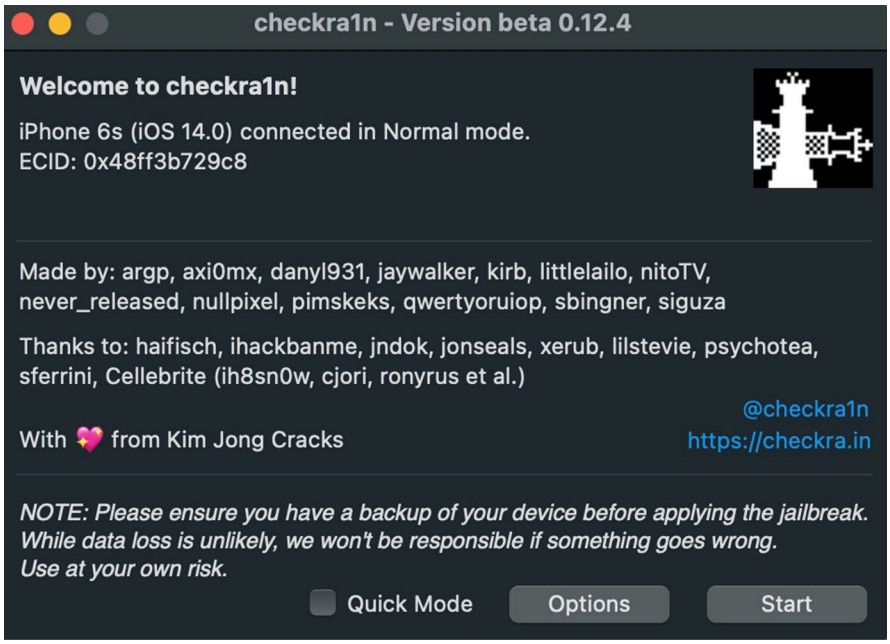
The terms matter and there are differences; checkm8 is the exploit that was discovered by axi0mX. This exploit is built into the UFED Touch2 and UFED4PC and is leveraged to temporarily jailbreak the device using checkra1n. The keyword here is temporary. Cellebrite will not permanently jailbreak the device. After acquisition, the device will reboot as normal and checkra1n will not be present. On the other hand, devices can be jailbroken using checkra1n, the first public jailbreak leveraging the checkm8 exploit. This will leave a permanent trace on the device as you are making a modification to the file system. Tools that support full file system extractions on a checkra1n'd device often require the installation of AFC2 and cydia, which makes additional changes to the device. Cellebrite UFED does not do this as AFC2 and cydia are not needed. Later in the course, you will get to see a device that was extracted using checkm8 in Lab 3.2. Lab 3.3 includes files that were extracted from the checkm8 dump, forcing you to leverage community efforts to parse the data. We hope you enjoy this.

Checkra1n Example (1)

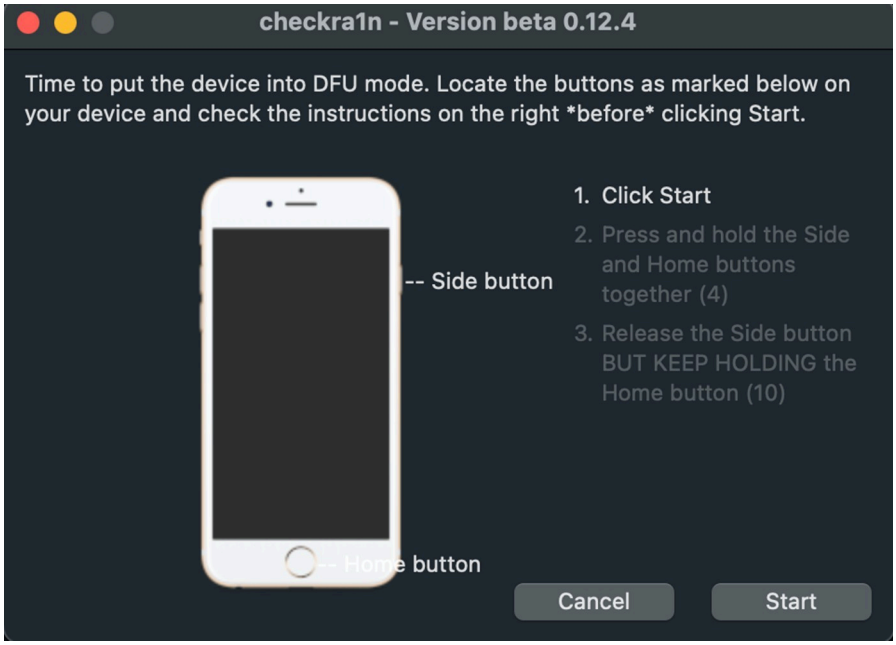




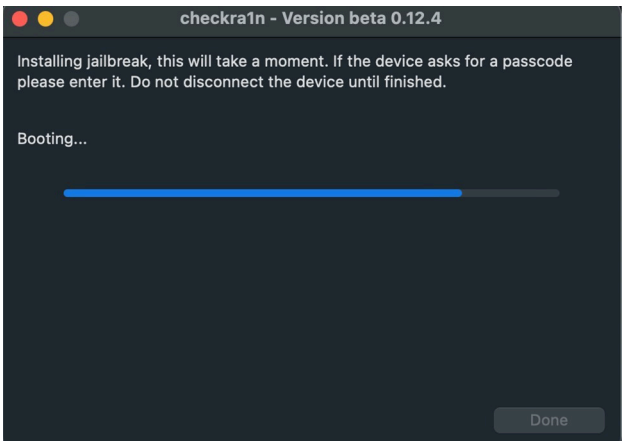
Once you jailbreak your device with checkra1n, you have a tethered jailbreak. This means the jailbreak is tethered to power. If it loses power, you must re-jailbreak the device. In this example, the device was jailbroken on a Mac using checkra1n.

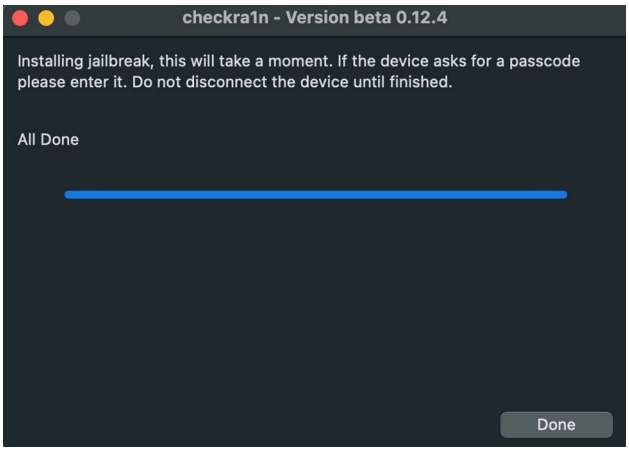


Once you press start, the device will enter recovery mode. From there, you will be promoted to enter DFU (Device Firmware Upgrade) mode.



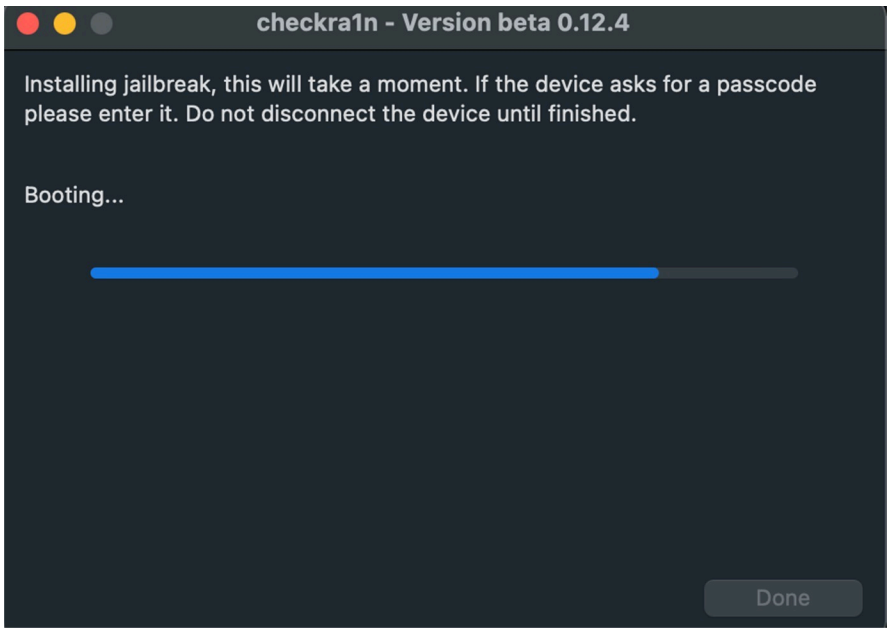
Checkra1n Example (2)



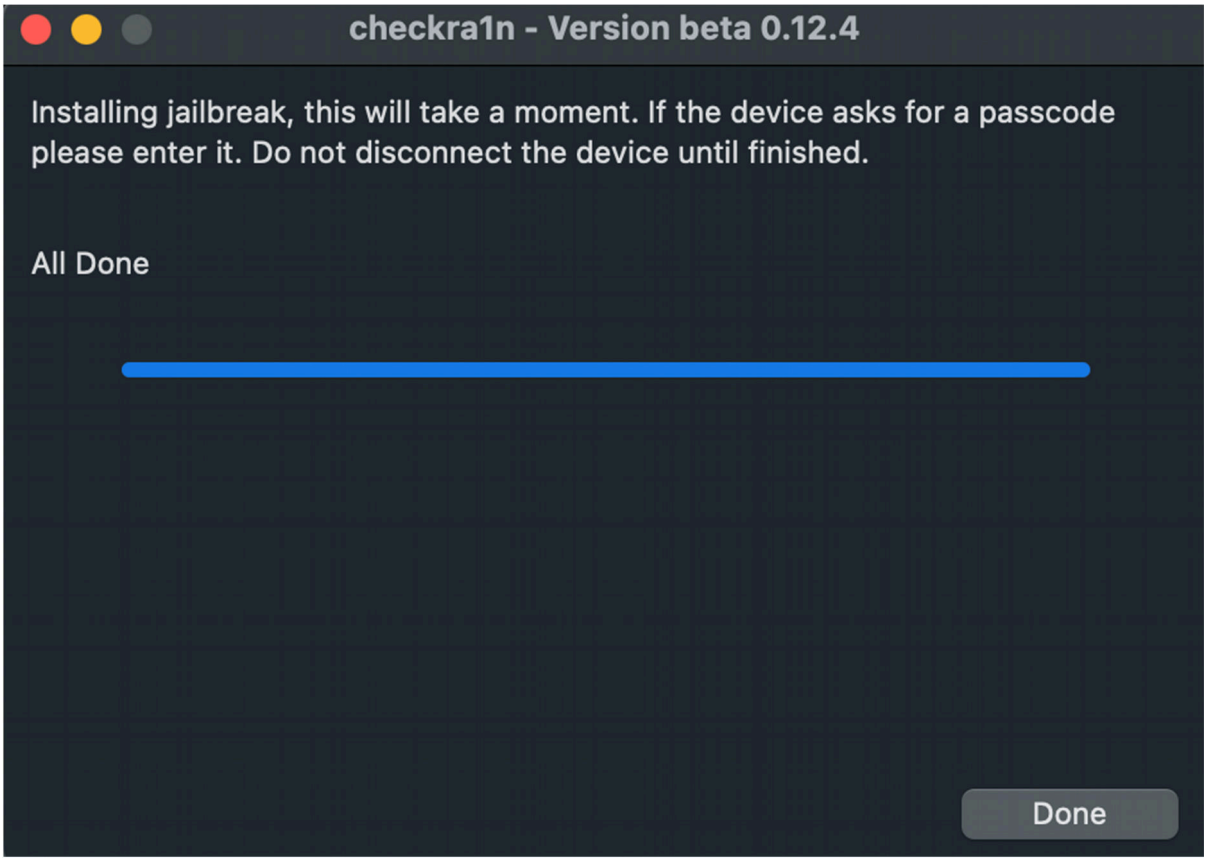


this time unless prompted to do so by checkra

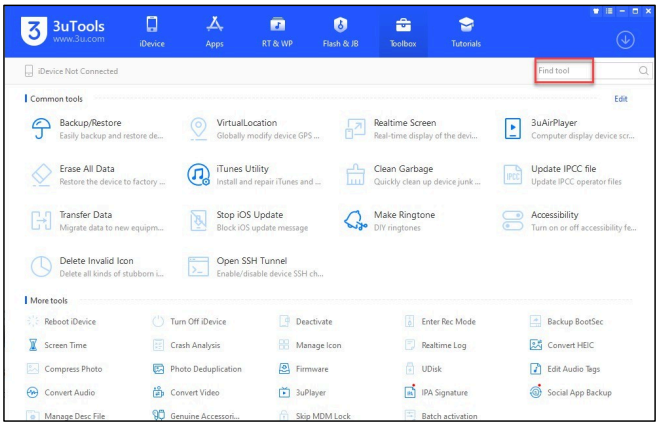
Once in DFU mode, the jailbreak will appear on the device screen. Do not touch the screen or the device during

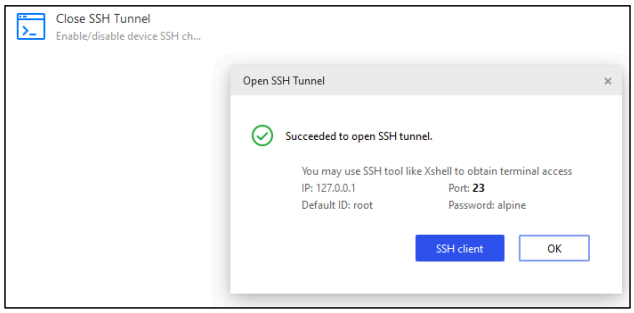


Once in DFU mode, the jailbreak will appear on the device screen. Do not touch the screen or the device during this time unless prompted to do so by checkra1n.



How to Manually Examine a Jailbroken iPhone

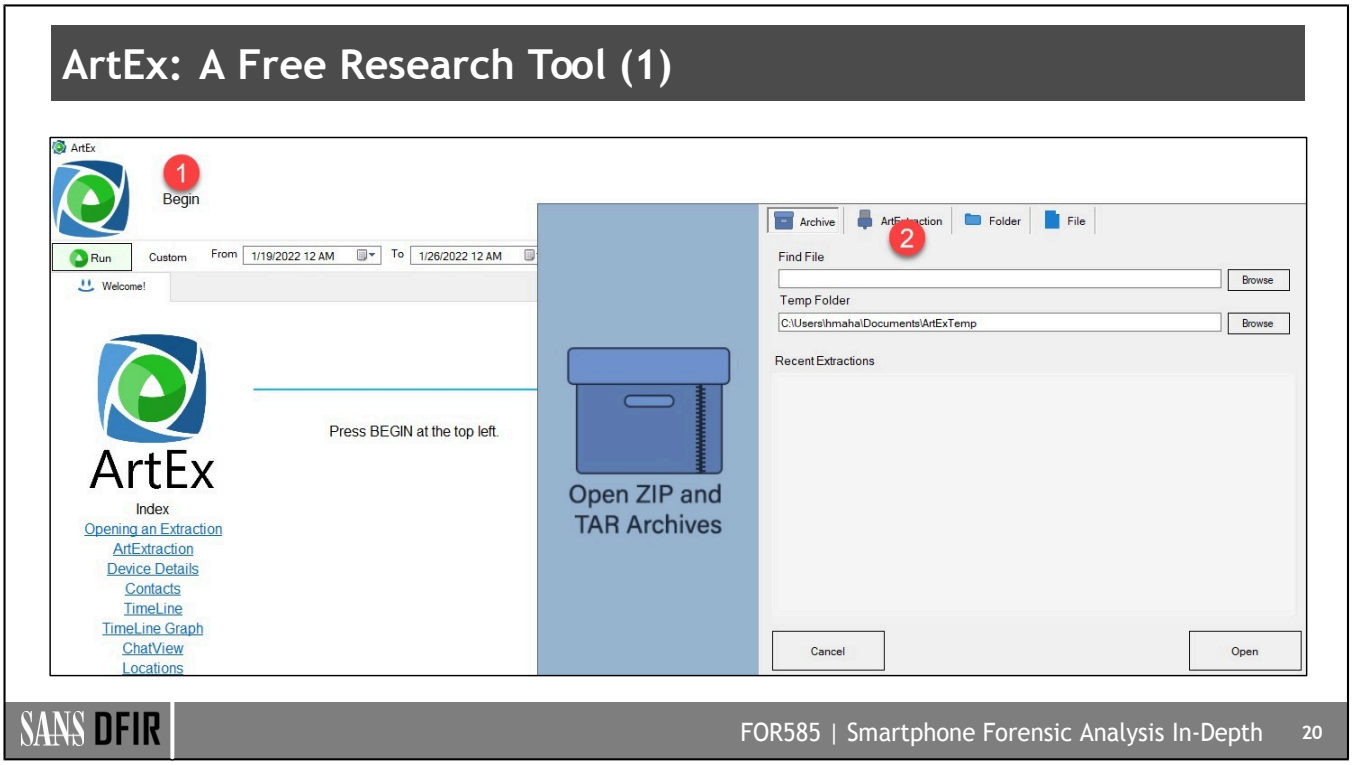




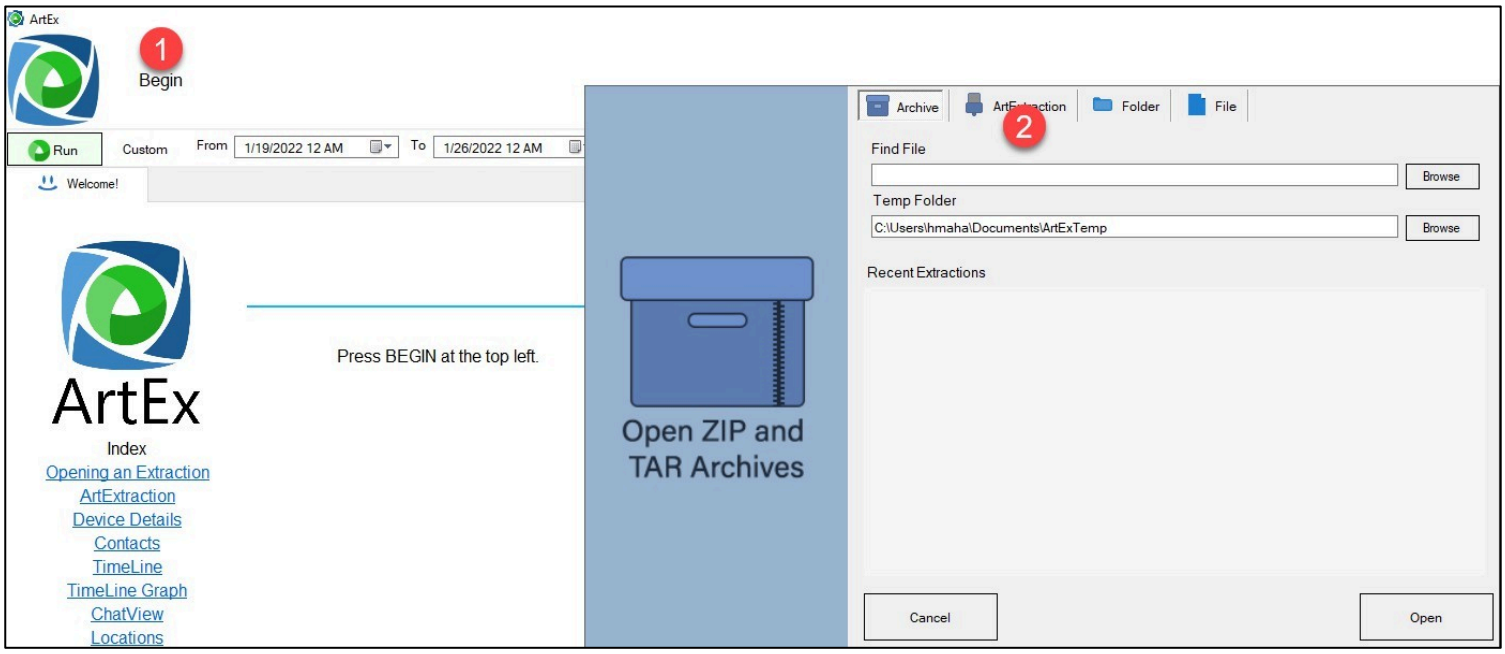
When you have a jailbroken device as evidence, we strongly recommend getting a Full File System extraction. When you have a jailbroken device that you can leverage for research or that has already been acquired, this solution is for you. ArtEx is a tool built for validation. However, it also offers support beyond validation to include acquisition of jailbroken devices and the opportunity to experience a live GUI view of a jailbroken device.

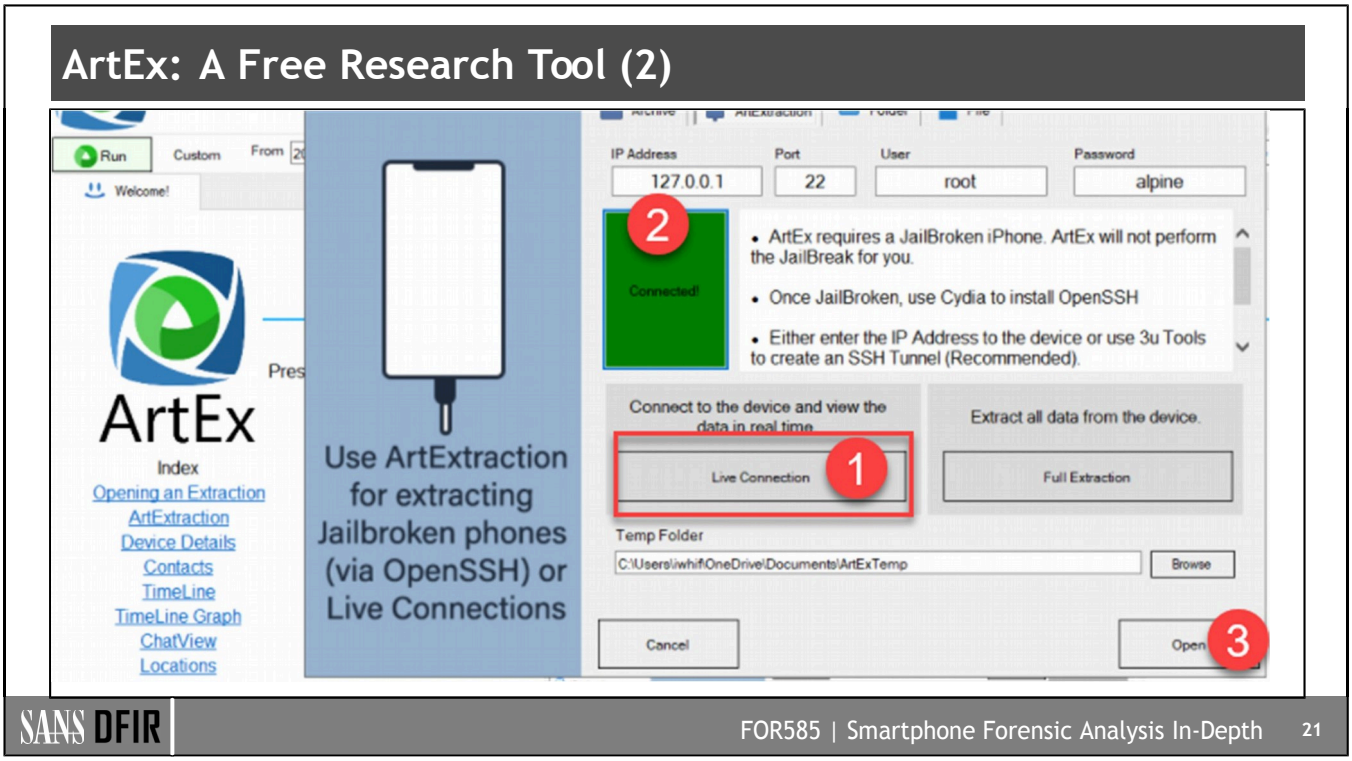
Think about it. You are asked what a deleted contact looks like. You create a contact and delete it. Wouldn't it be so much faster to just manually look at the data on the fly vs. creating a full forensic extraction and waiting for it to parse? With a jailbroken device, this can be done.

First, launch 3uTools and select **Open SSH Tunnel**. If you don't immediately see the option, type SSH into **Find** Tool. Once open, select **OK.** You may get errors if Cydia isn't installed on the device.



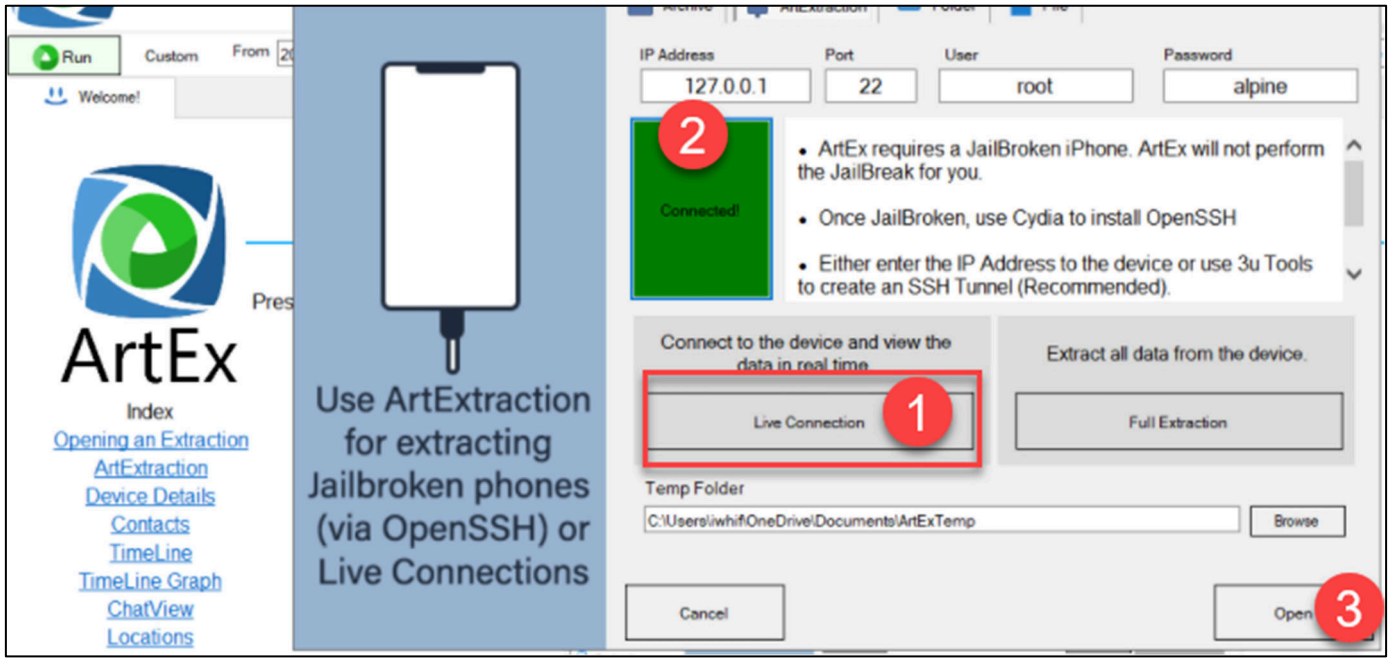
ArtEx, a tool developed by Ian Whiffin, is a great resource for interacting with live iOS devices that are in a jailbroken state. His blog is https://for585.com/artex. Some of the screenshots used in the next few slides are from his blog. Once the SSH Tunnel is open, **Begin**. Next, select **ArtEx Extraction**.



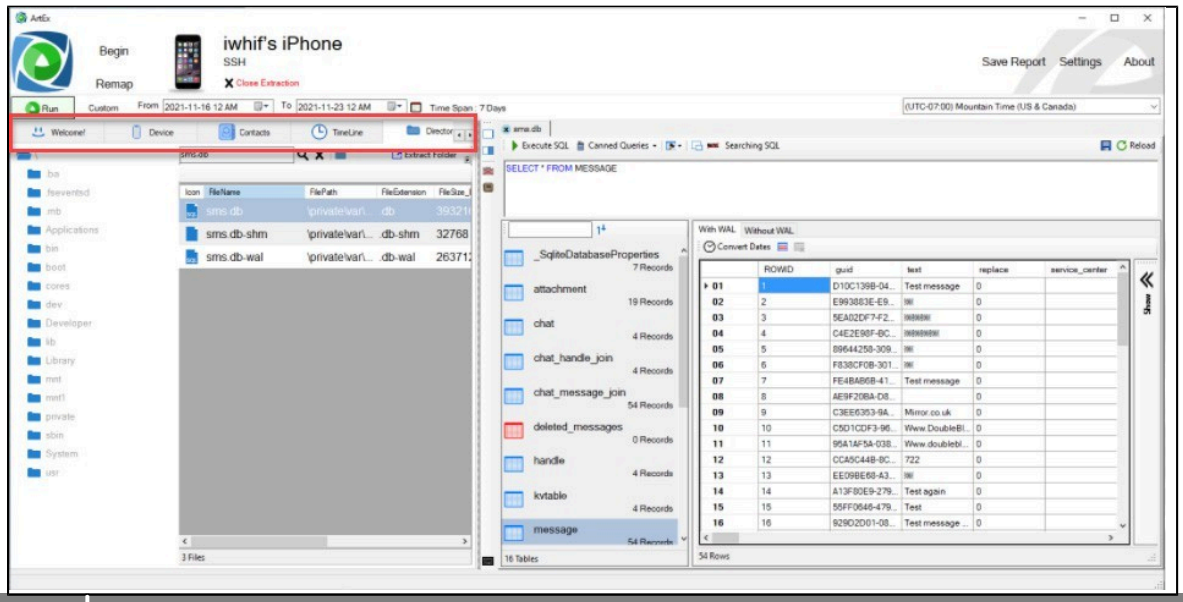


application.

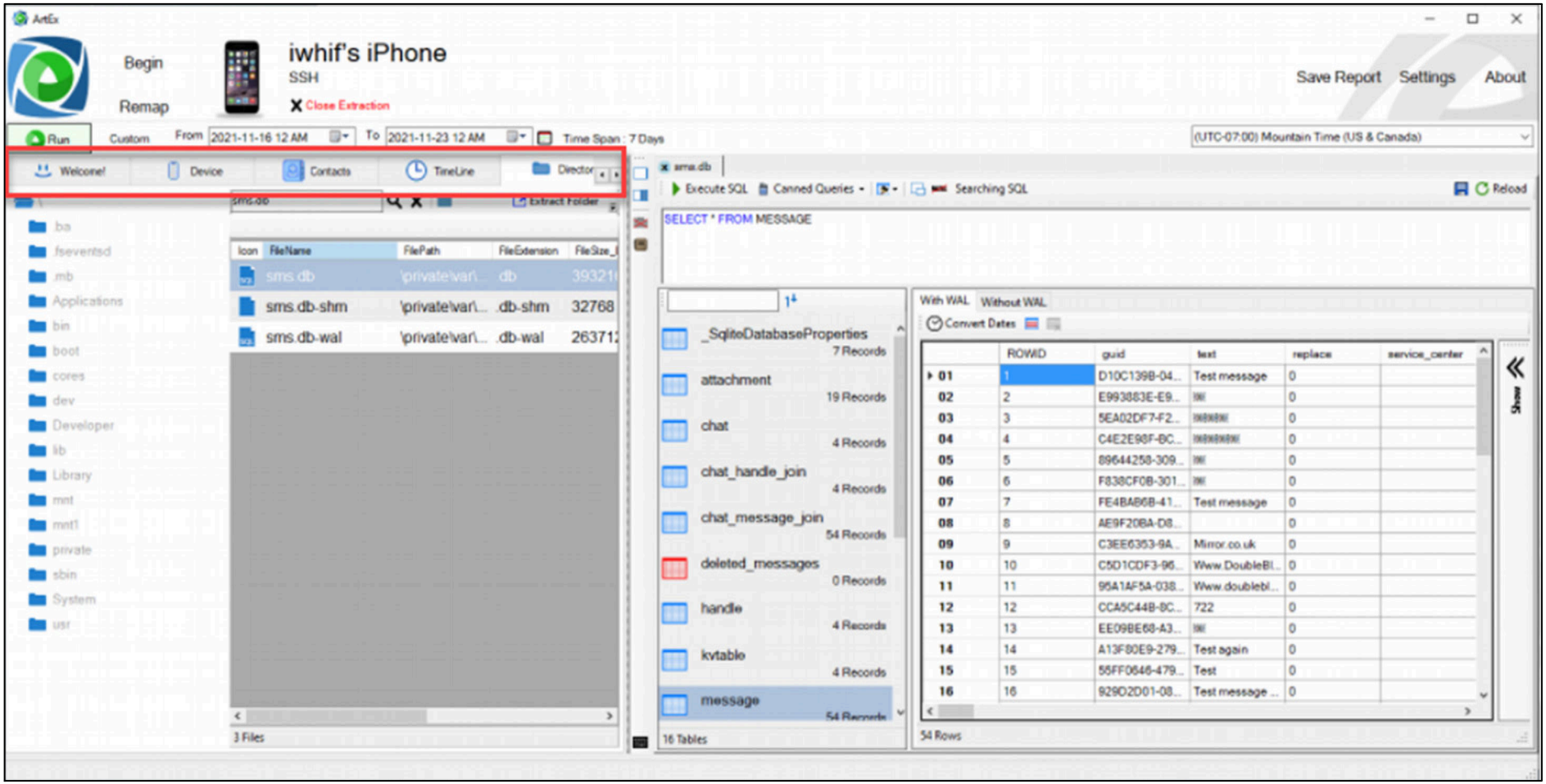
Next, select **Live Connection**. ArtEx requires OpenSSH be installed on the device through the Cydia



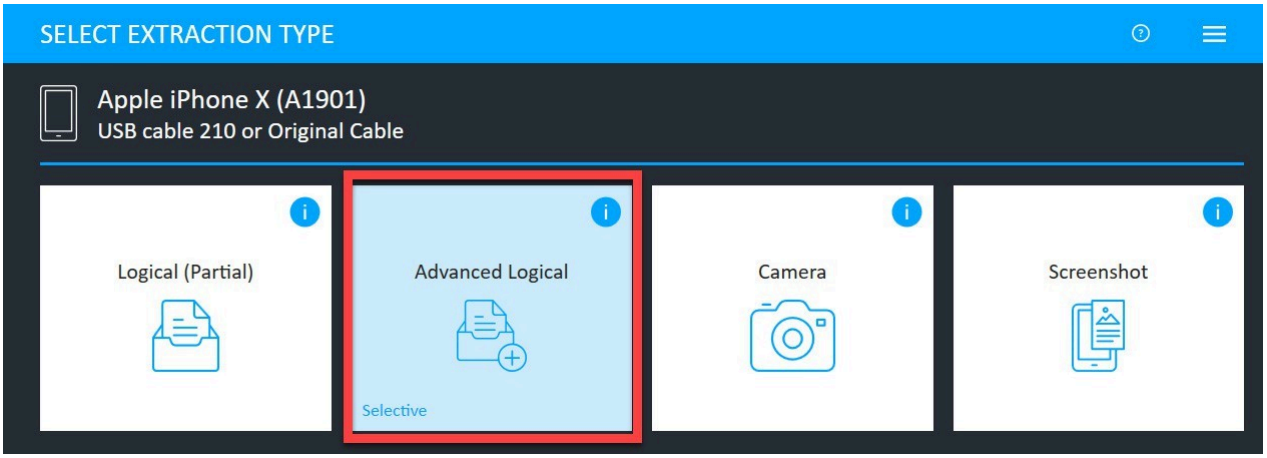
ArtEx: A Free Research Tool (4)



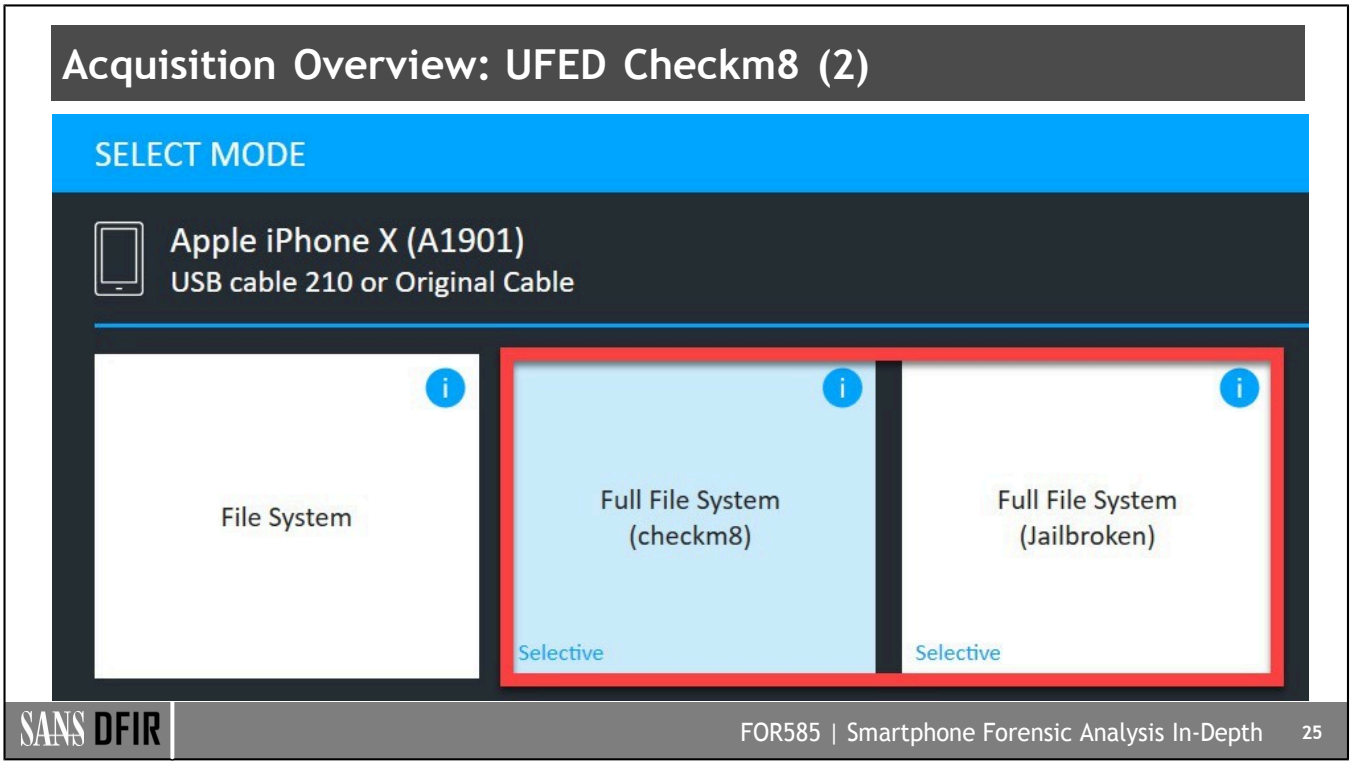
Your live device is now connected to ArtEx. From here, select a timeframe of interest or navigate between Databases, Timeline, Contacts, Device, etc. To open a database, simply click on the file and the database viewer opens on the right. This tool is fantastic, and we recommend using it the next time you are researching or come across a jailbroken device.


ArtEx
Begin
Remap
iwhif's iPhone
SSH
Close Extraction
Save Report
Settings
About
Run
Custom
From 2021-11-16 12 AM
To 2021-11-23 12 AM
Time Span : 7 Days
(UTC-07:00) Mountain Time (US & Canada)
Welcome!
Device
Contacts
TimeLine
Director
sms.db
Extract Folder
Icon
FileName
FilePath
FileExtension
sms.db
sms.db-shm
sms.db-wal
3 Files
Execute SQL
Canned Queries
Searching SQL
Reload
SELECT * FROM MESSAGE
With WAL
Without WAL
Convert Dates
_SqliteDatabaseProperties
7 Records
attachment
19 Records
chat
4 Records
chat_handle_join
4 Records
chat_message_join
54 Records
deleted_messages
0 Records
handle
4 Records
kvtable
4 Records
message
16 Tables
ROWID
guid
text
replace
service_center
54 Rows
Show

Acquisition Overview: UFED Checkm8 (1)

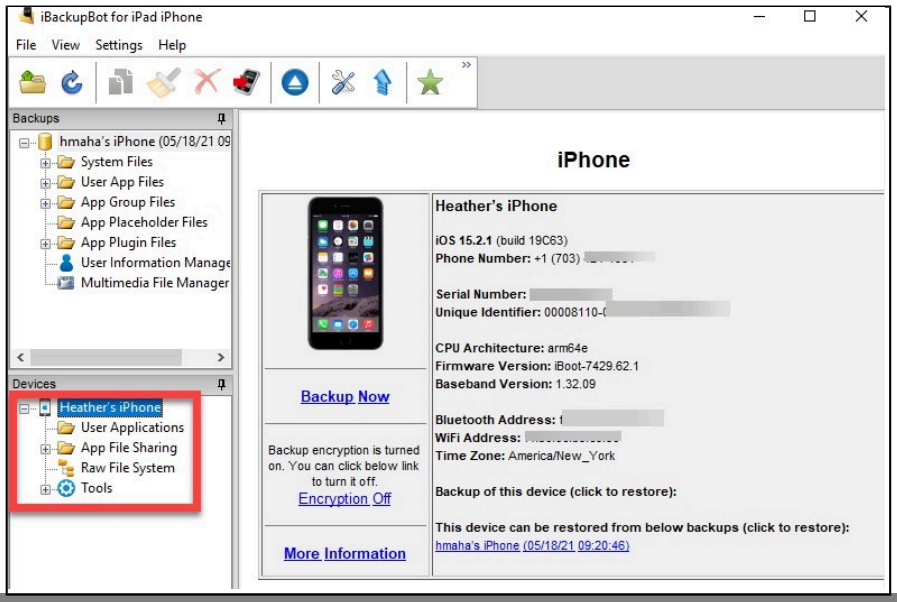


Most commercial tools will support a Full File System extraction for devices that are jailbroken with checkra1n. UFED is no exception to this method. If you have a device that is already jailbroken, select Full File System to obtain either a full or selective extraction, meaning you select the components of the phone you want to extract based upon your investigation. Where possible, a Full File System is recommended because it provides access to the richest dataset.



For non-jailbroken devices, Full File System (checkm8) can be leveraged to obtain either a full or selective file system extraction. The method UFED uses for this extraction runs the jailbreak in RAM, meaning the device is not permanently altered. When rebooted, the token

iBackupBot Method for Obtaining Log Files



If you have the device in your possession and can obtain a file system or logical image, it means you can unlock the device. iBackupBot is a free tool that can be used to capture unique data from iOS devices that the commercial tools seem to overlook. This is always a quick way to triage the files and applications on an iOS device.

In order for iBackupBot to recognize your iOS device, you must launch iTunes and ensure your device is unlocked and connected to iTunes. Once connected, the System Log and Crash Report can be recovered under the Tools section in the bottom left pane. These files show approximately two days' worth of activity that may be relevant to usage of the device.[1] For more information on analyzing these logs, refer to *Learning iOS Forensics*, Second Edition, by Epifani and Stirparo.[1]

Reference:

[1] Epifani and Stirparo, *Learning iOS Forensics*, Second Edition (Birmingham, UK: Packt, 2015).


iBackupBot for iPad iPhone
File View Settings Help
Backups
hmaha's iPhone (05/18/21 09
System Files
User App Files
App Group Files
App Placeholder Files
App Plugin Files
User Information Manage
Multimedia File Manager
Devices
Heather's iPhone
User Applications
App File Sharing
Raw File System
Tools
iPhone
Heather's iPhone
iOS 15.2.1 (build 19C63)
Phone Number: +1 (703)
Serial Number:
Unique Identifier: 00008110-
CPU Architecture: arm64e
Firmware Version: iBoot-7429.62.1
Baseband Version: 1.32.09
Bluetooth Address:
WiFi Address:
Time Zone: America/New_York
Backup of this device (click to restore):
This device can be restored from below backups (click to restore):
hmaha's iPhone (05/18/21 09:20:46)
Backup Now
Backup encryption is turned on. You can click below link to turn it off.
Encryption Off
More Information

Sysdiagnose - The Apple Logging Profiles

Logs created by Apple to help diagnose issues

Crash logs are automatic, sysdiagnose must be triggered by the user or examiner

MOST commercial tools will not extract them

Forensic implications are being researched

Some additional profiles can be installed

- The research documentation:

https://www.for585.com/sysdiagnose

- These scripts are available from GitHub:

https://github.com/cheeky4n6monkey/iOS_sysdiagnose_forensic_scripts

Mattia Epifani, Adrian Leong and Heather Mahalik conducted research and developed scripts to parse some of the logs. There is so much research to be done! This was first presented at OSDFCON 2019, and all slides can be found on smarterforensics.com/presentations. We recommend you read our entire 80+ page document on these logs and their relevance to forensic examination at https://www.for585.com/sysdiagnose.

We plan to keep researching these topics and adding scripts to the GitHub.
https://github.com/cheeky4n6monkey/iOS_sysdiagnose_forensic_scripts

Methods to create the logs, extract the logs and parse the logs are detailed in our research paper. If you are interested in conducting testing with us, please reach out to Heather, Mattia or Adrian. This may help you in a lab later in this section.

Wi-Fi Crash Logs

Wi-Fi goes far back and includes cell towers, too

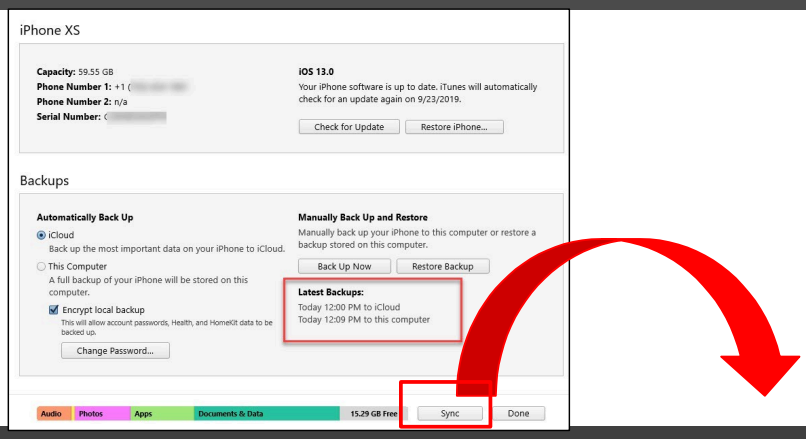
```
12/01/2018 15:32:45.963 WiFiManagerSetKnownNetworksCache: Updated 232 known networks in the cache
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: skipping disabled network _Heathrow Wi-Fi
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: skipping disabled network Google Starbucks
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: skipping disabled network BWHbgMechanicsburgInn&Suites
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: skipping disabled network Power Play Guest
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: skipping disabled network Caesars_Resorts
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: skipping disabled network MGMResorts-WiFi
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: Apple Store is disabled until first user join
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: skipping disabled network attwifi
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: skipping disabled network AmtrakConnectStation
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: skipping disabled network ResidenceInn_GUEST
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: skipping disabled network WineKitchen_EXT
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: skipping disabled network Marriott_CONFERENCE
12/01/2018 15:32:45.965 WiFiDeviceManagerSetNetworks: skipping disabled network Westin Reston Heights
12/01/2018 15:32:45.965 WiFiDeviceManagerSetNetworks: skipping disabled network Hilton Honors
12/01/2018 15:32:45.965 WiFiDeviceManagerSetNetworks: skipping disabled network Millennium_Guest
12/01/2018 15:32:45.965 WiFiDeviceManagerSetNetworks: skipping disabled network FLAMINGO
12/01/2018 15:32:45.965 WiFiDeviceManagerSetNetworks: skipping disabled network AmtrakConnect
12/01/2018 15:34:41.517 age diff 0.000202, accuracy diff 0.000000
12/01/2018 15:34:41.517 didUpdateLocations: latitude=+59.330741, longitude=+18.056106 Accuracy=1115.611906  timeIntervalSinceNow=0.002604secs
source=Cell
12/01/2018 15:34:41.522 age diff 0.005653, accuracy diff -0.000016
12/01/2018 15:34:41.522 didUpdateLocations: latitude=+59.331031, longitude=+18.056121 Accuracy=1115.611922  timeIntervalSinceNow=0.002017secs
source=Cell
```


The Wi-Fi logs may or may not exist by default. To be honest, we are not sure why some devices have them and others do not. What we do know, is that the Wi-Fi log contains a ton of data that goes back far in history. It tracks both Wi-Fi and cellular and includes cleared connections and devices that the iPhone/Apple Watch were in range of. Some things we need to test – what if the user loses service or puts the device into airplane mode. What happens if there is a space issue on the device? From the initial tests, it seems as if the device will make space for the logs as they are not written to the user partition. Finally, does GrayKey and Cellebrite Premium provide access to these logs, or do you have to dump them yourself? Stay tuned for more research.

```
12/01/2018 15:32:45.963 WiFiManagerSetKnownNetworksCache: Updated 232 known networks in the cache
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: skipping disabled network _Heathrow Wi-Fi
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: skipping disabled network Google Starbucks
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: skipping disabled network BWHbgMechanicsburgInn&Suites
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: skipping disabled network Power Play Guest
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: skipping disabled network Caesars_Resorts
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: skipping disabled network MGMResorts-WiFi
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: Apple Store is disabled until first user join
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: skipping disabled network attwifi
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: skipping disabled network AmtrakConnectStation
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: skipping disabled network ResidenceInn_GUEST
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: skipping disabled network WineKitchen_EXT
12/01/2018 15:32:45.964 WiFiDeviceManagerSetNetworks: skipping disabled network Marriott_CONFERENCE
12/01/2018 15:32:45.965 WiFiDeviceManagerSetNetworks: skipping disabled network Westin Reston Heights
12/01/2018 15:32:45.965 WiFiDeviceManagerSetNetworks: skipping disabled network Hilton Honors
12/01/2018 15:32:45.965 WiFiDeviceManagerSetNetworks: skipping disabled network Millennium_Guest
12/01/2018 15:32:45.965 WiFiDeviceManagerSetNetworks: skipping disabled network FLAMINGO
12/01/2018 15:32:45.965 WiFiDeviceManagerSetNetworks: skipping disabled network AmtrakConnect
```

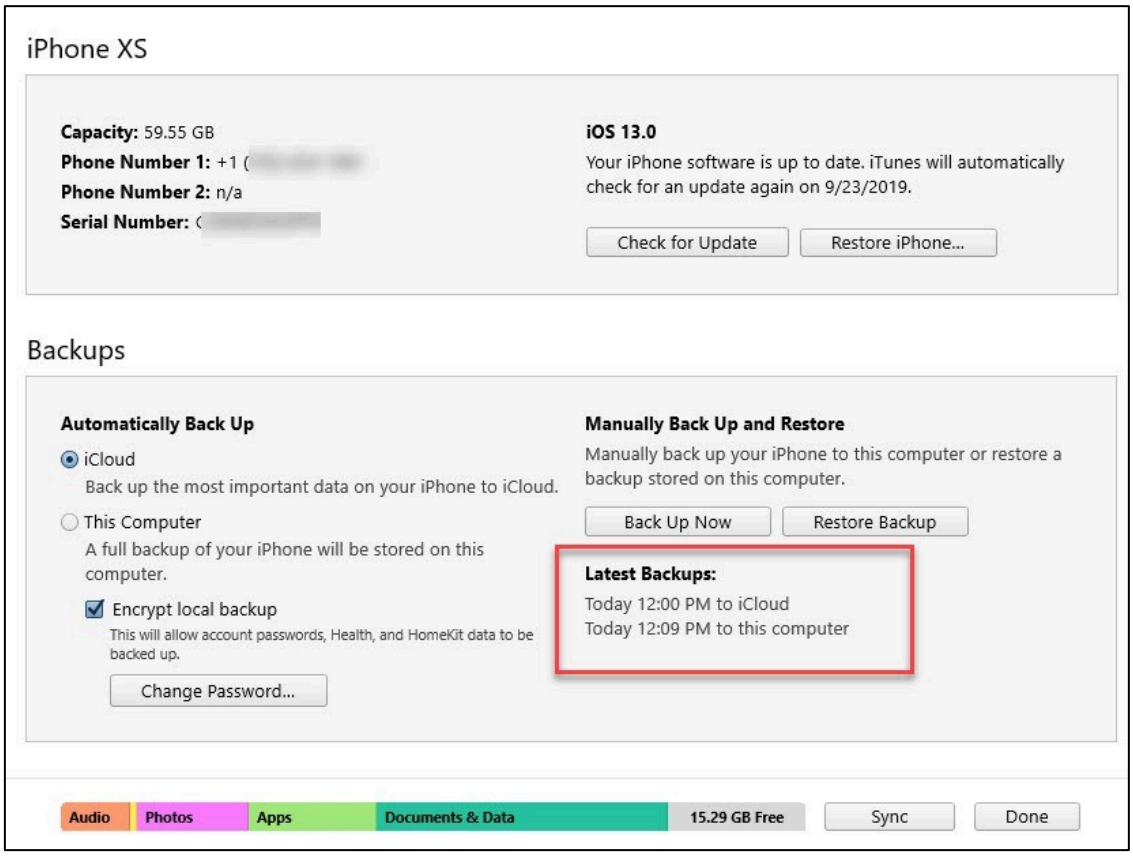
```
12/01/2018 15:34:41.517 age diff 0.000202, accuracy diff 0.000000
12/01/2018 15:34:41.517 didUpdateLocations: latitude=+59.330741, longitude=+18.056106 Accuracy=1115.611906  timeIntervalSinceNow=0.002604secs
source=Cell
12/01/2018 15:34:41.522 age diff 0.005653, accuracy diff -0.000016
12/01/2018 15:34:41.522 didUpdateLocations: latitude=+59.331031, longitude=+18.056121 Accuracy=1115.611922  timeIntervalSinceNow=0.002017secs
source=Cell
```

Sysdiagnose (3)



OS	Path
macOS	/Users/<username>/Library/Logs/CrashReporter/MobileDevice/[Device_Name]/
Windows	C:\Users\<username>\AppData\Roaming\Apple Computer\Logs\CrashReporter\MobileDevice\[Device_Name]\

Elcomsoft supports the extraction of the log files, both crash and user/examiner generated. This can also be done with iTunes, which is shown above. Make sure you have a fresh install of iTunes to prevent cross contamination. If you have updated your Mac, this can be done via the Finder. Simply select Sync and navigate to the logs and run the scripts that have been provided for parsing.



Acquisition Considerations and Suggested Steps

Determine the iOS type

Can you checkm8/checkra1n the device? Get an FFS

- If not, obtain at least a File System/Advanced Logical Acquisition

Obtain a backup if you are paranoid

- Make sure to encrypt it!

Obtain the Crash Logs from iBackupBot

Make sure your tools did not encrypt the backup and *forget* to decrypt it – if returning the phone

Consider cloud data

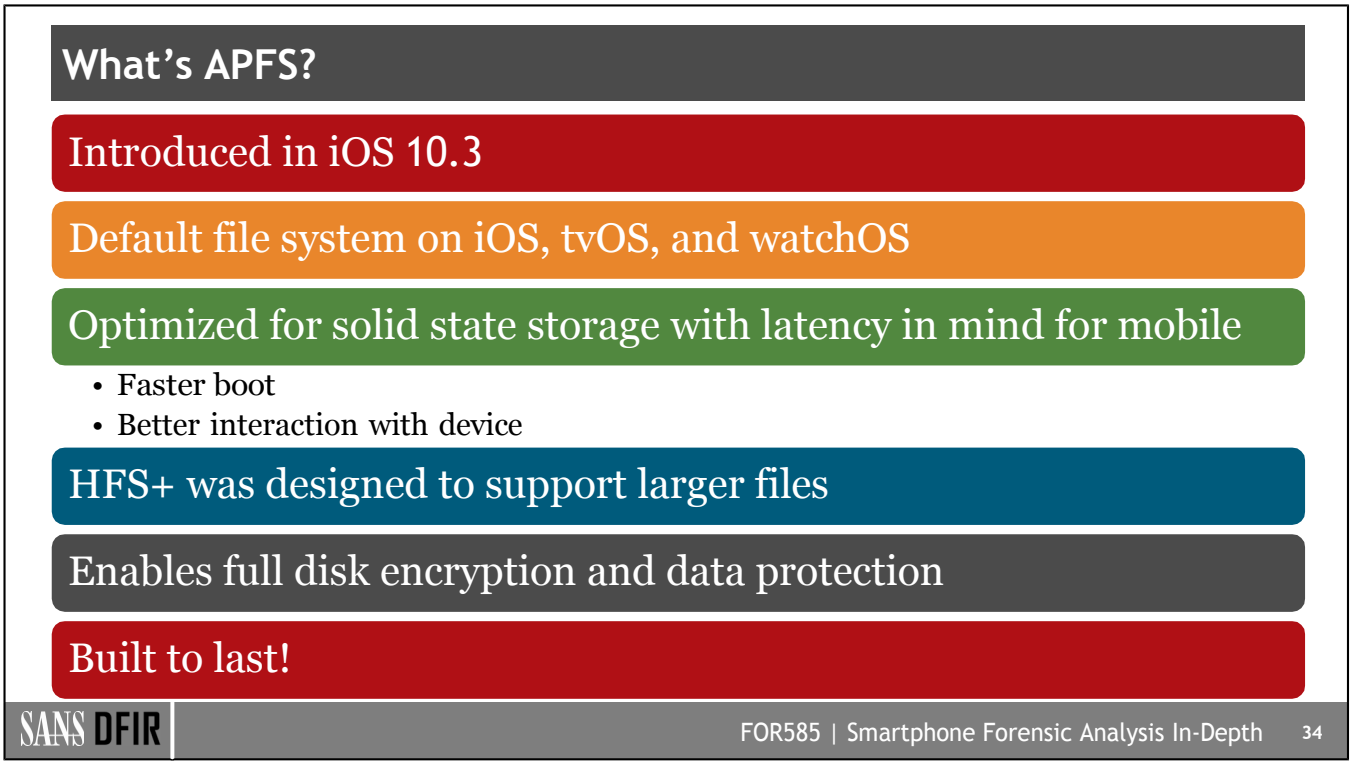
Consider Sysdiagnose and other Apple Profiles

All of these topics were just touched upon with the exception of cloud data and removing the iTunes encryption. This will be covered later in this section and in Section 4. For more information on iOS and Android acquisition, read the blog written by Heather Mahalik: https://smarterforensics.com/2021/12/android-and-ios-acquisition-recommendations/.

iOS Forensics Agenda

- Section 3.1: iOS Forensics Overview
- Section 3.2: iOS Device Acquisition Considerations
- Section 3.3: iOS File System Structures
- Section 3.4: iOS Evidentiary Locations
- Section 3.5: Handling Locked iOS Devices
- Section 3.6: Advanced Decoding and Traces of User Activity

This page intentionally left blank.



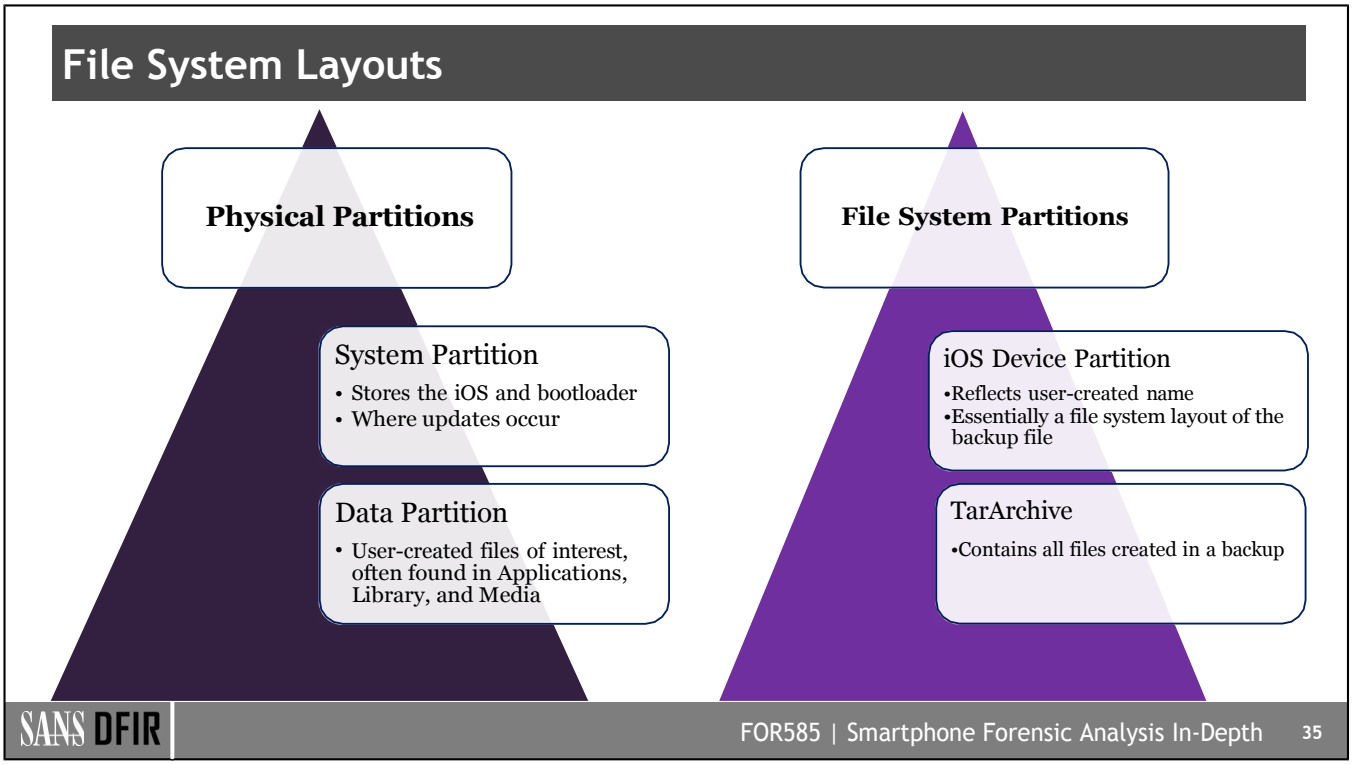
APFS was introduced in 2017 as the replacement or upgrade from the 30-year-old HFS file system. While HFS still exists on iOS devices, it is no longer being used on newly manufactured products, and older devices are often capable of upgrading to this new file system. APFS is new and we examiners are constantly evolving with it—meaning we don't know everything just yet!

What we do know is that APFS is now the default file system for iOS (starting with 10.3), tvOS, and watchOS. It's as if mobile was in mind when APFS was designed. The file system is built to function on solid state storage with latency in mind (faster boot and no need for storing extremely large files, which kill precious mobile space) and provides an overall better experience for the user. HFS+, on the other hand, was built to support really large files, which we don't commonly see on mobile devices. Bulkiness is not a friend for mobile.

Of course, Apple maintained the protection it has been offering to the user for quite some time, and APFS doesn't disappoint. Full disk encryption is enabled, and data protection is in place. These topics will be covered later in this section.[1]

Reference:

[1]: https://for585.com/apfs



A Physical acquisition of an iOS device provides access to the full file system of the device. By default, the System partition is set read-only by Apple, preventing access to the user. The Data partition is used to store user-created files, applications, and more.

The Library, Applications, and Media directories are where a majority of the user data gets stored, and most of this directory is contained in a backup file when the user backs up the device using iTunes.

Jailbreaking an iOS device provides the user with read/write access to the System partition. Accessing the System partition provides users with more control and the ability to install non-authorized third-party applications but voids the Apple warranty. Old methods for Physical acquisition used to force forensic examiners into jailbreaking a device in order to gain access.

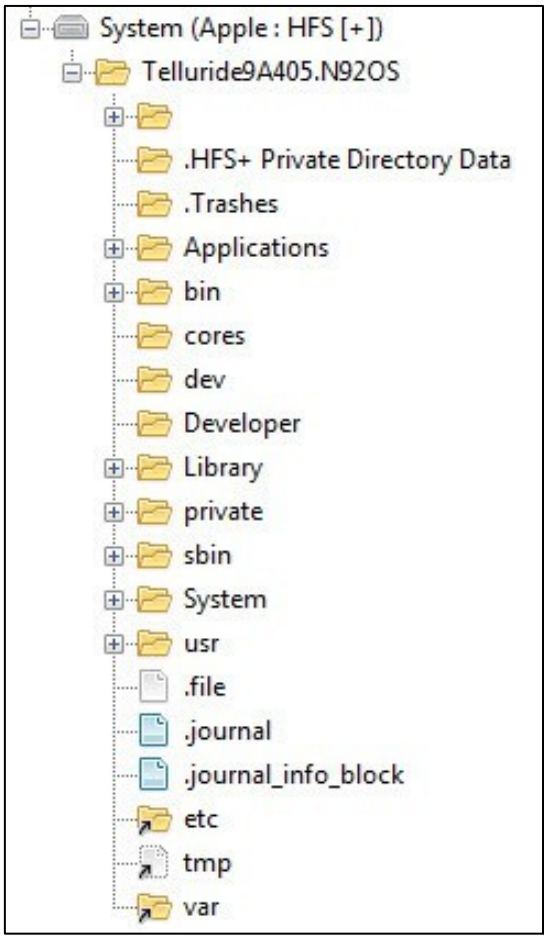
The /private/var/mobile holds most of the user data for the iOS device. The Applications folder holds all downloaded apps and associated data. Each folder under the applications is named according to the application identifier. The remaining data is stored in the Library and Media folders.

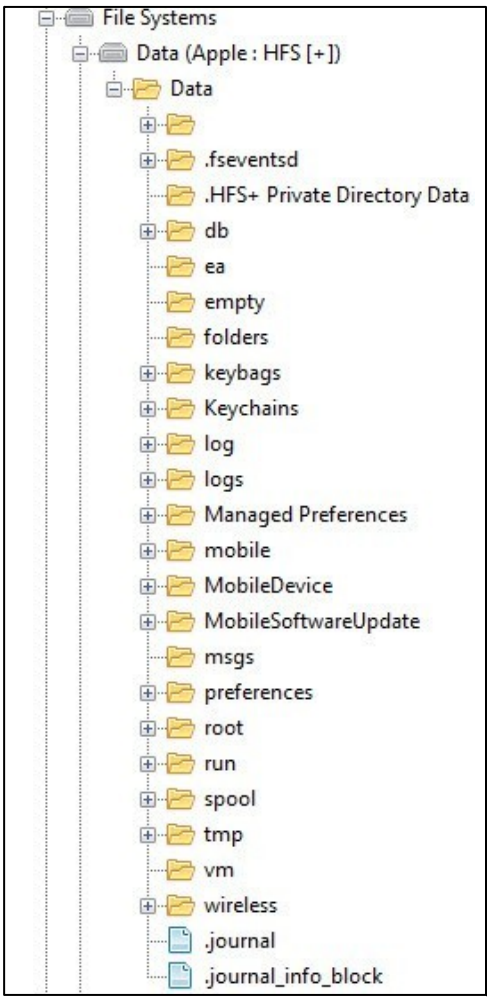
Examples of the Physical partitions are shown below. Notice that both the System and Data partition rely on the HFS+ file system. The System partition is displayed on the left and the Data partition is displayed on the right. Again, the method of acquisition may affect the data that is shown in Physical Analyzer. The System partition has the volume name of "Telluride9A405.N92OS." This volume name can be broken down as follows:[1]

Codename = Telluride
Build = 9A405

Reference:

[1] https://for585.com/iwiki (The iPhone Wiki Page)



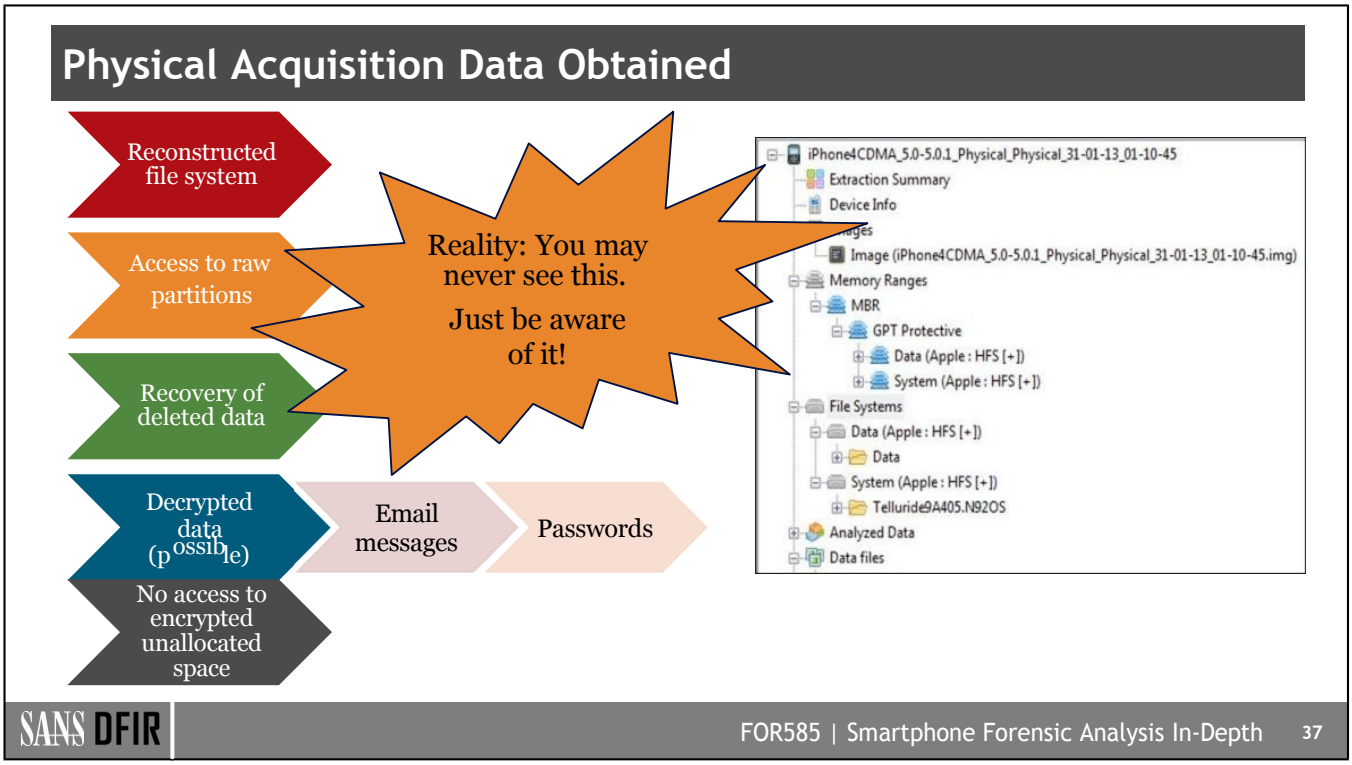


When examining an iOS file system dump (for those devices not using the A5 chip) in Physical Analyzer, two partitions are listed under the File Systems. The first partition is named for the iOS device itself. For example, "Heather's iPhone" would represent an iPhone that was named Heather by the user. The second partition is the TarArchive, which contains all the backup files, similar to what is created when backed up with iTunes.

For iOS devices using the A5+ chip, the file system may simply be one zip file containing the AFC Service, Backup Service, and Lockdown Service data.

An advanced logical/file system dump may not provide access to deleted data during parsing. However, deleted data may be residing in the database files and can be recovered using methods taught in this course.

Keep in mind that a “physical” keyword search does not work on advanced logical/file system dumps. Because the data is a .tar archive of a backup, there isn’t a raw container or image file to search. Thus, logical searching provides the most results. Oxygen is a better tool to use when it comes to searching content on iOS advanced logical or file system dumps.

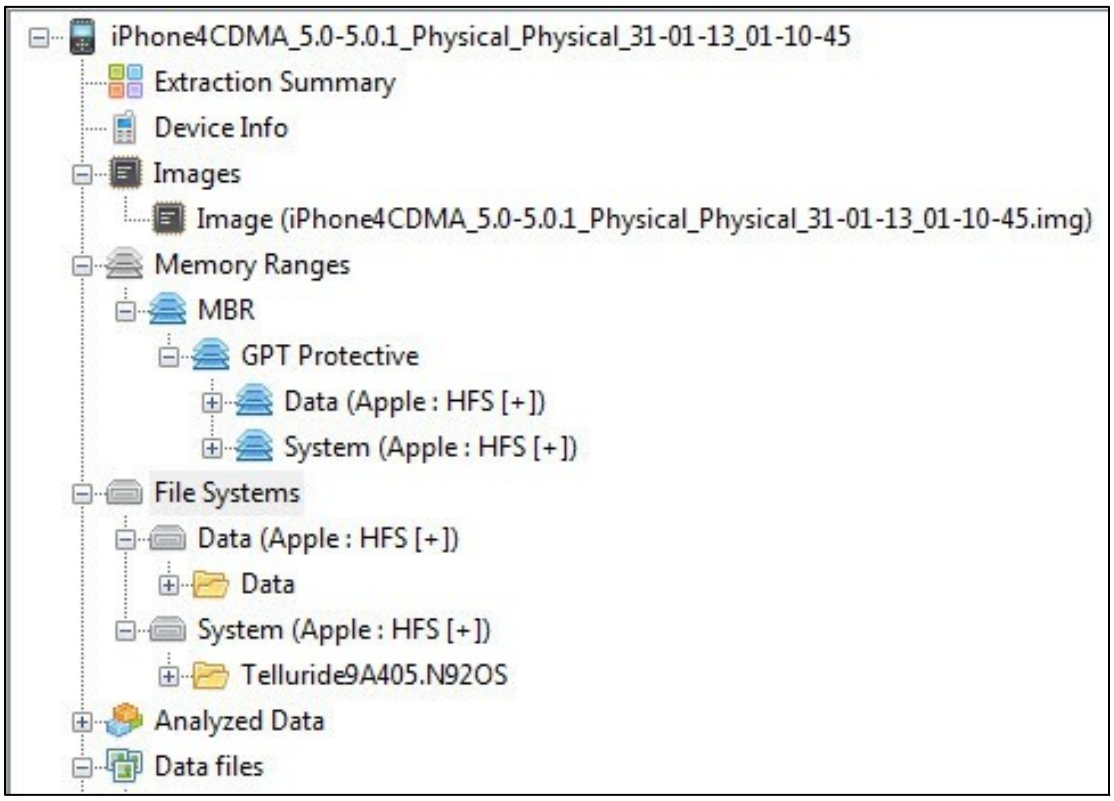


Chances are you may never see this type of acquisition unless a company or service provides you access, such as CAS. Chances are better if you are willing and able to jailbreak the device! Physical acquisition captures the raw file image, listed under Images in Physical Analyzer, as shown in this slide. This provides the examiner access to the entire raw Hex dump of the data pulled from the iOS device. NOTE: The only areas of unallocated space provided are the areas that can be decrypted by the forensic tool. Starting with iOS 4, encryption was enabled on the device. Older versions of iOS devices allow the full forensic capture of unallocated space for analysis and data carving.

The best and most comprehensive search should be conducted within the raw image file. The Memory Ranges are comprised of the Data and System partitions in their raw form, while the File System shows a normalized version of both the Data and the System partition, including a folder structure for each. The File System area of the data shown in Physical Analyzer is the reconstructed file system, as interpreted by the tool. Data is stored natively within the partitions and can be manually examined and decoded for relevance, which is discussed later in this section.

The Analyzed Data section in Physical Analyzer shows all data that Physical Analyzer knows how to parse. Again, use caution here and verify their findings rather than just trusting the tool. More details on how to verify your results and manually decode data are covered in the analytical portion of this section. The Analyzed Data section does not parse all forms of data available on the iOS device. The examiner must dig deeper to recover the overlooked data.

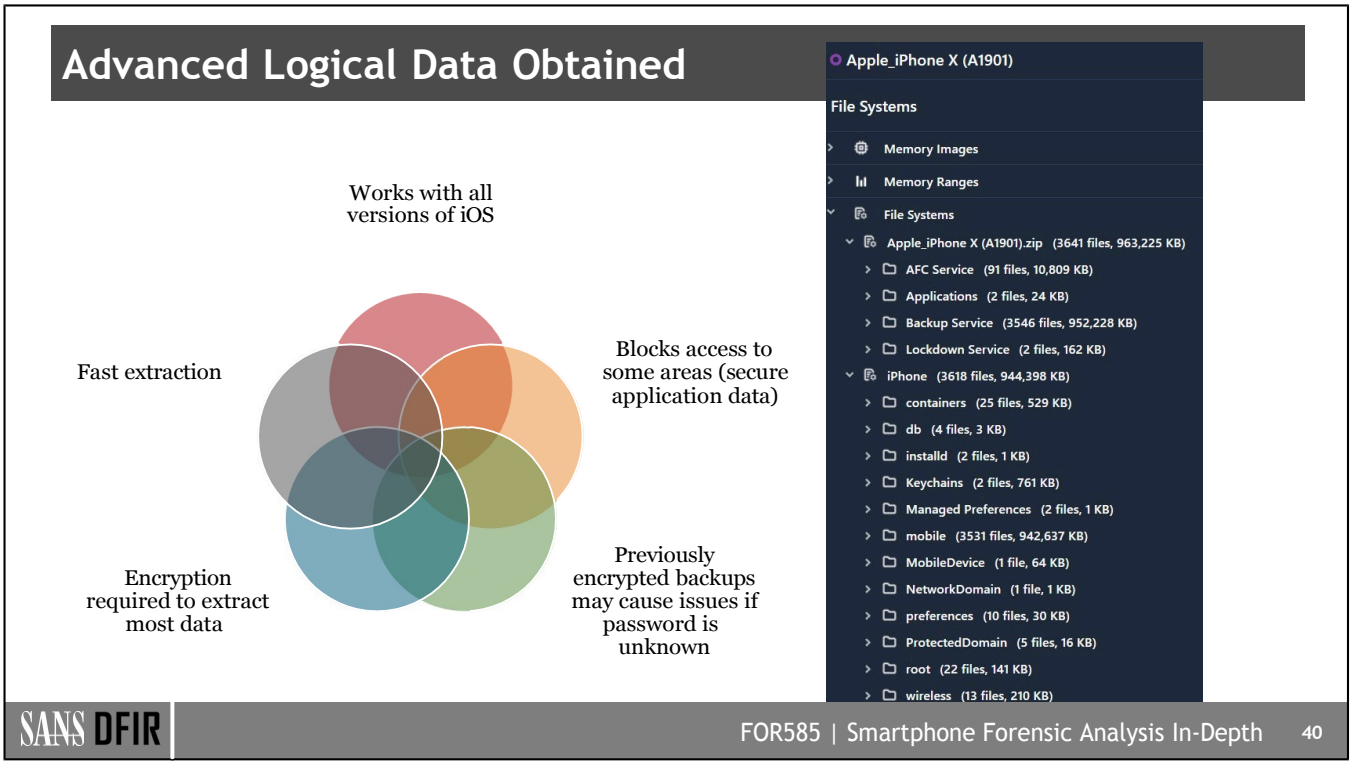
If the iOS device is locked with a simple four-digit passcode, iOS Physical Mode recovers the passcode for you on iOS devices released prior to 2011 (refer to the iOS Forensic Acquisition Support chart). If the iOS device is locked with a complex passcode, you can manually try as many passcodes as you like using Physical Analyzer or continue the extraction without being able to decrypt some of the saved passwords and email. Handling locked devices is covered in the next section. If the device isn't locked with a passcode, all data is extractable for iOS devices not using the A5+ chip.


iPhone4CDMA_5.0-5.0.1_Physical_Physical_31-01-13_01-10-45
Extraction Summary
Device Info
Images
Image (iPhone4CDMA_5.0-5.0.1_Physical_Physical_31-01-13_01-10-45.img)
Memory Ranges
MBR
GPT Protective
Data (Apple : HFS [+])
System (Apple : HFS [+])
File Systems
Data (Apple : HFS [+])
Data
System (Apple : HFS [+])
Telluride9A405.N92OS
Analyzed Data
Data files

Full File System Data Obtained

- KnowledgeC
- Health
- Mobile Installation Logs
- Battery Consumption Logs
- Interaction Logs
- Event Logs
- The best of the file system you will see!

A full file system extraction is our best method of acquisition on current devices. This is the closest we get to a physical extraction. Here, you can see that we have access to application usage files, health, event logs and more. Your labs will really cement the difference in a full file system acquisition vs a traditional advanced logical acquisition.



The option to conduct an Advanced Logical/File System dump was introduced by Cellebrite. Other tools now offer similar access to the device. This type of acquisition requires that the device be unlocked. When conducting an extraction, encrypting the extraction will capture more data and is recommended. A comparison is shown in the next slide.

As shown in the screenshot, a normal (non-jailbroken) File System acquisition captures and provides access to the following data, all stored in one .zip file:[1]

- **Apple File Conduit or Apple File Connection (AFC)**: The AFC is a service that runs on iOS devices and is used by iTunes to exchange files. The AFC is an option during a File System acquisition on the UFED Touch but is automatically captured during an Advanced Logical acquisition using Physical Analyzer.
- **Apple File Conduit or Apple File Connection 2 (AFC2)**: The AFC2 provides access to the entire file system and the lockdown service. This is commonly captured on jailbroken devices, but the lockdown service file is captured during all Advanced Logical/File System acquisitions.
- **Backup Services**: Backup Services are essentially a backup created by iTunes that Physical Analyzer normalized for your ease of analysis. The data contained here include databases, plists, and other backup file data that can be manually examined and carved for data not shown in the Analyzed Data section of Physical Analyzer.

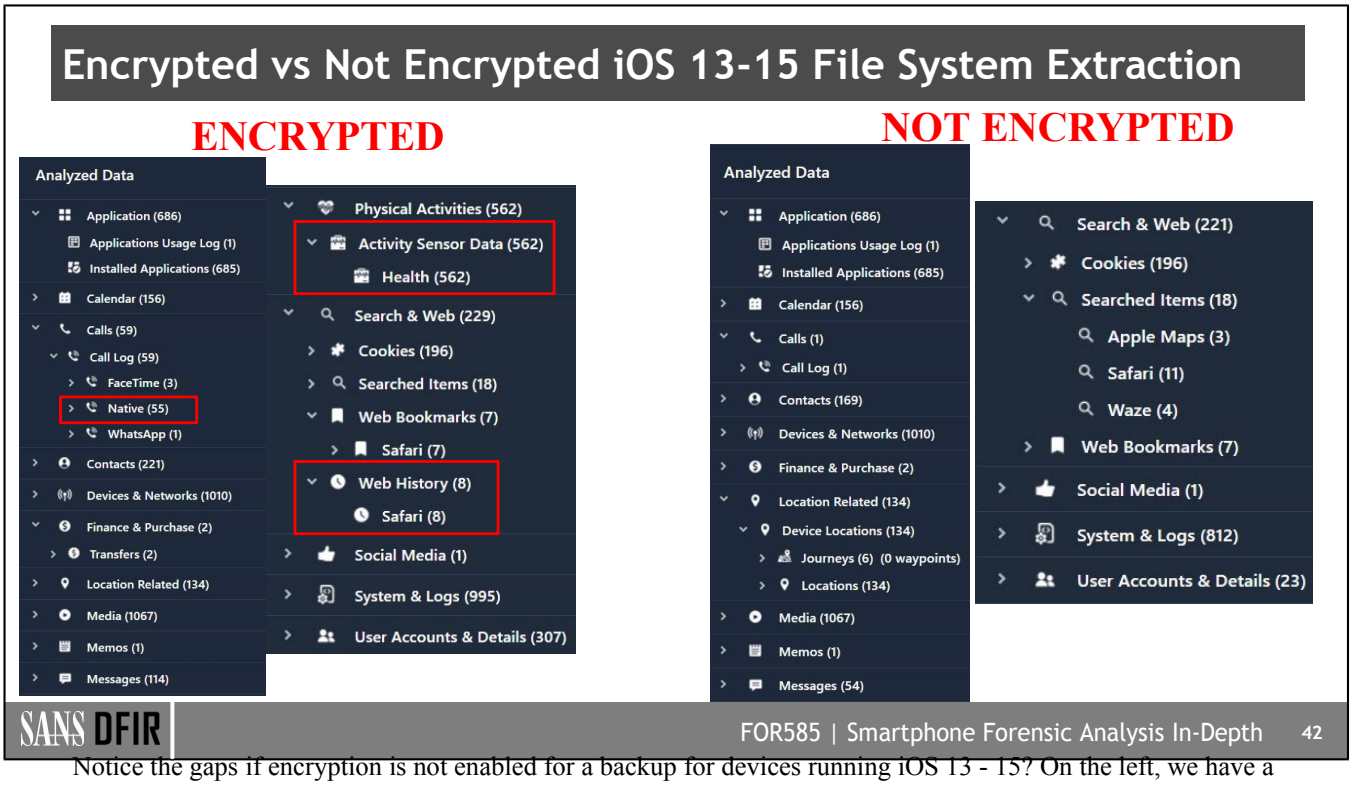
Essentially, a File System acquisition is a backup of all the data on the iOS device, including the raw files, but not the unallocated space.

As shown in the slide, a multitude of data can be captured using Logical or backup file acquisition. The Analyzed Data portion of Physical Analyzer is shown and reveals the data captured during a Logical acquisition of an iPhone 6s that was unlocked. In this example, email was captured. It all depends on the device! Email can be acquired and parsed on jailbroken devices during Logical and backup acquisitions. Again, deleted data may not be parsed, but it can be accessed if a backup file is created and the database files containing the raw data are accessible for carving.

Backup file forensics on iOS devices are covered in Section 4.

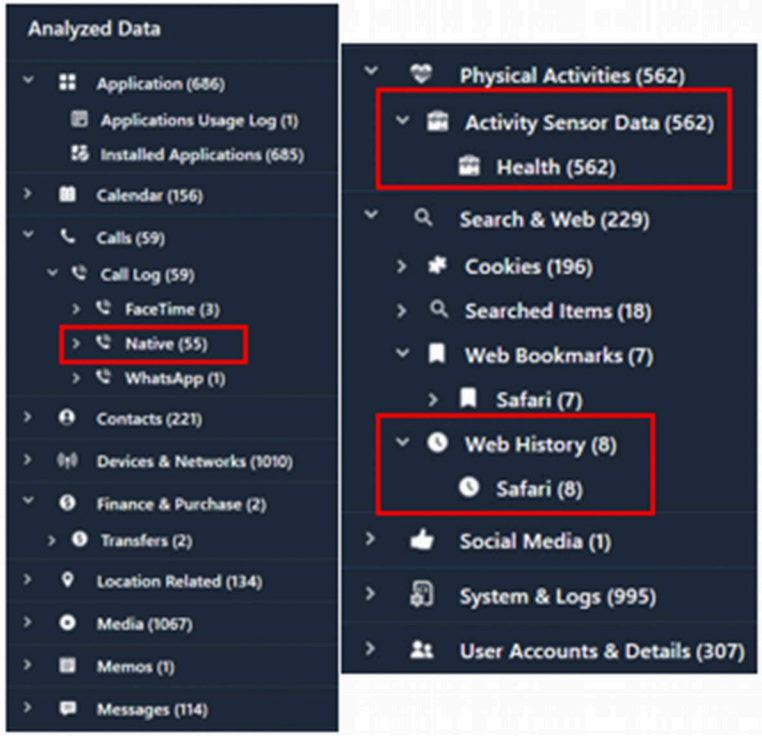
Reference:

[1] https://for585.com/iwiki

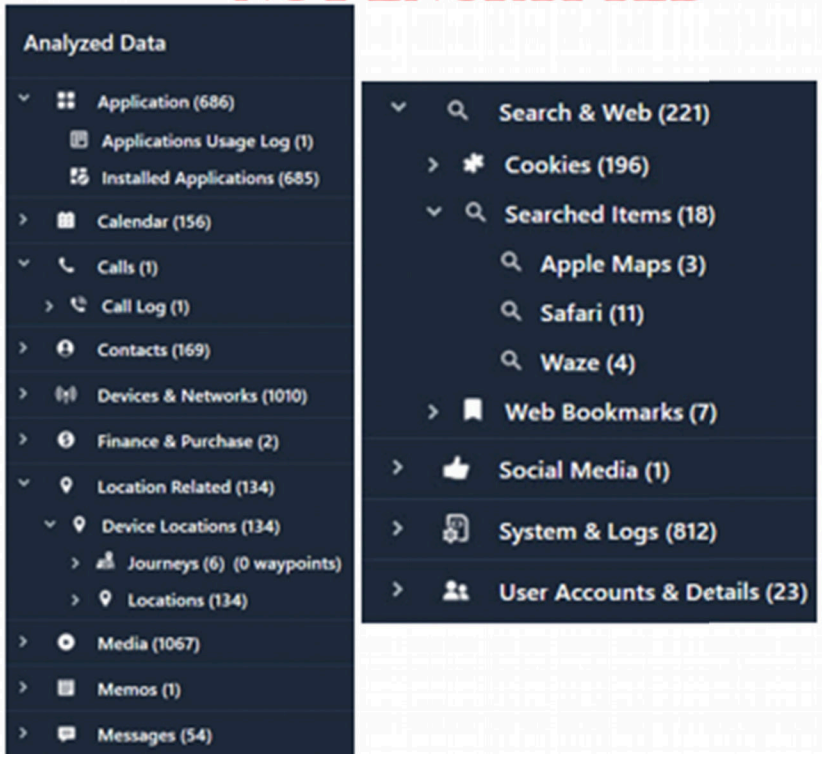


Notice the gaps if encryption is not enabled for a backup for devices running iOS 13 - 15? On the left, we have a full extraction using iTunes. This backup was encrypted, so we were able to extract Call logs, Safari, Maps, Health and Keychain. On the right, the same device was backed up without encryption and there are several gaps. Take note of the difference. For iOS 13 - 15, encryption is key!

ENCRYPTED



NOT ENCRYPTED



iOS Forensics Agenda

- Section 3.1: iOS Forensics Overview
- Section 3.2: iOS Device Acquisition Considerations
- Section 3.3: iOS File System Structures
- Section 3.4: iOS Evidentiary Locations
- Section 3.5: Handling Locked iOS Devices
- Section 3.6: Advanced Decoding and Traces of User Activity

This page intentionally left blank.

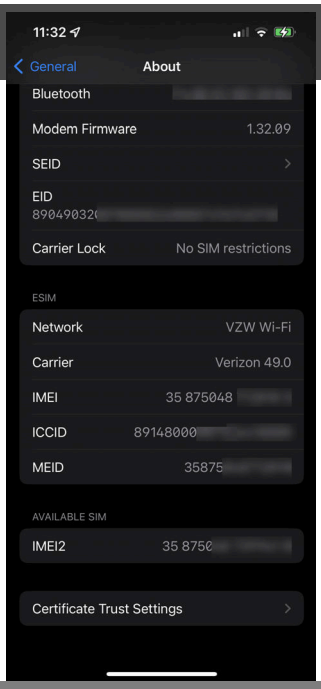
SIM Card Artifacts

Can alert us to various numbers the user had at different points in time

Dual SIM is present via eSIM on the latest devices

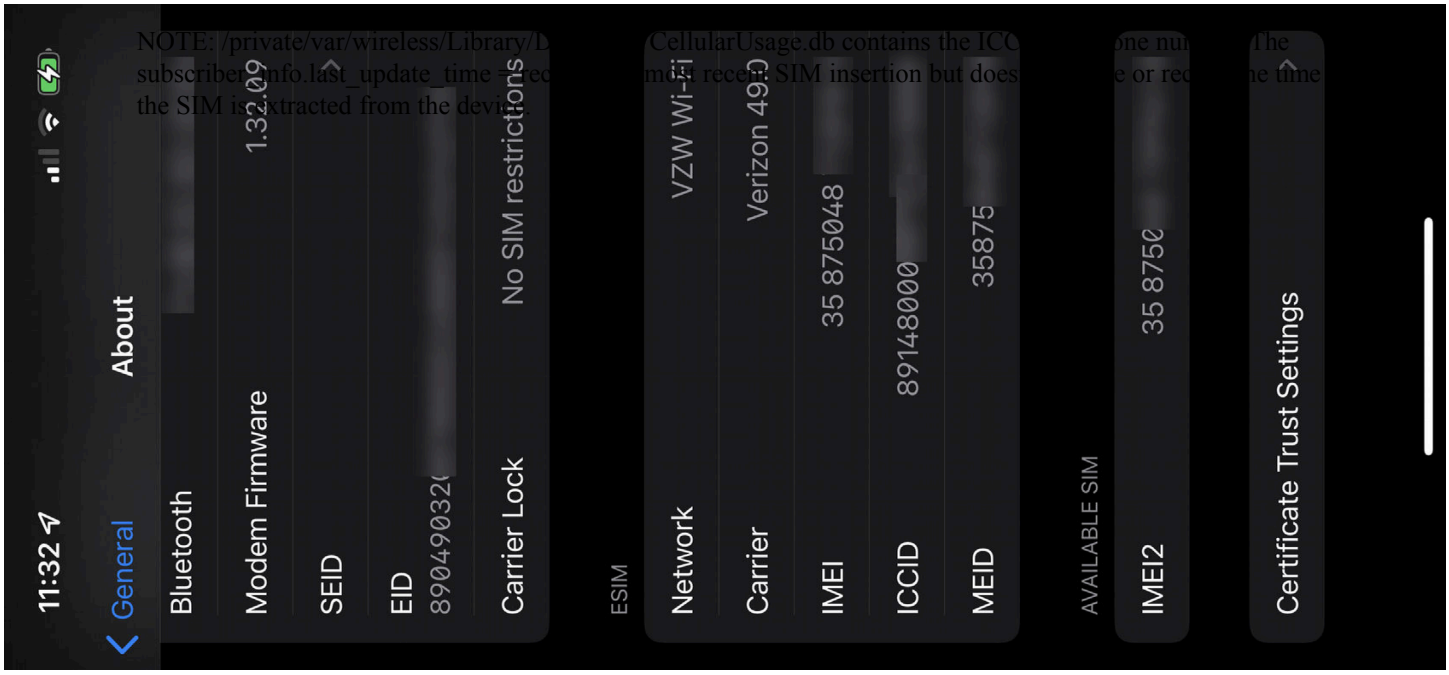
The following files store info:

- CellularUsage.db
- com.apple.commcenter.plist
- com.apple.commcenter.data.plist



As discussed in Section 1, SIM cards can provide us with information that may aid our investigations, such as phone numbers used and location information associated with the SIM card. In the past, we didn't have to worry about more than one SIM in an iPhone. This is no longer the case. Dual SIM, aka eSIM, was introduced with the iPhone XS. The user can now leverage more than one SIM and have more than one phone number similar to what we saw in Android. The files that track this information are /Library/Databases/CellularUsage.db and /wireless/Library/Preferences/com.apple.commcenter.plist. The file /wireless/Library/Preferences/com.apple.commcenter.data.plist also stores SIM card information.

NOTE: /private/var/wireless/Library/D… CellularUsage.db contains the ICC… one nu… The subscriber_info.last_update_time … rec… m… receive SIM insertion but does… e or rec… e time the SIM is extracted from the devi…



Call Logs, Contacts, and Messages

Call logs (including FaceTime)

- /Library/CallHistoryDB/callhistory.storedata (introduced in iOS 8)
  - zcallrecord
- /Library/Preferences/com.apple.cmfsyncagent.plist (Blocked Dialers)

Contacts

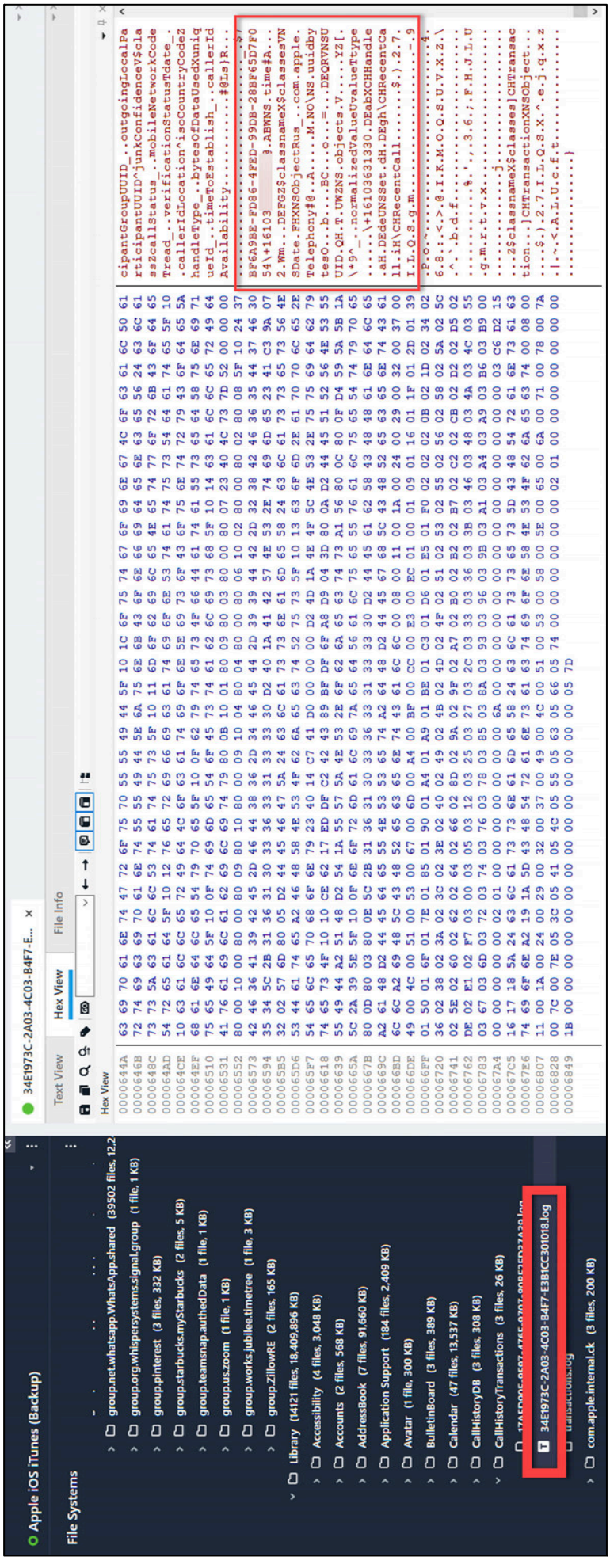
- /Library/AddressBook/AddressBook.sqlitedb
  - ABPerson
- /Library/AddressBook/AddressBookImages.sqlitedb

Communication data can be recovered from the /Library directory and is normally parsed by the mobile device forensic tools. Again, data marked for deletion may be parsed by some tools, but sometimes the data is overlooked and missed completely by the tool. The examiner must examine each database in question and then examine the data in Hex to ensure that nothing was overlooked. Free scripts are available to parse deleted data from the free pages of SQLite. Should more data be recovered, it should be carved and reported accordingly. If data is missing from the SQLite database, one may have luck recovering deleted calls from: /Library/CallHistoryTransactions/<GUID>.log

Important files for Call History

- /private/var/mobile/Library/CallHistoryDB/CallHistory.storedata
- /private/var/mobile/Library/CallHistoryTransactions/<GUID>.log

An example of the <GUID>.log is shown here.



SMS and iMessages

Stored in the same database

- /Library/SMS/sms.db

/Library/SMS/Attachments/*

/Library/SMS/Drafts/*

/Library/Preferences/com.apple.MobileSMS.plist

iMessage was introduced to iOS devices in iOS 5. Both SMS and iMessages are stored in the same SMS.db file. SMS messages on iOS devices are stored as standard SMS, MMS, or iMessages. While the database has remained the same, iOS 11 introduced changes to the timestamp format and added columns to the sms.db that some tools may not support. More on this will be covered in a few slides.

Prior to iOS 5 and the introduction of the iMessage, all dates and times for messages were stored in UNIX Epoch formats. iOS 5 and the iMessage introduced Mac Epoch timestamp formats. The names associated with the message and most iOS data are stored in plain ASCII, as are the phone numbers and message content. The database fields containing the phone number or email address for the SMS or iMessage may vary and are defined below.

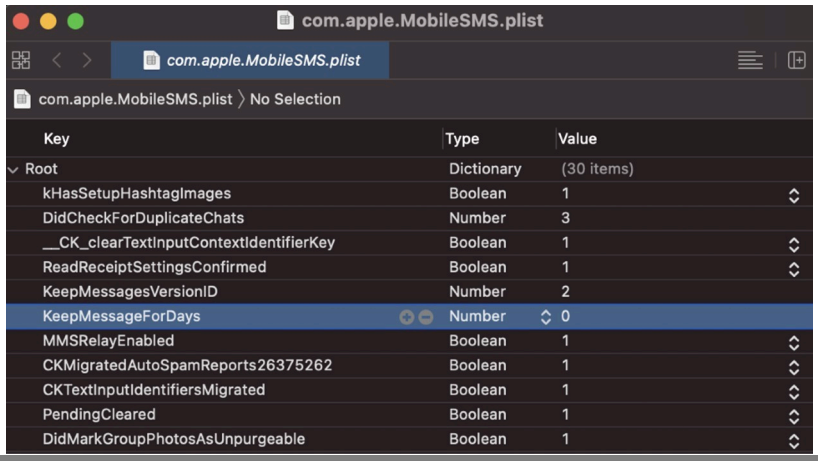
- Address or Chat_identifier: SMS Message
- Recipient: MMS Message
- Madrid_handle: iMessage (introduced with iOS 5)

Files of Interest for SMS

/Library/SMS/sms.db
/Library/SMS/Attachments/*
/Library/SMS/Drafts/*
/Library/Preferences/com.apple.MobileSMS.plist

com.apple.MobileSMS.plist

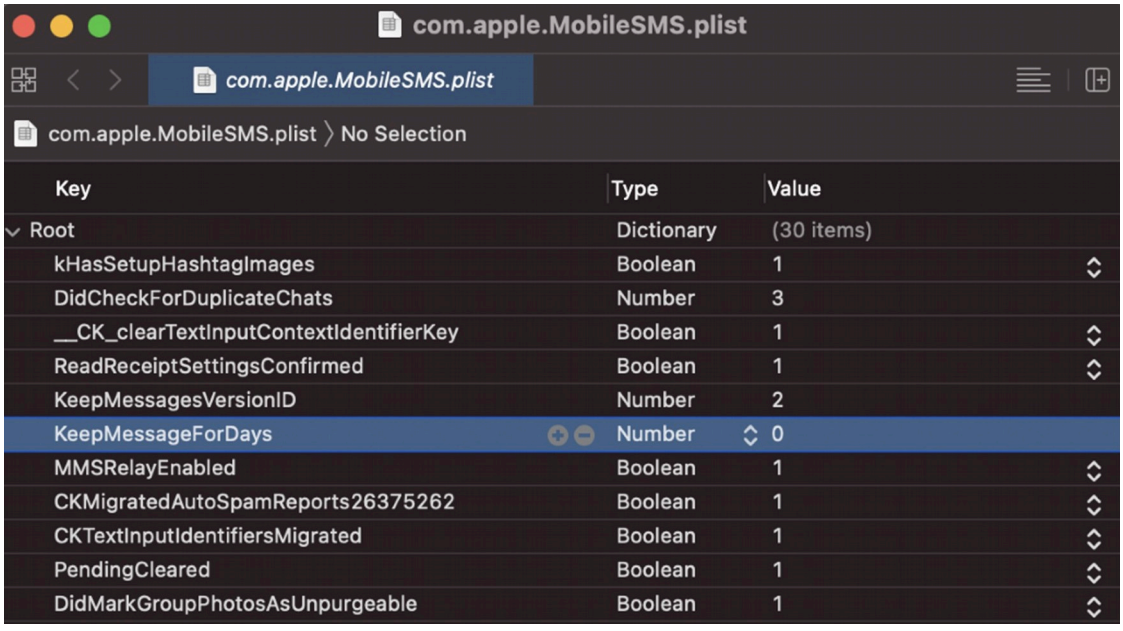
- KeepMessageForDays 0 = Forever, 365 = 1 year, 30 = 30 days – iOS 14+
- KeepMessageForDays NULL Value = Forever, 365 = 1 year, 30 = 30 days - iOS13 and below



/Library/Preferences/com.apple.MobileSMS.plist is shown in this slide. If you are missing message from the device, this is a great place to start.

KeepMessageForDays 0 = Forever, 365 = 1 year, 30 = 30 days – iOS 14+

KeepMessageForDays NULL Value = Forever, 365 = 1 year, 30 = 30 days - iOS13 and below



iOS 11+ SMS

iOS 11 introduced new columns to the sms.db

Apple started using 18-digit Mac datetime stamps for iMessage *if* the conversation *started* in iOS 11

The sms.db **will** contain both 9-digit and 18-digit Mac datetime stamps

Most commercial tools parse it correctly but verify!

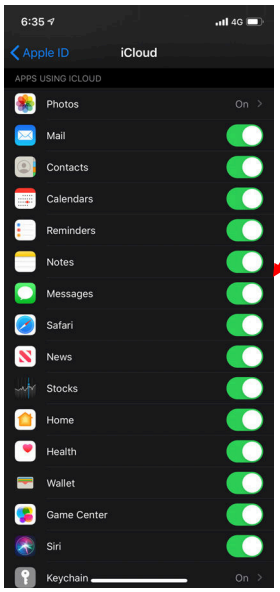
```
SELECT
message.rowid,
chat_message_join.chat_id,
message.handle_id,
message.text,
message.service,
message.account,
chat.account_login,
chat.chat_identifier AS "Other Party",
datetime(message.date/1000000000 + 978307200,'unixepoch','localtime') AS "conv start date",
case when LENGTH(chat_message_join.message_date)=18 then
datetime(chat_message_join.message_date/1000000000 + 978307200,'unixepoch','localtime')
when LENGTH(chat_message_join.message_date)=9 then
datetime(chat_message_join.message_date + 978307200,'unixepoch','localtime')
else 'N/A'
END AS "conversation start date",
datetime(message.date_read + 978307200,'unixepoch','localtime') AS "date read",
message.is_read AS "1=Incoming, 0=Outgoing",
case when LENGTH(chat.last_read_message_timestamp)=18 then
datetime(chat.last_read_message_timestamp/1000000000+978307200,'unixepoch','localtime')
when LENGTH(chat.last_read_message_timestamp)=9 then
datetime(chat.last_read_message_timestamp + 978307200,'unixepoch','localtime')
else 'N/A'
END AS "last date read",
attachment.filename,
attachment.created_date,
attachment.mime_type,
attachment.total_bytes
FROM
message
left join chat_message_join on chat_message_join.message_id=message.ROWID
left join chat on chat.ROWID=chat_message_join.chat_id
left join attachment on attachment.ROWID=chat_message_join.chat_id
order by message.date_read desc
```


Heather Mahalik wrote a detailed blog post on iOS 11 SMS and iMessage artifacts. The blog post "Time Is NOT on Our Side When It Comes to iOS 11 Messages" can be found at www.smarterforensics.com/blog. After discovering that the tools were not correctly parsing messages and/or not correctly decoding the 18-digit Mac datetime stamps, she worked with co-author Lee Crognale (Domenica) to create an SQLite query that will parse this information for you. The query shown above and the script that will parse the sms.db are available on Heather's GitHub and are included in your VM.[1]

Reference:

[1] https://github.com/hmahalik

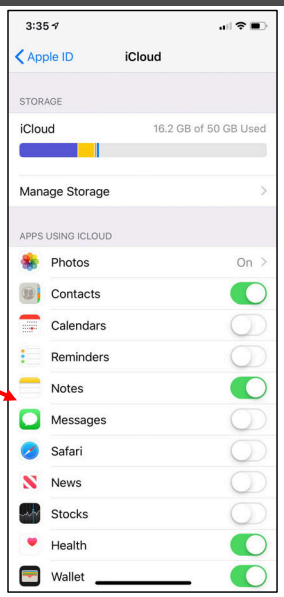
Missing Deleted SMS Messages



- The users settings will control if you can recover deleted SMS
- Once set – there is no going back

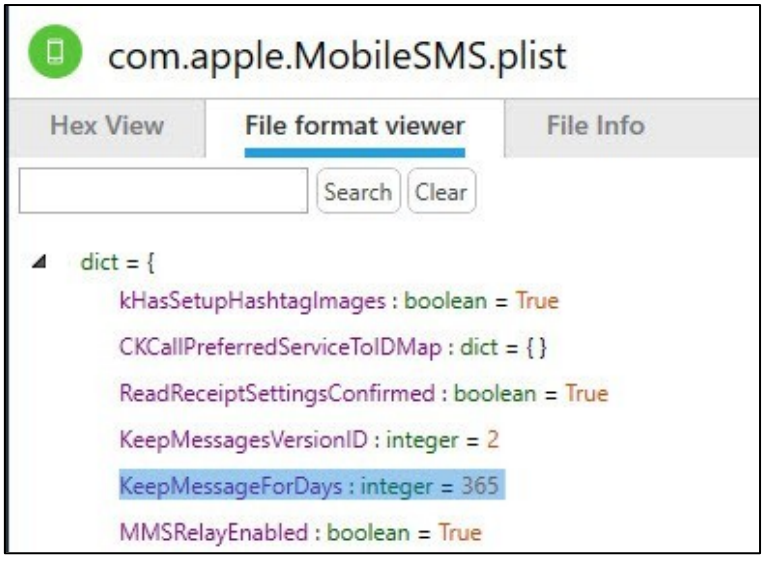
- If the iCloud switch is green:
  - ► SMS messages will be removed from the device after deletion
  - ►iMessages should persist

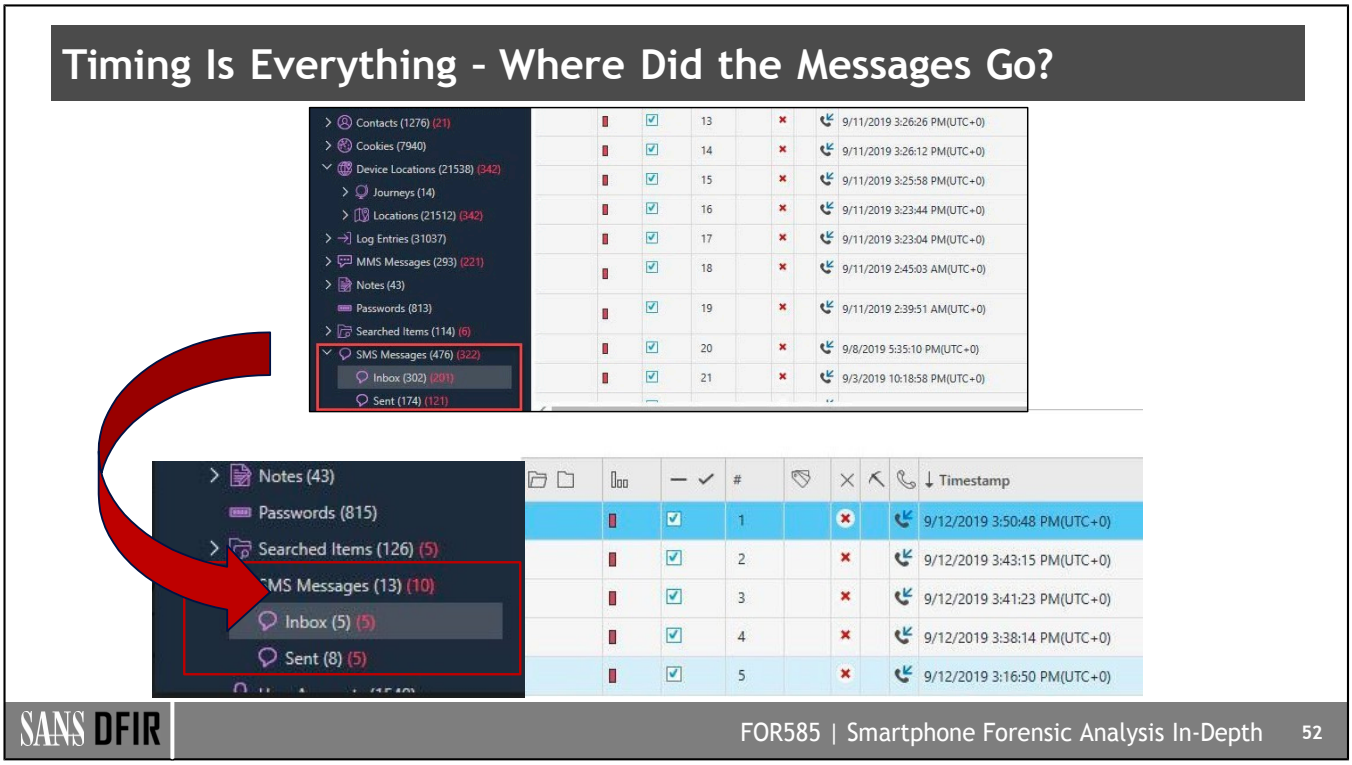
- If the iCloud switch is off:
  - ► SMS messages should be recoverable after deletion
  - ►iMessage should also persist



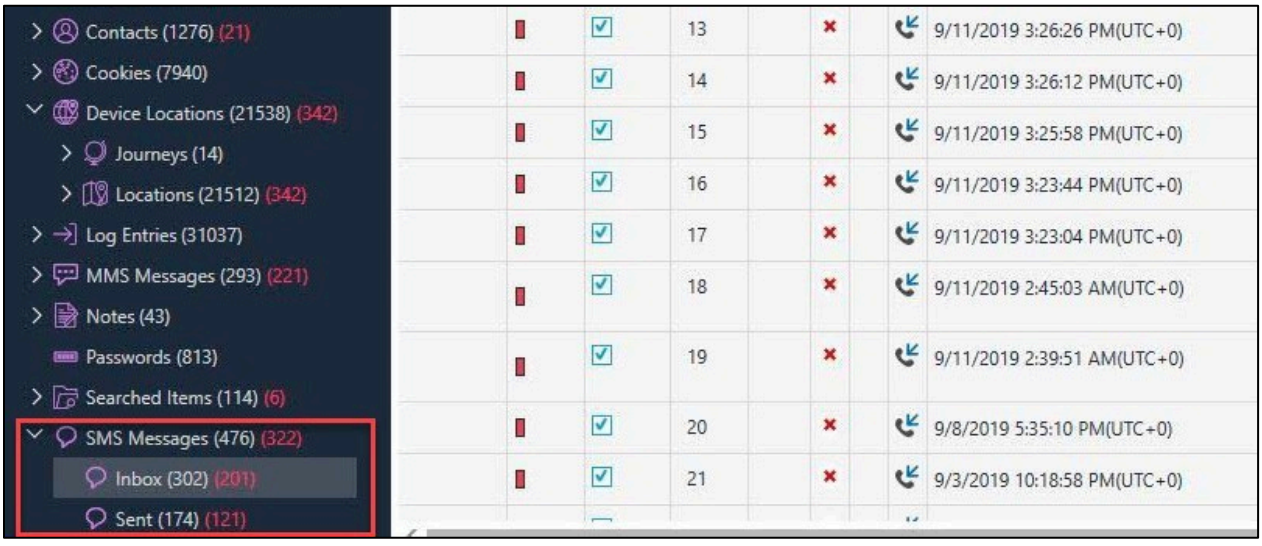
If you have ever examined an iPhone and wondered what accounts for the missing messages in the free pages, this slide may bring you some ease. The settings are controlling what we can recover. If the user enables messages in iCloud sync, messages that are deleted will be purged from the free pages. This does not include iMessage, which should persist. How long do the messages persist? Well, as the most popular phrase in digital forensics is used here – it depends. On what, to be honest, I have no idea. I thought it would be purged immediately and it always has been from my testing, but the next slide will prove that theory to be incorrect. If the Messages is set to "off" in the iCloud settings, all deleted messages will remain in the free pages.

Something important to note here is that once the change is made, there is no going back. The default is for this setting to remain off. Once you turn the switch on, SMS messages will always be purged from the free pages even if you switch the toggle to off again. Something else to consider is how long the messages are set to be kept! The default is forever but can be changed by the user. This user changed her device to save messages for one year.

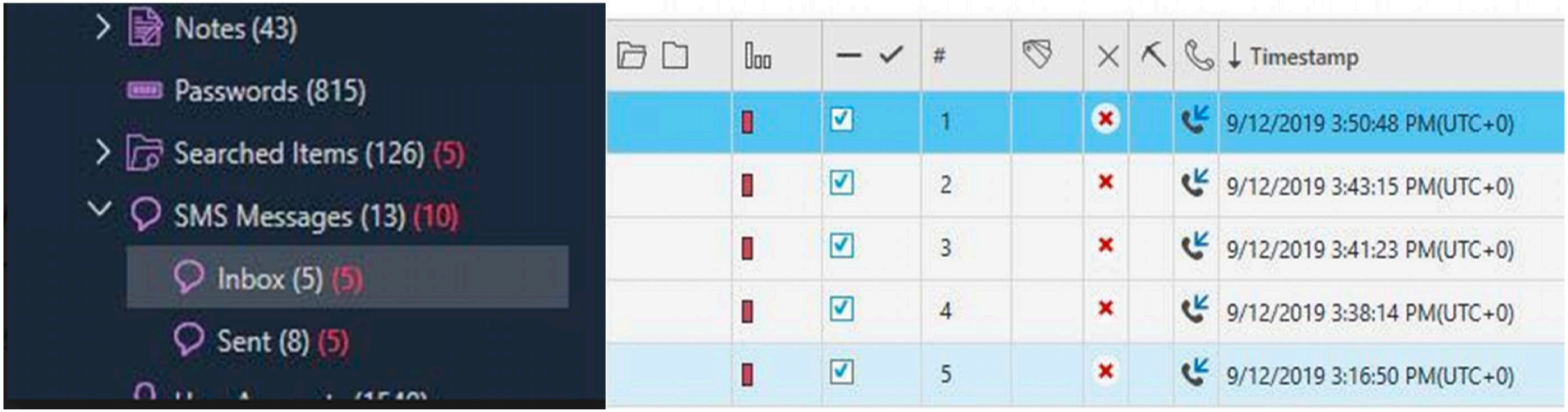


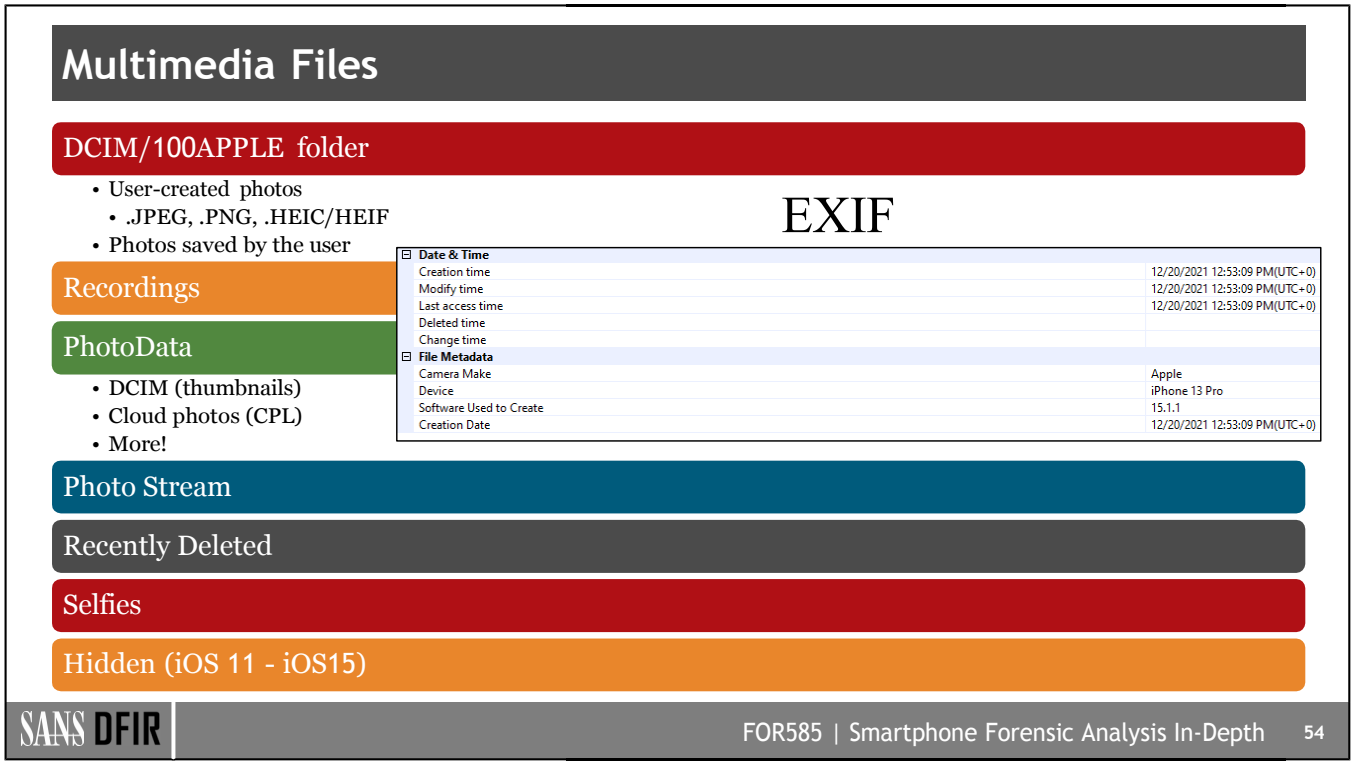


Once that button is turned on for Messages, there is no telling how much time you have to recover deleted SMS messages. On my tested device, the purge seemed to be as quick as me flipping the switch and dumping the device. However, a student in Vegas proved this to not be true. For this test, he enabled the Messages to be synced to iCloud, deleted several messages on Sept. 11, 2019, then dumped his device using Method 1 and Method 2 in Physical Analyzer, which is the old way of acquiring iOS devices. Currently, this would be done with an Advanced Logical extraction on UFED. Notice that all of the deleted SMS messages persisted. This blew my mind.



The next morning, I asked him to send a few more messages and delete them and dump his phone again. Guess what – all of the messages from Sept 11, 2019, were gone and only newly deleted messages were present! Clearly, there is a time frame in which the purging occurs. Could it be poor Wi-Fi? Poor cell service? Storage space? Sure, in smartphone forensics anything is possible. If this isn't reason enough for you to dump the phone as quickly as possible what is?


Notes (43)
Passwords (815)
Searched Items (126) (5)
SMS Messages (13) (10)
Inbox (5) (5)
Sent (8) (5)
#
Timestamp
1
9/12/2019 3:50:48 PM(UTC+0)
2
9/12/2019 3:43:15 PM(UTC+0)
3
9/12/2019 3:41:23 PM(UTC+0)
4
9/12/2019 3:38:14 PM(UTC+0)
5
9/12/2019 3:16:50 PM(UTC+0)



Digital cameras and smartphones store user-created photos and videos in the DCIM folder by default; this includes photos that were saved by the user from email, MMS, etc. If the user interacts with the photo, it is commonly saved here. The exception may be third-party applications where the images are stored local to the application directory. DCIM stands for Digital Camera Images. Examining this folder provides access to all the photos the user took with the smartphone. The acquisition method determines if deleted photos are placed into the Media folder for examination.

Up until iOS 11, the format was consistently JPEG for images captured with the iOS device. iOS 11 introduced HEIC, which leverages HEIF (High Efficiency Image Format) and uses advanced compression to retain great image quality.[1]

The PhotoData folder normally contains additional metadata and thumbnails for each photo. iOS 7 and iOS 8 introduced new features for photos to include Photo Stream, Moments, and Recently Deleted. Photo Stream allows the user to sync photos with multiple devices. The metadata for each file remains intact, so forensic examination allows the examiner to match the photo with the device for which it was recorded/taken. Moments allow the user to look at photos based on dates and locations and even allow searching on the device. Moments will be covered in a lab in section 4.

Recently Deleted photos was introduced with iOS 8. This was developed to allow a user to change their mind and recover photos that were recently deleted. The photos remain in this directory for ~30 days. After the ~30-day period, they are no longer recoverable to the user. This does not mean that file carving will not recover the data if the acquisition type supports it. Additionally, these deleted photos may exist in iCloud, Google cloud or elsewhere. Selfies, as previously discussed, were introduced with iOS 9. Finally, the ability to hide photos was introduced in iOS11. All of the features are available in iOS15.

Reference:

[1] https://for585.com/heic (What is HEIC?)

Multimedia Files EXIF Data and Deletion

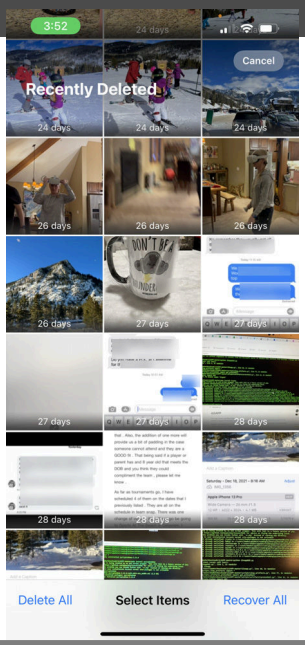
Name:	132E546C-142E-45BB-8726-66EF130C04D0.JPG
Type:	Images
Size (bytes):	2380548
Path:	Heather's iPhone/mobile/Media/PhotoData/CPLAssets/group59/132E546C-142E-45BB-8726-66EF130C04D0.JPG
Created:	9/25/2021 4:34:36 PM(UTC+0)
Accessed:	9/25/2021 4:34:55 PM(UTC+0)
Modified:	9/25/2021 4:34:55 PM(UTC+0)
Changed:	
Deleted:	
Extraction:	Legacy
MD5:	2e52cb2abd4c8f4e2d27af3e2fe83aec
Source file:	132E546C-142E-45BB-8726-66EF130C04D0.JPG

Metadata

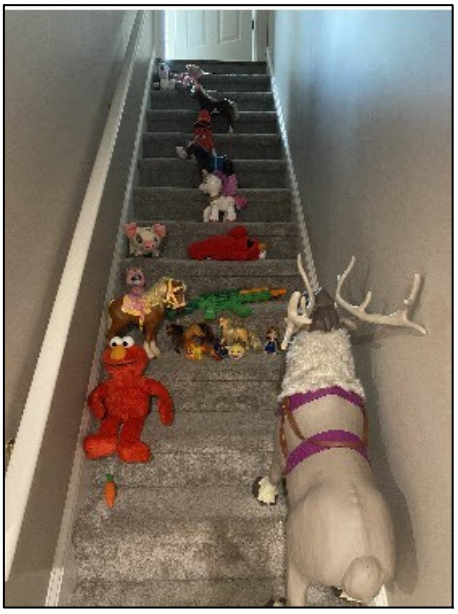
Camera Make:	Apple
Camera Model:	iPhone 11
Capture Time:	4/23/2020 11:07:22 AM
Pixel resolution:	4032x3024
Resolution:	72x72 (Unit: Inch)
Orientation:	Rotate 90 CW
Lat/Lon:	40. 34

Map

Position:	(40. 34)



Dates and times associated with Recently Deleted photos may not be as simple to recover. However, as shown in this screenshot, Apple tracks the amount of time the picture has been sitting in the “Recently Deleted” directory and preserves it for ~30 days! What is difficult is when a photo is placed into the deleted items and then recovered as it reverts the MAC back to the original timestamps. This is addressed in the next slide.



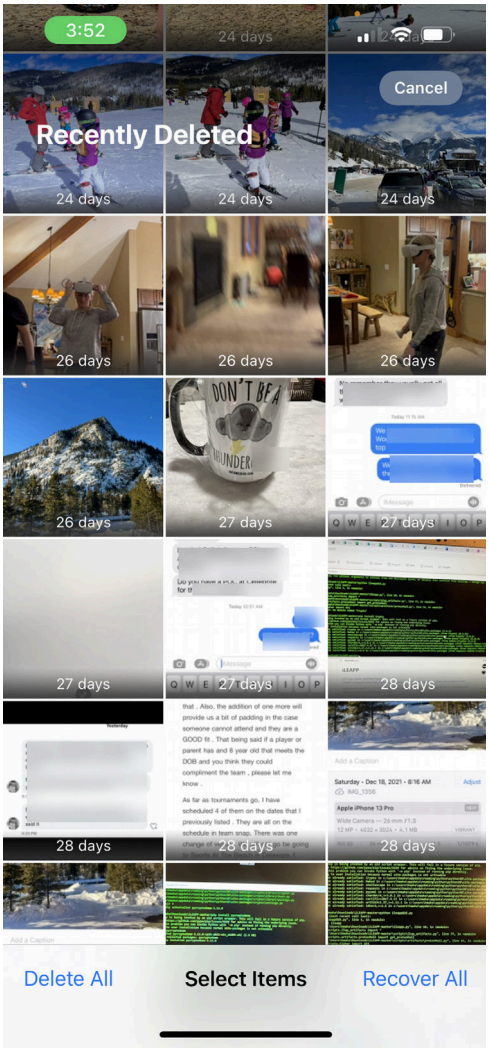
Name:	132E546C-142E-45BB-8726-66EF130C04D0.JPG
Type:	Images
Size (bytes):	2380548
Path:	Heather's iPhone/mobile/Media/PhotoData/CPLAssets/group59/132E546C-142E-45BB-8726-66EF130C04D0.JPG
Created:	9/25/2021 4:34:36 PM(UTC+0)
Accessed:	9/25/2021 4:34:55 PM(UTC+0)
Modified:	9/25/2021 4:34:55 PM(UTC+0)
Changed:	
Deleted:	
Extraction:	Legacy
MD5:	2e52cb2abd4c8f4e2d27af3e2fe83aec
Source file:	132E546C-142E-45BB-8726-66EF130C04D0.JPG

Metadata

Camera Make:	Apple
Camera Model:	iPhone 11
Capture Time:	4/23/2020 11:07:22 AM
Pixel resolution:	4032x3024
Resolution:	72x72 (Unit: Inch)
Orientation:	Rotate 90 CW
Lat/Lon:	40. 34

Map

Position:	(40. 34)



Photos.sqlite – Changed in iOS 14 & iOS 15

	Z_PK	File Name	Duration in Seconds	Is Deleted	Date Deleted	Date Added	Date Created	Date Modified	
7428	8349	IMG_1363.HEIC	0.0	Deleted	2022-01-26 12:17:00	2022-01-25 19:36:17	2022-01-25 19:36:16	2022-01-26 ...	DCIM/101APPLE
7429	8350	IMG_1364.HEIC	0.0	Deleted	2022-01-26 12:17:00	2022-01-25 20:05:56	2022-01-25 20:05:55	2022-01-26 ...	DCIM/101APPLE
7430	8351	IMG_1365.PNG	0.0	Deleted	2022-01-26 07:52:34	2022-01-26 07:50:13	2022-01-26 07:50:13	2022-01-26 ...	DCIM/101APPLE
7431	8352	IMG_1366.HEIC	0.0	Deleted	2022-01-26 15:13:25	2022-01-26 08:10:29	2022-01-26 08:10:27	2022-01-26 ...	DCIM/101APPLE
7432	8353	IMG_1367.PNG	0.0	Deleted	2022-01-26 09:14:41	2022-01-26 09:12:55	2022-01-26 09:12:55	2022-01-26 ...	DCIM/101APPLE
7433	8354	IMG_1368.PNG	0.0	Deleted	2022-01-26 12:17:00	2022-01-26 12:14:41	2022-01-26 12:14:41	2022-01-26 ...	DCIM/101APPLE
7434	8355	IMG_1369.PNG	0.0	Deleted	2022-01-26 12:17:00	2022-01-26 12:15:44	2022-01-26 12:15:44	2022-01-26 ...	DCIM/101APPLE
7435	8356	IMG_1370.PNG	0.0	Deleted	2022-01-26 12:16:49	2022-01-26 12:16:20	2022-01-26 12:16:19	2022-01-26 ...	DCIM/101APPLE
7436	8357	IMG_1371.PNG	0.0	Deleted	2022-01-26 15:13:25	2022-01-26 12:17:00	2022-01-26 12:17:00	2022-01-26 ...	DCIM/101APPLE
7437	8358	IMG_1372.HEIC	0.0	Deleted	2022-01-26 15:13:25	2022-01-26 13:39:23	2022-01-26 13:39:22	2022-01-26 ...	DCIM/101APPLE
7438	8359	IMG_1373.HEIC	0.0	N/A	NULL	2022-01-26 18:07:42	2022-01-26 18:07:41	2022-01-26 ...	DCIM/101APPLE
7439	8360	IMG_1374.JPG	0.0	N/A	NULL	2022-01-26 19:04:52	2022-01-26 19:04:52	2022-01-26 ...	DCIM/101APPLE
7440	8361	IMG_1375.JPG	0.0	N/A	NULL	2022-01-26 19:04:52	2022-01-26 19:04:52	2022-01-26 ...	DCIM/101APPLE
7441	8362	IMG_1376.JPG	0.0	N/A	NULL	2022-01-26 19:04:52	2022-01-26 19:04:52	2022-01-26 ...	DCIM/101APPLE
7442	8363	IMG_1377.HEIC	0.0	Deleted	2022-01-26 19:26:25	2022-01-26 19:10:07	2022-01-26 19:10:06	2022-01-26 ...	DCIM/101APPLE

The Photos.sqlite database tracks photos, videos, screenshots, and so on that were taken with the iOS device and the interactions with each multimedia file. For example, if the user uploads the data to cloud, this is stored here. If the user backs up her data to cloud, including multimedia, then the device must retrieve it from cloud to view the files on her device. These cloud-based multimedia files are also tracked in this database. Most importantly, multimedia files that are deleted are tracked in this database. Some of the most valuable information you will obtain pertaining to photos and videos will be found in Photos.sqlite. There are many queries for you in www.for585.com/notebook. Don't use them this week until the Section 6 challenge. Take the time to learn the hard way during the labs and try it manually!

Let's consider image files for this example. A photo is taken with the iPhone. The creation data is stored in the Photos.sqlite database. The user deletes this image. When this occurs, the date of deletion is tracked in the ZTRASHEDDATE, as highlighted in the slide. The state is changed from 0 (not deleted) to 1 (deleted). This is how Apple counts down the time you have remaining to recover the deleted photo.

What doesn't make sense is what happens when that image is recovered by the user. If the user recovers the image from the deleted images directory, the ZTRASHEDDATE is cleared and the ZTRASHEDSTATE reverts back to 0. Thus, it is very difficult to state that the photo was deleted and recovered. Clear as mud, right? This is why you need to test and validate what you are reporting because the artifacts may appear one way and not be telling the full story.

The query below will make examining this database easier. It is also available in your course notebook.

```
Select
z_pk,
zfilename AS "File Name",
zduration AS "Duration in Seconds",
case
when ztrashedstate = 1 then "Deleted"
else "N/A"
end AS "Is Deleted",
datetime(ztrasheddate+978307200,'unixepoch','localtime') AS "Date Deleted",
datetime(zaddeddate+978307200,'unixepoch','localtime') AS "Date Added",
datetime(zdatecreated+978307200,'unixepoch','localtime') AS "Date Created",
datetime(zmodificationdate+978307200,'unixepoch','localtime') AS "Date Modified",
zdirectory AS "File Path"
from zgenericasset
```

The table of interest in iOS 14 changed from **zgenericasset** to **zasset**. This will be a part of Lab 4.2 in section 4.

	Z_PK	File Name	Duration in Seconds	Is Deleted	Date Deleted	Date Added	Date Created	Date Modified	
7428	8349	IMG_1363.HEIC	0.0	Deleted	2022-01-26 12:17:00	2022-01-25 19:36:17	2022-01-25 19:36:16	2022-01-26 ...	DCIM/101APPLE
7429	8350	IMG_1364.HEIC	0.0	Deleted	2022-01-26 12:17:00	2022-01-25 20:05:56	2022-01-25 20:05:55	2022-01-26 ...	DCIM/101APPLE
7430	8351	IMG_1365.PNG	0.0	Deleted	2022-01-26 07:52:34	2022-01-26 07:50:13	2022-01-26 07:50:13	2022-01-26 ...	DCIM/101APPLE
7431	8352	IMG_1366.HEIC	0.0	Deleted	2022-01-26 15:13:25	2022-01-26 08:10:29	2022-01-26 08:10:27	2022-01-26 ...	DCIM/101APPLE
7432	8353	IMG_1367.PNG	0.0	Deleted	2022-01-26 09:14:41	2022-01-26 09:12:55	2022-01-26 09:12:55	2022-01-26 ...	DCIM/101APPLE
7433	8354	IMG_1368.PNG	0.0	Deleted	2022-01-26 12:17:00	2022-01-26 12:14:41	2022-01-26 12:14:41	2022-01-26 ...	DCIM/101APPLE
7434	8355	IMG_1369.PNG	0.0	Deleted	2022-01-26 12:17:00	2022-01-26 12:15:44	2022-01-26 12:15:44	2022-01-26 ...	DCIM/101APPLE
7435	8356	IMG_1370.PNG	0.0	Deleted	2022-01-26 12:16:49	2022-01-26 12:16:20	2022-01-26 12:16:19	2022-01-26 ...	DCIM/101APPLE
7436	8357	IMG_1371.PNG	0.0	Deleted	2022-01-26 15:13:25	2022-01-26 12:17:00	2022-01-26 12:17:00	2022-01-26 ...	DCIM/101APPLE
7437	8358	IMG_1372.HEIC	0.0	Deleted	2022-01-26 15:13:25	2022-01-26 13:39:23	2022-01-26 13:39:22	2022-01-26 ...	DCIM/101APPLE
7438	8359	IMG_1373.HEIC	0.0	N/A	*NULL*	2022-01-26 18:07:42	2022-01-26 18:07:41	2022-01-26 ...	DCIM/101APPLE
7439	8360	IMG_1374.JPG	0.0	N/A	*NULL*	2022-01-26 19:04:52	2022-01-26 19:04:52	2022-01-26 ...	DCIM/101APPLE
7440	8361	IMG_1375.JPG	0.0	N/A	*NULL*	2022-01-26 19:04:52	2022-01-26 19:04:52	2022-01-26 ...	DCIM/101APPLE
7441	8362	IMG_1376.JPG	0.0	N/A	*NULL*	2022-01-26 19:04:52	2022-01-26 19:04:52	2022-01-26 ...	DCIM/101APPLE
7442	8363	IMG_1377.HEIC	0.0	Deleted	2022-01-26 19:26:25	2022-01-26 19:10:07	2022-01-26 19:10:06	2022-01-26 ...	DCIM/101APPLE

iOS 15 - Photo Adjustments

iOS 15 introduced the ability to adjust the following:

- Capture Time
- Location

The original EXIF data is reflected on the device that took the photo

The user see's what they "adjusted" it to be in the gallery

The receiving device will show the "adjusted" time and location

- Test results coming in next slides
- Looking at the adjusted image in Hex will reveal the original creation date

iCloud will reflect the "adjusted" time and location

Photos.sqlite is impacted for certain values

The "adjusted" time's seconds are zero'd out (06:13:00)

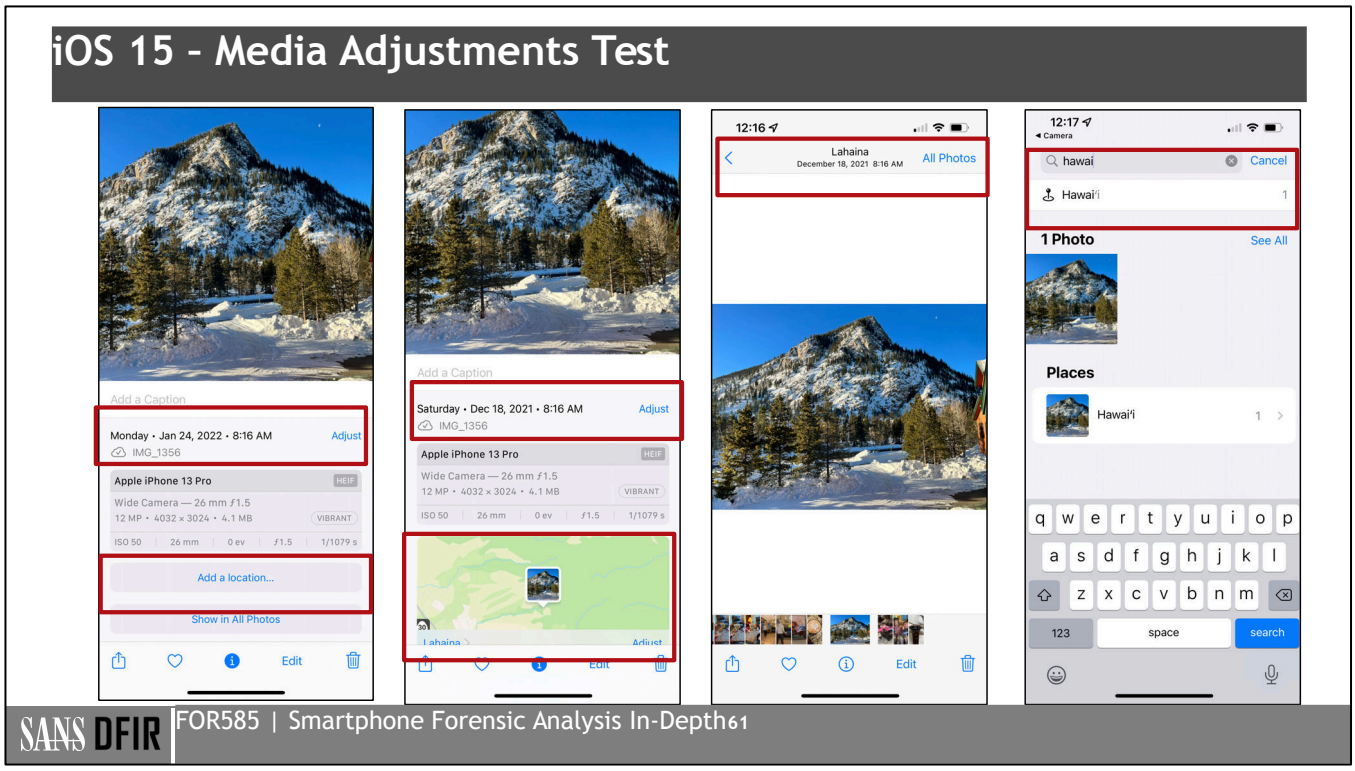
Photo adjustments were introduced in iOS 15. This feature enables the user to change or modify the capture time and the location for which the pictures was taken. When a user adjusts a photo, the original EXIF data persists, however the Photos.sqlite is updated for some values to show the original timestamp and the adjusted. As we will see in the upcoming slides, the timestamp that the photo was adjusted to will have the seconds zeroed out.

In regard to the iOS device gallery, the photo will show the location and timestamp for which it was adjusted. The original timestamp and location will not be visible to the user. If the photo syncs to iCloud, the adjusted timestamp and location will be used, and the image will be stored with that metadata.

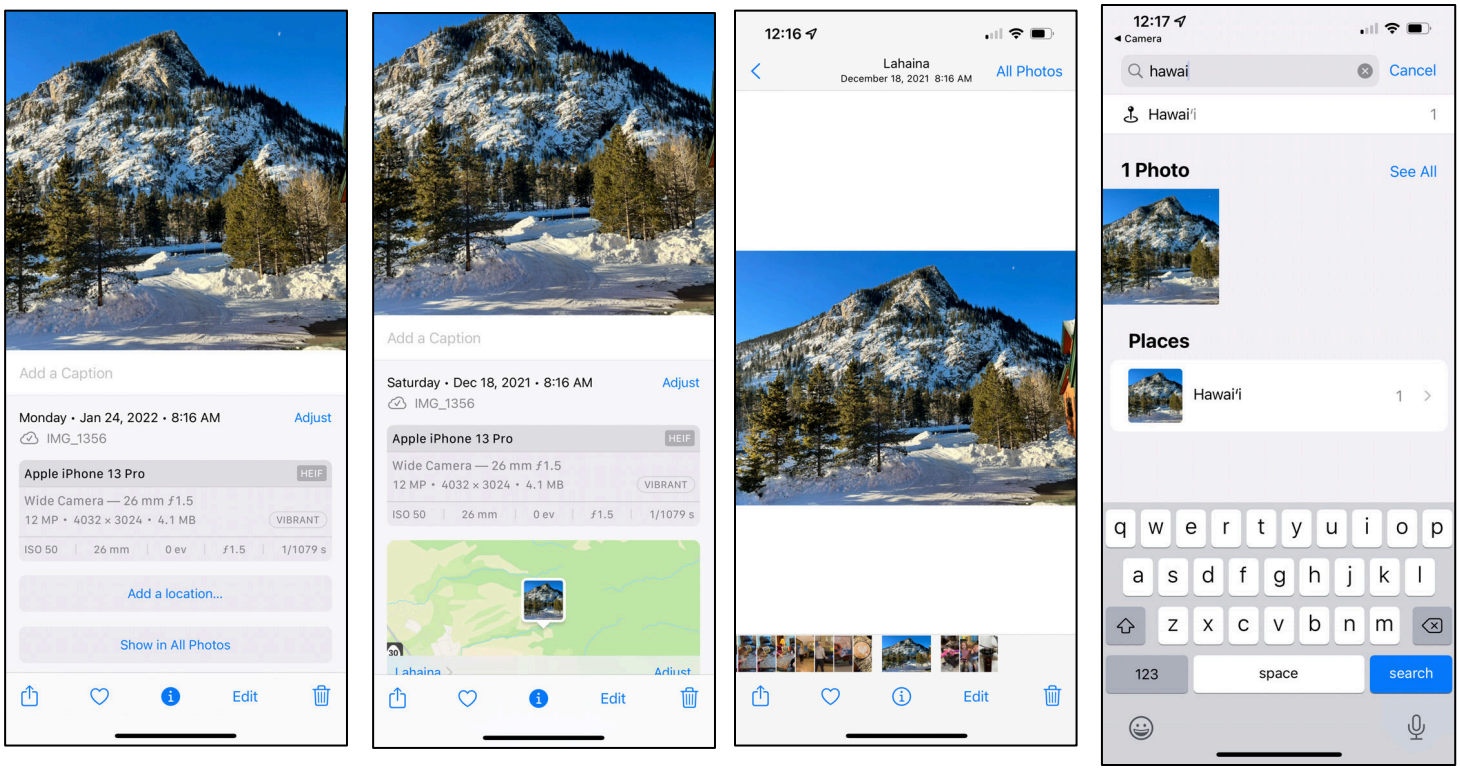
The new few slides will walk you through the logic behind Ian Whiffin's blog, which were validated by Heather Mahalik.[1]

Reference:

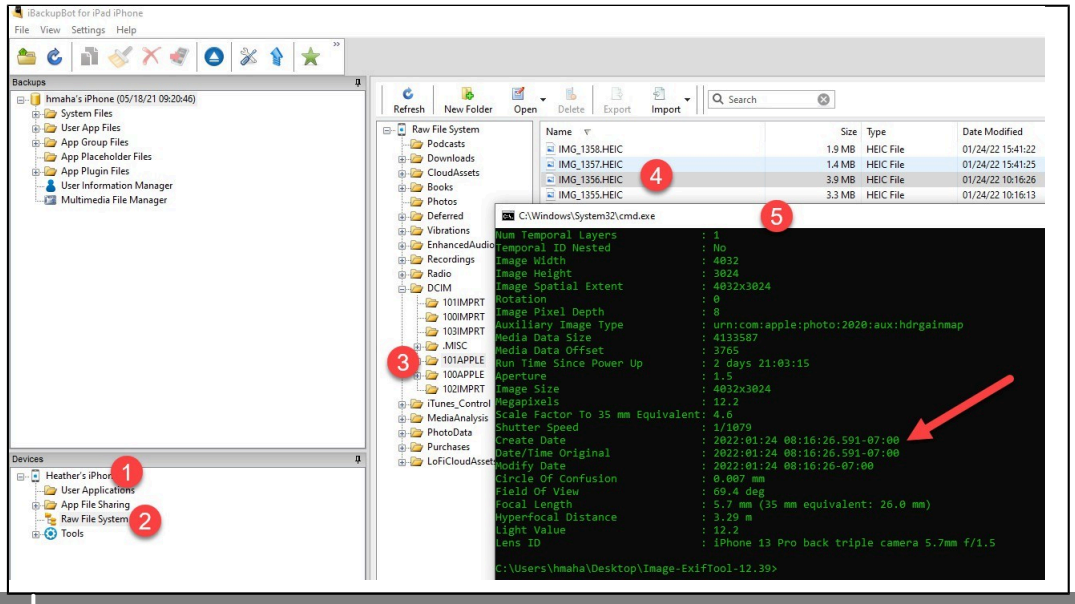
[1] https://for585.com/adjusted (Ian Whiffin's blog on Photo Adjustments)



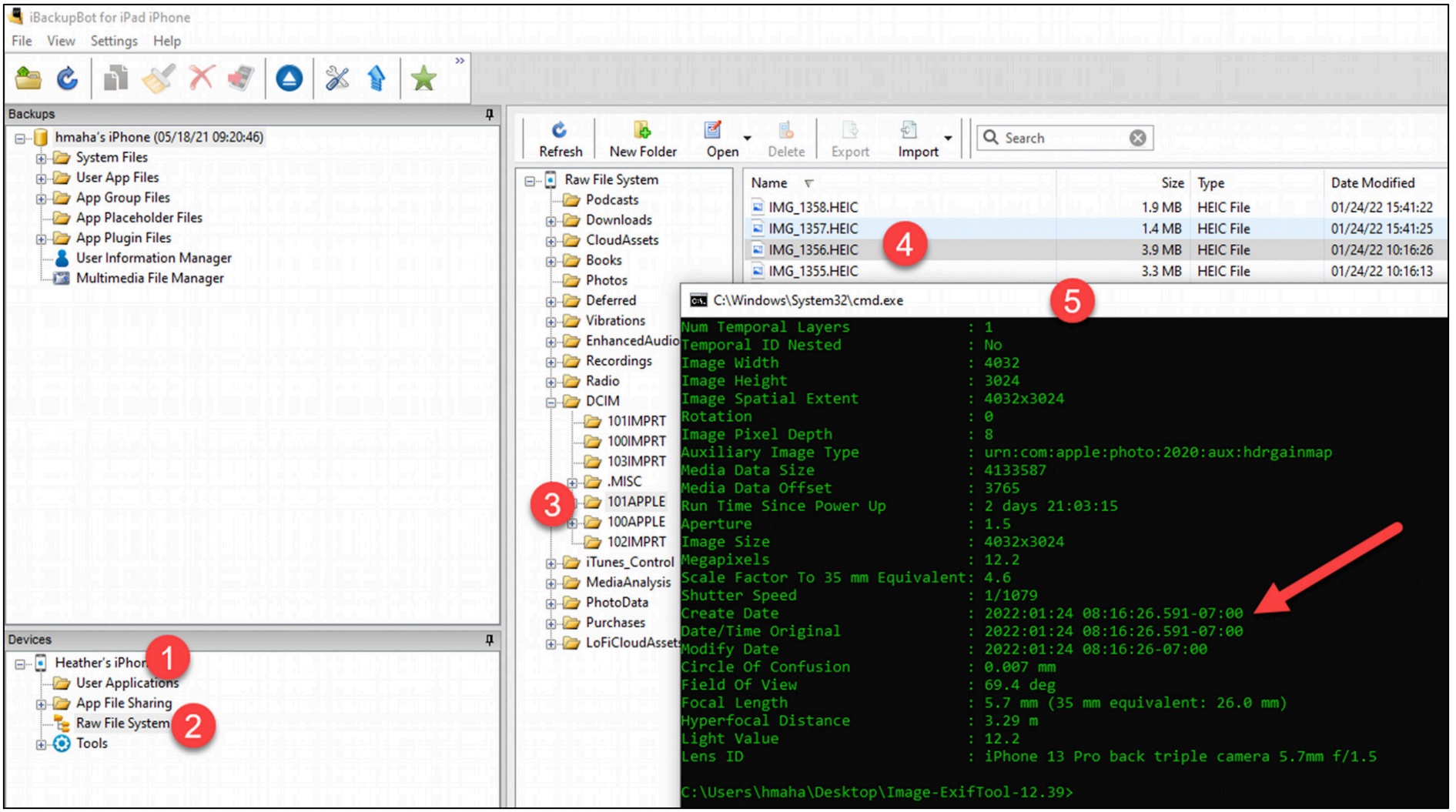
This is a walkthrough of the logic behind photo adjustments on iOS 15. In the first picture, a photo was taken in Frisco, Colorado on January 24, 2022, at 8:16 AM. The location was NOT being tracked by the iPhone at the time this photo was taken. In the second photo, the image was adjusted to the capture time of December 18, 2021, at 8:16 AM and the location was changed to Maui Hawai'i. When the user looks at the photo in their gallery, they will only see the second image. It's as if the first never occurred and Apple honors the adjustments. The third photo shows how the photo looks for the user on their iOS device. The final image shows how the location is honored by Apple and when the user searches for "Hawai" that image of Colorado appears as if it were taken in Hawai'i.



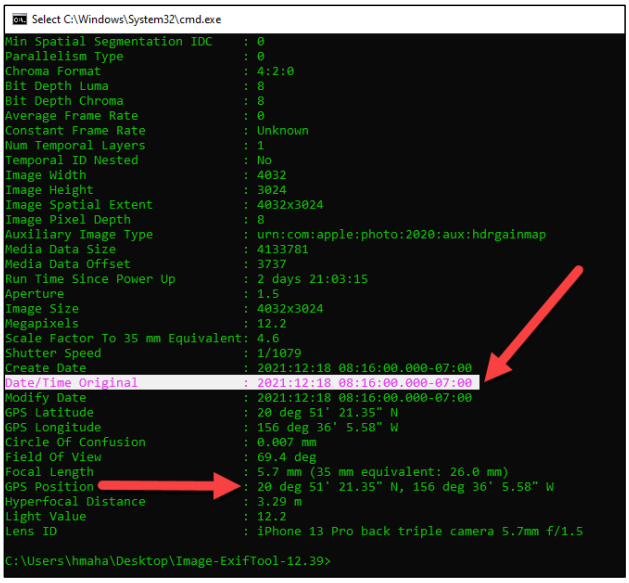
iOS 15 - Photo Adjustments - Native Device Artifacts

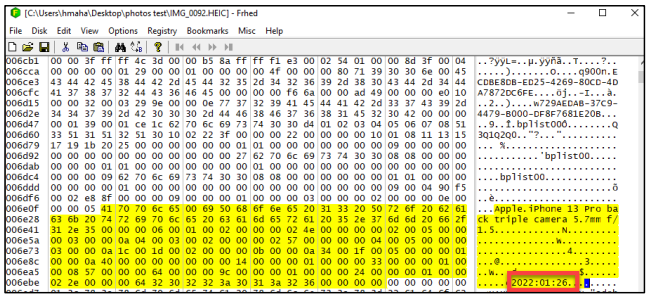


In the slide above we loaded the original device that took and adjusted the photo shown in the previous slide. The goal is to export the image and examine the EXIF data. Here, iBackupBot was used to connect with the iPhone, as shown in step 1. In step 2 Raw file system is selected. Step 3 is when you navigate to the appropriate DCIM directory. Step 4 shows the image, as identified on the device and in the previous slides. Step 5 shows the results from EXIF tool. As seen here, the metadata shows the original (non-adjusted) timestamp of the photo. This means that the iOS device is leveraging another file to track the adjustments.

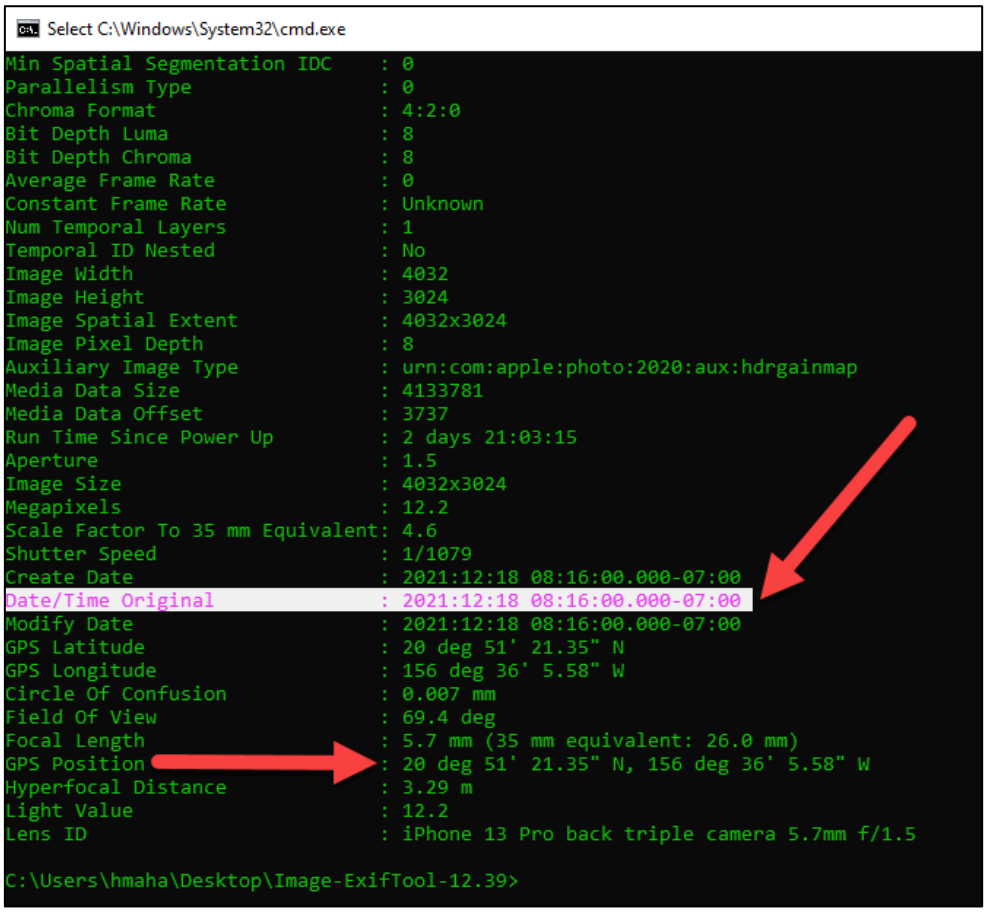

iBackupBot for iPad iPhone
File View Settings Help
Backups
hmaha's iPhone (05/18/21 09:20:46)
System Files
User App Files
App Group Files
App Placeholder Files
App Plugin Files
User Information Manager
Multimedia File Manager
Devices
Heather's iPhon
User Applications
App File Sharing
Raw File System
Tools
1
2
Refresh
New Folder
Open
Delete
Export
Import
Search
Raw File System
Podcasts
Downloads
CloudAssets
Books
Photos
Deferred
Vibrations
EnhancedAudio
Recordings
Radio
DCIM
101IMPRT
100IMPRT
103IMPRT
.MISC
101APPLE
100APPLE
102IMPRT
iTunes_Control
MediaAnalysis
PhotoData
Purchases
LoFiCloudAsset
3
Name
Size
Type
Date Modified
IMG_1358.HEIC 1.9 MB HEIC File 01/24/22 15:41:22
IMG_1357.HEIC 1.4 MB HEIC File 01/24/22 15:41:25
IMG_1356.HEIC 3.9 MB HEIC File 01/24/22 10:16:26
IMG_1355.HEIC 3.3 MB HEIC File 01/24/22 10:16:13
4
C:\Windows\System32\cmd.exe
5
Num Temporal Layers : 1
Temporal ID Nested : No
Image Width : 4032
Image Height : 3024
Image Spatial Extent : 4032x3024
Rotation : 0
Image Pixel Depth : 8
Auxiliary Image Type : urn:com:apple:photo:2020:aux:hdrgainmap
Media Data Size : 4133587
Media Data Offset : 3765
Run Time Since Power Up : 2 days 21:03:15
Aperture : 1.5
Image Size : 4032x3024
Megapixels : 12.2
Scale Factor To 35 mm Equivalent: 4.6
Shutter Speed : 1/1079
Create Date : 2022:01:24 08:16:26.591-07:00
Date/Time Original : 2022:01:24 08:16:26.591-07:00
Modify Date : 2022:01:24 08:16:26-07:00
Circle Of Confusion : 0.007 mm
Field Of View : 69.4 deg
Focal Length : 5.7 mm (35 mm equivalent: 26.0 mm)
Hyperfocal Distance : 3.29 m
Light Value : 12.2
Lens ID : iPhone 13 Pro back triple camera 5.7mm f/1.5
C:\Users\hmaha\Desktop\Image-ExifTool-12.39>

iOS 15 - Photo Adjustments - Receiving Device Artifacts





After the test was complete on the previous slide, a photo was AirDropped to another device. The AirDropped EXIF data showed the adjusted timestamp and adjusted location. The only way to even hunt down the original timestamp was to look in a Hex tool, which is, where we see 2022:01:24



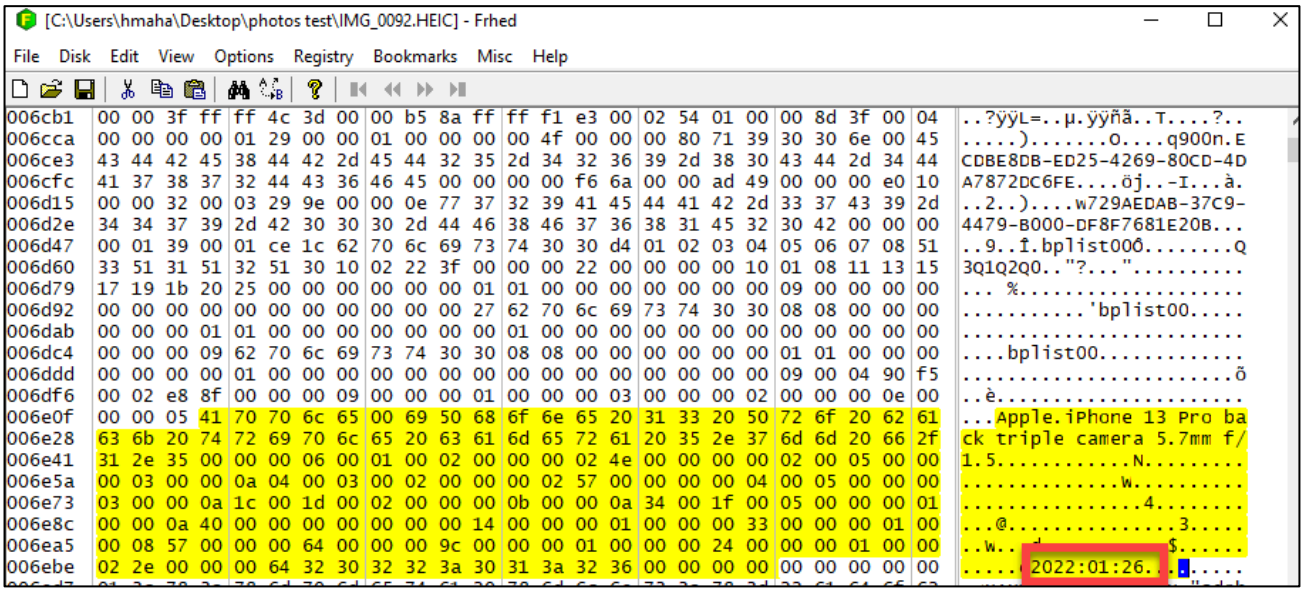
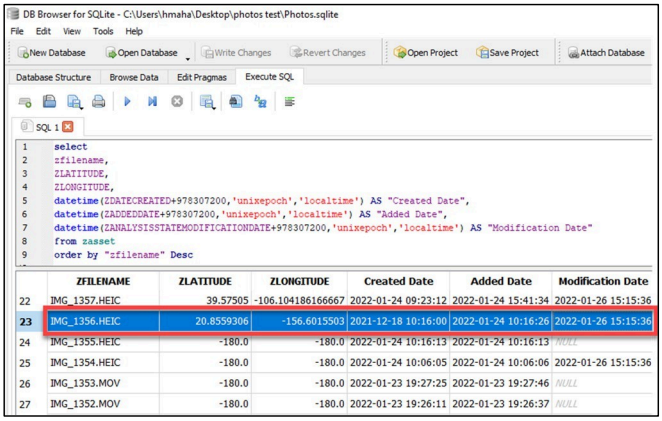

[C:\Users\hmaha\Desktop\photos test\IMG_0092.HEIC] - Frhed
File Disk Edit View Options Registry Bookmarks Misc Help
006cb1 00 00 3f ff ff 4c 3d 00 00 b5 8a ff ff f1 e3 00 02 54 01 00 00 8d 3f 00 04 ..?ÿÿL=..µ.ÿÿñã..T....?..
006cca 00 00 00 00 01 29 00 00 01 00 00 00 00 4f 00 00 00 80 71 39 30 30 6e 00 45 .....).....O....q900n.E
006ce3 43 44 42 45 38 44 42 2d 45 44 32 35 2d 34 32 36 39 2d 38 30 43 44 2d 34 44 CDBE8DB-ED25-4269-80CD-4D
006cfc 41 37 38 37 32 44 43 36 46 45 00 00 00 00 f6 6a 00 00 ad 49 00 00 00 e0 10 A7872DC6FE....öj..-I...à.
006d15 00 00 32 00 03 29 9e 00 00 0e 77 37 32 39 41 45 44 41 42 2d 33 37 43 39 2d ..2..)....w729AEDAB-37C9-
006d2e 34 34 37 39 2d 42 30 30 30 2d 44 46 38 46 37 36 38 31 45 32 30 42 00 00 00 4479-B000-DF8F7681E20B...
006d47 00 01 39 00 01 ce 1c 62 70 6c 69 73 74 30 30 d4 01 02 03 04 05 06 07 08 51 ..9..î.bplist00Ô.....Q
006d60 33 51 31 51 32 51 30 10 02 22 3f 00 00 00 22 00 00 00 00 10 01 08 11 13 15 3Q1Q2Q0.."?..."..........
006d79 17 19 1b 20 25 00 00 00 00 00 00 00 01 01 00 00 00 00 00 00 00 09 00 00 00 00 ... %.....................
006d92 00 00 00 00 00 00 00 00 00 00 00 27 62 70 6c 69 73 74 30 30 08 08 00 00 00 ....'bplist00.....
006dab 00 00 00 01 01 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 ....
006dc4 00 00 00 09 62 70 6c 69 73 74 30 30 08 08 00 00 00 00 00 00 01 01 00 00 00 ....bplist00.............
006ddd 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 09 00 04 90 f5 ....õ
006df6 00 02 e8 8f 00 00 00 09 00 00 00 01 00 00 00 03 00 00 00 02 00 00 00 0e 00 ..è.....................
006e0f 00 00 05 41 70 70 6c 65 00 69 50 68 6f 6e 65 20 31 33 20 50 72 6f 20 62 61 ...Apple.iPhone 13 Pro ba
006e28 63 6b 20 74 72 69 70 6c 65 20 63 61 6d 65 72 61 20 35 2e 37 6d 6d 20 66 2f ck triple camera 5.7mm f/
006e41 31 2e 35 00 00 00 06 00 01 00 02 00 00 00 02 4e 00 00 00 00 02 00 05 00 00 1.5.....N.....
006e5a 00 03 00 00 0a 04 00 03 00 02 00 00 00 02 57 00 00 00 00 04 00 05 00 00 00 ....W.....
006e73 03 00 00 0a 1c 00 1d 00 02 00 00 00 0b 00 00 0a 34 00 1f 00 05 00 00 00 01 ....4.....
006e8c 00 00 0a 40 00 00 00 00 00 00 00 14 00 00 00 01 00 00 00 33 00 00 00 01 00 ...@.....3.....
006ea5 00 08 57 00 00 00 64 00 00 00 9c 00 00 00 01 00 00 00 24 00 00 00 01 00 00 ..W...d.....$.....
006ebe 02 2e 00 00 00 64 32 30 32 32 3a 30 31 3a 32 36 00 00 00 00 00 00 00 00 00 .....d2022:01:26.....
Here, the tool Frhed was used to scan the Hex inal timestamp.

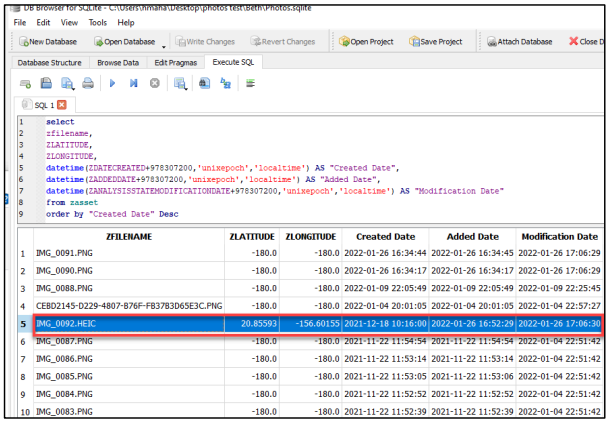
Photo Adjustments - Photos.sqlite

Original Device



	ZFILENAME	ZLATITUDE	ZLONGITUDE	Created Date	Added Date	Modification Date
22	IMG_1357.HEIC	39.57505	-106.104186166667	2022-01-24 09:23:12	2022-01-24 15:41:34	2022-01-26 15:15:36
23	IMG_1356.HEIC	20.8559306	-156.6015503	2021-12-18 10:16:00	2022-01-24 10:16:26	2022-01-26 15:15:36
24	IMG_1355.HEIC	-180.0	-180.0	2022-01-24 10:16:13	2022-01-24 10:16:13	NULL
25	IMG_1354.HEIC	-180.0	-180.0	2022-01-24 10:06:05	2022-01-24 10:06:06	2022-01-26 15:15:36
26	IMG_1353.MOV	-180.0	-180.0	2022-01-23 19:27:25	2022-01-23 19:27:46	NULL
27	IMG_1352.MOV	-180.0	-180.0	2022-01-23 19:26:11	2022-01-23 19:26:37	NULL

Receiving Device

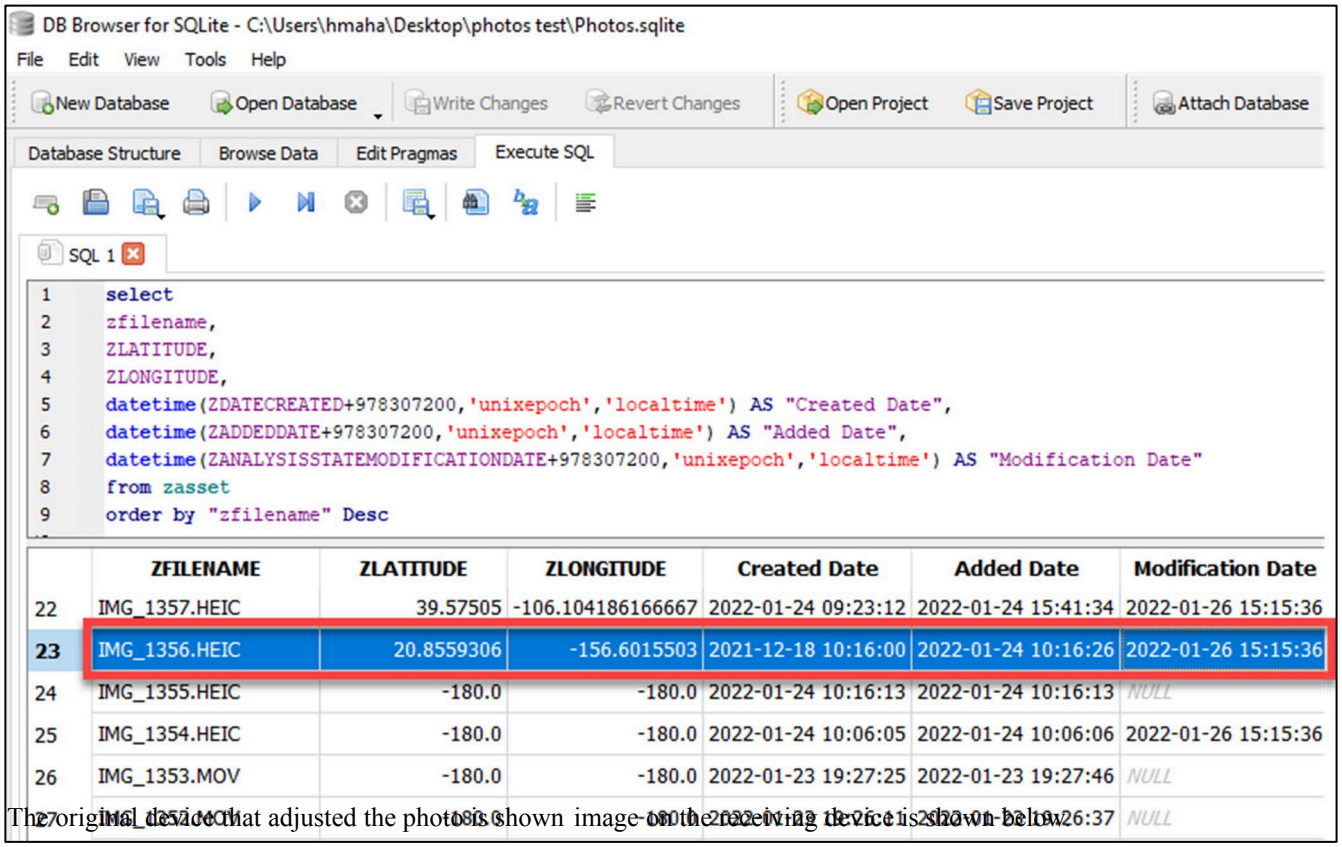


	ZFILENAME	ZLATITUDE	ZLONGITUDE	Created Date	Added Date	Modification Date
1	IMG_0091.PNG	-180.0	-180.0	2022-01-26 16:34:44	2022-01-26 16:34:45	2022-01-26 17:06:29
2	IMG_0090.PNG	-180.0	-180.0	2022-01-26 16:34:17	2022-01-26 16:34:17	2022-01-26 17:06:29
3	IMG_0088.PNG	-180.0	-180.0	2022-01-09 22:05:49	2022-01-09 22:05:49	2022-01-09 22:25:45
4	CEBD2145-D229-4807-B76F-FB37B3D65E3C.PNG	-180.0	-180.0	2022-01-04 20:01:05	2022-01-04 20:01:05	2022-01-04 22:57:27
5	IMG_0092.HEIC	20.85593	-156.60155	2021-12-18 10:16:00	2022-01-26 16:52:29	2022-01-26 17:06:30
6	IMG_0087.PNG	-180.0	-180.0	2021-11-22 11:54:54	2021-11-22 11:54:54	2022-01-04 22:51:42
7	IMG_0086.PNG	-180.0	-180.0	2021-11-22 11:53:14	2021-11-22 11:53:14	2022-01-04 22:51:42
8	IMG_0085.PNG	-180.0	-180.0	2021-11-22 11:53:05	2021-11-22 11:53:06	2022-01-04 22:51:42
9	IMG_0084.PNG	-180.0	-180.0	2021-11-22 11:52:52	2021-11-22 11:52:52	2022-01-04 22:51:42
10	IMG_0083.PNG	-180.0	-180.0	2021-11-22 11:52:39	2021-11-22 11:52:39	2022-01-04 22:51:42

The truth is often found in a database. Here we are looking at Photos.sqlite to see how the data exists on the original device that took and adjusted the photo as well as the device that received the AirDrop. The query below was used to obtain quick information needed for our test.

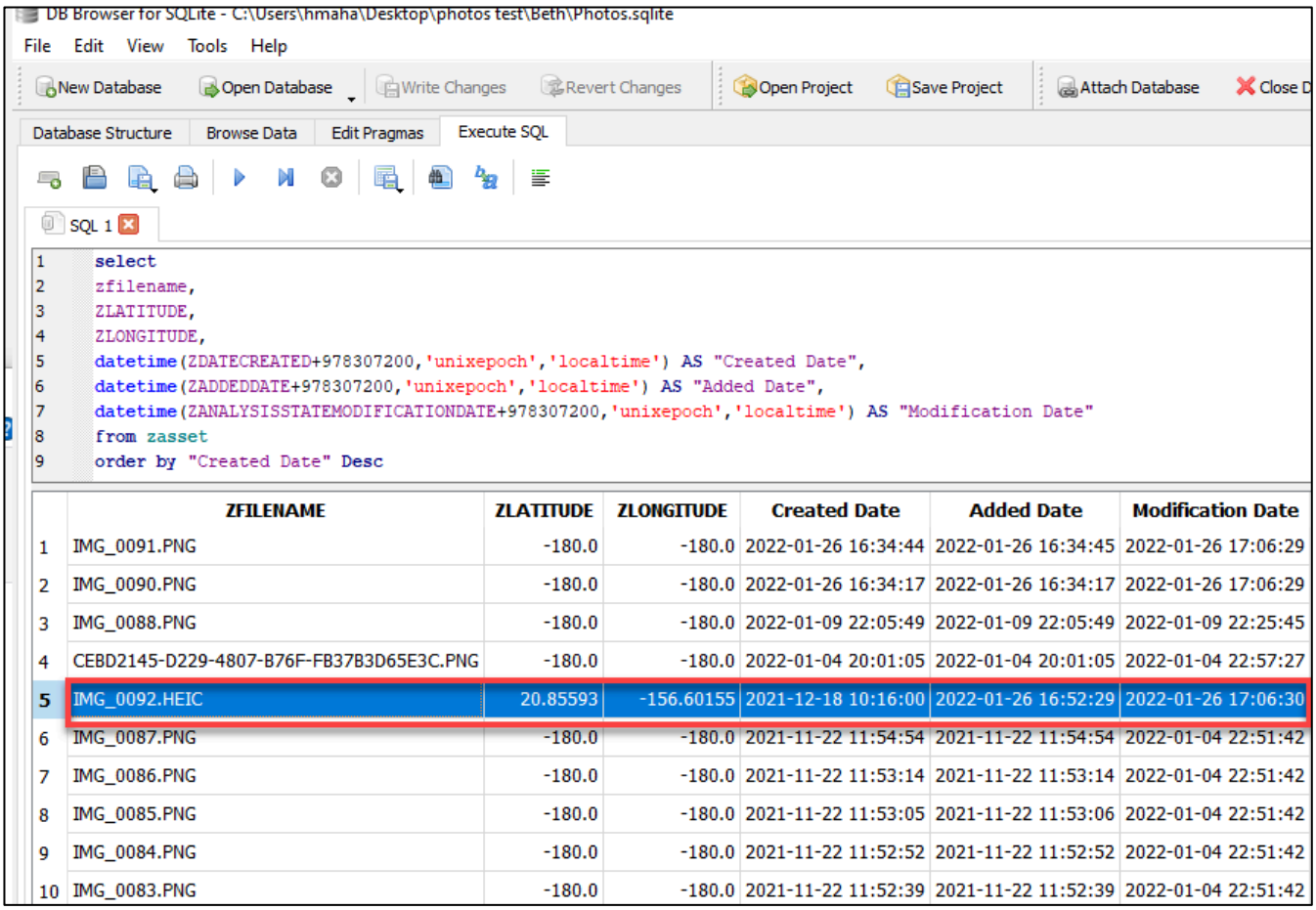
```
select
zfilename,
ZLATITUDE,
ZLONGITUDE,
datetime(ZDATECREATED+978307200,'unixepoch','localtime') AS "Created Date",
datetime(ZADDEDDATE+978307200,'unixepoch','localtime') AS "Added Date",
datetime(ZANALYSISSTATEMODIFICATIONDATE+978307200,'unixepoch','localtime') AS "Modification Date"
from zasset
```

The Original device shows that the Latitude, Longitude and Created Date were updated to match the adjustment made by the user. The Receiving device shows the adjusted data and the date it was added to the device.



	ZFILENAME	ZLATITUDE	ZLONGITUDE	Created Date	Added Date	Modification Date
22	IMG_1357.HEIC	39.57505	-106.104186166667	2022-01-24 09:23:12	2022-01-24 15:41:34	2022-01-26 15:15:36
23	IMG_1356.HEIC	20.8559306	-156.6015503	2021-12-18 10:16:00	2022-01-24 10:16:26	2022-01-26 15:15:36
24	IMG_1355.HEIC	-180.0	-180.0	2022-01-24 10:16:13	2022-01-24 10:16:13	NULL
25	IMG_1354.HEIC	-180.0	-180.0	2022-01-24 10:06:05	2022-01-24 10:06:06	2022-01-26 15:15:36
26	IMG_1353.MOV	-180.0	-180.0	2022-01-23 19:27:25	2022-01-23 19:27:46	NULL
27	IMG_1352.MOV	-180.0	-180.0	2022-01-23 19:26:1	2022-01-23 19:26:37	NULL

The original device that adjusted the photo is shown image on the receiving device is shown below.



	ZFILENAME	ZLATITUDE	ZLONGITUDE	Created Date	Added Date	Modification Date
1	IMG_0091.PNG	-180.0	-180.0	2022-01-26 16:34:44	2022-01-26 16:34:45	2022-01-26 17:06:29
2	IMG_0090.PNG	-180.0	-180.0	2022-01-26 16:34:17	2022-01-26 16:34:17	2022-01-26 17:06:29
3	IMG_0088.PNG	-180.0	-180.0	2022-01-09 22:05:49	2022-01-09 22:05:49	2022-01-09 22:25:45
4	CEBD2145-D229-4807-B76F-FB37B3D65E3C.PNG	-180.0	-180.0	2022-01-04 20:01:05	2022-01-04 20:01:05	2022-01-04 22:57:27
5	IMG_0092.HEIC	20.85593	-156.60155	2021-12-18 10:16:00	2022-01-26 16:52:29	2022-01-26 17:06:30
6	IMG_0087.PNG	-180.0	-180.0	2021-11-22 11:54:54	2021-11-22 11:54:54	2022-01-04 22:51:42
7	IMG_0086.PNG	-180.0	-180.0	2021-11-22 11:53:14	2021-11-22 11:53:14	2022-01-04 22:51:42
8	IMG_0085.PNG	-180.0	-180.0	2021-11-22 11:53:05	2021-11-22 11:53:06	2022-01-04 22:51:42
9	IMG_0084.PNG	-180.0	-180.0	2021-11-22 11:52:52	2021-11-22 11:52:52	2022-01-04 22:51:42
10	IMG_0083.PNG	-180.0	-180.0	2021-11-22 11:52:39	2021-11-22 11:52:39	2022-01-04 22:51:42

Ian Whiffin summarizes the relevant tables from Photos.sqlite and which are affected by the adjustments in his blog www.for585.com/adjusted. This table is shown below.

ZASSET	
ZCREATEDDATE	Affected by the change
ZLATITUDE	Affected by the change
ZLONGITUDE	Affected by the change
ZADDEDDATE	Unaffected
ZANALYSISSTATEMODIFICATIONDATE	Unaffected
ZADDITIONALASSETATTRIBUTES	
ZSCENEANALYSISTIMESTAMP	Affected by the change
ZTIMEZONEOFFSET	Affected by the change
ZTIMEZONENAME	Affected by the change
ZREVERSELOCATIONDATA	Affected by the change
ZGPSHORIZONTACCURACY	Affected by the change
ZEXIFTIMESTAMPSTRING	Unaffected
ZCLOUDMASTERMEDIAMETADATA	
ZDATA	Unaffected
ZEXTENDEDATTRIBUTES	
ZLATITUDE	Unaffected
ZLONGITUDE	Unaffected

As we discussed in Section 2 on Android, iOS snapshots may be your best evidence. With jailbreaks now available, chances are good that we can recover screenshots from iOS devices! You never know what you may get. Having some relevant information supporting the screenshot may prove to be fruitful. A good blog to read about this is: https://for585.com/iossnaps and the script to parse it is: https://github.com/abrignoni/iOS-Snapshot-Triage-Parser/blob/master/SnapshotTriage.py.

Snapshots are taken on iOS devices when the application in use is minimized by the user or is interrupted by another activity. For example, if I am playing Words With Friends and a call comes in, my application will snapshot itself and fade into the background while I take the call. The format is .png for iOS 10 and earlier and .ktx for iOS 11+.

If Apple saves the screenshot, it will be stored in:

Library/Caches/Snapshots/<bundleID>/<bundleID>/

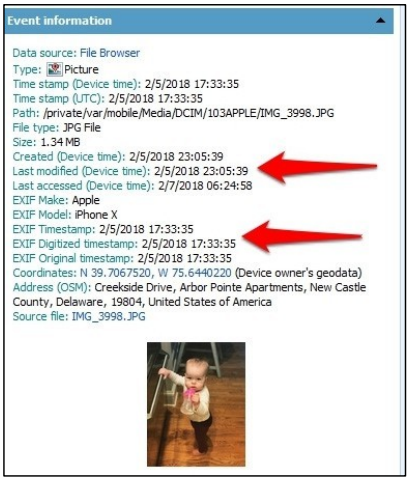
or

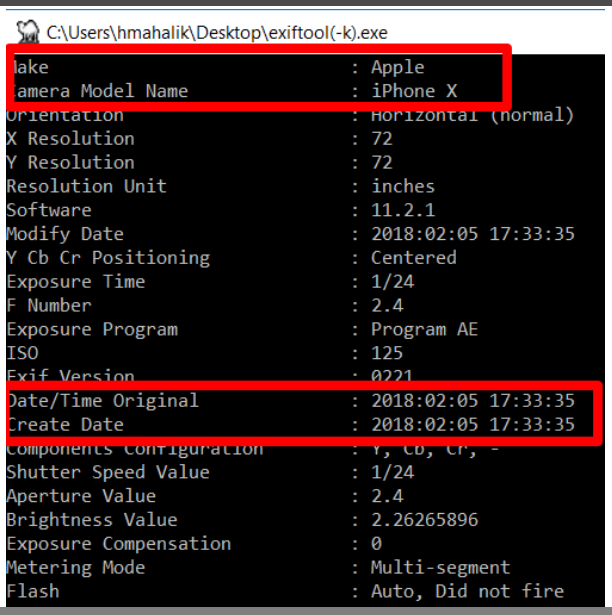
/mobile/Containers/Data/Application/<ApplicationUUID>/Library/Caches/Snapshots/<bundleID>

If the user creates the screenshot, which is shown on the slide above where we see Instagram from my iOS 15 device, the file would be saved as .png in the /Media/DCIM/101APPLE directory because I (the user) chose to save it to photos.

Multimedia Files EXIF Data

Don't let EXIF artifacts confuse your investigation…





Sometimes it's easy to get lost in what your tool is trying to tell you. When we look at the EXIF data for the picture shown on the left, we can see several timestamps. We have Original, Digitized, EXIF and Device timestamps. How can you make sense of all of this? Sometimes it can be as easy as using more than one tool; other times you may have to create test data. The example on the right is ExifTool parsing the original photo, not the one on the left, which is pulled from my device. Why are we showing you both? So, you learn what you can trust in your tool. A photo was sent to my device. The screenshot on the right from ExifTool shows the original metadata. I opened the MMS and saved the image to my device. The EXIF parsed by Oxygen (the left image) shows the original timestamps as well as my device time:

Created (Device time): when the photo was saved on my device
Last modified (Device time): the last time the photo was modified
Last accessed (Device time): the last time the photo was viewed or accessed

The metadata all reflects the original device that took the photo, which is also shown by ExifTool. Now where this can get confusing is the Coordinates and Address. Notice how Oxygen states "Device owner's geodata"? This is where Heather was when she opened and saved that photo! This is not only close to where Heather was when she received the photo but shows the train tracks of Amtrak that run right behind Arbor Point Apartments in New Castle County, Delaware. When we first saw this, we were confused. We thought the metadata meant the photo was taken there (on the Amtrak tracks), but it doesn't. It is where the user of the device was located when that photo was saved/created on their device. This could really help an investigation if location and pictures matter!

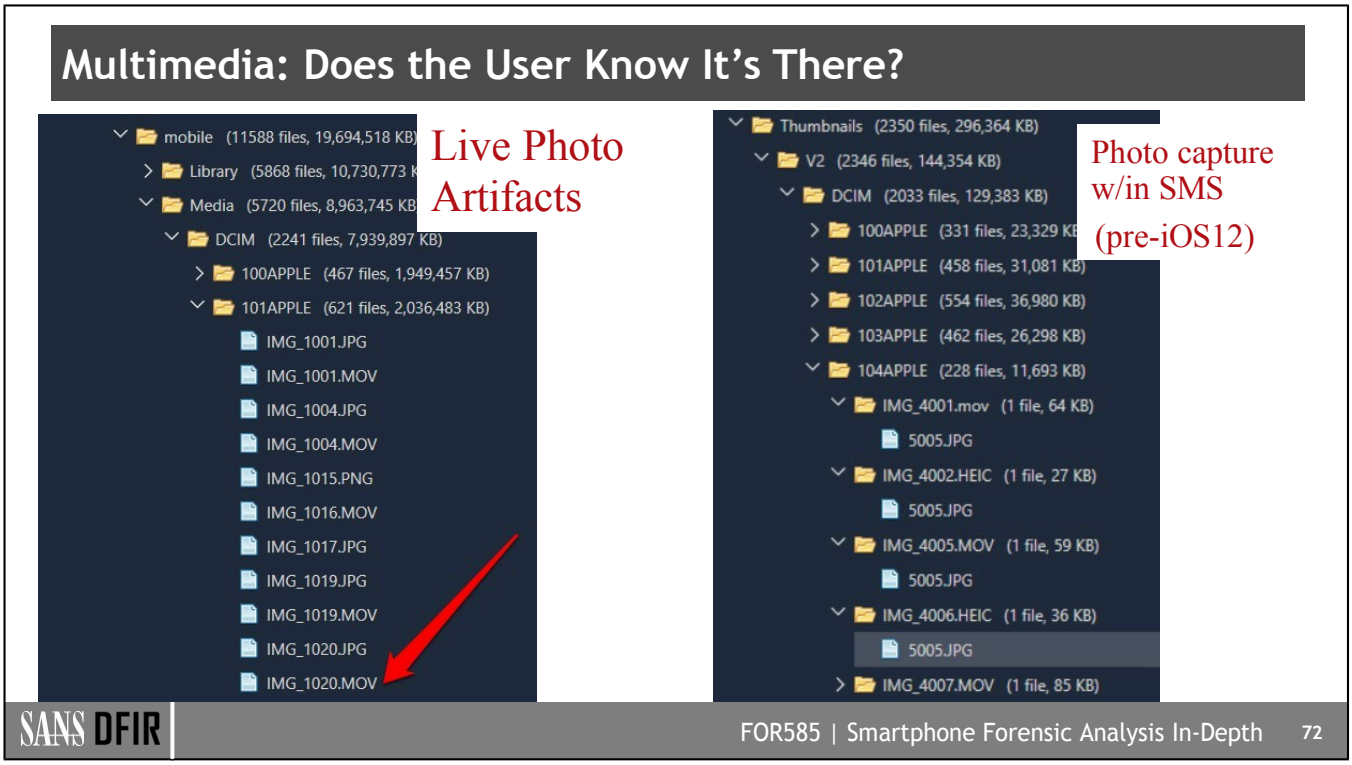
Remember, location services may be turned off, and you may get little to no metadata, depending on the device that took the photo and the settings on the device. If you are questioning what you are seeing or even not seeing, export the photo and look at the EXIF in a second tool. If you find images that are of interest, try ExifTool. It's a free way to verify what you believe the tool is telling you. This program is on your VM in C:\ProgramData\chocolatey\bin.

Event information

Data source: File Browser
Type: Picture
Time stamp (Device time): 2/5/2018 17:33:35
Time stamp (UTC): 2/5/2018 17:33:35
Path: /private/var/mobile/Media/DCIM/103APPLE/IMG_3998.JPG
File type: JPG File
Size: 1.34 MB
Created (Device time): 2/5/2018 23:05:39
Last modified (Device time): 2/5/2018 23:05:39
Last accessed (Device time): 2/7/2018 06:24:58
EXIF Make: Apple
EXIF Model: iPhone X
EXIF Timestamp: 2/5/2018 17:33:35
EXIF Digitized timestamp: 2/5/2018 17:33:35
EXIF Original timestamp: 2/5/2018 17:33:35
Coordinates: N 39.7067520, W 75.6440220 (Device owner's geodata)
Address (OSM): Creekside Drive, Arbor Pointe Apartments, New Castle County, Delaware, 19804, United States of America
Source file: IMG_3998.JPG

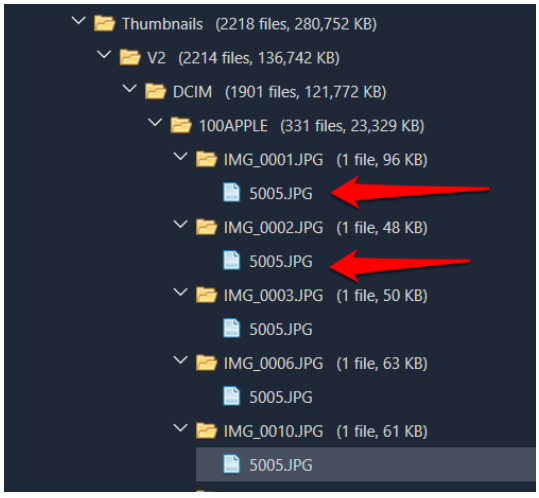
C:\Users\hmahalik\Desktop\exiftool(-k).exe

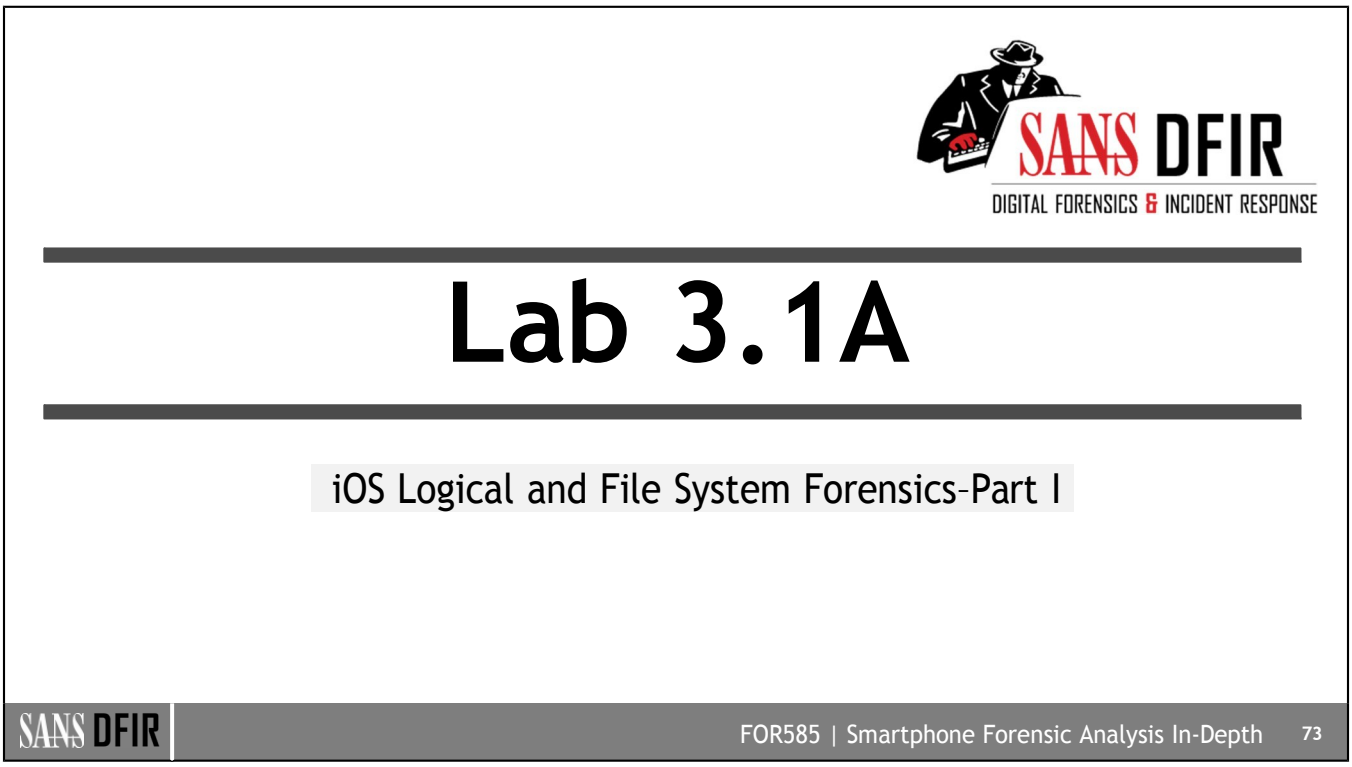
```
lake                            : Apple
Camera Model Name               : iPhone X
Orientation                     : Horizontal (normal)
X Resolution                    : 72
Y Resolution                    : 72
Resolution Unit                 : inches
Software                        : 11.2.1
Modify Date                     : 2018:02:05 17:33:35
Y Cb Cr Positioning             : Centered
Exposure Time                   : 1/24
F Number                        : 2.4
Exposure Program                : Program AE
ISO                             : 125
Exif Version                    : 0221
Date/Time Original              : 2018:02:05 17:33:35
Create Date                     : 2018:02:05 17:33:35
Components Configuration        : Y, Cb, Cr, -
Shutter Speed Value             : 1/24
Aperture Value                  : 2.4
Brightness Value                : 2.26265896
Exposure Compensation           : 0
Metering Mode                   : Multi-segment
Flash                           : Auto, Did not fire
```



The PhotoData directory contains files the user may not even realize exist on their iPhone. On the left, we see artifacts associated with Live Photos on iOS devices. Most users do not realize that a .MOV file is created for every Live Photo taken on an iOS device. This data is saved in the PhotoData directory and can be reviewed for relevance. Audio is also included for these files. Pretty amazing evidence when trying to place a person or subject behind a mobile device. The .JPG above the .MOV file is the image the user sees or saw on the device.

The screenshot on the right shows the artifacts left behind when a user snaps a picture from within the SMS application. The location for these photos is /var/mobile/Media/PhotoData/Thumbnails/V2/DCIM/*. These photos are not stored in the user's gallery (except in iOS 12); however, the thumbnails and pictures are captured and saved to the device. This data cannot be viewed easily by the user unless they back up their device—again, fantastic artifacts here! Use caution when searching for a file name, as you will notice that the images all have the same file name, but the path will differ. Also, take notice that HEIC image directories are saved as JPG. This means our tool can render the graphics for us. One thing that is odd is that the .JPG files all have the same name. The only thing unique is the directory containing that file, which may be a better search term.





This page intentionally left blank.

Location Information

Cell towers

Wi-Fi networks – Changed in iOS 14 – randomized by Apple

Media locations

Maps

Searched items

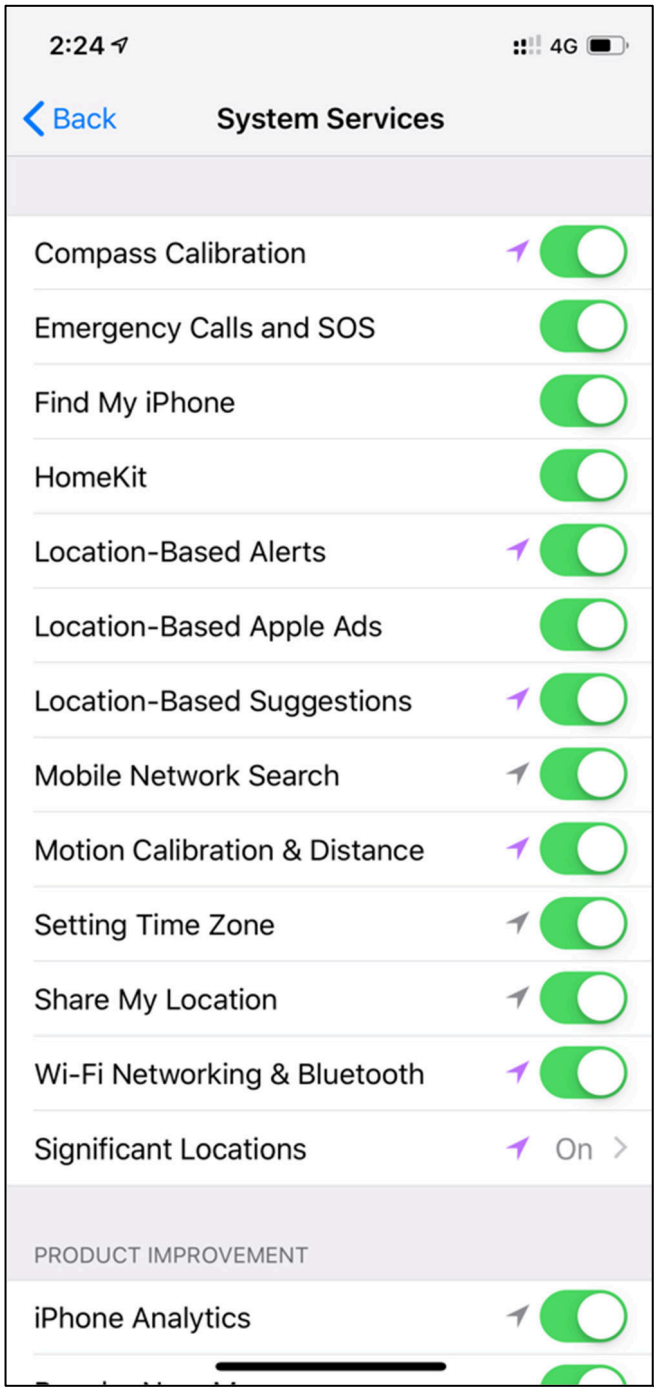
- Apple Maps, Google Maps, Navigation apps

Third-party applications and more

Location information is saved to the iOS device via multiple channels. The iOS device tracks cell towers and Wi-Fi networks that come into close proximity with the device, as well as those to which the iOS device connects. Location information is also recovered from media files, maps, internet searches, Apple Maps, Google Maps, and third-party applications. The examiner must know the difference between locations that the user connects to versus those that were simply in close proximity to the device. However, even the latter can be helpful if we can put the device in a region when activities occurred. Make sure you look at Installed Applications and consider that browsers may even keep Google searches for locations directly inside of its database! You really need to look everywhere, which is why searching for terms like “position” and “location” in a tool that can search within all content is super helpful.

One thing that we can do is leverage what the tool detects and then dig for artifacts we know we can trust. Later in this section, we will look closely at what we can rely on and what we cannot trust.

Frequent locations/Significant Locations: This is not always consistent, and we cannot state why something is saved here versus other locations. Essentially, we do not understand Apple’s algorithm. For this data to be tracked, the user must enable the settings in Settings > Privacy > Location Services > System Services >Significant Locations. This is shown in the next slide. More information on location data and how the artifacts exist can be found on Sarah Edwards' blog: https://for585.com/518. There are tons of files that store location information available on your iOS cheat sheet.



Reference:

[1] https://for585.com/518

Location Services Settings

You need to determine if location services are enabled on the device

What if it is changed by the user (turned off and back on)?

com.apple.locationd.plist

- If the value is 0x01 (Hex view) or "True", then location services are enabled

We are going to talk about locations coming up, but we need to address this here because EXIF information commonly stores location artifacts. If you are examining a device and you aren't finding locations, verify the settings in the com.apple.locationd.plist to see if the settings shows 0x01, which means enabled. A great reference blog is https://theforensicscooter.com/2021/09/20/ios-location-services-and-system-services-on-or-off/ .

Wi-Fi – Key Files of Interest

All devices:

/preferences/com.apple.Wi-Fi.known-networks.plist
/preferences/SystemConfiguration/com.apple.wifi.plist
/root/Library/Preferences/com.apple.preferences.network.plist
• Tracks if airplane mode is enabled

Jailbroken devices:

/db/dhcpclient/leases/en0.plist
/root/Library/ApplicationSupport/com.apple.wifianalyticsd/DeviceAnalyticsModel.sqlite
/wireless/Library/Preferences/com.apple.commcenter.device_specific_nobackup.plist
/root/Library/Application Support/WiFiNetworkStoreModel.sqlite

Cell towers and Wi-Fi provide solid location information for mobile devices. Apple is a bit more organized with how Wi-Fi is stored, and this information can be found in either /preferences.com.apple.wifi.known-networks.plist (iOS14 and iOS 15) and /preferences/SystemConfiguration/com.apple.wifi.plist for older iOS versions. When examining, make sure you look for key artifacts like SSID, BSSID, timestamps, password requirements, etc. The “HwMacAddressMacRandomisation” value is device's actual hardware Wi-Fi MAC address, not randomized. If device is being used as a mobile hotspot, connected devices can be identified here. Device's connecting to the mobile hotspot can be located within the nested keys of "InterfaceDataUsageV1“. These findings may help you place a suspect in a location at a specific time. A good reference is here https://ciofecaforensics.com/2020/10/24/apple-private-addresses/.

The file /root/Library/Preferences/com.apple.preferences.network.plist - Contains one boolean value for AirplaneModeEnabled (0 = No, 1 = Yes). In theory, this file can be checked to make sure the examiner had the device in Airplane Mode at the time of extraction. If value is ever 1, this may open the line of questioning about data integrity for not using Airplane Mode (unless other mitigations are employed).

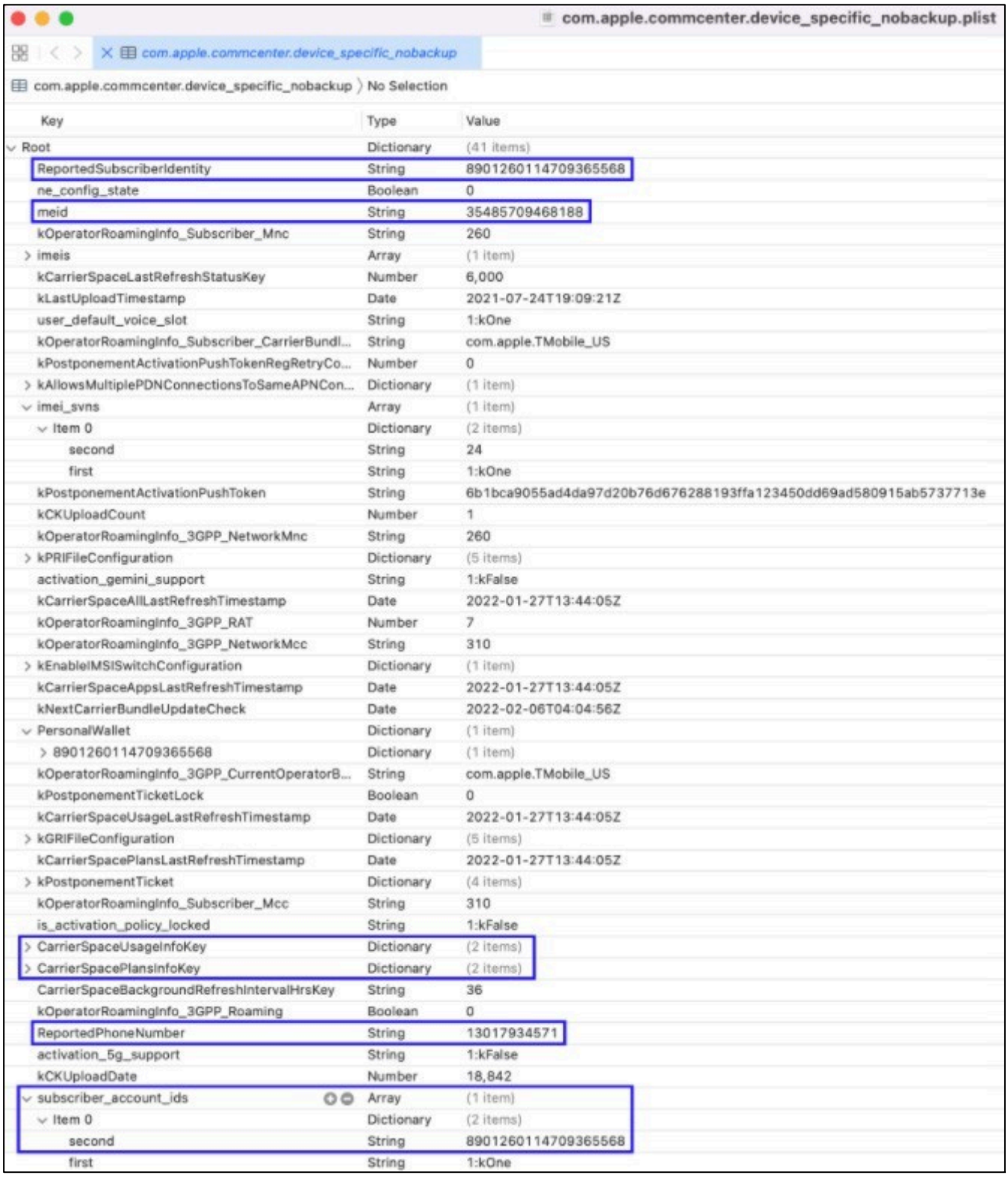
/private/var/db/dhcpclient/leases/en0.plist - The screenshot below is data from iOS 14.5.1. This is the Wi-Fi network connected to when data was extracted or the most recent Wi-Fi connection. Private MAC is close but has a leading '01' and the Router Hardware Address is correct except the first byte of 76 is actually 74 on the hardware, in this example. Last byte of address is explained by channel. Bottom line, the router information may be a bit off.

/private/var/root/Library/ApplicationSupport/com.apple.wifianalyticsd/DeviceAnalyticsModel.sqlite -

Contains a lot of information related to Wi-Fi including join and leave times, last seen time, geo-coordinates, signal strength, and more. Needs additional testing to validate the behavior of the join / leave times.

```
select
ZGEOTAG.Z_PK,
ZGEOTAG.Z_ENT,
datetime(zgeotag.zdate + 978307200,'unixepoch') as "Date",
datetime(zbss.ZLASTSEEN + 978307200,'unixepoch') as "Last Seen",
ZGEOTAG.ZLATITUDE as "Latitude",
ZGEOTAG.ZLONGITUDE as "Longitude",
zbss.ZBSSID as "BSSID",
ZNETWORK.ZSSID
from ZGEOTAG
left join zbss on zbss.Z_PK=ZGEOTAG.ZBSS
left
join znetwork on znetwork.Z_PK=zbss.ZNETWORK
```

/private/var/wireless/Library/Preferences/com.apple.commcenter.device_specific_nobackup.plist -
Contains phone number, ICCID, MNC, MCC, and potentially information about the specific cellular plan subscribed to.

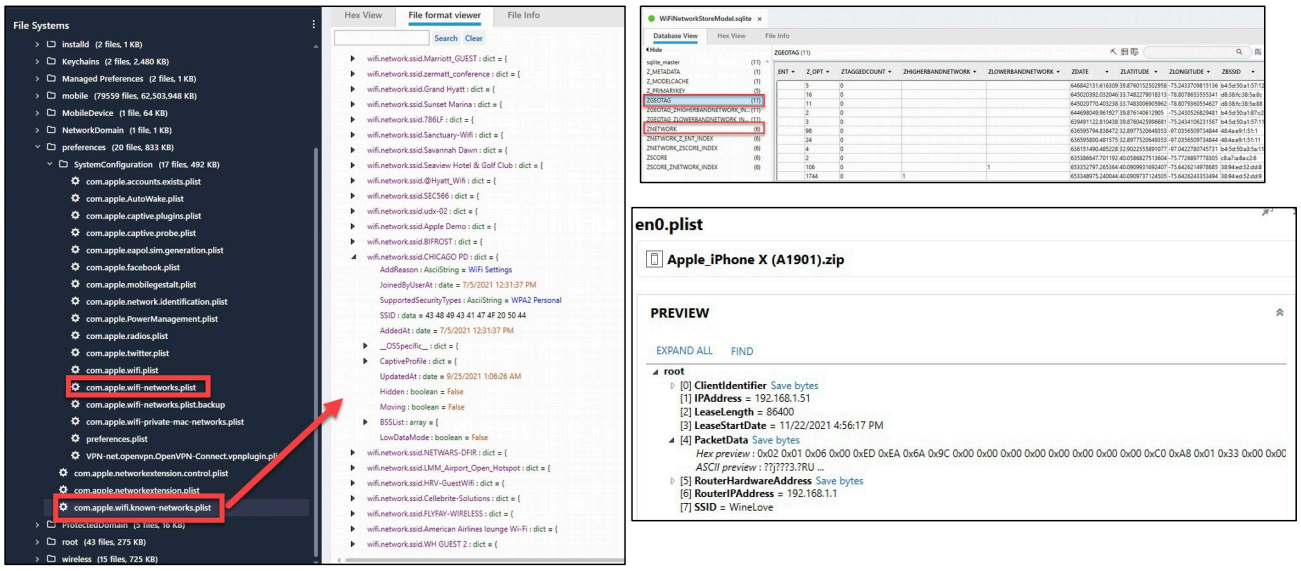


Key	Type	Value
Root	Dictionary	(41 items)
ReportedSubscriberIdentity	String	8901260114709365568
ne_config_state	Boolean	0
meid	String	35485709468188
kOperatorRoamingInfo_Subscriber_Mnc	String	260
imeis	Array	(1 item)
kCarrierSpaceLastRefreshStatusKey	Number	6,000
kLastUploadTimestamp	Date	2021-07-24T19:09:21Z
user_default_voice_slot	String	1:kOne
kOperatorRoamingInfo_Subscriber_CarrierBundl...	String	com.apple.TMobile_US
kPostponementActivationPushTokenRegRetryCo...	Number	0
kAllowsMultiplePDNConnectionsToSameAPNCon...	Dictionary	(1 item)
imei_svns	Array	(1 item)
Item 0	Dictionary	(2 items)
second	String	24
first	String	1:kOne
kPostponementActivationPushToken	String	6b1bca9055ad4da97d20b76d676288193ffa123450dd69ad580915ab5737713e
kCKUploadCount	Number	1
kOperatorRoamingInfo_3GPP_NetworkMnc	String	260
kPRIFileConfiguration	Dictionary	(5 items)
activation_gemini_support	String	1:kFalse
kCarrierSpaceAllLastRefreshTimestamp	Date	2022-01-27T13:44:05Z
kOperatorRoamingInfo_3GPP_RAT	Number	7
kOperatorRoamingInfo_3GPP_NetworkMcc	String	310
kEnableIMSISwitchConfiguration	Dictionary	(1 item)
kCarrierSpaceAppsLastRefreshTimestamp	Date	2022-01-27T13:44:05Z
kNextCarrierBundleUpdateCheck	Date	2022-02-06T04:04:56Z
PersonalWallet	Dictionary	(1 item)
8901260114709365568	Dictionary	(1 item)
kOperatorRoamingInfo_3GPP_CurrentOperatorB...	String	com.apple.TMobile_US
kPostponementTicketLock	Boolean	0
kCarrierSpaceUsageLastRefreshTimestamp	Date	2022-01-27T13:44:05Z
kGRIFileConfiguration	Dictionary	(5 items)
kCarrierSpacePlansLastRefreshTimestamp	Date	2022-01-27T13:44:05Z
kPostponementTicket	Dictionary	(4 items)
kOperatorRoamingInfo_Subscriber_Mcc	String	310
is_activation_policy_locked	String	1:kFalse
CarrierSpaceUsageInfoKey	Dictionary	(2 items)
CarrierSpacePlansInfoKey	Dictionary	(2 items)
CarrierSpaceBackgroundRefreshIntervalHrsKey	String	36
kOperatorRoamingInfo_3GPP_Roaming	Boolean	0
ReportedPhoneNumber	String	13017934571
activation_5g_support	String	1:kFalse
kCKUploadDate	Number	18,842
subscriber_account_ids	Array	(1 item)
Item 0	Dictionary	(2 items)
second	String	8901260114709365568
first	String	1:kOne

Another file of interest is /Library/Application Support/WiFiNetworkStoreModel.sqlite. The query below will parse this information for you.

```
select
ZGEOTAG.z_pk,
datetime(zgeotag.zdate+978307200,'unixepoch') as "Date Joined",
ZGEOTAG.ZLATITUDE as "Latitude",
ZGEOTAG.ZLONGITUDE as "Longitude",
ZGEOTAG.zbssid as "BSSID",
ZGEOTAG.ZHIGHERBANDNETWORK as "Higher Band Network",
ZGEOTAG.ZLOWERBANDNETWORK as "Lower Band Network"
from ZGEOTAG
```

Wi-Fi Examples



This slide shows examples of the files discuss

WiFiNetworkStoreModel.sqlite

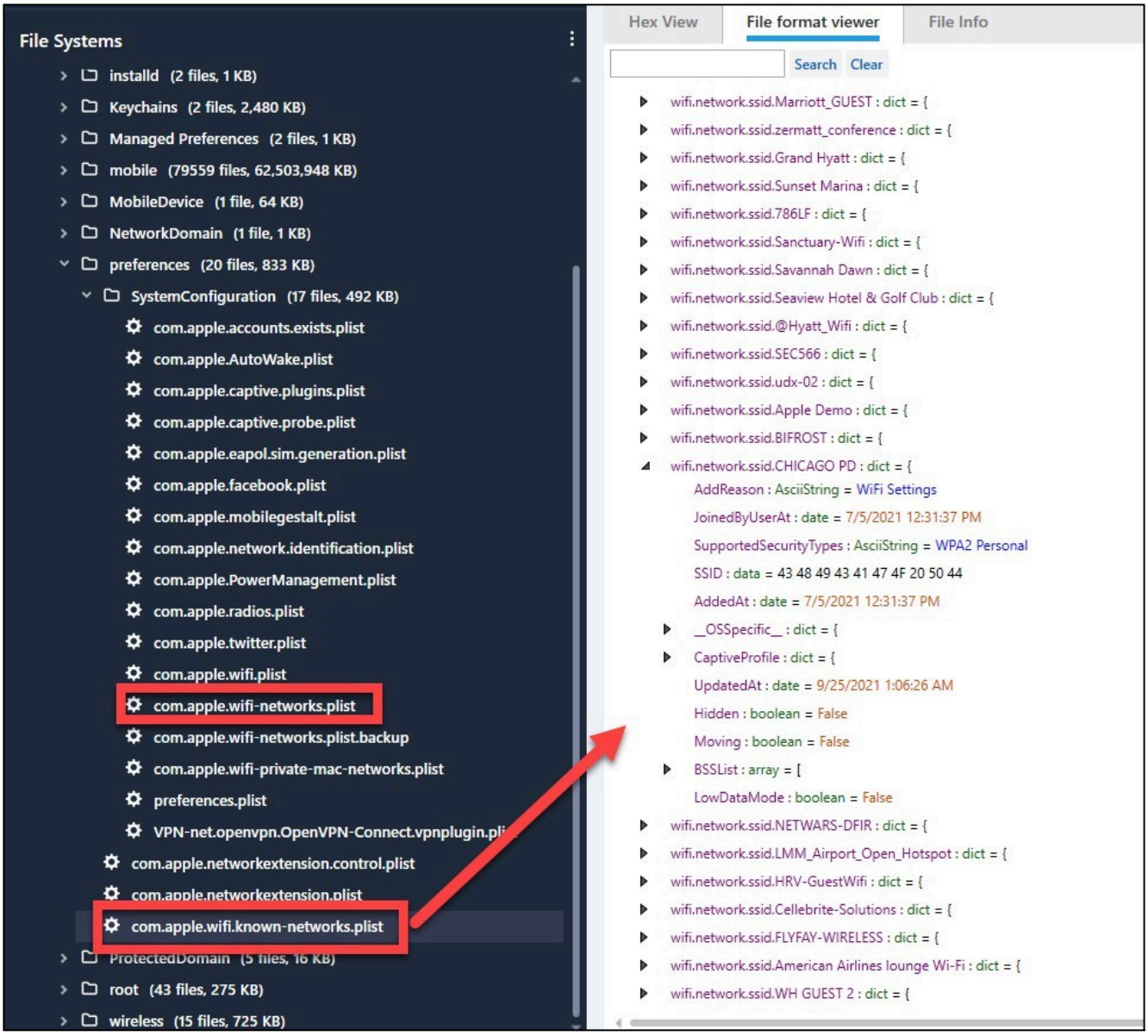
Database View | Hex View | File Info

Hide

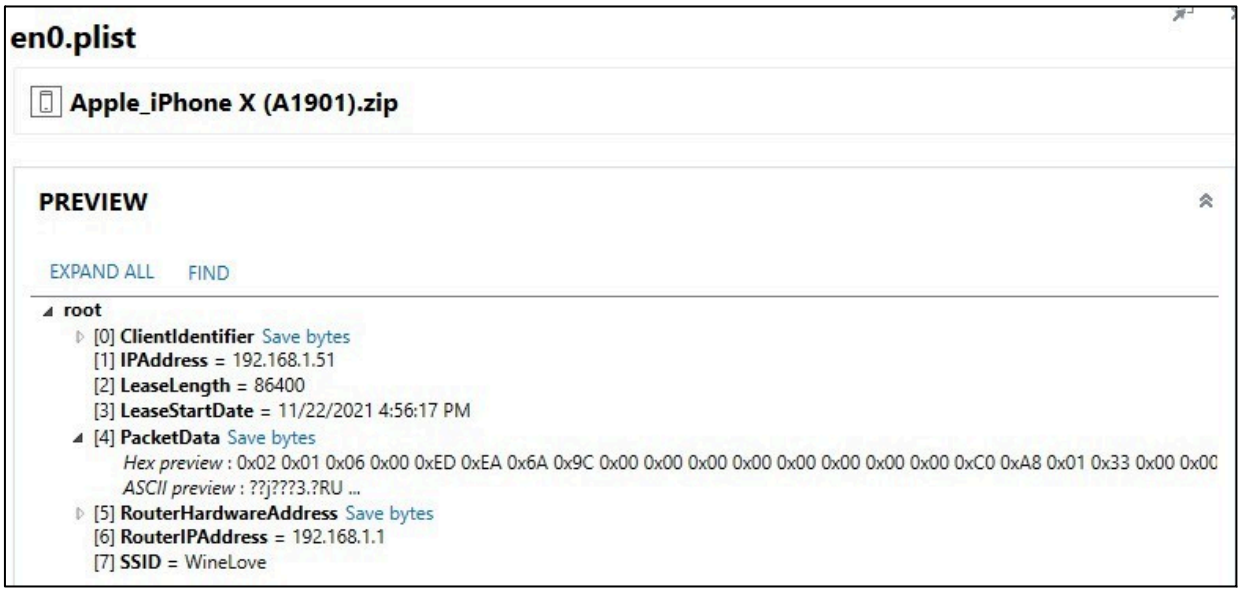
ZGEOTAG (11)

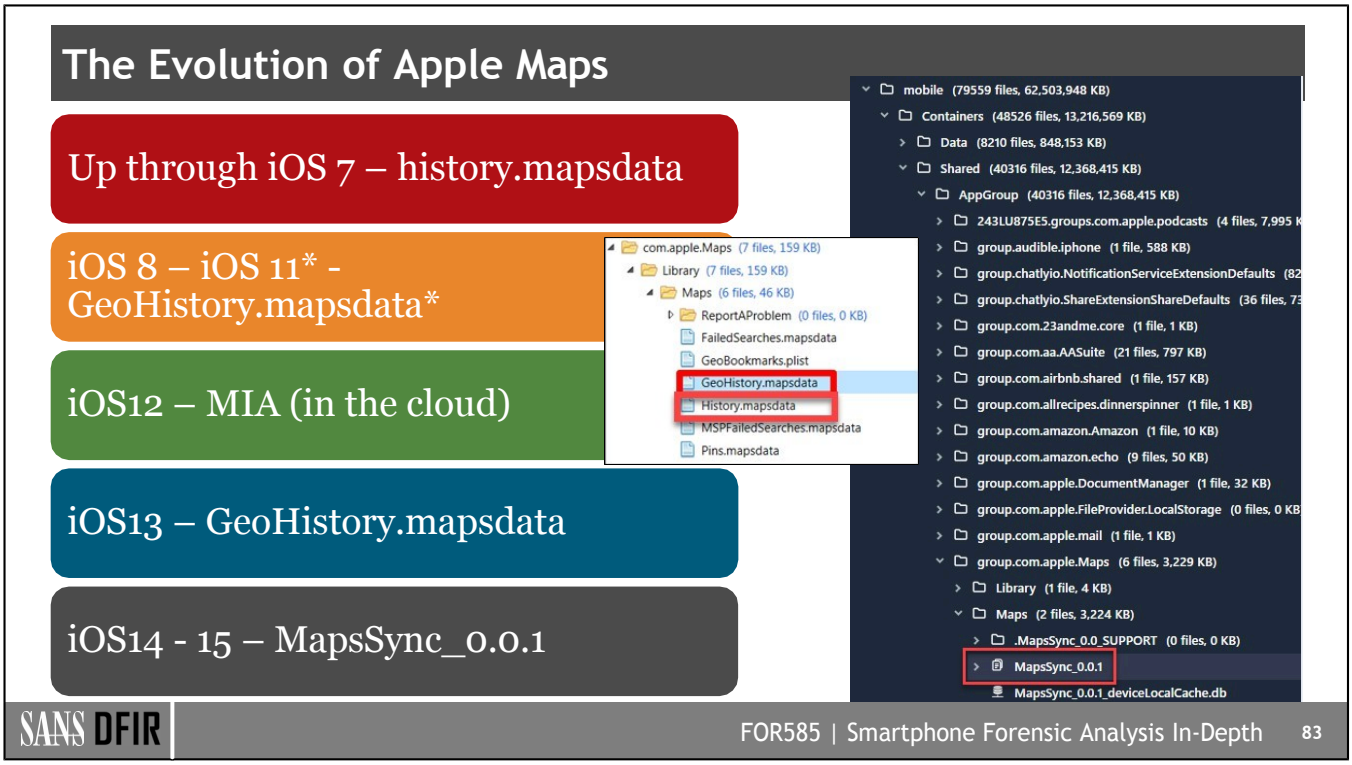
sqlite_master	(11)
Z_METADATA	(1)
Z_MODELCACHE	(1)
Z_PRIMARYKEY	(5)
ZGEOTAG	(11)
ZGEOTAG_ZHIGHERBANDNETWORK_IN...	(11)
ZGEOTAG_ZLOWERBANDNETWORK_IN...	(11)
ZNETWORK	(6)
ZNETWORK_Z_ENT_INDEX	(6)
ZNETWORK_ZSCORE_INDEX	(6)
ZSCORE	(6)
ZSCORE_ZNETWORK_INDEX	(6)

_ENT	Z_OPT	ZTAGGEDCOUNT	ZHIGHERBANDNETWORK	ZLOWERBANDNETWORK	ZDATE	ZLATITUDE	ZLONGITUDE	ZBSSID
	5	0			646842131.616309	39.8760152502958	-75.2433709815136	b4:5d:50:a1:57:12
	16	0			645020392.032046	33.7482279018313	-78.8078653555341	d8:38:fc:38:5a:8c
	11	0			645020770.403238	33.7483006905962	-78.8079360554627	d8:38:fc:38:5a:88
	2	0			644698049.961927	39.876140612905	-75.2430526829481	b4:5d:50:a1:87:c2
	3	0			639491122.810438	39.8760425996681	-75.2434106231567	b4:5d:50:a1:57:11
	98	0			636595794.838472	32.8977520649353	-97.0356509734844	48:4a:e9:1:51:1
	24	0			636595800.481575	32.8977520649353	-97.0356509734844	48:4a:e9:1:51:11
	4	0			636151490.485228	32.9022555891077	-97.0422780745731	b4:5d:50:a3:5a:11
	2	0			635386647.701192	40.0586827513604	-75.7726897778305	c8:a7:a:8a:c2:6
	106	0		1	653352797.265364	40.0909931692407	-75.6426214978685	38:94:ed:52:dd:8
	1744	0	1		653348975.240044	40.0909737124505	-75.6426243353494	38:94:ed:52:dd:9



At the time of writing this, most commercial tools did not parse this information.





Apple Maps on iOS versions has changed over time. Up until iOS 7, the History.mapsdata file was leveraged for Apple Maps artifacts. This file was located at /private/var/mobile/Containers/Data/Application/<Apple_Maps_GUID>/Library/Maps/History.mapsdata.

For iOS 8 through iOS 11, the updated Apple Maps storage file was GeoHistory.mapsdata. This file is located in /private/var/mobile/Containers/Data/Application/<Apple_Maps_GUID>/Library/Maps/GeoHistory.mapsdata. Notice the top image contains both GeoHistory.mapsdata and History.mapsdata; this is because the user upgraded to iOS 11 from iOS 7. Even a gradual upgrade will show these artifacts. Finding traces like this is helpful to profile how long the device user has been using iOS.

There were inconsistencies in iOS10 and *sometimes* the GeoHistory.mapsdata file was present, sometimes it was used by Apple Maps and often it was there and just not leveraged. This left us wondering where the artifacts were being stored. Everything depended on how iOS 10 ended up on the device. Was it purchased with that version? Upgraded from a previous version? Or possibly a jailbroken device? For more details on this, read the blog by Heather Mahalik on "How the Grinch Stole Apple Maps".[1]

Since iOS 10, we may only be able to recover Apple Maps from iCloud. And this may not be pulled via an iCloud backup, but instead by accessing the syncing data in iCloud. These methods will be shown in Section 4. Also, the GeoHistory.mapsdata may be missing from any iOS image running 11.2.6 or later. Read the blog article by Heather Mahalik called "First the Grinch and Now the Easter Bunny! Where Is Apple Maps Hiding?"[2] To make matters even more confusing, GeoHistory.mapsdata is missing completely in iOS 12.

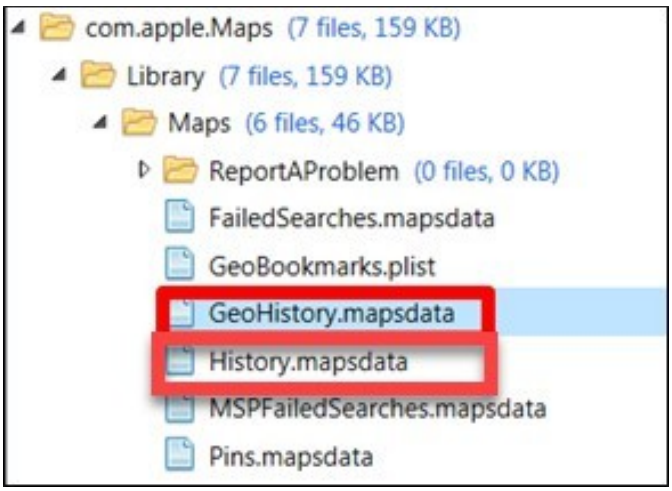
iOS 14 made another change which is still present in iOS 15, and a new file was introduced. The /private/var/mobile/Containers/Shared/AppGroup/group.com.apple.Maps/Maps/MapsSync_0.0.1 was introduced, (NOTE, there may be various numbers at the end of the filename) and at the time of writing this material, it was not supported by any tools. Thus, we created a query for you and Adrian Leong wrote a script.[3] Adrian's script was based on Heather Mahalik's initial research on iOS 14.[4]

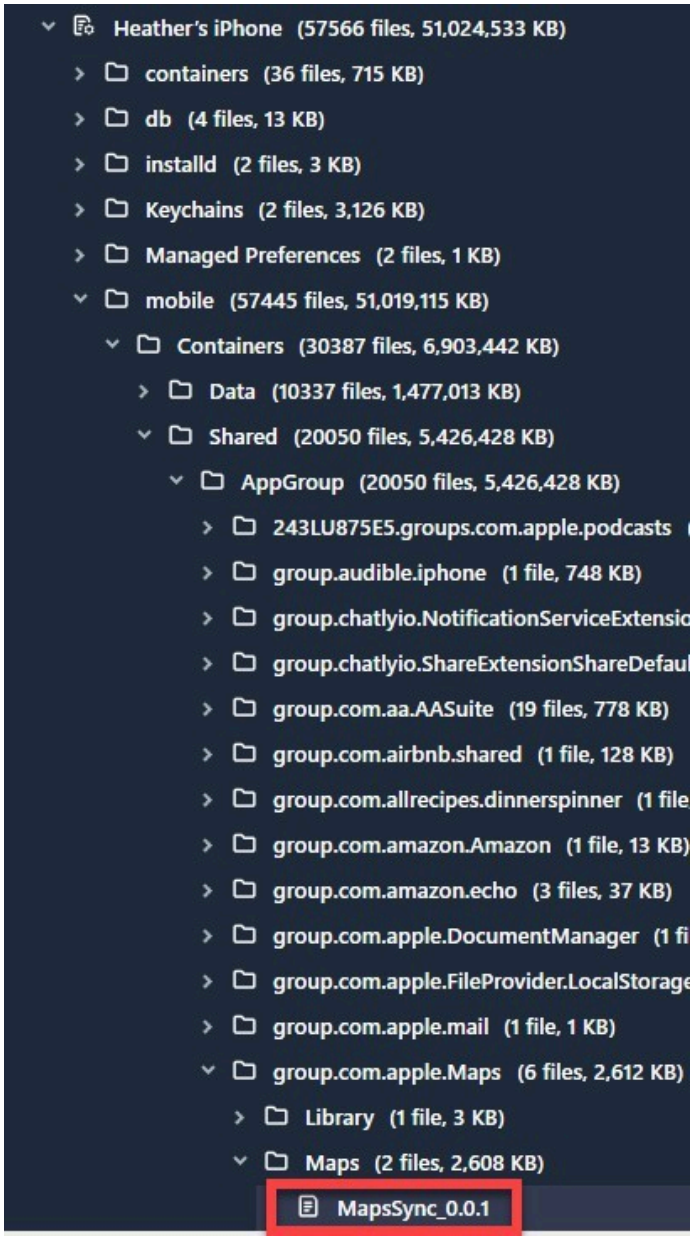
Bottom line, don’t assume you cannot parse Apple Maps data. You have to know where to look for it. Please refer to your cheat sheet and class poster to remember these key files.

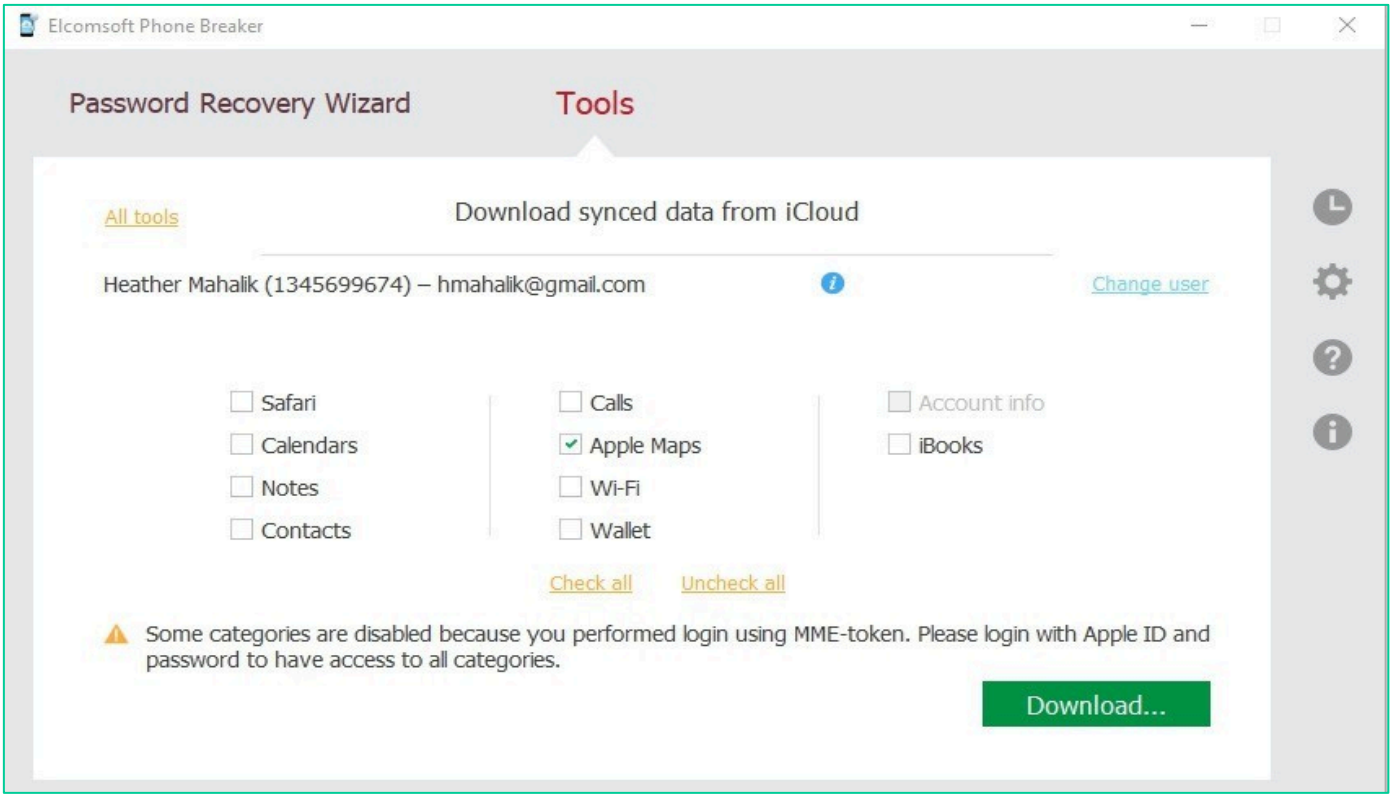
References:

[1] https://www.for585.com/grinch (iOS 10 blog)
[2] https://www.for585.com/easter (iOS 12 blog)
[3] https://for585.com/monkey14 (iOS 14 script)
[4] https://for585.com/ios14blog (iOS 14 blog)

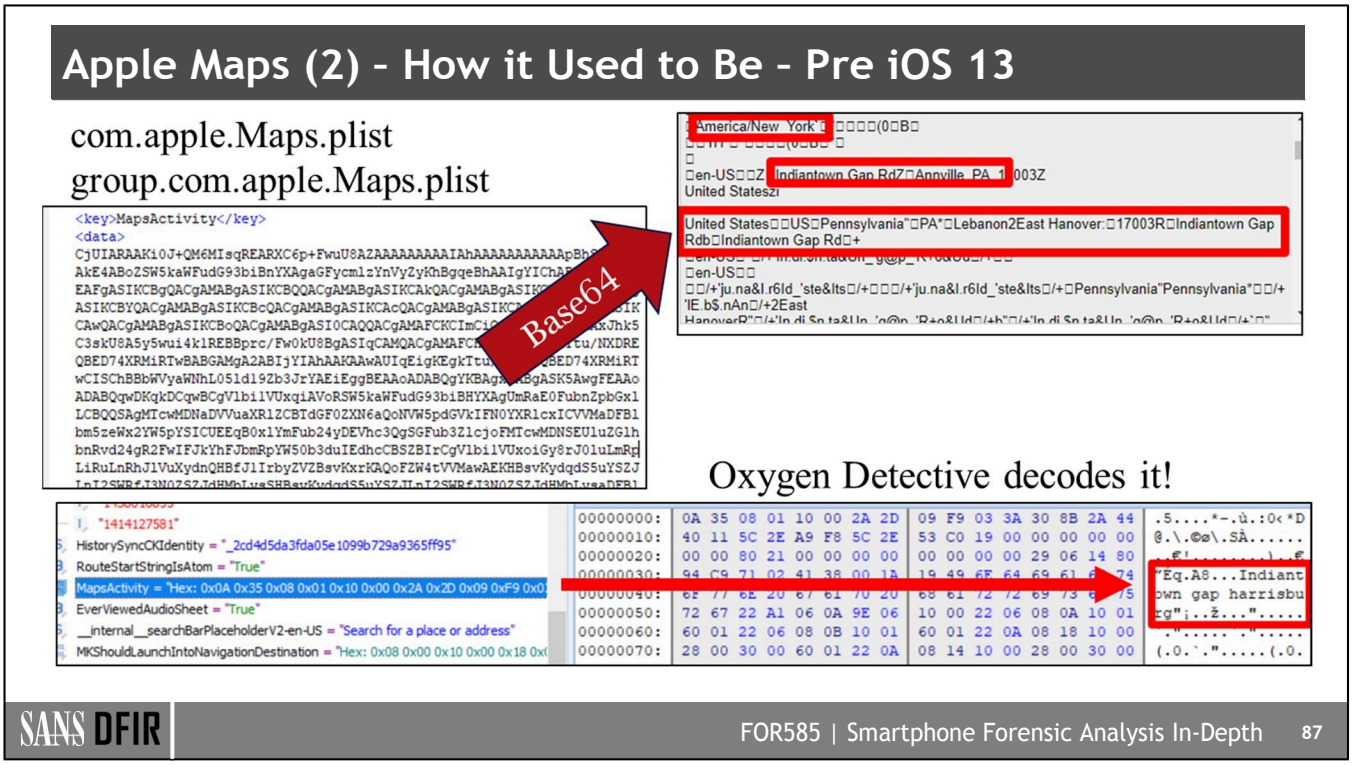
.


com.apple.Maps (7 files, 159 KB)
Library (7 files, 159 KB)
Maps (6 files, 46 KB)
ReportAProblem (0 files, 0 KB)
FailedSearches.mapsdata
GeoBookmarks.plist
GeoHistory.mapsdata
History.mapsdata
MSPFailedSearches.mapsdata
Pins.mapsdata


Heather's iPhone (57566 files, 51,024,533 KB)
containers (36 files, 715 KB)
db (4 files, 13 KB)
installd (2 files, 3 KB)
Keychains (2 files, 3,126 KB)
Managed Preferences (2 files, 1 KB)
mobile (57445 files, 51,019,115 KB)
Containers (30387 files, 6,903,442 KB)
Data (10337 files, 1,477,013 KB)
Shared (20050 files, 5,426,428 KB)
AppGroup (20050 files, 5,426,428 KB)
243LU875E5.groups.com.apple.podcasts
group.audible.iphone (1 file, 748 KB)
group.chatlyio.NotificationServiceExtensio
group.chatlyio.ShareExtensionShareDefaul
group.com.aa.AASuite (19 files, 778 KB)
group.com.airbnb.shared (1 file, 128 KB)
group.com.allrecipes.dinnerspinner (1 file
group.com.amazon.Amazon (1 file, 13 KB)
group.com.amazon.echo (3 files, 37 KB)
group.com.apple.DocumentManager (1 fi
group.com.apple.FileProvider.LocalStorage
group.com.apple.mail (1 file, 1 KB)
group.com.apple.Maps (6 files, 2,612 KB)
Library (1 file, 3 KB)
Maps (2 files, 2,608 KB)
MapsSync_0.0.1

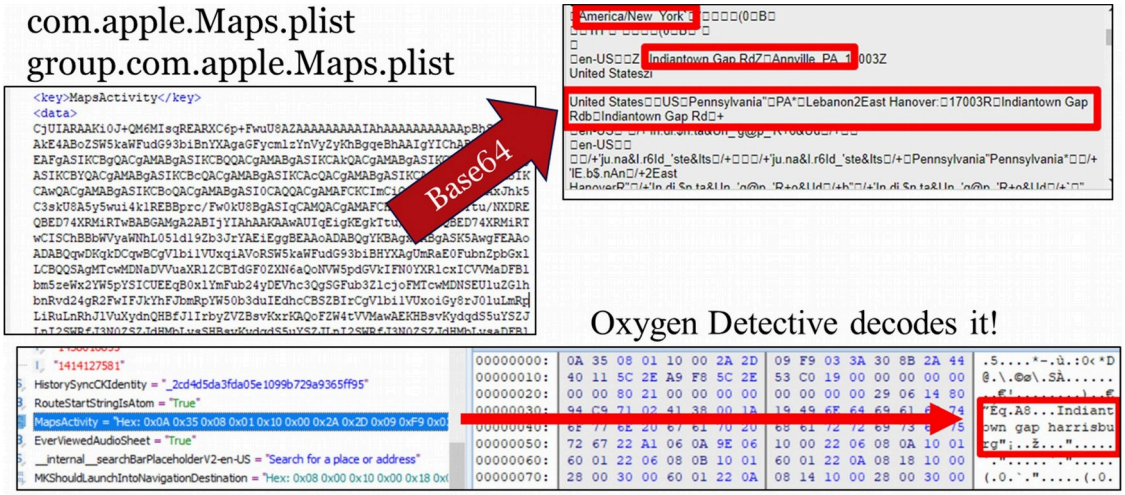


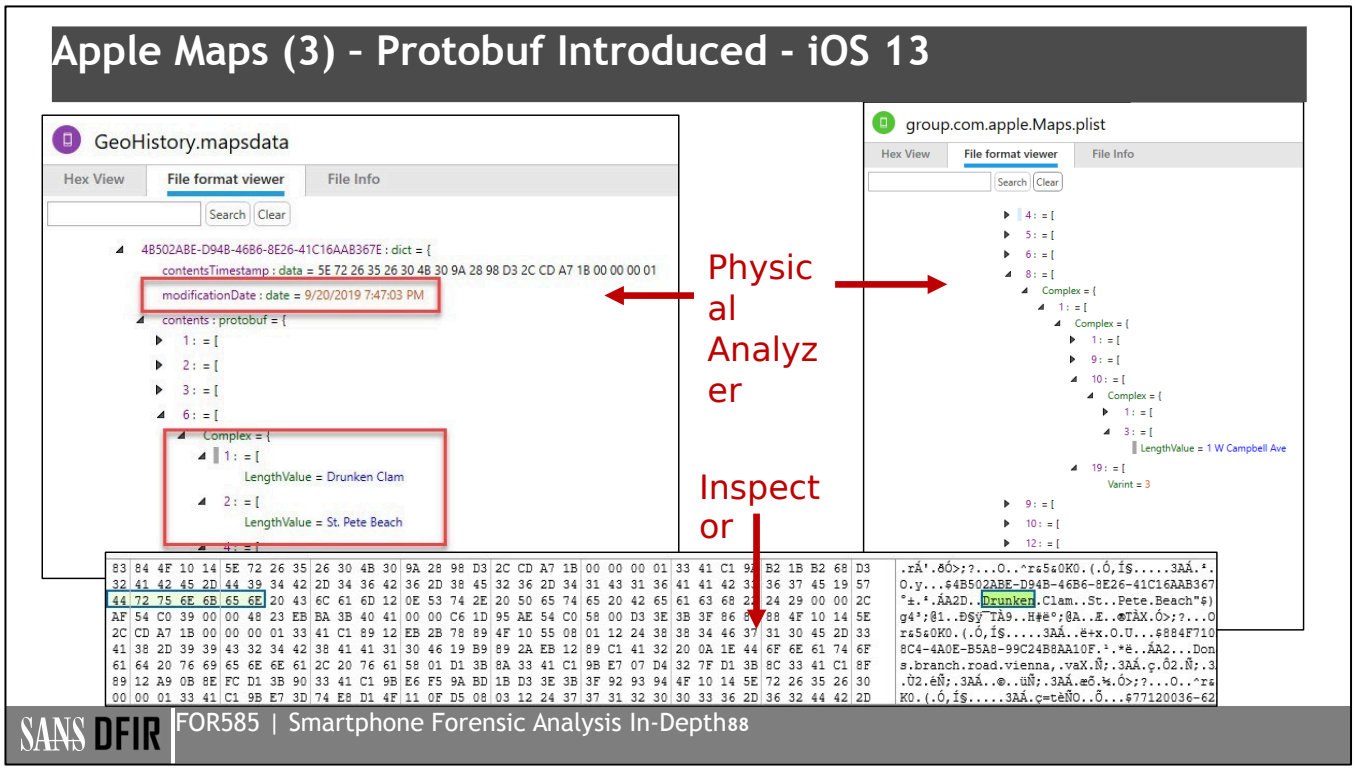
Elcomsoft Phone Breaker, which will be covered in more detail in Section 4, has the capability to pull data synced with iCloud. This includes Apple Maps! This may be your only chance to recover these artifacts, as they are the current searches conducted in Apple M on on this tool, stay tuned for section 4.



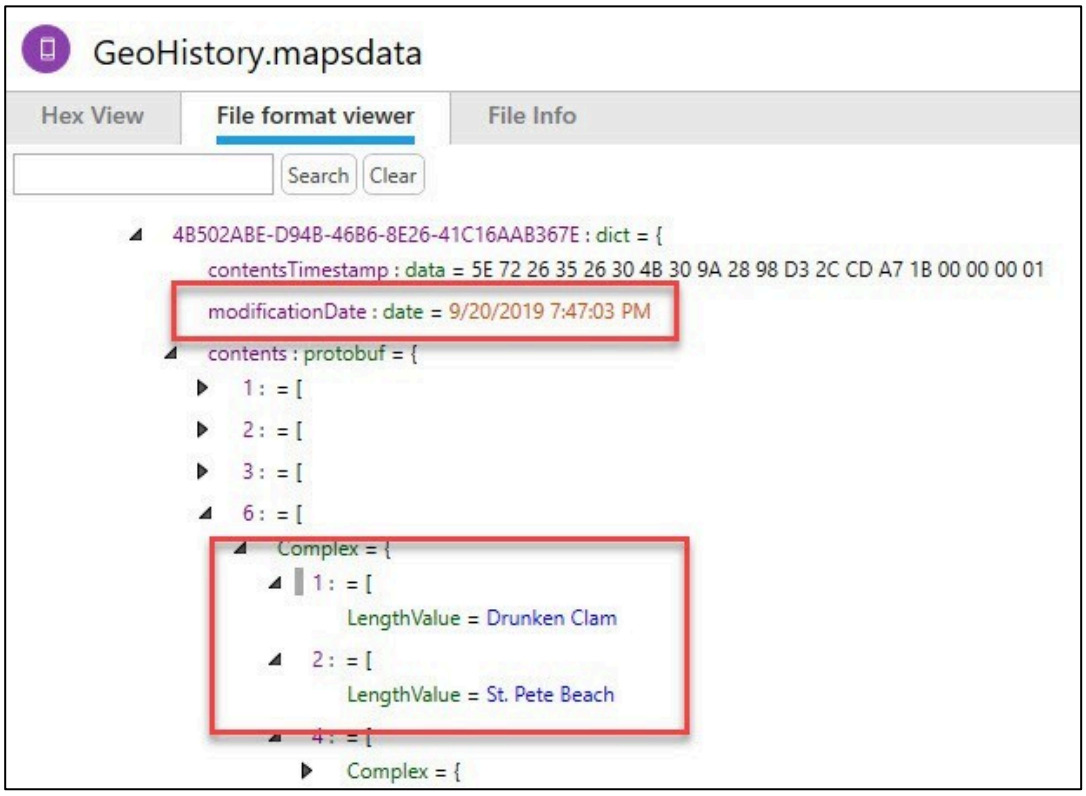
Prior to iOS 13, the good news is that regardless of your access to the Apple Maps database, the com.apple.Maps.plist and group.com.apple.Maps.plist still contain accessible information we can use to associate to the usage of Apple Maps. This file is located at ~/Applications/com.apple.Maps/Library/Preferences/com.apple.Maps.plist. Here we can, at a minimum, determine the last time Apple Maps was used on the device. Another good file to look for is ~/Applications/group.com.apple.Maps/Library/Preferences/group.com.apple.Maps.plist.

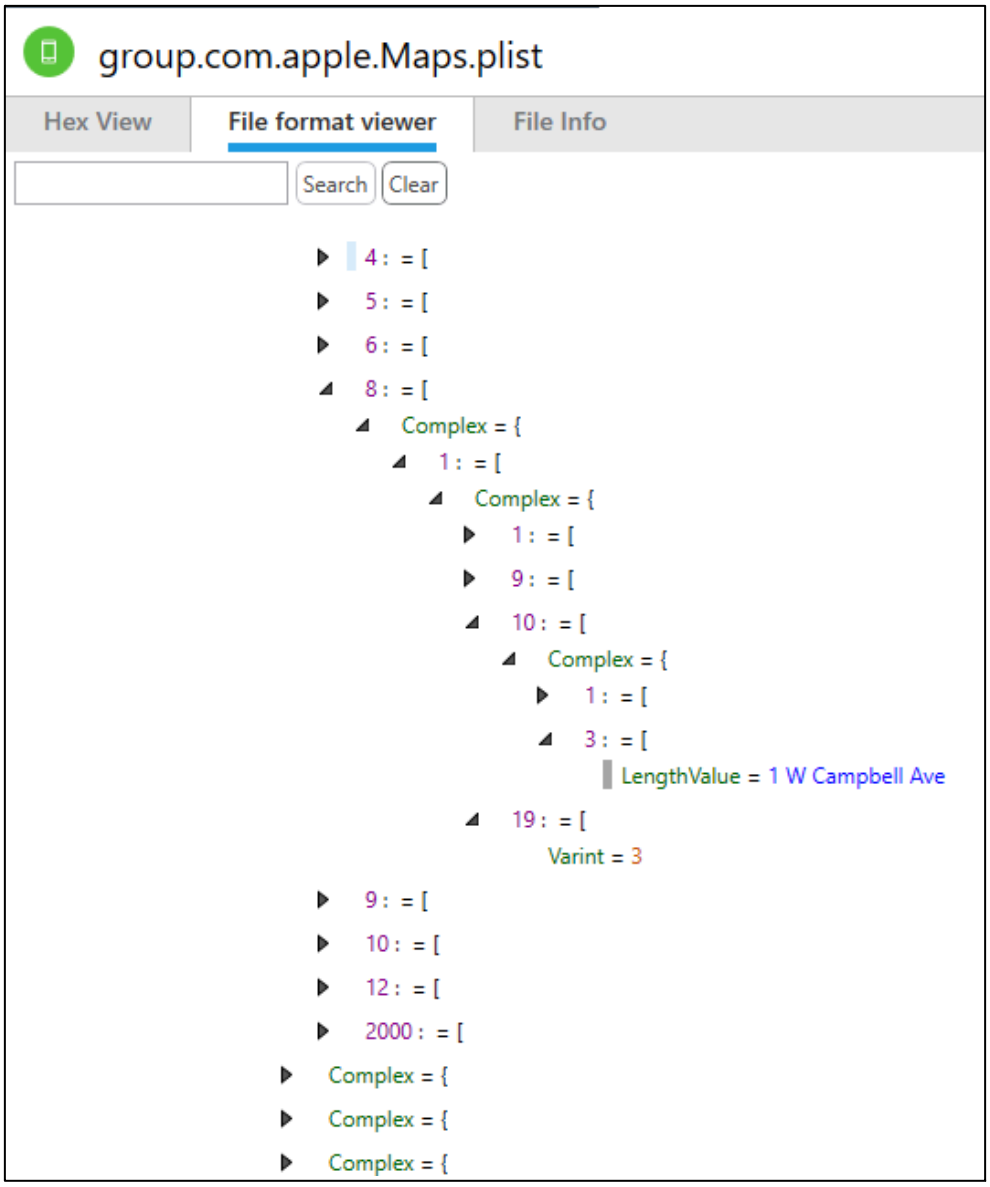
Apple Maps makes good use of com.apple.Maps.plist and group.com.apple.Maps.plist and will store the last coordinates used by Apple Maps, as well as the direction searches, Maps activity, last usage time, and more. The screenshot on the top left shows the native plist being examined in XML view. Here we can see MapsActivity in what appears to be Base64 encoding. An online Base64 decoder was used to reveal the time zone and searches in Maps. In the bottom screenshot, we can see that these values are hidden in "Hex" in the plist and can be converted internally in Oxygen. Now you have two solid options for examining Apple Maps plists.



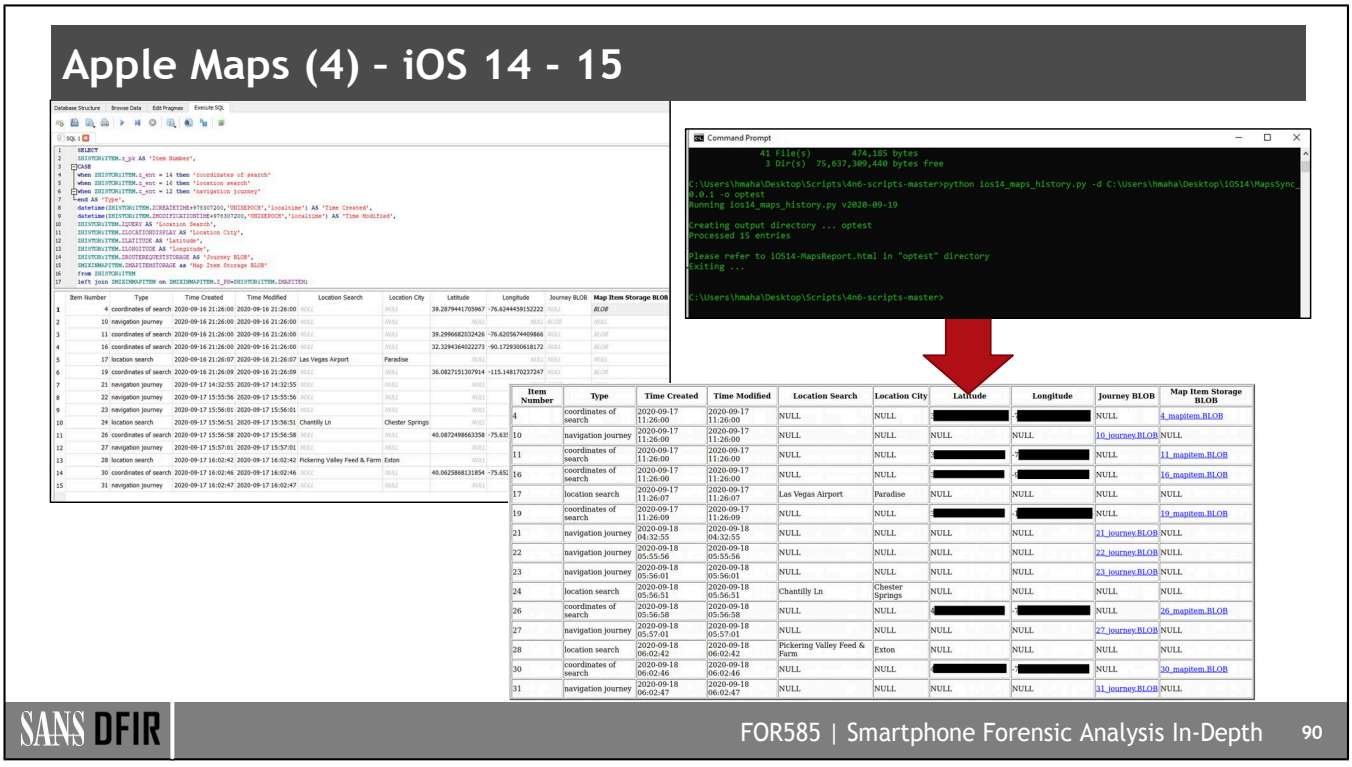


At the time of writing this, there weren't many tools that supported parsing Protobufs. With the introduction of iOS 13, we realized that the Apple Maps data was being stored in protocol buffers aka Protobufs. Yes, this isn't a recent discovery, yet many tools struggle with the parsing of this type of data. While Cellebrite parses the two primary Protobufs shown above, there are so many left untouched. We recommend using protoc to parse the unsupported ones.






```
83 84 4F 10 14 5E 72 26 35 26 30 4B 30 9A 28 98 D3 2C CD A7 1B 00 00 00 01 33 41 C1 9A B2 1B B2 68 D3   .rÁ'.8Ó>;?...O..^r&5&0K0.(.Ó,Í§.....3AÁ.².h
32 41 42 45 2D 44 39 34 42 2D 34 36 42 36 2D 38 45 32 36 2D 34 31 43 31 36 41 41 42 33 36 37 45 19 57   O.y...$4B502ABE-D94B-46B6-8E26-41C16AAB367E.W
44 72 75 6E 6B 65 6E 20 43 6C 61 6D 12 0E 53 74 2E 20 50 65 74 65 20 42 65 61 63 68 22 24 29 00 00 2C   °±.².ÁA2D..Drunken.Clam..St..Pete.Beach"$)..,
AF 54 C0 39 00 00 48 23 EB BA 3B 40 41 00 00 C6 1D 95 AE 54 C0 58 00 D3 3E 3B 3F 86 87 88 4F 10 14 5E   g4³;@1..Đ§ÿ¯TÀ9..H#ë°;@A..Æ..®TÀX.Ó>;?...O..^
2C CD A7 1B 00 00 00 01 33 41 C1 89 12 EB 2B 78 89 4F 10 55 08 01 12 24 38 38 34 46 37 31 30 45 2D 33   r&5&0K0.(.Ó,Í§.....3AÁ..ë+x.O.U...$884F710E-3
41 38 2D 39 39 43 32 34 42 38 41 41 31 30 46 19 B9 89 2A EB 12 89 C1 41 32 20 0A 1E 44 6F 6E 61 74 6F   8C4-4A0E-B5A8-99C24B8AA10F.¹.*ë..ÁA2...Donato
61 64 20 76 69 65 6E 6E 61 2C 20 76 61 58 01 D1 3B 8A 33 41 C1 9B E7 07 D4 32 7F D1 3B 8C 33 41 C1 8F   s.branch.road.vienna,.vaX.Ñ;.3AÁ.ç.Ô2.Ñ;.3AÁ.
89 12 A9 0B 8E FC D1 3B 90 33 41 C1 9B E6 F5 9A BD 1B D3 3E 3B 3F 92 93 94 4F 10 14 5E 72 26 35 26 30   .Ù2.éÑ;.3AÁ..©..üÑ;.3AÁ.æõ.¼.Ó>;?...O..^r&5&0
00 00 01 33 41 C1 9B E7 3D 74 E8 D1 4F 11 0F D5 08 03 12 24 37 37 31 32 30 30 33 36 2D 36 32 44 42 2D   K0.(.Ó,Í§.....3AÁ.ç=tèÑO..Õ...$77120036-62DB-
```



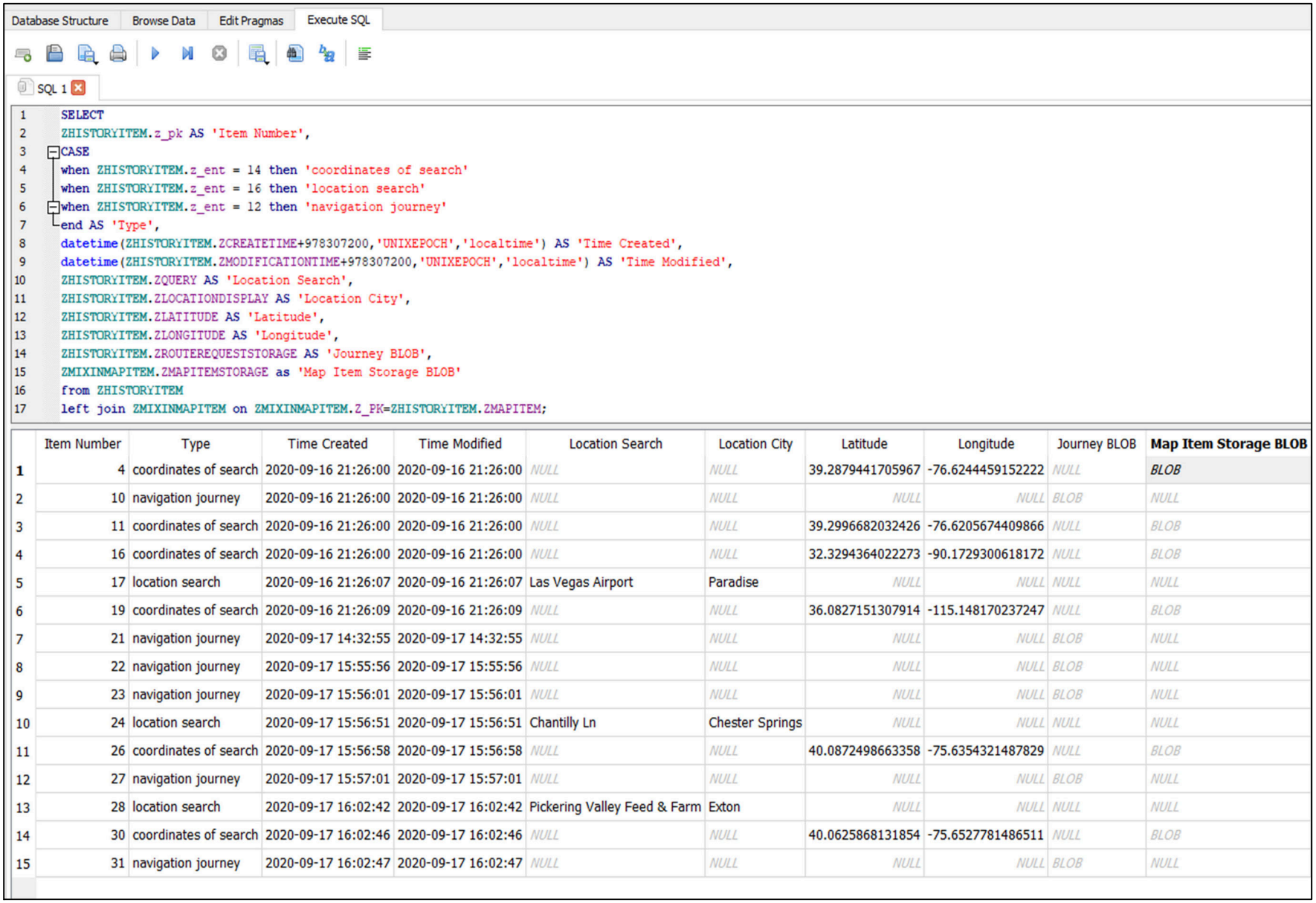
Item Number	Type	Time Created	Time Modified	Location Search	Location City	Latitude	Longitude	Journey BLOB	Map Item Storage BLOB
4	coordinates of search	2020-09-17 11:26:00	2020-09-17 11:26:00	NULL	NULL	[redacted]	[redacted]	NULL	4_mapitem.BLOB
10	navigation journey	2020-09-17 11:26:00	2020-09-17 11:26:00	NULL	NULL	NULL	NULL	10_journey.BLOB	NULL
11	coordinates of search	2020-09-17 11:26:00	2020-09-17 11:26:00	NULL	NULL	[redacted]	[redacted]	NULL	11_mapitem.BLOB
16	coordinates of search	2020-09-17 11:26:00	2020-09-17 11:26:00	NULL	NULL	[redacted]	[redacted]	NULL	16_mapitem.BLOB
17	location search	2020-09-17 11:26:07	2020-09-17 11:26:07	Las Vegas Airport	Paradise	NULL	NULL	NULL	NULL
19	coordinates of search	2020-09-17 11:26:09	2020-09-17 11:26:09	NULL	NULL	[redacted]	[redacted]	NULL	19_mapitem.BLOB
21	navigation journey	2020-09-18 04:32:55	2020-09-18 04:32:55	NULL	NULL	NULL	NULL	21_journey.BLOB	NULL
22	navigation journey	2020-09-18 05:55:56	2020-09-18 05:55:56	NULL	NULL	NULL	NULL	22_journey.BLOB	NULL
23	navigation journey	2020-09-18 05:56:01	2020-09-18 05:56:01	NULL	NULL	NULL	NULL	23_journey.BLOB	NULL
24	location search	2020-09-18 05:56:51	2020-09-18 05:56:51	Chantilly Ln	Chester Springs	NULL	NULL	NULL	NULL
26	coordinates of search	2020-09-18 05:56:58	2020-09-18 05:56:58	NULL	NULL	[redacted]	[redacted]	NULL	26_mapitem.BLOB
27	navigation journey	2020-09-18 05:57:01	2020-09-18 05:57:01	NULL	NULL	NULL	NULL	27_journey.BLOB	NULL
28	location search	2020-09-18 06:02:42	2020-09-18 06:02:42	Pickering Valley Feed & Farm	Exton	NULL	NULL	NULL	NULL
30	coordinates of search	2020-09-18 06:02:46	2020-09-18 06:02:46	NULL	NULL	[redacted]	[redacted]	NULL	30_mapitem.BLOB
31	navigation journey	2020-09-18 06:02:47	2020-09-18 06:02:47	NULL	NULL	NULL	NULL	31_journey.BLOB	NULL

As discussed, iOS 14 brought on the new file MapsSync_0.0.1. This database stores 15 entries and many are different representations of the same location artifact. In this database, you will find a column called **Type** that stores "coordinates of search", "navigation journey", and "location search". This database is transactional in nature, which means the first written data is overwritten when a new search is created. At most, you will have the last five searches. We are hopeful that a cloud extraction will pull additional artifacts. This file has not changed in iOS 15.

Heather Mahalik wrote a query that is available in your class notebook: https://for585.com/notebook and Adrien Leong wrote a script that implements Heather's query into HTML format and exports the BLOBS.[1]

Reference:

[1] https://for585.com/monkey14 (iOS 14 script)

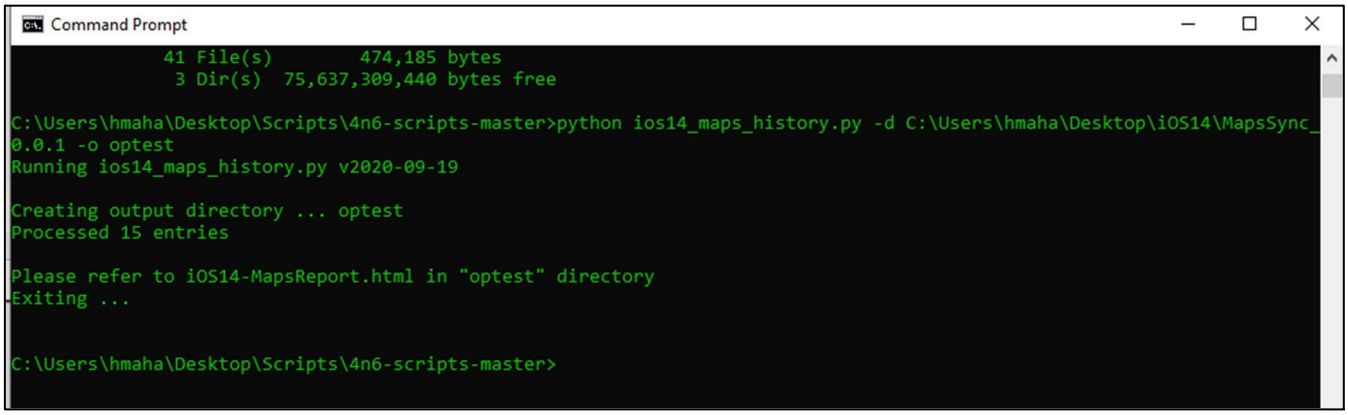


Database Structure | Browse Data | Edit Pragmas | Execute SQL

SQL 1

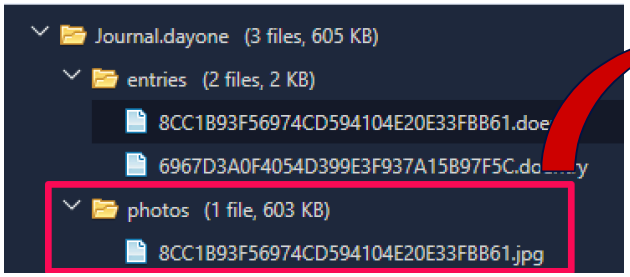
```
SELECT
ZHISTORYITEM.z_pk AS 'Item Number',
CASE
 when ZHISTORYITEM.z_ent = 14 then 'coordinates of search'
 when ZHISTORYITEM.z_ent = 16 then 'location search'
when ZHISTORYITEM.z_ent = 12 then 'navigation journey'
 end AS 'Type',
 datetime(ZHISTORYITEM.ZCREATETIME+978307200,'UNIXEPOCH','localtime') AS 'Time Created',
 datetime(ZHISTORYITEM.ZMODIFICATIONTIME+978307200,'UNIXEPOCH','localtime') AS 'Time Modified',
 ZHISTORYITEM.ZQUERY AS 'Location Search',
 ZHISTORYITEM.ZLOCATIONDISPLAY AS 'Location City',
 ZHISTORYITEM.ZLATITUDE AS 'Latitude',
 ZHISTORYITEM.ZLONGITUDE AS 'Longitude',
 ZHISTORYITEM.ZROUTEREQUESTSTORAGE AS 'Journey BLOB',
 ZMIXINMAPITEM.ZMAPITEMSTORAGE as 'Map Item Storage BLOB'
 from ZHISTORYITEM
 left join ZMIXINMAPITEM on ZMIXINMAPITEM.Z_PK=ZHISTORYITEM.ZMAPITEM;
```

	Item Number	Type	Time Created	Time Modified	Location Search	Location City	Latitude	Longitude	Journey BLOB	Map Item Storage BLOB
1	4	coordinates of search	2020-09-16 21:26:00	2020-09-16 21:26:00	NULL	NULL	39.2879441705967	-76.6244459152222	NULL	BLOB
2	10	navigation journey	2020-09-16 21:26:00	2020-09-16 21:26:00	NULL	NULL	NULL	NULL	BLOB	NULL
3	11	coordinates of search	2020-09-16 21:26:00	2020-09-16 21:26:00	NULL	NULL	39.2996682032426	-76.6205674409866	NULL	BLOB
4	16	coordinates of search	2020-09-16 21:26:00	2020-09-16 21:26:00	NULL	NULL	32.3294364022273	-90.1729300618172	NULL	BLOB
5	17	location search	2020-09-16 21:26:07	2020-09-16 21:26:07	Las Vegas Airport	Paradise	NULL	NULL	NULL	NULL
6	19	coordinates of search	2020-09-16 21:26:09	2020-09-16 21:26:09	NULL	NULL	36.0827151307914	-115.148170237247	NULL	BLOB
7	21	navigation journey	2020-09-17 14:32:55	2020-09-17 14:32:55	NULL	NULL	NULL	NULL	BLOB	NULL
8	22	navigation journey	2020-09-17 15:55:56	2020-09-17 15:55:56	NULL	NULL	NULL	NULL	BLOB	NULL
9	23	navigation journey	2020-09-17 15:56:01	2020-09-17 15:56:01	NULL	NULL	NULL	NULL	BLOB	NULL
10	24	location search	2020-09-17 15:56:51	2020-09-17 15:56:51	Chantilly Ln	Chester Springs	NULL	NULL	NULL	NULL
11	26	coordinates of search	2020-09-17 15:56:58	2020-09-17 15:56:58	NULL	NULL	40.0872498663358	-75.6354321487829	NULL	BLOB
12	27	navigation journey	2020-09-17 15:57:01	2020-09-17 15:57:01	NULL	NULL	NULL	NULL	BLOB	NULL
13	28	location search	2020-09-17 16:02:42	2020-09-17 16:02:42	Pickering Valley Feed & Farm	Exton	NULL	NULL	NULL	NULL
14	30	coordinates of search	2020-09-17 16:02:46	2020-09-17 16:02:46	NULL	NULL	40.0625868131854	-75.6527781486511	NULL	BLOB
15	31	navigation journey	2020-09-17 16:02:47	2020-09-17 16:02:47	NULL	NULL	NULL	NULL	BLOB	NULL



Item Number	Type	Time Created	Time Modified	Location Search	Location City	Latitude	Longitude	Journey BLOB	Map Item Storage BLOB
4	coordinates of search	2020-09-17 11:26:00	2020-09-17 11:26:00	NULL	NULL	[redacted]	[redacted]	NULL	4_mapitem.BLOB
10	navigation journey	2020-09-17 11:26:00	2020-09-17 11:26:00	NULL	NULL	NULL	NULL	10_journey.BLOB	NULL
11	coordinates of search	2020-09-17 11:26:00	2020-09-17 11:26:00	NULL	NULL	[redacted]	[redacted]	NULL	11_mapitem.BLOB
16	coordinates of search	2020-09-17 11:26:00	2020-09-17 11:26:00	NULL	NULL	[redacted]	[redacted]	NULL	16_mapitem.BLOB
17	location search	2020-09-17 11:26:07	2020-09-17 11:26:07	Las Vegas Airport	Paradise	NULL	NULL	NULL	NULL
19	coordinates of search	2020-09-17 11:26:09	2020-09-17 11:26:09	NULL	NULL	[redacted]	[redacted]	NULL	19_mapitem.BLOB
21	navigation journey	2020-09-18 04:32:55	2020-09-18 04:32:55	NULL	NULL	NULL	NULL	21_journey.BLOB	NULL
22	navigation journey	2020-09-18 05:55:56	2020-09-18 05:55:56	NULL	NULL	NULL	NULL	22_journey.BLOB	NULL
23	navigation journey	2020-09-18 05:56:01	2020-09-18 05:56:01	NULL	NULL	NULL	NULL	23_journey.BLOB	NULL
24	location search	2020-09-18 05:56:51	2020-09-18 05:56:51	Chantilly Ln	Chester Springs	NULL	NULL	NULL	NULL
26	coordinates of search	2020-09-18 05:56:58	2020-09-18 05:56:58	NULL	NULL	[redacted]	[redacted]	NULL	26_mapitem.BLOB
27	navigation journey	2020-09-18 05:57:01	2020-09-18 05:57:01	NULL	NULL	NULL	NULL	27_journey.BLOB	NULL
28	location search	2020-09-18 06:02:42	2020-09-18 06:02:42	Pickering Valley Feed & Farm	Exton	NULL	NULL	NULL	NULL
30	coordinates of search	2020-09-18 06:02:46	2020-09-18 06:02:46	NULL	NULL	[redacted]	[redacted]	NULL	30_mapitem.BLOB
31	navigation journey	2020-09-18 06:02:47	2020-09-18 06:02:47	NULL	NULL	NULL	NULL	31_journey.BLOB	NULL

Third-Party Application Location Artifacts (1)



When your app asks if you want to include location information and you say "no," what happens?

Third-party applications leave location artifacts in places one may not expect. While each application is different, a good example is the Day One application. This application is essentially a multimedia journal application that enables the user to post journal entries with attached photos and videos. I downloaded the application, created two journal entries, and attached a photo. The first journal entry included the text "first tomatoes from my garden." I took a photo of the tomatoes in the application and was prompted to include my location in the photo. I selected to NOT save the location. Do you think the application respected my wishes?

When we examine the EXIF associated with the application, we will NOT see coordinates containing location information. But what happens when your phone thinks? When your phone suggests something or uses its brain, that data must be stored somewhere.

Third-Party Application Location Artifacts (2)

docid	c0entry_id	c1text	c2modified_date
1	8CC1B93F56974CD594104E20E33FBB61	First tomatoes from my garden!	1373325781
2	6967D3A0F4054D399E3F937A15B97F5C	Test	


```
<?xml version="1.0" encoding="UTF-8"?>.<!DOCTYP
E plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "h
ttp://www.apple.com/DTDs/PropertyList-1.0.dtd">
.<plist version="1.0">.<dict>..<key>Creation Da
te</key>..<date>2013-07-08T23:22:35Z</date>..<k
ey>Entry Text</key>..<string>First tomatoes fro
m my garden!</string>..<key>Location</key>..<di
ct>...<key>Administrative Area</key>...<string>
Virginia</string>...<key>Country</key>...<strin
g>United States</string>...<key>Latitude</key>.
..<real>38.897663774005039</real>...<key>Locali
ty</key>...<string>Dunn Loring</string>...<key>
Longitude</key>...<real>-77.240605317128114</re
al>...<key>Place Name</key>...<string>8521 Mine
rva Ct</string>..</dict>..<key>Starred</key>..<
true/>..<key>Time Zone</key>..<string>America/N
ew_York</string>..<key>UUID</key>..<string>8CC1
B93F56974CD594104E20E33FBB61</string>..<key>Wea
ther</key>..<dict>...<key>Celsius</key>...<stri
ng>29</string>...<key>Description</key>...<stri
ng>Partly Cloudy</string>...<key>Fahrenheit</ke
y>...<string>84</string>...<key>IconName</key>.
..<string>pcloudy.png</string>..</dict>.</dict>
```


View in Hex!

Here we are looking at the Journal.sqlite, which stores the journal entries for the Day One application. The database appears to look normal, but if you look closely, you will see a binary plist and zlocation column. This binary plist contains all of the "thoughts" your phone had when it asked if you wanted to include your location information. In the screenshot above, we can see the binary plist from the zlocation in Hex. Notice that we see time zone, creation date, UUID, latitude, longitude, Place Name (home address for this user), and more! Even the weather is documented even though I said "no" to including location artifacts.

This only happened because location services was enabled on the device. If location services for the device was not enabled, this would not have happened, as the user would have to turn on the settings.

docid	c0entry_id	c1text	c2modified_date
1	8CC1B93F56974CD594104E20E33FBB61	First tomatoes from my garden!	1373325781
2	6967D3A0F4054D399E3F937A15B97F5C	Test	1373325858

<?xml version="1.0" encoding="UTF-8"?>.<!DOCTYP
E plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "h
ttp://www.apple.com/DTDs/PropertyList-1.0.dtd">
.<plist version="1.0">.<dict>..<key>Creation Da
te</key>..<date>2013-07-08T23:22:35Z</date>..<k
ey>Entry Text</key>..<string>First tomatoes fro
m my garden!</string>..<key>Location</key>..<di
ct>...<key>Administrative Area</key>...<string>
Virginia</string>...<key>Country</key>...<strin
g>United States</string>...<key>Latitude</key>.
..<real>38.897663774005039</real>...<key>Locali
ty</key>...<string>Dunn Loring</string>...<key>
Longitude</key>...<real>-77.240605317128114</re
al>...<key>Place Name</key>...<string>8521 Mine
rva Ct</string>..</dict>..<key>Starred</key>..<
true/>..<key>Time Zone</key>..<string>America/N
ew_York</string>..<key>UUID</key>..<string>8CC1
B93F56974CD594104E20E33FBB61</string>..<key>Wea
ther</key>..<dict>...<key>Celsius</key>...<stri
ng>29</string>...<key>Description</key>...<stri
ng>Partly Cloudy</string>...<key>Fahrenheit</ke
y>...<string>84</string>...<key>IconName</key>.
..<string>pcloudy.png</string>..</dict>.</dict>

Internet Activity

History

Searched items

- Location information

Bookmarks

Cookies

Images

iOS devices store internet activity in several locations across the device. Although the cookies, web bookmarks, web history, and image files are commonly parsed and decoded by tools like Physical Analyzer, AXIOM, and Oxygen, the database files for each should be examined for more information to include deleted data. We take a closer look at mobile browsing applications in Section 5 of this course.

iOS Safari - Files of Interest

/mobile/Library/Safari/CloudTabs.db

/mobile/Library/Safari/Bookmarks.db

/mobile/Library/Safari/SafariTabs.db

/mobile/Library/Safari/History.db

/mobile/Containers/Data/Application/<GUID>/Library/Safari/Thumbnails/

/mobile/Containers/Data/Application/<GUID>/Library/Favicons/Favicons/

/mobile/Containers/Data/Application/<GUID>/Caches/Cache.db

/mobile/Containers/Data/Application/<GUID>/Library/Preferences/com.apple.mobilesafari.plist

The following files are key files of interest for iOS Safari:

- /mobile/Library/Safari/CloudTabs.db - will only exist if icloud sync is enabled for Safari
- /mobile/Library/Safari/Bookmarks.db
- /mobile/Library/Safari/SafariTabs.db
- /mobile/Library/Safari/History.db
- /mobile/Containers/Data/Application/<GUID>/Library/Safari/Thumbnails/
- /mobile/Containers/Data/Application/<GUID>/Library/Favicons/Favicons/
- /mobile/Containers/Data/Application/<GUID>/Caches/Cache.db
- /mobile/Containers/Data/Application/<GUID>/Library/Preferences/com.apple.mobilesafari.plist

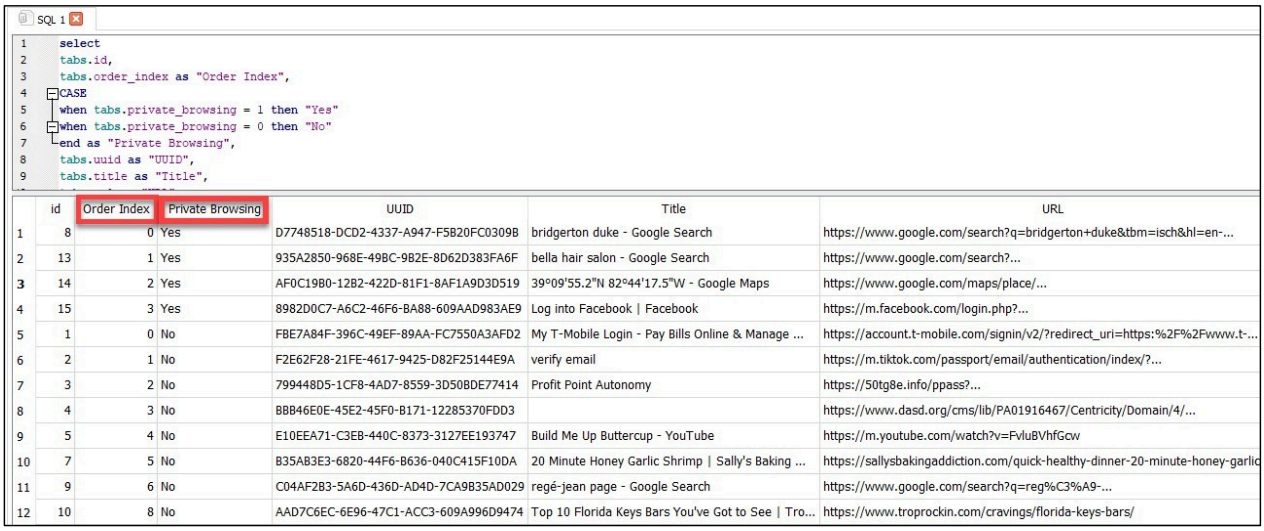
While most of the files are self-explanatory, here are some descriptions that may help you in your investigations.
The database /mobile/Library/Safari/SafariTabs.db

```
SELECT
bookmarks.id as "ID",
bookmarks.parent as "Parent",
bookmarks.title as "Title",
bookmarks.url as "URL",
windows_tab_groups.active_tab_id as "Active Tab ID",
--Active Tab ID is the last active tab or tab being viewed for each group
bookmarks.num_children as "# of Children",
bookmarks.order_index as "Order Index"
from bookmarks
left join bookmark_title_words on bookmark_title_words.bookmark_id=bookmarks.id
left join windows_tab_groups on windows_tab_groups.tab_group_id=bookmarks.id
```

The history database is parsed by most commercial tool. The path is /private/var/mobile/Library/Safari/History.db. This database tracks the searches made using the Safari URL bar in the regular browsing tabs (Private browsing searches are not recorded here).

```
select
history_visits.id,
datetime(history_visits.visit_time + 978307200,'unixepoch') as "Visit Time",
history_visits.title as "Title",
CASE
when history_visits.load_successful = 1 then "Yes"
when history_visits.load_successful = 0 then "No"
end as "Load Successful",
history_items.url as "URL",
history_items.domain_expansion as "Domain Expansion",
history_items.visit_count as "Visit Count"
from history_visits
left join history_items on history_items.id=history_visits.history_item
```

iOS Safari - BrowserState.db



SQL 1

```
select
tabs.id,
tabs.order_index as "Order Index",
CASE
when tabs.private_browsing = 1 then "Yes"
when tabs.private_browsing = 0 then "No"
end as "Private Browsing",
tabs.uuid as "UUID",
tabs.title as "Title",
```

	id	Order Index	Private Browsing	UUID	Title	URL
1	8	0	Yes	D7748518-DCD2-4337-A947-F5B20FC0309B	bridgerton duke - Google Search	https://www.google.com/search?q=bridgerton+duke&tbm=isch&hl=en-...
2	13	1	Yes	935A2850-968E-49BC-9B2E-8D62D383FA6F	bella hair salon - Google Search	https://www.google.com/search?...
3	14	2	Yes	AF0C19B0-12B2-422D-81F1-8AF1A9D3D519	39°09'55.2"N 82°44'17.5"W - Google Maps	https://www.google.com/maps/place/...
4	15	3	Yes	8982D0C7-A6C2-46F6-BA88-609AAD983AE9	Log into Facebook | Facebook	https://m.facebook.com/login.php?...
5	1	0	No	FBE7A84F-396C-49EF-89AA-FC7550A3AFD2	My T-Mobile Login - Pay Bills Online & Manage ...	https://account.t-mobile.com/signin/v2/?redirect_uri=https:%2F%2Fwww.t-...
6	2	1	No	F2E62F28-21FE-4617-9425-D82F25144E9A	verify email	https://m.tiktok.com/passport/email/authentication/index/?...
7	3	2	No	799448D5-1CF8-4AD7-8559-3D50BDE77414	Profit Point Autonomy	https://50tg8e.info/ppass?...
8	4	3	No	BBB46E0E-45E2-45F0-B171-12285370FDD3		https://www.dasd.org/cms/lib/PA01916467/Centricity/Domain/4/...
9	5	4	No	E10EEA71-C3EB-440C-8373-3127EE193747	Build Me Up Buttercup - YouTube	https://m.youtube.com/watch?v=FvluBVhfGcw
10	7	5	No	B35AB3E3-6820-44F6-B636-040C415F10DA	20 Minute Honey Garlic Shrimp | Sally's Baking ...	https://sallysbakingaddiction.com/quick-healthy-dinner-20-minute-honey-garlic
11	9	6	No	C04AF2B3-5A6D-436D-AD4D-7CA9B35AD029	regé-jean page - Google Search	https://www.google.com/search?q=reg%C3%A9-...
12	10	8	No	AAD7C6EC-6E96-47C1-ACC3-609A996D9474	Top 10 Florida Keys Bars You've Got to See | Tro...	https://www.troprockin.com/cravings/florida-keys-bars/

The database in the screenshot above is /private/var/mobile/Library/Safari/BrowserState.db. This database tracks the open tabs within regular and Private browsing. The data within this file can be used to determine how many individual Safari tabs are open, if any of them are Private tabs, and when the tabs were last viewed. If any Private tabs are visible in this database, it means the tabs are still logically visible within the Private tabs within the Safari application. Once the user removes / clears the Private tabs, they are purged from this database without leaving records behind.

The Order Index represents both Private and regular browsing, where 0 is the oldest tab opened and 3 is the newest of the tabs that are still accessible in Private Browsing, as seen in the screenshot. Regular browsing is stored in the same manner. The query below will work well for this database.

```
select
tabs.id,
tabs.order_index as "Order Index",
CASE
when tabs.private_browsing = 1 then "Yes"
when tabs.private_browsing = 0 then "No"
end as "Private Browsing",
tabs.uuid as "UUID",
tabs.title as "Title",
tabs.url as "URL",
tabs.user_visible_url as "User Visible URL",
datetime(tabs.last_viewed_time + 978307200,'unixepoch') as "Last Viewed Time",
CASE
when tabs.opened_from_link = 1 then "Yes"
when tabs.opened_from_link = 0 then "No"
end as "Opened From Link",
tab_sessions.session_data as "Session Data (BLOB)",
tabs.browser_window_uuid as "Browser Window UUID",
tabs.browser_window_id as "Browser Window ID"
from tabs
left join tab_sessions on tab_sessions.tab_uuid=tabs.uuid
order by tabs.last_viewed_time DESC
```

```
SQL 1
select
    tabs.id,
    tabs.order_index as "Order Index",
    CASE
        when tabs.private_browsing = 1 then "Yes"
        when tabs.private_browsing = 0 then "No"
    end as "Private Browsing",
    tabs.uuid as "UUID",
    tabs.title as "Title",
```

	id	Order Index	Private Browsing	UUID	Title	URL
1	8	0	Yes	D7748518-DCD2-4337-A947-F5B20FC0309B	bridgerton duke - Google Search	https://www.google.com/search?q=bridgerton+duke&tbm=isch&hl=en-...
2	13	1	Yes	935A2850-968E-49BC-9B2E-8D62D383FA6F	bella hair salon - Google Search	https://www.google.com/search?...
3	14	2	Yes	AF0C19B0-12B2-422D-81F1-8AF1A9D3D519	39°09'55.2"N 82°44'17.5"W - Google Maps	https://www.google.com/maps/place/...
4	15	3	Yes	8982D0C7-A6C2-46F6-BA88-609AAD983AE9	Log into Facebook | Facebook	https://m.facebook.com/login.php?...
5	1	0	No	FBE7A84F-396C-49EF-89AA-FC7550A3AFD2	My T-Mobile Login - Pay Bills Online & Manage ...	https://account.t-mobile.com/signin/v2/?redirect_uri=https:%2F%2Fwww.t-...
6	2	1	No	F2E62F28-21FE-4617-9425-D82F25144E9A	verify email	https://m.tiktok.com/passport/email/authentication/index/?...
7	3	2	No	799448D5-1CF8-4AD7-8559-3D50BDE77414	Profit Point Autonomy	https://50tg8e.info/ppass?...
8	4	3	No	BBB46E0E-45E2-45F0-B171-12285370FDD3		https://www.dasd.org/cms/lib/PA01916467/Centricity/Domain/4/...
9	5	4	No	E10EEA71-C3EB-440C-8373-3127EE193747	Build Me Up Buttercup - YouTube	https://m.youtube.com/watch?v=FvluBVhfGcw
10	7	5	No	B35AB3E3-6820-44F6-B636-040C415F10DA	20 Minute Honey Garlic Shrimp | Sally's Baking ...	https://sallysbakingaddiction.com/quick-healthy-dinner-20-minute-honey-garlic
11	9	6	No	C04AF2B3-5A6D-436D-AD4D-7CA9B35AD029	regé-jean page - Google Search	https://www.google.com/search?q=reg%C3%A9-...
12	10	8	No	AAD7C6EC-6E96-47C1-ACC3-609A99609474	Top 10 Florida Keys Bars You've Got to See | Tro...	https://www.troprockin.com/cravings/florida-keys-bars/

Table "tab_sessions" has a "session_data" column which contains mostly BLOB's. The BLOBs are binary .bplists with a 4-byte buffer at the beginning. Removing the 4 bytes and printing the .bplist to screen produces additional information about the browsing activity.

iOS Notes

iOS devices store notes in two different files

/mobile/Library/Notes/notes.sqlite

- Third-party app notes
- Includes Google and other apps

/mobile/Containers/Shared/AppGroup/<GUID>/NoteStore.sqlite

- Apple/iCloud notes
- Both those stored natively on the device and those syncing with cloud

iOS devices allow third-party applications to store notes from the application onto the device. These notes are located in /mobile/Library/Notes/notes.sqlite, which is often parsed by the forensic tools. For native notes and iCloud notes, the location is /mobile/Containers/Shared/AppGroup/<GUID>/NoteStore.sqlite. These files will be covered further in a lab in section 4.

Preference Files of Interest

/preferences/SystemConfiguration/com.apple.accounts.exists.plist

/Library/DataAccess/AccountInformation.plist

/Library/Preferences/

- com.apple.locationd.plist: If 0x01, location services is enabled
- com.apple.homesharing.plist: iCloud sharing information
- com.apple.assistant.backedup.plist: iCloud sync settings
- com.apple.coreduetd.plist: Synced devices
- com.apple.NanoRegistry.plist: Apple Watch information
- com.apple.commcenter.plist: Device Info (IMEI, Phone Number, SIM/eSIM, etc.)
- com.apple.MobileSMS.plist: Days to keep SMS
- com.apple.identityservices.idstatuscache.plist: Snippets of contacts, phone numbers, etc.
- com.apple.accountsettings.plist: Email accounts pushed to device

The files listed above are all included on your cheat sheet. We are going to be examining some of these in the labs, and you will find many of them to be useful. Keep an eye out for a blog on Smarterforensics.com on the most helpful plists and why.

- /preferences/SystemConfiguration/com.apple.accounts.exists.plist: Account information
- /Library/DataAccess/AccountInformation.plist: Email Sync information
- /Lockdown/device_values.plist: Tons of info—EXAMINE THIS FILE—Activated State, Bluetooth address, and more
- /Library/Preferences/
  - com.apple.locationd.plist: If 0x01, location services is enabled
  - com.apple.homesharing.plist: iCloud sharing information
  - com.apple.assistant.backedup.plist: iCloud sync settings
  - com.apple.coreduetd.plist: Synced devices
  - com.apple.NanoRegistry.plist: Apple Watch information
  - com.apple.commcenter.plist: Device Info (IMEI, Phone Number, SIM/eSIM, etc.)
  - com.apple.MobileSMS.plist: Days to keep SMS
  - com.apple.identityservices.idstatuscache.plist: Snippets of contacts, phone numbers, etc.
  - com.apple.accountsettings.plist: Email accounts pushed to device

Caution: You may need Method 2 to pull some of these files via Physical Analyzer.

Other Databases & Files of Interest

Recents communication

- /mobile/Library/Recents/Recents

Voicemail

- /mobile/Library/Voicemail/voicemail.db

Passes

- /mobile/Library/Passes/passes23.sqlite

Bluetooth

- /containers/Shared/SystemGroup/<GUID>/Library/Databases/com.apple.MobileBluetooth.ledevices.other.db
- /containers/Shared/SystemGroup/<GUID>/Library/Databases/com.apple.MobileBluetooth.ledevices.paired.db
- /containers/Shared/SystemGroup/<GUID>/Library/Preferences/com.apple.MobileBluetooth.devices.plist

Calendar

- /mobile/Library/Calendar/Calendar.sqlitedb
- /mobile/Library/Calendar/Extras.db
- /mobile/Library/Calendar/Notifications.Calendar.Protected

Accounts

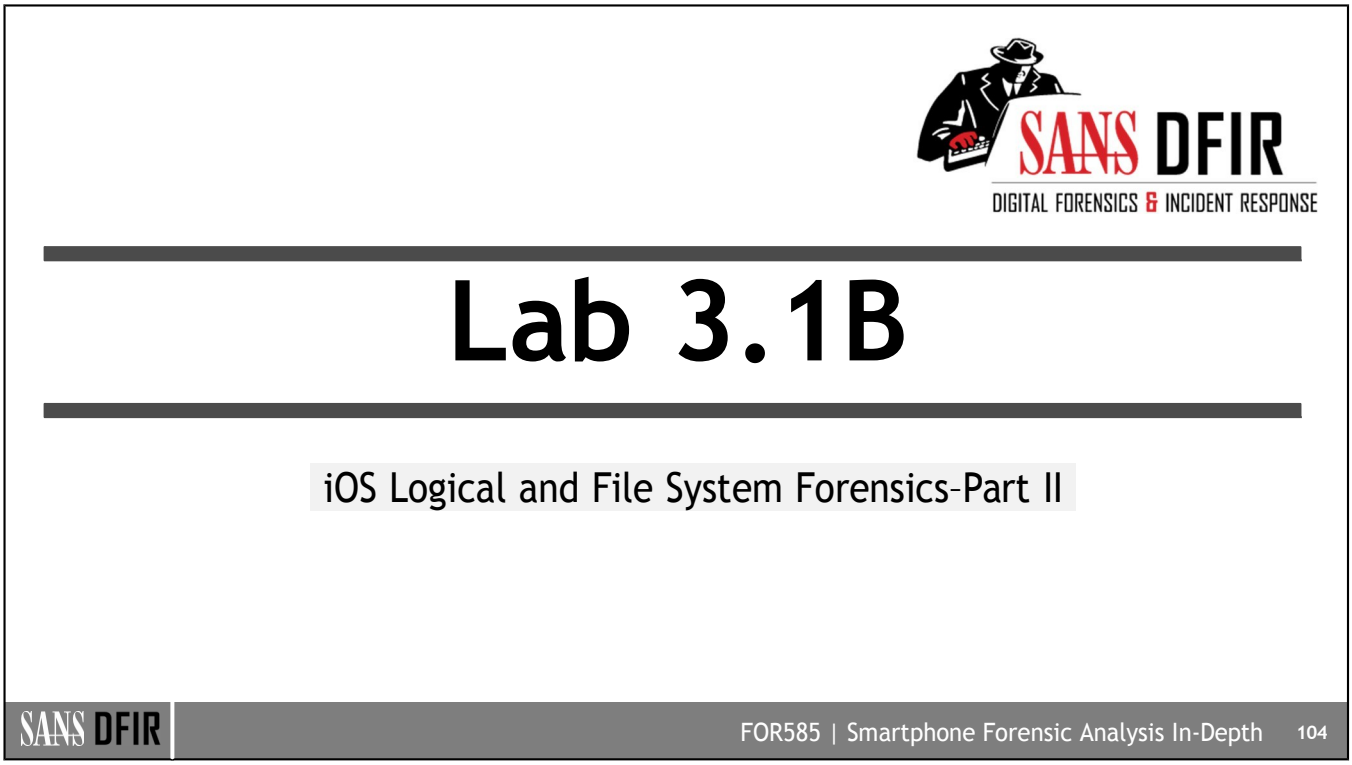
- /mobile/Library/Accounts/Accounts3.sqlite

Aggregated Dictionary

- /mobile/Library/AggregateDictionary/ADDataStore.sqlitedb

The databases and files above are important, and most are self explanatory. These files are on your cheat sheet and should be examined manually in your investigations should your tool of choice not parse them.

The Bluetooth is covered in your labs in this section. A blog worth reading is one written by Heather Mahalik and Matt Goeckel which includes 3 years of research and screenshots from your lab. This blog went through the DFIR Peer Review process and won the Forensic 4:Cast Blog of the Year in 2020: https://dfir.pubpub.org/pub/frknihlg/release/1.



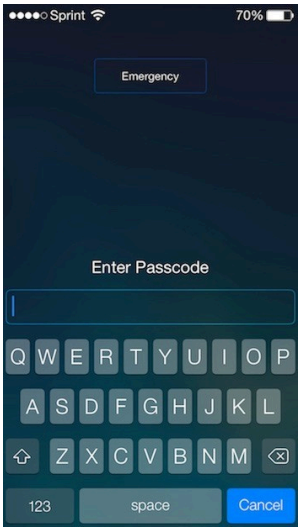
This page intentionally left blank.

iOS Forensics Agenda

- Section 3.1: iOS Forensics Overview
- Section 3.2: iOS Device Acquisition Considerations
- Section 3.3: iOS File System Structures
- Section 3.4: iOS Evidentiary Locations
- Section 3.5: Handling Locked iOS Devices
- Section 3.6: Advanced Decoding and Traces of User Activity

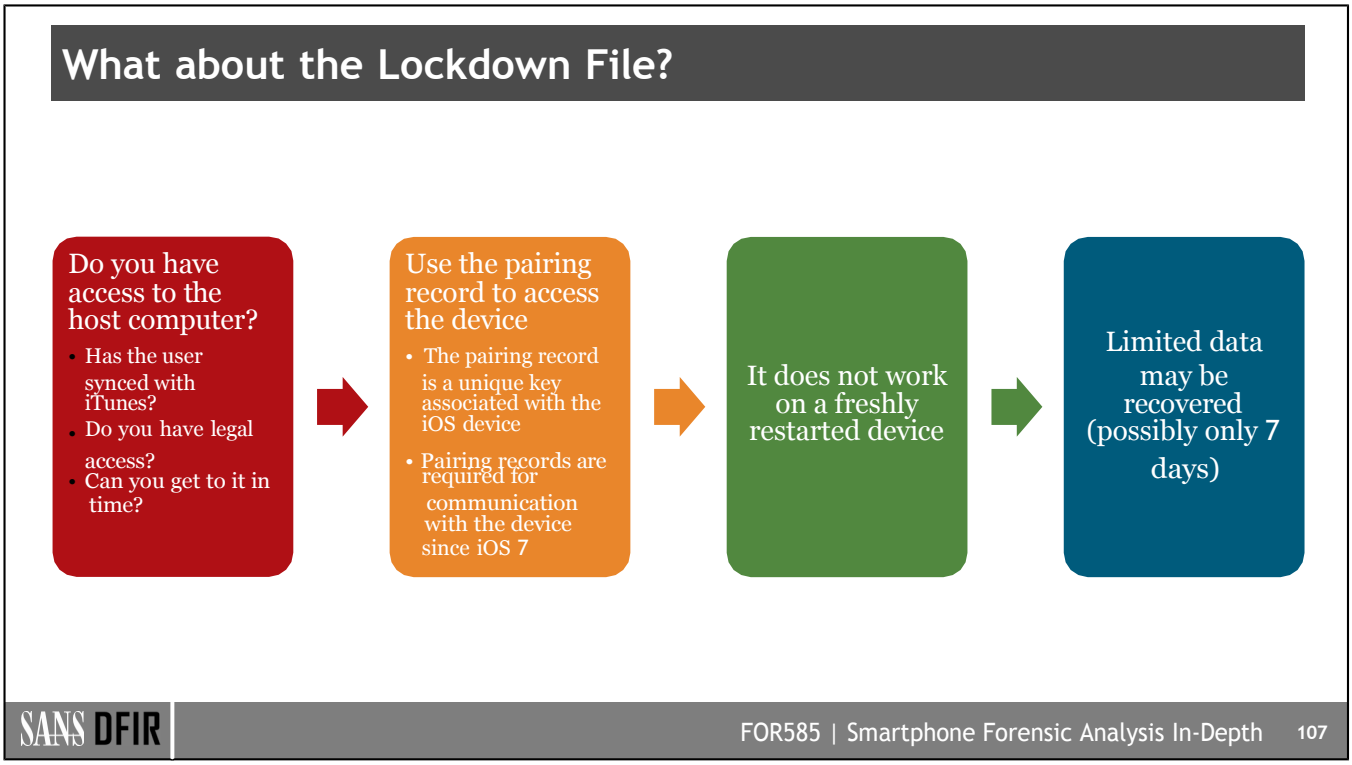
This page intentionally left blank.

iOS Lock Codes



iOS devices have four lock options: simple four-digit, six-digit, complex, and biometric (fingerprint or face) scan. A simple passcode consists of four digits and is the default offered on iOS devices. The screenshot on the left shows an example of a simple passcode. The screenshot in the middle shows a complex passcode. A complex passcode can be a combination of any numbers, letters, and symbols. Most forensic tools can crack a simple passcode but cannot currently crack a complex passcode. At best, complex passcodes can be bypassed if the iOS device can enter DFU mode. The screenshot on the right is the fingerprint scan that was introduced with the iPhone 5s. The face unlock was introduced with the unveiling of the iPhone X. A backup passcode must be set for biometric scan errors and, more importantly, to unlock the biometric data in the secure enclave. Thus, the biometric lock is only as strong as the backup password or PIN. If the fingerprint or face unlock is keeping you out, use dictionary files from similar devices to attempt to obtain the lock bypass.

Currently, a complex passcode is more secure than a simple passcode, face unlock, or fingerprint scan.



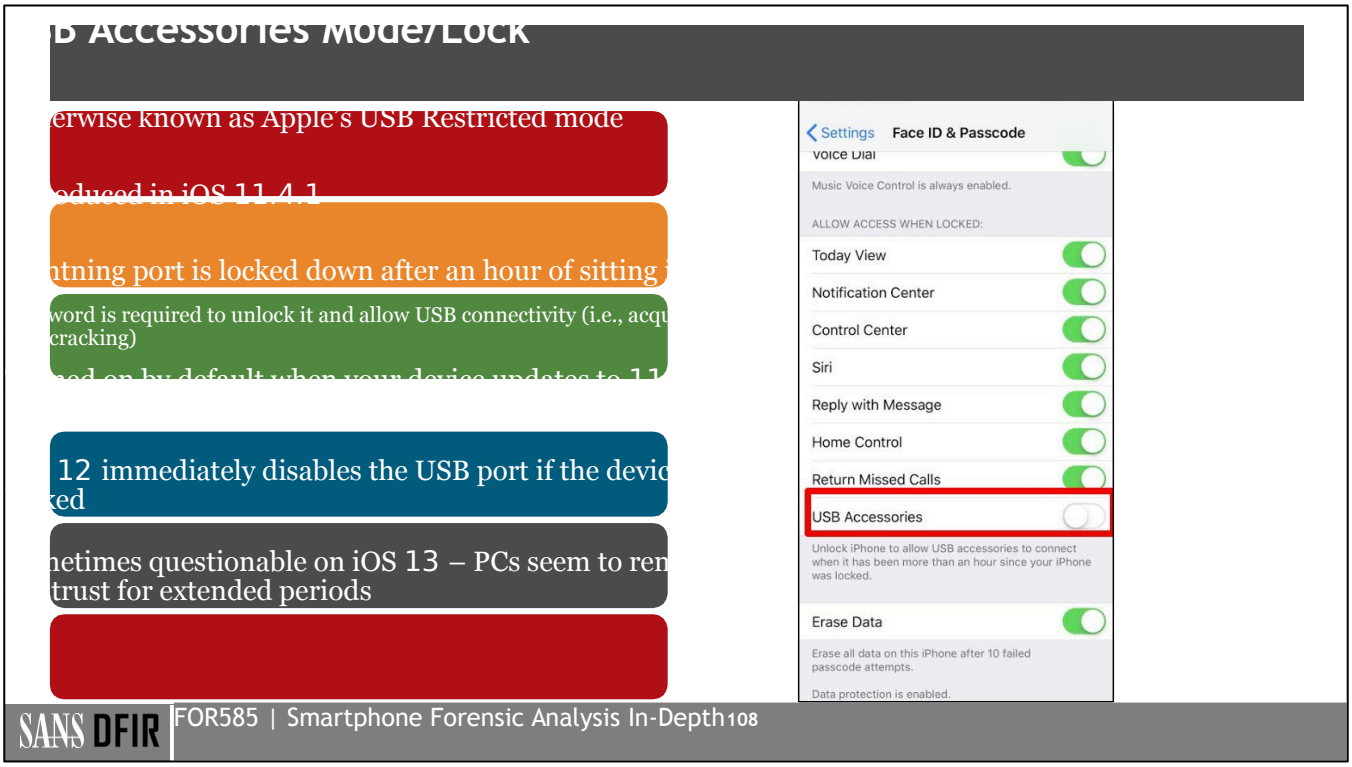
The lockdown file found on the host computer may contain the pairing record required to access the locked iOS device. The pairing record is a unique key associated with the iOS device that is synced to the host computer to allow access every time the device is connected. The concept of a pairing record and trust/pair relationship was introduced with iOS 7.

One caveat is that the lockdown file will not work with a freshly started device and may not work on later iOS versions. Additionally, it may not work after 7 days of not being connected! This means that we need to consider what we learned in Section 1. If the device is on and we power it off, the chance of using the pairing record from the lockdown file is gone. The phone must remain booted and in an active state for the lockdown file to work.

Even if you get access to the device by utilizing the lockdown file, limited data may be retrieved, if it even works. It is common to only capture third-party application data from the device via iTunes. Email, calls, SMS, and other native iOS data files are not synced with iTunes and cannot be pulled by simply connecting the iOS device to the host machine. The way around this is to possibly create a backup file using iTunes. Best practices include trying it all anyway. Using the lockdown file is so quick to determine if you can get access that it's worth the effort to try.

The location on a Mac (OS X) for the lockdown directory is /var/db/lockdown. On a Windows computer, the paths are:

Windows XP: C:\Documents and Settings\username\Application Data\Apple Computer\Lockdown
Windows Vista: C: \Users\username\AppData\Roaming\Apple Computer\Lockdown\
Windows 7 and later: C:\ProgramData\Apple\Lockdown



Apple USB Restricted Mode or USB Accessories Mode/Lock was introduced with iOS 11.4.1. There was a rumor that the lightning port would be locked down, preventing brute force unlocks (Grayshift and CAS, Premium), and it came about when iOS 11.4.1 was released into beta. When the user updates to iOS 11.4.1, the USB Restricted Mode is automatically enabled, which means USB Accessories is turned off. To view this setting on your device, go to Settings>Face ID & Passcode > USB Accessories—it's actually enabled when the toggle is turned off (to the left, not green). Magnet Forensics and Elcomsoft wrote blogs on cables that may prevent this mode from becoming enabled. Essentially, when you get the device, you immediately plug it in. At the time of writing these edits, iOS 13 seemed to remember the trust on PCs longer than expected.

Grayshift GrayKey/Cellebrite Premium

- Grayshift GrayKey is an unlocking box for law enforcement only
- Cellebrite Premium is an unlocking box for law enforcement only
- Non-LE can get the FFS feature, but not unlocking portion (Premium/Premium ES)
- Hardware and OS versions may cause problems

GrayKey came out early in 2018, introduced by Atlanta, Georgia-based Grayshift as a law enforcement–only tool. The device can be connected to two iPhones at the same time. The devices are connected for about two minutes, after which they are disconnected from the device, still in a locked state. At some later point in time, the phones will display a black screen showing the passcode and additional information. The device is also supposed to be able to break passwords. Depending on the length of the passcode, it can take up to three days or longer for six-digit passcodes, and the time needed for longer passphrases is unknown.[1] Law enforcement officials can request more information at https://graykey.grayshift.com. Cellebrite Premium[2] is the box designed to unlock and provide Full file system access to both iOS and Android devices.

References:

[1] https://for585.com/m5e9q (Grayshift GrayKey write-up by Malwarebytes)

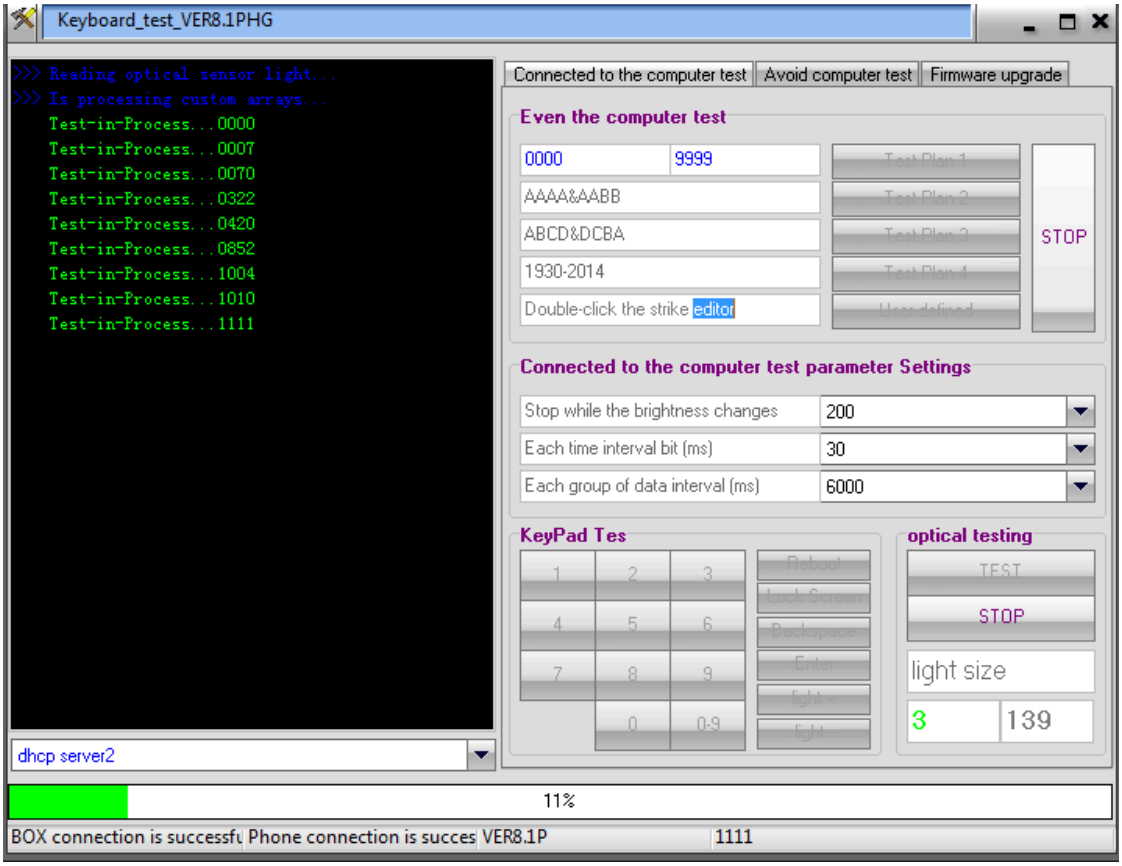
[2] https://for585.com/premium (Cellebrite Premium)

There were several methods and services for attempting to gain access to a locked iOS device that are available to everyone. Beginning in November 2014, a "black box"[1, 2] came out in the phone unlocking/hacking community called the iP-BOX. This device (and its successors the iP-BOX 2 and iP-BOX 3) can be used to defeat four- and six-digit passcodes on some iOS devices through the iPhone 7, running some versions of iOS up to iOS 10. For some devices (running iOS 6), a process must be followed in order to set the device into "an infinite unlock condition" to run the iP-BOX, but other versions of iOS require no further exploit.

For iOS 6 devices, to access the infinite unlock condition, follow these steps:[2]

1. Access the Emergency Dialing interface and dial 112.
2. Press home key to get back to interface "Slide to Unlock."
3. Press the bottom of the keyboard.
4. Slide the screen up to get to the calculator.
5. There appears a green stripe on top of the screen, which shows "***the line is busy now***."
6. Press the green stripe indicator to get back to the calling interface. In the middle of the screen, there is the address book.
7. Press the home key and address book simultaneously to access the infinite unlock condition.
8. Connect iP-BOX to the phone, move the sensor to the screen, and press the black key of the equipment to unlock.

This process requires that the phone is out of Airplane mode and that a call to the 911 Emergency center might be completed. Obviously, this is not an ideal situation because you will be adding to the call history on the phone, are connected to the network, and are directly interacting with the phone. This is a deviation from a normal forensic process, but in the event that it is the only option, such deviation may be necessary to defeat the device lock. ***If you need to perform these steps, be sure to document your actions carefully!***



IP-BOX devices of all versions consist of the box itself, old and new iPhone cables, an optical sensor, and the associated software. The device can be operated either in a standalone fashion after being set up via the software, or it can be connected to a computer during operation.[3]

The iP-BOX software is used to configure passcode test patterns. The user can choose from several free patterns, such as a birthday attack (which contains patterns that match DDMM/MMDD) or all numbers between 0000 and 9999 or can create a custom number-based dictionary attack. Each attempt takes approximately 6 seconds to perform, and so the dictionary containing all numbers between 0000 and 9999 would take between 6 seconds and approximately 17 hours to complete. Custom dictionaries can cut down on the amount of total time used to attack the passcode and may be used before a full attack if desired. Six-digit brute force attacks using newer versions of iP-BOX can take far longer.

The optical sensor is the tool's trigger to signal success because when a correct code is encountered, the screen changes and the background light output changes. The box signals success by beeping and flashing. If the optical sensor is placed in the wrong place or the screen is touched and changed by the user during the attack process, the box may stop and provide a false success signal. To restart the attack, simply press the button twice to restart the attack at the next sequential number.

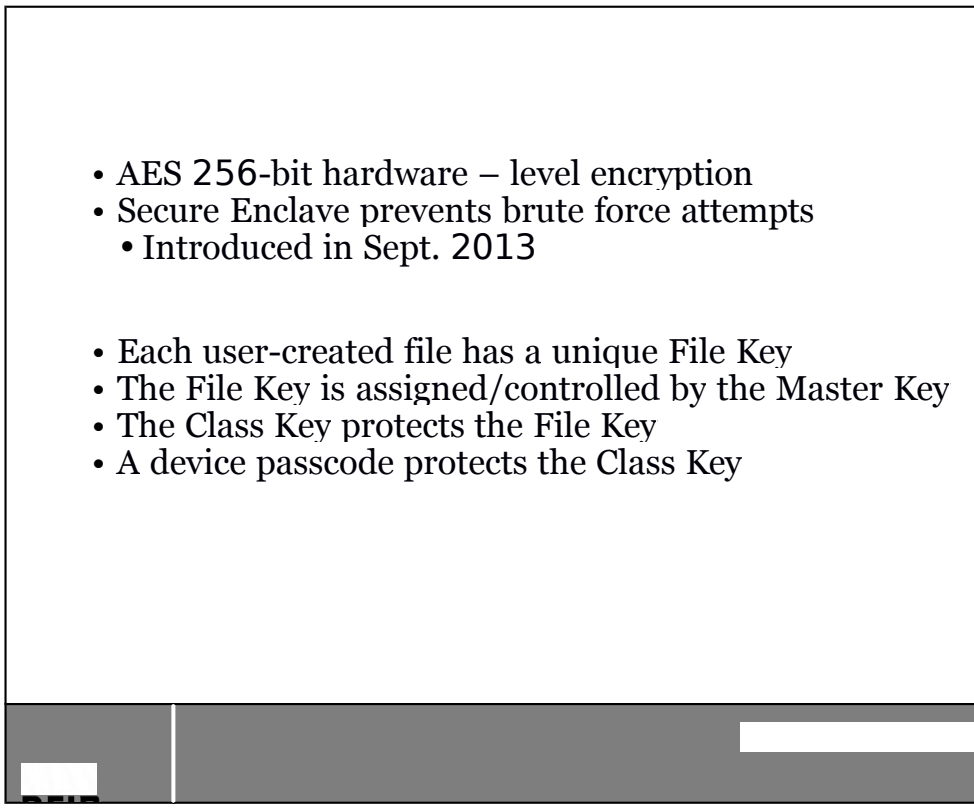
The software is then used to interface with the box and load the chosen test patterns for an offline attack of the device, or the attack can be run directly using the software through the box to the phone. The screenshot on the previous page shows an attack in process using the software through the box to the device. A custom passcode list is being used in that example.

If the attack is allowed to run completely and fails, try slowing down the interval between attacks and restarting the attack. Also, if you used the software through the device for the original attack, try rerunning the attack with the box only, or vice versa.

References:

[1] https://for585.com/hack
[2] https://for585.com/teelbox
[3] https://for585.com/pef

The fear of wiping a device after guessing too many times is a valid concern. Here is what that setting looks like.



iOS devices offer greater levels of encryption with each release. AES 256-bit encryption is hardware-level encryption stored between the flash memory and the system area in effaceable storage. The Apple A5–A13 chips offer the greatest amount of encryption and are the hardest to access when locked. The iOS device is encrypted with 256-bit AES encryption at the hardware level of the device. This encryption key is stored between the flash memory and system area on the iOS device. This area is referred to as "effaceable storage."[1] Brute force attempts are no longer possible since Sept. 2013 when Secure Enclave was introduced – the exception here – GrayKey, Cellebrite Premium and CAS. They can get beyond the Secure Enclave.

The iPhone 4s, which debuted the A5 chip, featured beefed-up security, which decreased the number of commands available in the bootloader,

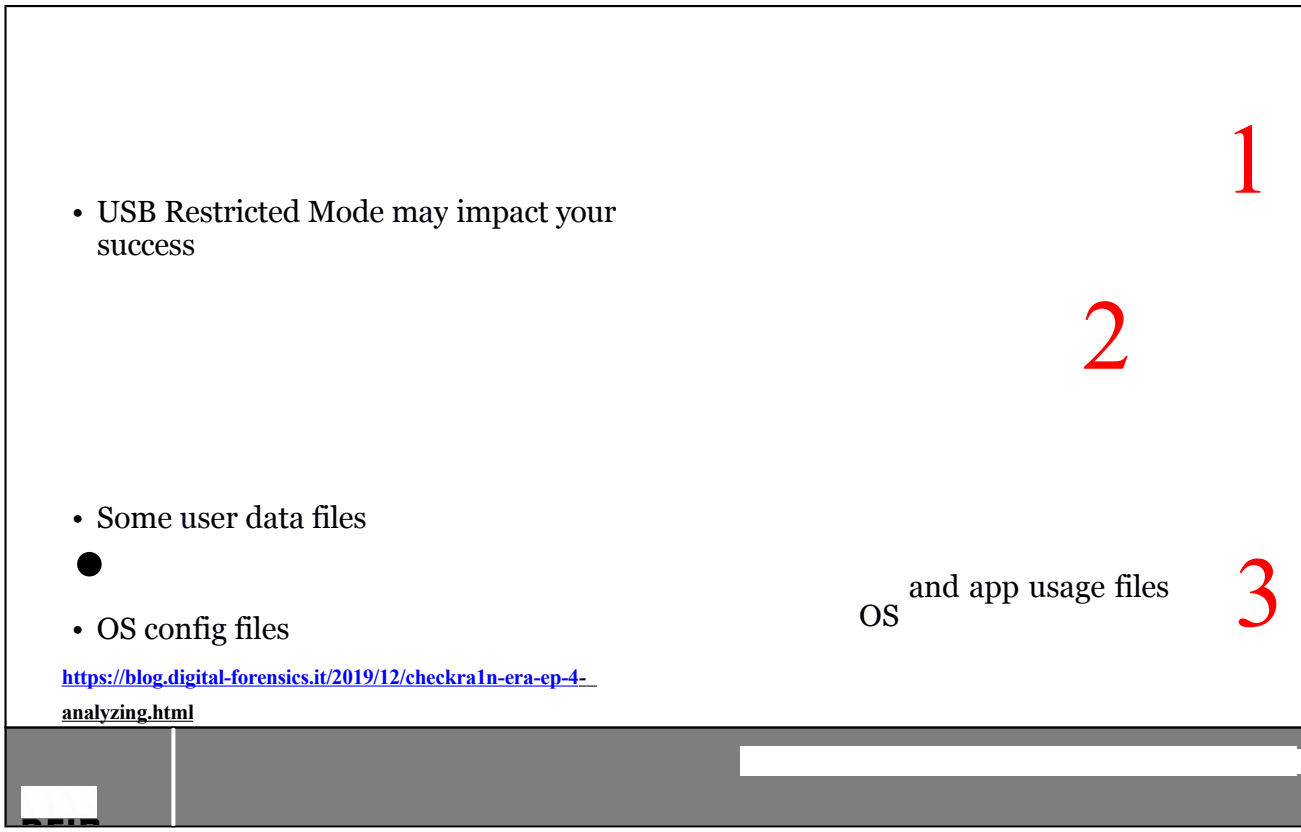
making it more difficult to physically acquire the device. The A6–A13 chips have followed suit and have become even more secure. The iPhone 4 has an exploit (24kpwn) that was used to physically access the device and bypass a lock. In the A5, A6, and A7 chips, this exploit has been patched. This patch occurred well before the release of the iPhone 6 and the A8 chip. For the longest time, the A11 chip found in the iPhone X is extremely secure – until the checkm8 exploit was identified. Currently, there is no known exploit for the A13 chips in the iPhone 11 series devices.

iOS data encryption makes our lives even more difficult when attempting to recover files from unallocated space. When a file is created by the user, a unique encryption key is created for each file. This is referred to as the File Key. The File Key is created and controlled by the Master Key (File Protection Key), which resides in the effaceable storage on the iOS device. The Master Key can be wiped, which destroys the knowledge of the File Keys, thus encrypting all user data and rendering it unrecoverable. This prevents access to decrypting the user-created files on the iOS device. Thus, when a device wipe occurs, only the Master Key is erased. This makes wiping fast and effective.

The Class Key protects the File Key. The Class Key is stored in the metadata of the file. Finally, the device passcode and the UDID of the iOS device protect the Class Key. iOS encryption is explained in detail in *Learning iOS Forensics*, Second Edition.[1]

Reference:

- Epifani and Stirparo, *Learning iOS Forensics*, Second Edition (Birmingham, UK: Packt, 2015).



BFU, or before first unlock, is a method for gaining access to an otherwise inaccessible device. Guessing the password on a locked device to obtain a full file system extraction is risky and you could disable or wipe the device. Do not just guess passcodes, even using UFED and checkm8. BFU mode can be leveraged via checkra1n (refer to the blog by Mattia Epifani: https://blog.digital-forensics.it/2019/12/checkra1n-era-ep-4- analyzing.html) or UFED. Checkm8 has some restrictions that may impact if BFU is available or not.

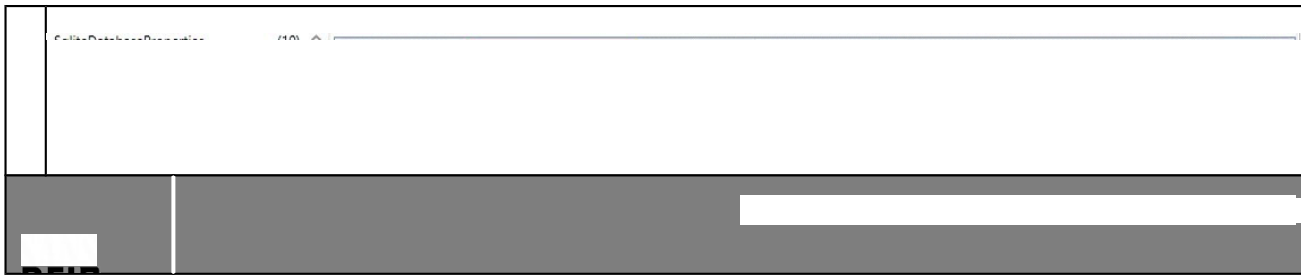
- For Checkm8 to work on iPhone 8 and iPhone X, starting with iOS 14, the passcode must be removed before starting the extraction. Thus, BFU is irrelevant.
- For previous models (iPhone 6, 6s and 7) a BFU extraction is possible with iOS 14 and iOS 15.

Thus, iPhone 7 and below, BFU is possible regardless of the iOS version. On iPhone 8 and iPhone X, BFU is possible up to iOS 13.

In order to obtain a BFU image using UFED, it's pretty simple. When prompted to enter the password and you don't know it – select Cancel and then opt to complete a BFU extraction. From there, dig through the dump for clues as to what the password may be.

This page intentionally left blank.

- She can do it all for us
- Keep in mind that every interaction with Siri leaves a trace
- You just need to dig to find it
- iOS 13 - 15 made it easier with messages – it's in a blob in sms.db

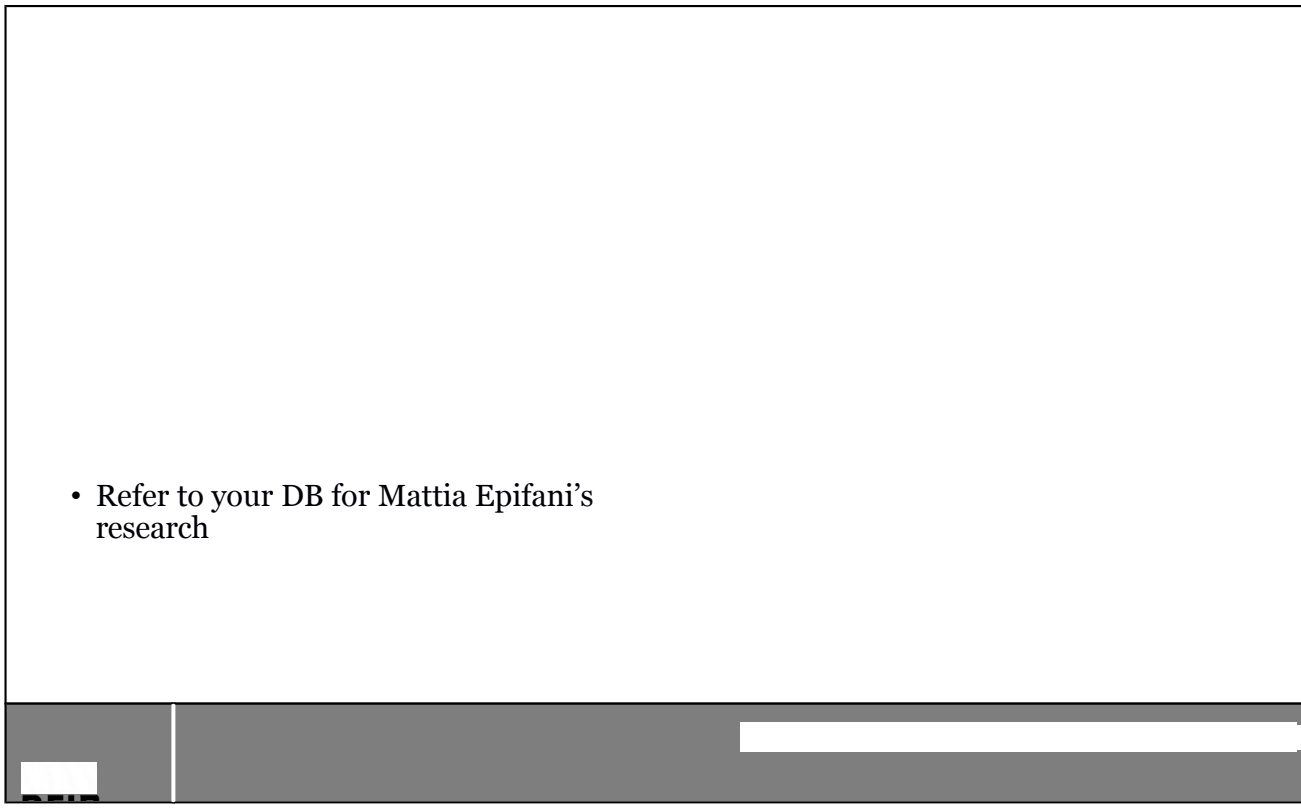


The first step in searching for traces of Siri use should be to validate if the user was using Siri. To do this, simply search for “Siri” in the tool or navigate to Library/Preferences/com.apple.SiriViewService.plist, as shown in this slide, and verify that Siri is active. Once validated, you must think of ways Siri is used. For example, if she is used to set a reminder, that data should be stored in the same location as tasks or calendar items. If she is used to get directions, the information is stored with the maps or within the app she used to get the directions. Thus, it’s a needle-in-the-haystack search!

Consider some aspects of the “Hey Siri” feature. Even on a locked device (if Siri is enabled this way), you can ask Siri to send a text, update your Facebook status, call someone, and even request addresses. Although this isn’t a forensic solution per se, it is a method to access data on a locked iOS device that is otherwise useless for your investigation.

/Library/Preferences/com.apple.assistant* is another file that may be useful here.

If the user leverages Siri to send an SMS message in iOS 13, there will be a BLOB in the column “message_summary_info” that contains “com.apple.siri” which means Siri was leveraged to send the message as shown in fine print on your iPhone message screen.



The Apple Watch was released in spring 2015. As expected, the Apple Watch craze began. Sarah Edwards, the author of FOR518, and I spent a few hours picking apart the watch to see what we could find. I recently upgraded to the Apple Watch Series 3 with cellular, and a lot has changed. Before we dive into that, let's start with the first and second Apple Watches.

The Apple Watch must be paired with an iPhone running iOS 8.2 or later for full functionality. True, anyone can wear and use certain features of the Apple Watch, but full functionality requires that the device be paired with an iPhone. The Apple Watch is paired to the iPhone via Bluetooth. The watch connects to Wi-Fi, which means that the device continuously syncs data that the iPhone accepts when connected to the same Wi-Fi. For example, my Apple Watch is connected to my personal iPhone. I use this iPhone for both work and personal use. I sync personal email, calendars, and notes for both work and personal accounts. When my watch is connected to my work Wi-Fi (as is my iPhone), the calendar continues to update even if Bluetooth is turned off. See any issues with this? We address those in a few slides.

The Apple Watch runs a Watch OS, which is similar to iOS. The watch has at least 512 MB of RAM and 8 GB of internal flash storage separate from iPhone device storage.[1] Thus, a good chunk of data is saved on the Apple Watch. Currently, the easiest way to get data from the Apple Watch

is to parse the backup file for the iPhone from iTunes or iCloud. We cover parsing backup files in Section 4 but will take a quick glimpse at what the data will look like. As shown in the screenshot, a diagnostics port is available, and we believe we can pull data directly from the Apple Watch; however, I was not willing to ruin my watch should I be incorrect!

There are both forensic and security implications when wearing, examining, and handling an Apple Watch. Two things to consider is that when we parse a backup of an iPhone to obtain data from an Apple Watch, what if:

- The watch is no longer synced, and the data is missing?
- The backup is encrypted?

Another aspect is physical access. What if the watch is lost, or stolen, or worse, the iPhone it is paired with is? What happens when an Apple Watch enters a secure or prohibited facility?

Security is something else we, as users and examiners, must consider. With Wi-Fi and Bluetooth always being enabled, we are putting our devices (Watch and iPhone) at risk for unwanted access. We are now visible to others who may be scanning for Wi-Fi devices! In addition, the permissions we give our watch can be taken and used by others if the watch is stolen or borrowed. Further research is required to determine just how vulnerable we are by wearing an Apple Watch.

Each Apple Watch has its own Synced Data Directory under /var/mobile/Library/DeviceRegistry/<GUID>. Under this directory, a plethora of information pertaining to the Apple Watch can be recovered. Be careful when examining a first or second edition of the Apple Watch because a lot of the files are exact copies of what is on the device and do not represent what happened on the watch. Some examples of data you can find in the DeviceRegistry directory of the first two Apple Watches include:

- AddressBook
- GeoServices
- Health
- Mail
- Maps
- Passes
- Preferences
- PairedSync
- Photos

This list is just a sampling of what can be recovered. Some other important locations to examine include:

Email:
/var/mobile/Library/DeviceRegistry/<GUID>/NanoMail/registry.sqlite
AddressBook:
/var/mobile/Library/DeviceRegistry/<GUID>/AddressBook **Voicemail**:
/var/mobile/Library/DeviceRegistry/<GUID>/PreferencesSync/NanoDomains/com.apple.mobilephone

This binary plist file is stored on the iPhone as /mobile/Library/DeviceRegistry.state/properties.bin. Here, we can see information pertaining to the Apple Watch.

The properties.bin file contains paired Apple Watch information, including:

- Watch Name
- Make
- Model
- OS
- GUID

The Synced Data Directory contains a list of installed applications on the Apple Watch. This file is a binary plist with an embedded plist. Fun, right? This embedded binary plist is located in /var/mobile/Library/DeviceRegistry/<GUID>/NanoPreferencesSync/NanoDomains/com.apple.Carousel.

Make sure you know how the data is being populated on the device under the DeviceRegistry for the Apple Watch. The question is this: Was the activity initiated on the Apple Watch or the iPhone? Does it matter? Why does the Watch data contain information that occurred before the Watch was released?

Make sure you familiarize yourself with ways to extract data directly from the Apple watch (more information is provided in the course Dropbox)!

Reference:

[1] https://for585.com/watch

The release of the Apple Watch Series 3 had us hoping that the device would finally track calls that were made from the watch, while acting independently from the iPhone. This is not the case and still isn't with the latest release. All of the calls I made from my Apple Watch were placed right into my callhistory.storedata with the rest of the calls. Worse yet, there isn't a status flag stating where the call was made. This makes putting a person behind a device extremely difficult.

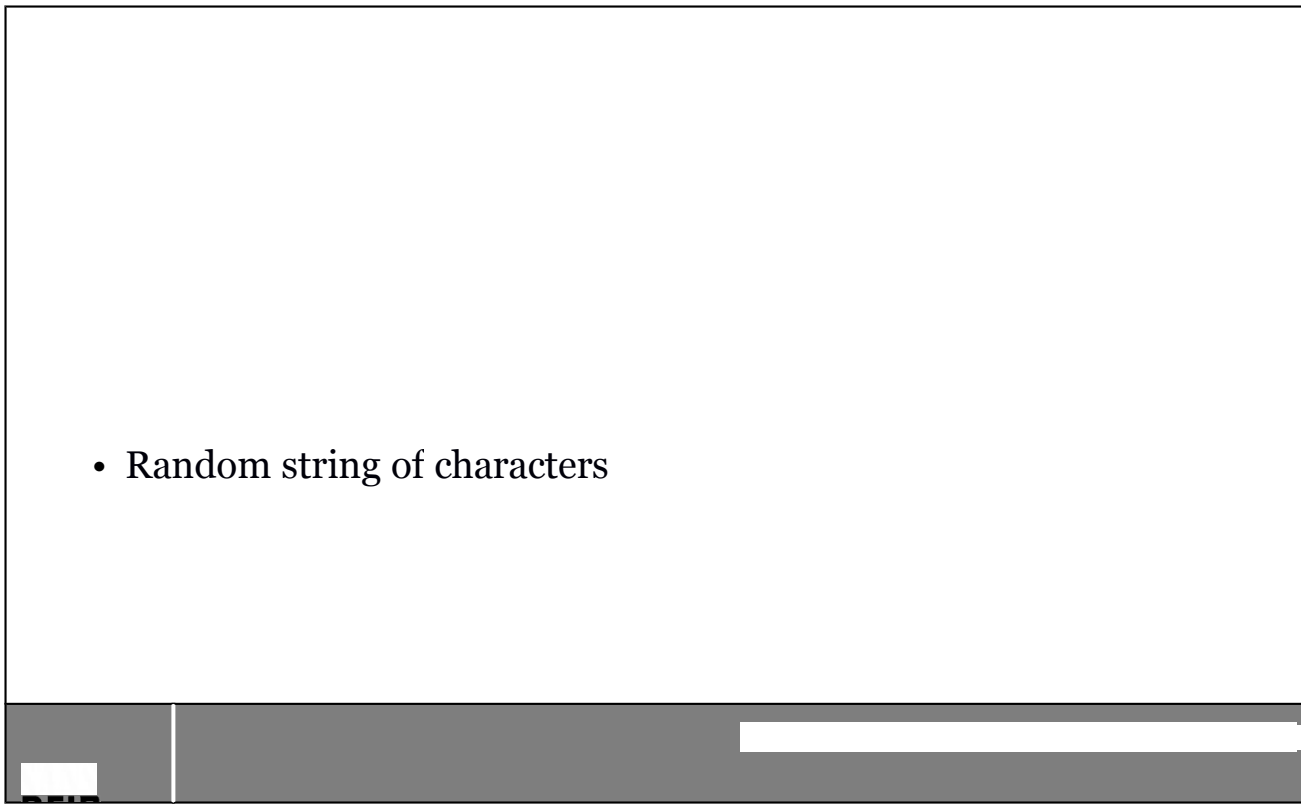
When examining artifacts from the latest Apple Watch, we see that the files of interest still exist in
/var/mobile/Library/DeviceRegistry/<GUID> but are primarily plist and dat file. Apple watch data also exists in

/mobile/Library/NanoBackup. You will find a few databases, and

those should be examined for relevance. A tip for iOS examinations,

if you see the words "nano" or "DeviceRegistry" an Apple watch is

involved.



All applications are stored in the data partition of the iOS device. The acquisition method used to obtain the forensic image may affect the access provided to the application data for examination. The applications and associated user data may be stored in more than one location on the device. Although the most common area to recover application data is the /private/var/mobile/Containers/Data/Application directory, traces may exist in other locations.[1, 2] Another common location is /private/var/mobile/Containers/Shared/AppGroup. This directory may store additional files and databases of interest. The best method for recovering all data relating to a specific application is to conduct a physical keyword search in a tool such as Physical Analyzer or a content search in Oxygen Forensics. If you simply have a logical dump to examine, the applications will be listed under /applications.

iOS applications are stored in the /private/var/mobile/Containers/Data/Application and

/private/var/mobile/Containers/Shared/AppGroup on the data partition. The Application folder is named according to the application identifier. For example, Facebook may be listed as FEE2B8A6-C70C-4BAA-AE79-D3A4996FF496. That folder contains the Facebook.app folder, Documents, Library, and tmp folders. It is best to examine each directory if databases and files of interest cannot be located.

References:

- Andrew Hoog and Katie Strzempka, *iPhone and iOS Forensics: Investigation, Analysis and Mobile Security for Apple iPhone, iPad and iOS Devices* (Waltham, MA: Elsevier, 2011).
- Heather Mahalik, Rohit Tamma, and Satish Bommisetty, *Practical Mobile Forensics*, Second Edition (Birmingham, UK: Packt, 2016).

The Application folder contains four separate folders, each of which are explained here:[1]

- **Documents folder**: Contains the following files, which are required by the application to function; these are generally generic files (such as icons) that are present in the application itself:
  - plists
  - text docs
  - images
- **Library folder**: Shows traces of user activity
  - cached data
  - cookies
  - preferences (user login data may be found here)
  - WebKit data (if applicable)
- **Application code**
- **TMP folder**: Commonly empty, but may be used by the application to store temporary files

This slide intentionally left blank.

Shows access permission for each application on the iPhone

The TCC.db is an extremely important database. This file contains all the services (access permissions) that each application has privileges to use. Here we can see a few things. First, we are examining the **access** table of the /var/mobile/Library/TCC/TCC.db. We can see the services to Camera, which show applications that request access to the camera, and the **auth_value** column alerts us if the permission was granted (2), selected (3), or denied (0). The **service** values that are odd, but relevant are kTCCServiceLiverpool represents location data and kTCCServiceUbiquity represents cloud sync.

T

C

C

.

d

b

s

e

l

e
c
t
a
c
c
e
s
s
.
s
e
r
v
i
c
e
a
s
"
S
e
r
v
i
c
e
"
,
a
c
c
e
s
s
.
c
l

i
e
n
t
a
s
"
C
l
i
e
n
t
"
,
a
c
c
e
s
s
.
c
l
i
e
n
t
_
t
y
p
e
a
s
"
C
l

i
e
n
t
T
y
p
e
"
,
C
A
S
E
when
access.aut
h_value =
2 then
"Allowed"
when
access.aut
h_value =
0 then
"Not
Allowed"
when
access.aut
h_value =
3 then
"Selected
Only" end
as "Auth
Value",
a
c
c
e
s

s
.
a
u
t
h
_
r
e
a
s
o
n
a
s
"
A
u
t
h
R
e
a
s
o
n
"
,
a
c
c
e
s
s
.
a
u
t

h

_

v

e

r

s

i

o

n

a

s

"

A

u

t

h

V

e

r

s

i

o

n

"

,

datetime(access.last_modified,'unixe

poch','localtime') as "Last Modified"

from access

- Does the timeline make sense?
- When was the file created, and how does the internal content compare?

- Crash Logs from iOS device
- /Library/Databases/DataUsage.sqlite
- /Library/com.apple.itunesstored/itunesstored2.sqlitedb
- */installd/Library/MobileInstallation/UninstalledApplications.plist*
- */installd/Library/Logs/MobileInstallation/*.log*
- /Library/Logs/mobile_installation_helper.log*
- /Library/Preferences/addaily.plist
- /Library/Keyboard/langlikelihood.dat
- /Library/FrontBoard/applicationState.db
- */Library/Application Support/com.apple.remotemanagementd/RMAdminStore-Local.sqlite*
- */Library/CoreDuet/Knowledge/KnowledgeC.db*

There are tons of locations that track application usage on iOS devices. Here is just a sampling. Your cheat sheet is full of files that track application usage, so make sure you refer to it. Some of the files included here are only available on jailbroken devices. All of the files below offer some form of application information, whether it be the file path, date used, date installed, date uninstalled, bundle ID, and more.

- Crash Logs from iOS Device
- /Library/Databases/DataUsage.sqlite
- /var/mobile/Library/com.apple.itunesstored/itunesstored2.sqlitedb
- */installd/Library/MobileInstallation/UninstalledApplications.plist: Full file system extraction required*
- */installd/Library/Logs/MobileInstallation/*.log: Full file system extraction required*
- /Library/Logs/mobile_installation_helper.log*
- /Library/Preferences/addaily.plist
- /Library/Keyboard/langlikelihood.dat
- /Library/FrontBoard/applicationState.db
- */Library/Application Support/com.apple.remotemanagementd/RMAdminStore-Local.sqlite* - Screentime
- */Library/CoreDuet/Knowledge/KnowledgeC.db*

A parser created by Alexis Brignoni, a former FOR585 student, available here: https://github.com/abrignoni/iOS-Mobile-Installation-Logs-Parser

If you are lucky enough to have a jailbroken device, we recommend using his script! You can also use iLEAPP, which will parse these files. iLEAPP is available in your VM and is covered later in this section. APOLLO dives deep into knowledgec.db and is coming up later in this section.

Other files, like UninstalledApplications.plist and MobileInstallation/*, may only be available with a full file system dump, not an iTunes backup. The mobile_installation_helper.log will be available regardless, as will addaily.plist and langlikelihood.dat. All of the files listed in this slide add value to your examination if applications are a concern.

The /Library/com.apple.itunesstored/itunesstored2.sqlitedb stores apps that were downloaded via iTunes. This may contain traces of deleted applications. In the example in the slide, com.wunderground.weatherunderground was deleted by the user.

iBackupBot can be used to examine Crash Logs on an attached iOS device. Crash Logs can give us a good glimpse at what was running and "crashing" on the device. Notice the applications that are listed in the Crash Report? These can be examined to show apps that were running at specific points in time. These logs should be accessible on any iOS device as long as your workstation can communicate with the device.

Another location that stores applications that have been used, deleted, or even reinstalled on the iOS device is DataUsage.sqlite, which is located at /var/wireless/Library/Databases/Datausage.sqlite. This database is accessible in most forensic images and backups and is located in Library/databases. When examining this database, we can see one primary table of interest is **ZPROCESS**. All other tables should be examined for relevance to your investigation. In **ZPROCESS**, we can see the first time the app was launched, as well as the last time the app was launched and used. The Bundle name represents the application installer. Below is a sample query that extracts key data from this table.

```
Select ZPROCESS.Z_PK,
  DateTime(ZFIRSTTIMESTAMP + 978307200, 'UNIXEPOCH') AS
  "First Timestamp",
  DateTime(ZTIMESTAMP + 978307200,
  'UNIXEPOCH') AS "Last Timestamp",
  ZBUNDLENAME,
Z
P
R
O
```

CNAME

From

ZPROCESS

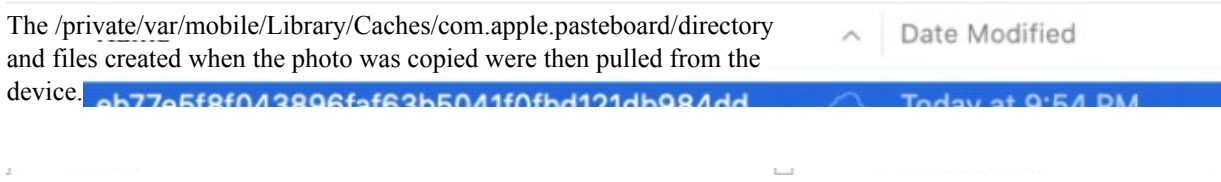
- The most recent “copy” will be tracked
- Any file or word that can be long pressed and copied
- CAUTION: Hash values of images will not match & file extensions may be dropped
  - File extensions may be converted for compatibility, which ruins the hash value

/mobile/Library/Caches/com.apple.Pasteboard/.* - This directory will contain one or more other directories with a naming convention that appears to be hash values without file extensions. The files without extensions can be anything the user has copied. The file can simply be ASCII text if a string was copied, or the file can be a photo or video if the user copied those.

An example of the copy/paste is shown below. Jared Barnhart provided these examples.

Copying a photo from the Photos app of Tito’s and a soda with lime in La Jolla - Torrey Pines (see image, following page). When the photo was copied, a directory was created and the copied file was written into it.

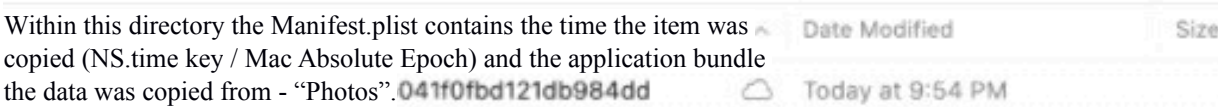
The /private/var/mobile/Library/Caches/com.apple.pasteboard/directory and files created when the photo was copied were then pulled from the device.

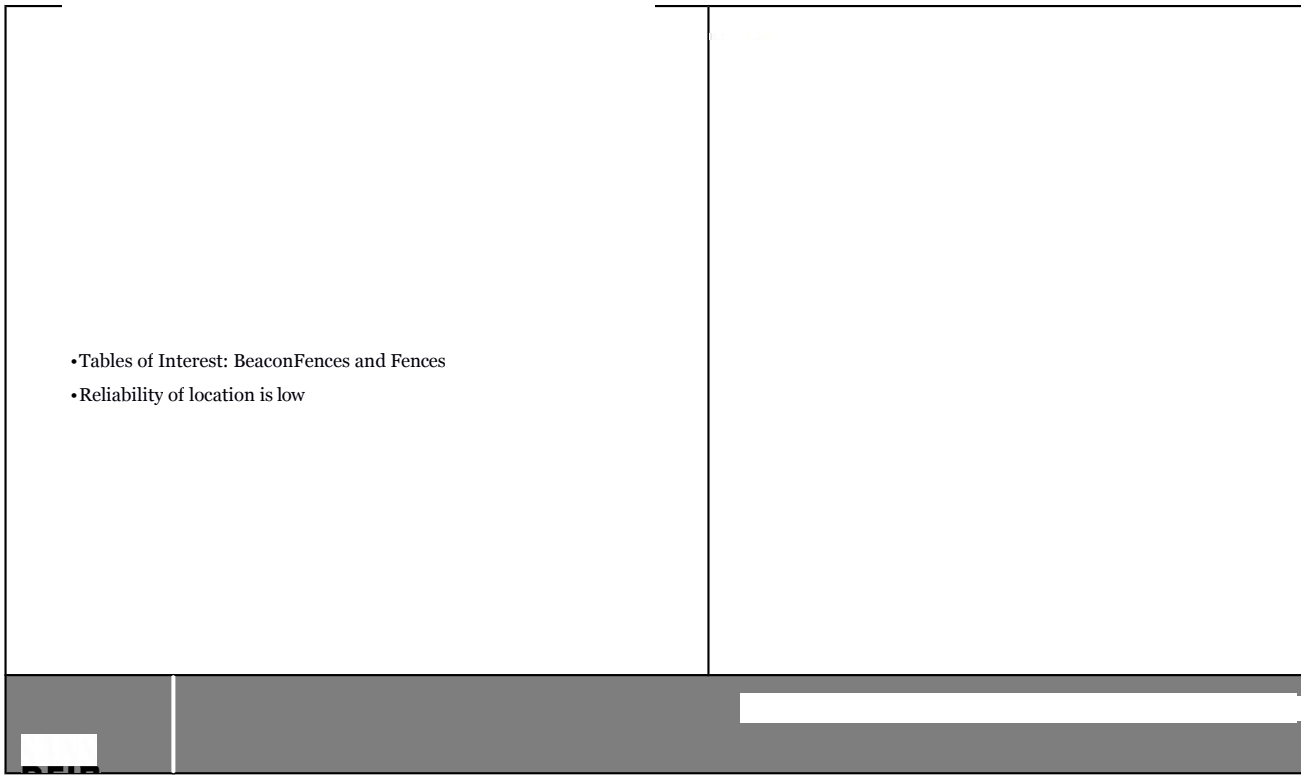


This file without an extension is a version of the image file that was copied.

The actual image file that was copied was IMG_0036.HEIC and the file size differs from the copied file and the hashes do not match.

Within this directory the Manifest.plist contains the time the item was copied (NS.time key / Mac Absolute Epoch) and the application bundle the data was copied from - “Photos”.





Trying to prove that the user deleted an application from a non-jailbroken device is more difficult because we can't access all of the uninstall/install logs. Here are some files that may help you do this. You are going to have to prove application usage and removal from smartphones. Section 5 of this course will cover recovering deleted applications in more detail.

The three files below will show which apps had access to spotlight and others including the location restrictions/settings:

- /Library/Preferences/com.apple.corespotlightui.plist
- /Library/TCC/TCC.db
- /Library/Caches/locationd/
  - clients.plist
  - consolidated.db > Tables of Interest: BeaconFences and Fences **NOTE: The location artifacts here should not be trusted. A great blog to read on this was written by Ian Whiffin: http://www.doubleblak.com/m/blogPosts.php?id=22.

Ever wonder how your device knows what is draining the battery? Yep, there are files to track that. Simple files like /Library/Preferences/com.apple.BatteryCenter.BatteryWidget.plist show if the user can see the battery icon. Other files like CurrentPowerLog.PLSQL will rock your investigation if you find an older backup or device.
The number of artifacts found in the
/
private/var/containers/Shared/SystemGroup/<GUID>/Library/BatteryLife
/CurrentPowerlog.PLSQL is astounding. This file provides us access to glimpses of user data that was draining the battery, supports application usage (even if deleted), how the data was used (Wi-Fi versus cellular), and potential timestamps that can link a user and an activity to the iOS device. Most tools support parsing of data from this file and a Full File System extraction is required to obtain it.

Sarah Edwards kicked off the research with her Bsides NOLA presentation. Check it out at https://for585.com/518. This file contains over 200 tables. Enjoy parsing that!

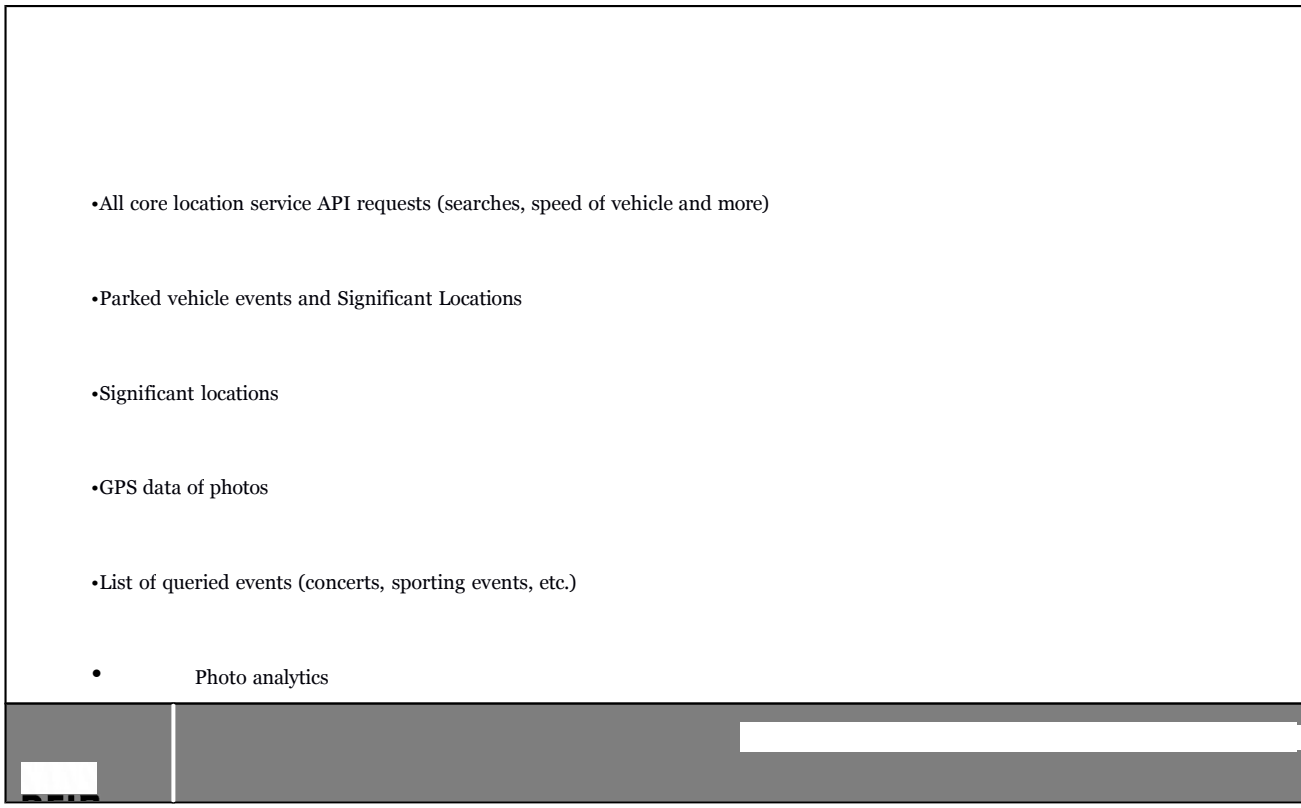
Screentime can be used in most investigations to put applications in use and for specific periods in time. Everything that is used to keep us "digitally healthy" tends to be helpful in forensic investigations. Most commercial tools are parsing artifacts from Screentime, and the files of interest are below. This artifact is going to be a part of your next lab!

/private/var/mobile/Library/Application Support/com.apple.remotemanagementd/RMAdminStore-Cloud.sqlite
/private/var/mobile/Library/Application Support/com.apple.remotemanagementd/RMAdminStore-Local.sqlite

Lab 3.2

iOS Forensic Analysis

This page intentionally left blank.



Jared Barnhart, Heather Mahalik, and Ian Whiffin dove into researching location artifacts on iOS and Android and rating them on what you can trust based upon timestamp and location accuracy. The full webinar is available on Cellebrite's website. https://cellebrite.com/en/episode-15-ibeg-to-dfir-location-data-on-ios-and- android-devices/. Some of these files require a jailbreak to obtain full file system access.

The Location cheat sheet for this webinar is available in your Dropbox under Day/Section 3. Some of these databases are parsed by the commercial tools, iLEAPP and APOLLO. Always make sure you verify that they are being parsed for all relevant information when your investigation involved location artifacts.

/
private/var/mobile/Library/Cache
s/com.apple.routined/Cache.sqlit
e Tables of Interest:
Z
R
T
C
L
L
O
C
A

T
I
O
N
M
O

Z
R
T
C
L
L
O
C
A
T
I
O
N
M
O

Z
R
T
V
I
S
I
T
M
O
ZRTLEARNEDLOCATIONOFINTERESTVISITMO

/
private/var/mobile/Library/Cache
s/com.apple.routined/Local.sqlite
Table of Interest:
ZRTVEHICLEEVENTMO

/
private/var/mobile/Library/Caches/co
m.apple.routined/Cloud-V2.sqlite
Tables of Interest:
Z
R
T
L
E
A
R

NEDVISTIMO

ZRTLEARNEDPLACEMO

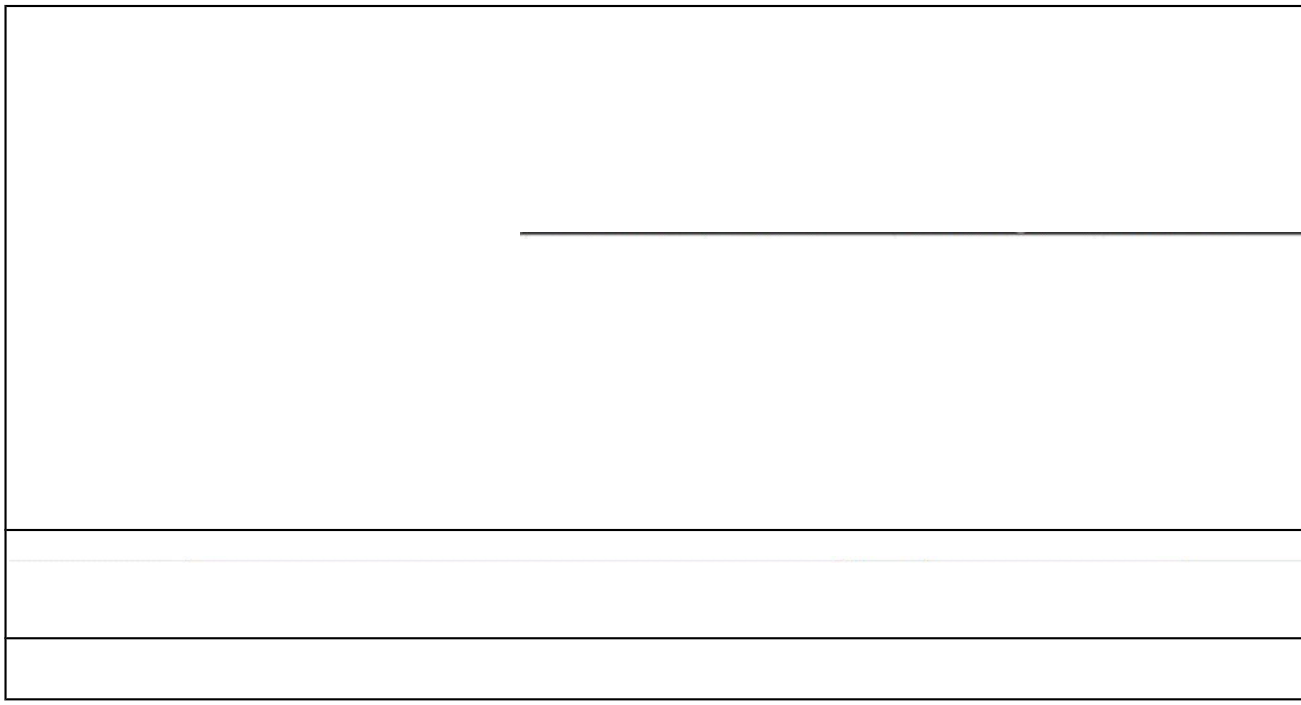
ZRTMAPITEMMO

/private/var/mobile/Media/PhotoData/Photos.sqlite

Tables of
Interest:
ZASSE
T
ZADDI
TIONA
LASSE
TATTR
IBUTE
S
ZCLO
UDMA
STER
MEDI
AMET
ADAT
A

/private/var/mobile/Media/PhotoData/Caches/GraphService/CLSPublicEventCache.sqlite

/
var/mobile/Media/PhotoData/Caches/GraphServi
ce/PhotosGraph/photosgraph.kgdb Table of
Interest:
EDGEVALUE



iOS location cache is stored in /private/var/mobile/Library/Caches/com.apple.routined/Cache.sqlite Tables of Interest:

ZRTCLLOCATIONMO

ZRTCLLOCATIONMO

Z

R

T

V

I

S

I

T

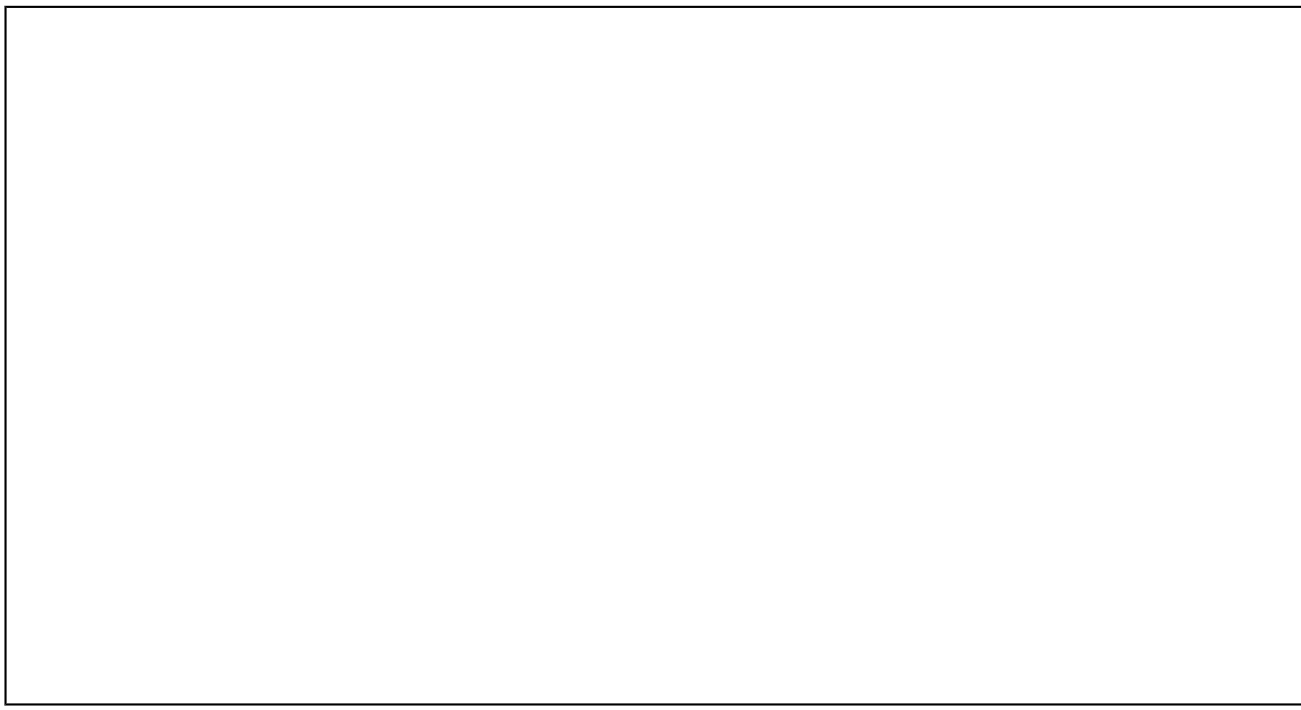
M

O

ZRTLEARNEDLOCATIONOFINTERESTVISITMO

All location requests conducted on the iOS device results in an entry being created in the database. The location and the timestamp are accurate based upon the research conducted by Jared, Heather and Ian. Refer to the same webinar and cheat sheet mentioned in the previous slide. Maps and camera are constantly recording for information to store here. System Services and Safari monitor on occasion.

ArtEx is shown in the slide with Cached location results from this database.



Significant locations are extremely relevant to iOS investigations. While the user can elect what they want to have tracked, often you will relevant information in this database. APOLLO, created by Sarah Edwards, parses this database nicely as you will see in your next lab.

- AirTag and more

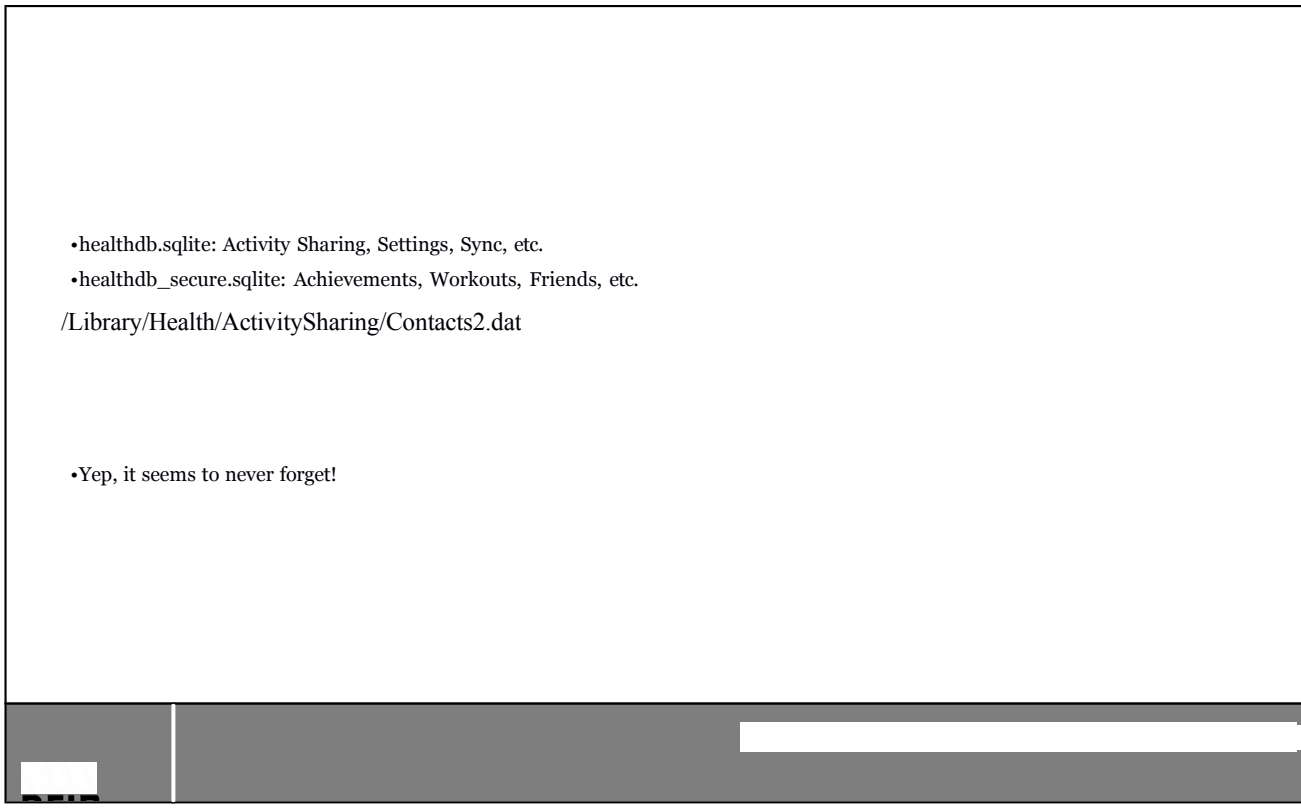
As previously discussed, location artifacts are everywhere on iOS devices. A sample list is provided in this slide. We recommend use of your tools to triage locations and gain an idea on where the user was when an event or crime occurred. From there, refer to your cheat sheet to dive into the paths that track user locations. Do not forget to leverage the Location Cheat Sheet available in Day/Section 3 of your course Dropbox.

Here is the sample list of other places to consider finding location data. This *does not* even scratch the surface of all of the location artifacts that will exist on iOS devices when we think of third-party application data.

Think of everything you learned up until this point. Even concepts from Android apply here. Go forth and enjoy the hunt.

- /Library/Preferences/com.apple.weather.plist
- /Library/Maps/Bookmarks.plist
- /com.apple.Maps/Maps
- /com.apple.Maps.plist
- /group.com.apple.Maps.plist
- /private/var/mobile/Library/Caches/com.apple.findmy.fmipcore/Devices.data
- /private/var/mobile/Library/Caches/com.apple.findmy.fmipcore/Items.data
- iCloud and Google cloud

Make sure you verify locations as the device may appear to be somewhere it wasn't. A great blog on the accuracy of locations by Ian Whiffin can be found at: https://doubleblak.com/blogPosts.php?id=14.



Health data is one of the hottest iOS topics, and I have spent a ton of time researching it to help former students put a person behind a crime. How? The two databases listed here track so much about the user. More than just a workout completed.

- /Library/Health/
  - healthdb.sqlite: Activity Sharing, Settings, Sync, etc.
  - healthdb_secure.sqlite: Achievements, Workouts, Friends, etc.
- /Library/Health/ActivitySharing/Contacts2.dat: friends you share workouts with
- */private/var/root/Library/Caches/locationd/cache_encrypteddC.db*: Steps, distance, floors climbed, timestamps and more (requires a jailbreak).
- */Library/Caches/locationd/cache_encryptedC.db*

Sarah Edwards and I presented #DFIRFIT or Bust at the DFIR Summit in Austin, TX, in 2018. This talk drummed up a lot of interest. Our talk was recorded and is available on the SANS YouTube channel. Since then, I have done several talks on Health and Making a Murderer: Health Edition. These presentations can be found on smarterforensics.com. Health data provides so much information and can help solve many different crimes. Even insurance companies are using data to prove that someone was healthy and active while collecting workers compensation

from the workplace. Think about it…. It's scary!

Sarah Edwards released APOLLO, which will parse some of the health data. Most commercial tools parse health as well.

Reference:

https://github.com/mac4n6/Apollo

The query below parsed the information in the healthdb_secure.sqlite and sorted by date to prove where the user was at specific periods of time. If you are interested in other scenarios, go out to my previous presentations where I demo dragging a body and more.

```
Select
datetime(samples.start_date+978307200,'unixepoch','loc
altime') as "Start Date",
datetime(samples.end_date+978307200,'unixepoch','loc
altime') as "End Date", samples. data_id,
case
w
h
e
n

d
a
t
a
_
t
y
p
e

=
```

3

then

"weight"

when

data_type

=

7

th

e
n

"
s
t
e
p
s
"

w
h
e
n

d
a
t
a
_
t
y
p
e

=

8

t
h
e
n

"
d
i

s
t

i
n

m
"
when
data_ty
pe = 9
then
"resting
energy"
when
data_ty
pe = 10
then
"active
energy"
when
data_ty
pe = 12
then
"flights
climbe
d"
when
data_type =
67 then
"weekly
calorie goal"
when
data_type =
70 then
"watch on"
w
h

e
n

d
a
t
a
_
t
y
p
e

=

7
5

t
h
e
n

"
s
t
a
n
d
"

w
h
e
n

d
a

t
a
_
t
y
p
e

=

7
6

t
h
e
n

"
a
c
t
i
v
i
t
y
"

w
h
e
n

d
a
t
a

_
t
y
p
e

=

7
9

t
h
e
n

"
w
o
r
k
o
u
t
"
when
data_ty
pe = 83
then
"some
workou
ts" end
as
"activit
y type",

```
quantity, original_quantity, unit_strings.unit_string, original_unit,
correlations.correlation, correlations.object,correlations.provenance
string_value, metadata_values.data_value,
metadata_values.numerical_value,
metadata_values.value_type,metadata_keys.key
from samples
left outer join quantity_samples on
samples.data_id = quantity_samples.data_id
left outer join unit_strings on
quantity_samples.original_unit =
unit_strings.RowID left outer join
correlations on samples.data_id =
correlations.object
left outer join metadata_values on
metadata_values.object_id =
samples.data_id  left outer join
metadata_keys on metadata_keys.ROWID =
metadata_values.key_id order by "Start
Date" desc
```

Most commercial tools are parsing iOS health data. However, not all tools are perfect, and most will interpret the data in different manners. It is up to you to validate the findings.

The following query will parse additional information to prove if the tool extracted it correctly. select :

datetime(samples.start_date+978307200,'unixepoch','localtime') as "Start Date",
datetime(samples.end_date+978307200,'unixepoch','localtime') as "End Date", samples. data_id, case
w
h
e
n

d
a
t
a
_
t
y
p
e

=

3

t
h
e
n

"
w

e
i
g
h
t
"

w
h
e
n

d
a
t
a
_
t
y
p
e

=

7

t
h
e
n

"
s
t
e
p
s
"

w
h
e
n

d
a
t
a
_
t
y
p
e

=

8

t
h
e
n

"
d
i
s
t

i
n

m
"
when
data_ty

pe = 9
then
"resting
energy"
when
data_ty
pe = 10
then
"active
energy"
when
data_ty
pe = 12
then
"flights
climbed
" when
data_ty
pe = 67
then
"weekly
calori
goal"
when
data_ty
pe = 70
then
"watch
on"
w
h
e
n

d
a
t
a

_
t
y
p
e

=

7
5

t
h
e
n

"
s
t
a
n
d
"

w
h
e
n

d
a
t
a
_
t
y
p
e

=

7
6

t
h
e
n

"
a
c
t
i
v
i
t
y
"

w
h
e
n

d
a
t
a
_
t
y
p
e

=

7
9

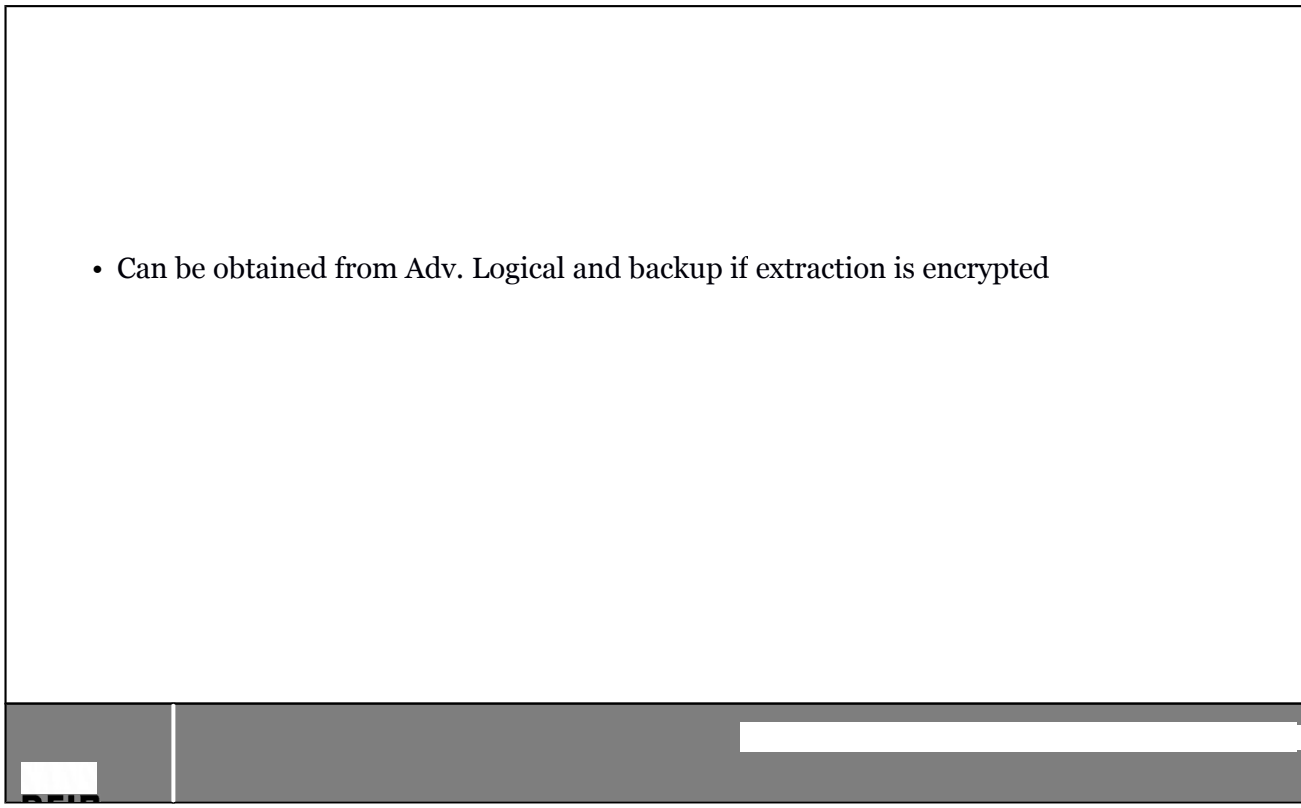
t
h
e
n

"
w
o
r
k
o
u
t
"
when
data_ty
pe = 83
then
"some
workou
ts" end
as
"activit
y type“,
quantity AS "Flights"

from samples

left outer join quantity_samples on
samples.data_id = quantity_samples.data_id
left outer join unit_strings on
quantity_samples.original_unit =
unit_strings.RowID left outer join

```
correlations on samples.data_id =
correlations.object
left outer join metadata_values on
metadata_values.object_id =
samples.data_id  left outer join
metadata_keys on metadata_keys.ROWID =
metadata_values.key_id where "Start Date"
like '%2017-01-19%'
order by "Start Date" desc
```



Proving that a device was in use may aid in your investigation. Obviously, you can search for powering events, determine if the device was unlocked, see if any activity was synching or updated, search for application and browser activity, and even see the last time it was booted. Some of these artifacts will require a full file system extraction. You should be able to dig far enough into the evidence that you can determine device usage. As always, refer to your cheat sheet for more information. There are so many artifacts left behind. Remember, every contact

leaves a trace.

The file /Library/CoreDuet/People/interactionC.db shows datetimes (both start and end), application used, sender (if relevant), and more. This file is parsed by most tools and associated to the activity in question.

/private/var/networkd/db/netusage.sqlite

Table of interest:
znetworkattachmentzkind 1 is Wi-Fi, 2 is Cellular.

A sampling of files that prove device activity include:

- /SystemConfiguration/preferences.plist
- /Library/CoreDuet/coreduetd.db: lock/device passcode
- /Library/Preferences/com.apple.identityservices.idstatuscache.plist
- /preferences/SystemConfiguration/com.apple.networkidentification.plist
- /Library/Preferences/
  - com.apple.aggregated.plist (Last boot time—not necessarily last time used!)
  - com.apple.contacts.donation-agent.plist (last metrics attempt datetime)
  - com.apple.contextstored.plist (activity throttling dates)
  - com.apple.contextstored.plist (activity throttling dates)

- /mobile/Library/CoreDuet/People/interactionC.db
- Requires at least an encrypted iTunes or Advanced logical extraction to obtain this file

Most tools parse InteractionC.db. In this slide, we see that AXIOM parses InteractionC.db for Contacts and Interactions. The Contacts are shown in the slide.

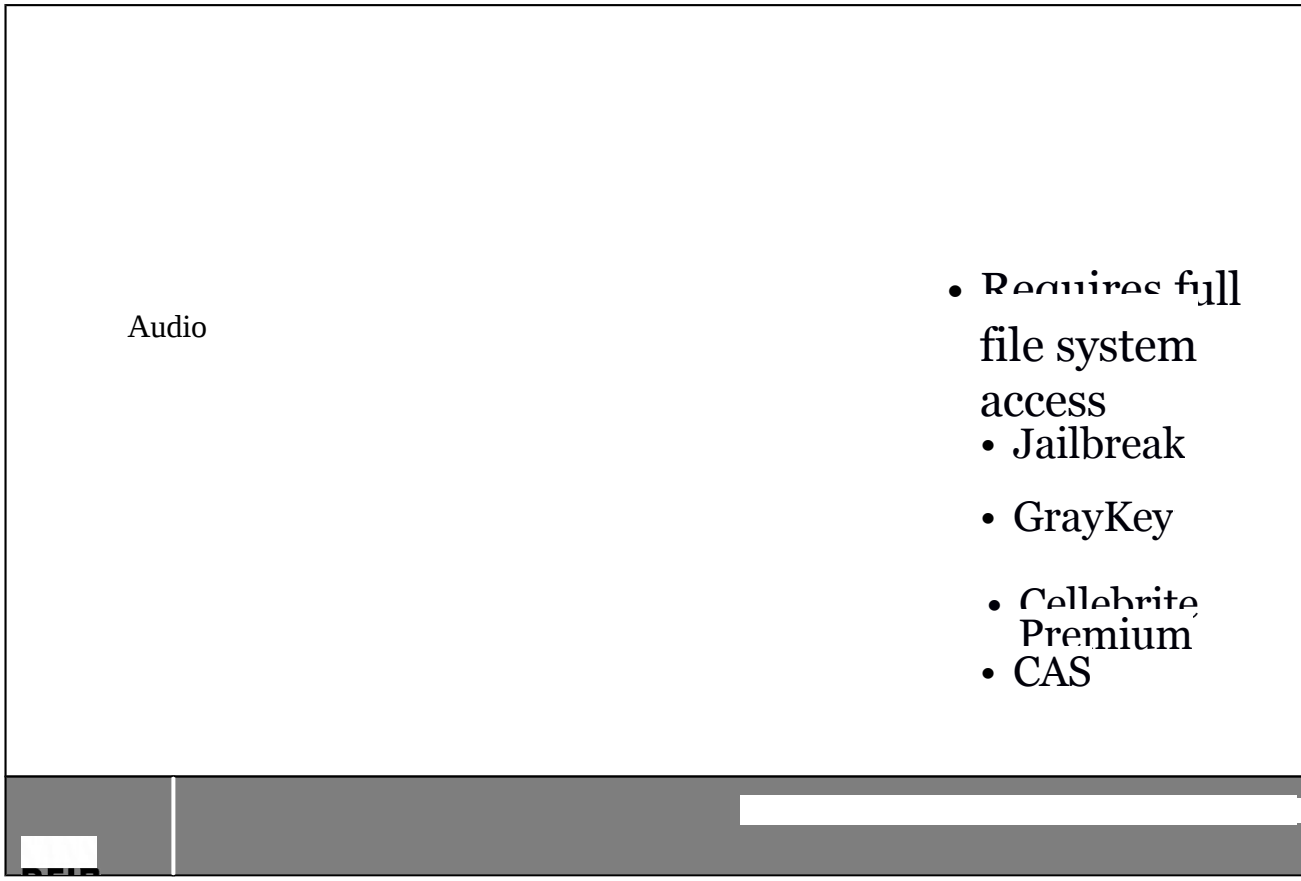
- Identity service is used by Apple to validate user credentials as they traverse Apple's ESS
- May contain deleted messages, FaceTime, and email
  - Metadata only, no content
- Contains metadata for many things:
  - iCloud
  - Fitness

The com.apple.identityservices.idstatuscache.plist is located at /var/mobile/Library/Preferences/com.apple.identityservices.idstatuscache.plist. This file is used by Apple to confirm the user's credentials as it travels across Apple's ESS (Enterprise Shared Services). This file stores so much information and it should be examined for every iOS device you come across. You will find account information, iCloud sync dates, metadata (email addresses, phone numbers) and a lookup date for each item. While content isn't available, this file may provide keywords and clues that are invaluable to your investigation.

The screenshot on the right shows an example of the last LookupDate for

FaceTime Audio. More research is required on this file, but the artifacts may provide hints on where to dive deeper in other files on the iOS device.

*Special thanks to Detective Chad Gish for uncovering this incredible artifact!

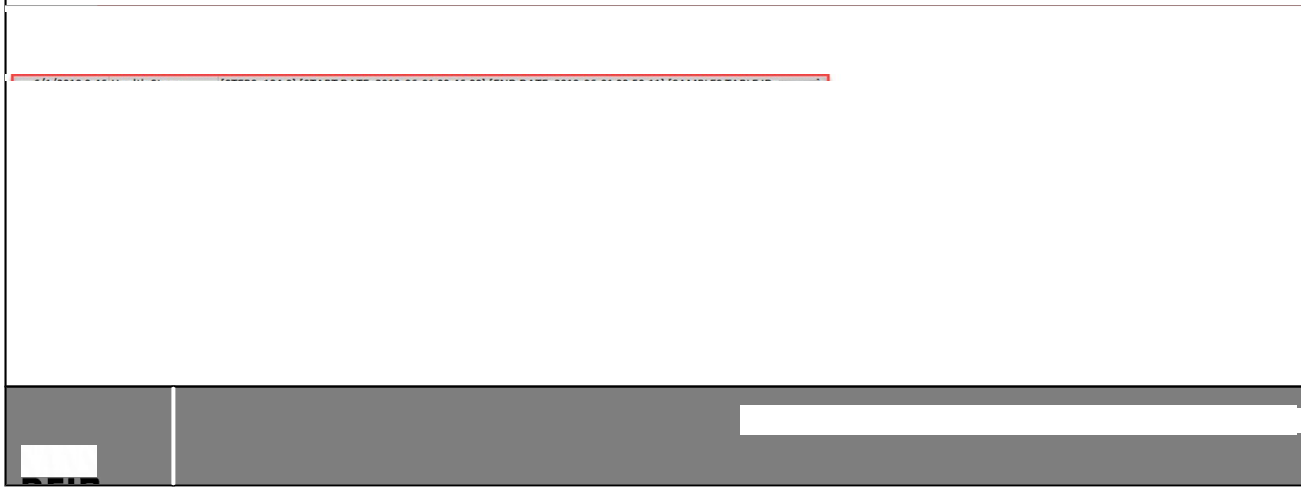


The KnowledgeC data base is extremely important to forensic investigations, but it does require additional access to obtain a full file system dump. This file is located in /private/var/mobile/Library/CoreDuet/Knowledge directory. You will find that this file is a merge of system artifacts and user artifacts that may be useful to your investigation.

Sarah Edwards wrote a helpful blog, which can be read here: https://www.mac4n6.com/blog/2018/8/5/knowledge-is-power-using-the-knowledgecdb-database-on-macos-and- ios-to-determine-precise-user-and-application-usage. Another helpful blog is explained by the view of Ian Whiffin and can be found here: https://doubleblak.com/blogPosts.php?id=2

As with most of these artifacts, the commercial tools will parse it as will iLEAPP and APOLLO. You will be examining this file in your upcoming lab.

- Developed by Sarah Edwards to help tie multiple artifacts together
- May need a full file system dump or jailbreak to obtain all relevant files
- Some files are still worth running to get a summary of activity



APOLLO is provided inside of your VM in Scripts for Class. Make sure you update to the current version, as needed because bugs are fixed as quickly as Sarah can manage. The command line shows how to run APOLLO on an iPhone. The first step is to export databases of interest from your tool of choice and then run APOLLO against that directory. Some tools have even implemented APOLLO, so the export isn't required. I selected to run "yolo" as the version because I assumed the user updated. A sampling of the results is provided.

iLEAPP is a community tool developed by Alexis Brigoni. His Twitter, Github and blog are shown in the slide. Alexis is always pushing to fill gaps and help the community extract data of interest. iLEAPP should be used in the next lab as you will easily uncover artifacts that you may have missed.

- Do they fit the timeline?
- Do they make sense?

- iOS is not consistent with data retention
- Understand that you may not be able to uncover all artifacts

This page intentionally left blank.

- What's draining the battery?
  - /Library/BatteryLife/CurrentPowerlog.PLSQL
- What apps are frequently used?
  - /Library/BatteryLife/CurrentPowerlog.PLSQL

Powerlog archives can be useful when attempting to track malware on an iOS device. CurrentPowerlog.PLSQL contains dates and times that can be associated with applications running in the background, applications draining the battery and surviving if the user deletes the app. This file should be examined in every investigation. Even if malware isn't a concern, this file tracks so many items that are logged without user knowledge. The data-retention time for each table may differ; iOS version and available space may determine how much is stored in this archive. For more information on this archive, refer to Sarah Edwards' presentation (https://for585.com/518).

Lab 3.3

iOS Forensic Analysis with Scripts

This page intentionally left blank.